Il Mainframe

Una grande tecnologia nel rispetto dell'ambiente

A. Barbarino
A. Corona
M. Moretti
F. Quarta
C. Troncone

Seconda Edizione – Novembre 2008

Indice

Gli autori

Angelo Barbarino Senior Consulting I/T Specialist del Systems & Technology Group, IBM Italia. Dal 2005 è impegnato a favorire la collaborazione didattica con le università italiane, promuovendo il corso "Sistemi Centrali" del quale ha tenuto molte edizioni in diversi atenei. Contribuisce alla realizzazione di progetti di ottimizzazione di infrastrutture informatiche a livello europeo. Laureato in Fisica a Catania nel 1981, segue da più di 25 anni il settore informatico. È membro del Technical Expert Council di IBM Italia e partecipa a diverse comunità e gruppi di lavoro a livello internazionale.

Andrea Corona I/T Specialist del Systems & Technology Group, IBM Italia. Ha responsabilità tecnico-commerciali sui sistemi z e si occupa dell'introduzione di nuove tecnologie in questa piattaforma. Laureato in Ingegneria Elettronica nel 2002 presso l'Università di Cagliari, è in IBM dal 2003 – inizialmente nella divisione di consulenza IBM Global Business Services – e realizza progetti complessi per clienti di diversi settori eterogenei. Partecipa a team di ricerca sulle performance e sulla sicurezza delle applicazioni Web J2EE e Linux; segue la valutazione delle prestazioni applicative e di sistema di soluzioni Linux su piattaforma z. E' coautore del libro "IBM System z Strengths and Values" e collabora alla IBM Academic Initiative.

Mario Moretti Si laurea in Matematica con indirizzo informatico, collaborando con il Centro Scientifico IBM. Attualmente è impegnato come I/T Specialist all'interno della struttura Systems & Technology Group, IBM Italia. Partecipa alla IBM Academic Initiative a supporto delle conoscenze e collaborazioni con l'ambiente universitario.
Dal 2007 tiene il corso "Sistemi Centrali" presso l'Università degli Studi La Sapienza di Roma.

Filippo Quarta Referente tecnico per i nuovi workload su IBM System z presso il Systems & Technology Group, IBM Italia. E' entrato in IBM nel 2007 svolgendo attività tecnico-commerciali in ambiente mainframe e zLinux. In precedenza ha ricoperto il ruolo di ricercatore industriale in Telecom Italia contribuendo allo sviluppo di diverse piattaforme per servizi mobili a valore aggiunto e partecipando a progetti di ricerca legati alla convergenza delle reti. Si è laureato in Ingegneria Informatica nel 2001 presso l'Università di Lecce.

Corrado Troncone Dal 2000 svolge attività di Brand Technical Support sui sistemi z presso il Systems & Technology Group, IBM Italia. Dopo aver concluso il biennio all'Università Federico II di Napoli, Facoltà d'Ingegneria, entra in IBM Italia nel 1983 come Tecnico di Manutenzione Software sui Sistemi Centrali di architettura IBM/370, sistemi operativi MVS e VM. Viene assegnato presso il Laboratorio di sviluppo VM a Kingston, NY(USA) e successivamente svolge attività sistemistica sui mainframe presso i clienti con progetti complessi. Partecipa alla IBM Academic Initiative per la diffusione delle conoscenze dell'architettura dei Sistemi Centrali presso le Università italiane.

Ringraziamenti

Gli autori ringraziano i colleghi **Fulvio Capogrosso,** Distinguished Engineer IBM Italia per la Premessa e l'Appendice dedicata ai Sistemi Ibridi; **Massimo Leoni,** Executive I/T Architect IBM Italia per l'Appendice dedicata a Sicurezza e Crittografia; **Gaetano Maretto**, Senior Consulting IT Specialist IBM Italia per il sostanziale supporto alla revisione tecnica del testo; **Delia La Volpe,** System z Technical Sales Support IBM Italia per il supporto alla revisione tecnica e per l'attenta e paziente rilettura del testo onde predisporlo per la stampa.

Premessa

[a cura di Fulvio Capogrosso]

La seconda edizione di questo volume "Il Mainframe" è motivata dal grande successo della prima edizione del 2007 con più di mille copie distribuite all'interno e all'esterno di IBM, dalla necessità di apportare miglioramenti al testo ed aggiornamenti di contenuto resi necessari dall'evoluzione tecnologica avvenuta nel corso dell'anno e dall'esigenza di raggiungere un pubblico ancora più numeroso anche a seguito della crescente popolarità della tecnologia del mainframe nell'industria informatica.

Le principali variazioni rispetto alla prima edizione riguardano i seguenti argomenti:

- *il mainframe come elemento di ottimizzazione dell'infrastruttura informatica poiché utilizza tecniche di virtualizzazione e di server consolidation con le relative metodologie di configurazione e criteri di scelta delle piattaforme elaborative*
- *il ruolo del sistema operativo zLinux*
- *il ruolo del sistema operativo z/VM*
- *la tecnologia di workload management del mainframe*

*La parte più propriamente tecnologica è stata aggiornata e comprende anche le innovazioni introdotte con **i nuovi modelli IBM System z10 presentati a febbraio e ottobre 2008**. Questi nuovi sistemi confermano la oramai consolidata tradizione di costante evoluzione tecnologica (tecnologia di base, packaging, capacità di input/output, potenza complessiva e strumenti per migliorare l'usabilità e la disponibilità) e introducono significativi elementi di innovazione quali una maggiore integrazione fra hardware e software per migliorare le prestazioni in ambienti altamente virtualizzati, la disponibilità di nuove istruzioni per l'esecuzione di operazioni floating point di tipo decimale come richiesto dagli standard internazionali e la possibilità di integrazione con processori specializzati tipo il Cell Broadband Engine (Cell BE) per l'esecuzione di applicazioni altamente innovative. Il Sistema z10 ci permette di affermare, riecheggiando il tema dell'annuncio ('The future runs on System z') che la*

tecnologia del mainframe, cui questo volume è dedicato, si conferma in posizione di assoluta leadership nel panorama dell'informatica moderna e, pur mantenendo un forte grado di compatibilità con le applicazioni esistenti, si propone come la piattaforma più adatta a soddisfare le sempre più complesse esigenze applicative e di sistema del prossimo futuro.

Il Mainframe

I Mainframe rappresentano una delle più rilevanti realizzazioni tecnologiche del ventesimo secolo. Definita nei suoi elementi di base dalla IBM negli anni '60, la tecnologia del Mainframe ha continuato nel corso degli anni ad arricchirsi di funzionalità sempre più avanzate e di innovazioni tecniche via via più sofisticate sempre anticipando o assecondando le esigenze di un'industria informatica in continua espansione. Oggi i Mainframe sono presenti - ed hanno un ruolo insostituibile in tutto il mondo - nelle infrastrutture informatiche delle più importanti aziende industriali e finanziarie, nelle società di servizi pubbliche e private e nelle grandi istituzioni nazionali ed internazionali. Il termine stesso "Mainframe" è diventato sinonimo di elaboratore di grandi capacità elaborative ed alto livello di qualità del servizio. Fare di questa tecnologia materia di studio nelle aule Universitarie è un progetto di grande valore e di vasta portata che offre agli studenti un programma di fondamenti avanzati di informatica con l'importante obiettivo di trasferire concetti e tecniche di grande valore formativo.

*Da un punto di vista storico la tecnologia del Mainframe nasce nel 1964 quando la IBM costruisce e commercializza gli elaboratori della famiglia S/360 determinando un momento di forte discontinuità ed innovazione nel mondo dell'informatica di allora. Molti sono gli elementi tecnologici di grande rilievo caratteristici del Mainframe e questo corso di lezioni universitarie ne fornisce una ampia panoramica con riferimento allo stato dell'arte più recente. Per comprendere a fondo l'importanza ed il valore di questa tecnologia è importante però soffermarci preliminarmente su alcuni punti qualificanti che fin dalla nascita ne determinarono l'assoluta unicità. Ci riferiamo alla definizione di una "**architettura**" di riferimento come livello di astrazione superiore rispetto agli elementi implementativi ("**il***

__disegno"__); alla promessa di una "__compatibilità binaria__" delle applicazioni (i "__programmi"__) estesa nel tempo a protezione degli investimenti degli utenti, e alla nozione di elaboratore "__general purpose__" cioè in grado di eseguire contemporaneamente ed in modo efficiente programmi ed applicazioni di tipo generalizzato. Tutto ciò rappresentò, allora, un grande momento di discontuinità nei confronti di una realtà esistente in cui i dettagli implementativi dell'elaboratore erano visibili al programmatore e di conseguenza i programmi risultavano legati ad una particolare implementazione e non facilmente esportabili su implementazioni diverse. Inoltre gli elaboratori, allora, erano di norma "specializzati", cioè venivano progettati e realizzati avendo come riferimento una particolare tipologia applicativa (per esempio gli elaboratori specializzati per il calcolo scientifico).

Il primo grande elemento di novità degli elaboratori della serie S/360 è la presenza di una Architettura. L' "Architettura" di un elaboratore ne definisce, ad un elevato livello di astrazione, i principi e le norme di funzionamento. Essa contiene, fra l'altro, le regole di indirizzamento della memoria, l'insieme delle istruzioni che la macchina è in grado di eseguire con il dettaglio del risultato atteso dall'esecuzione delle istruzioni stesse, l'insieme dei registri, i meccanismi ed i codici di interruzione del programma. Descritta per esteso in un documento pubblico dal titolo "Principles of Operations", l'architettura permette al programmatore di creare programmi eseguibili su tutti gli elaboratori che fanno riferimento a quella architettura (elaboratori "compatibili") senza dover necessariamente conoscere i dettagli costruttivi o la implementazione dell'elaboratore. Analogamente, per quanto riguarda i progettisti e gli ingegneri addetti al disegno ed alla costruzione degli elaboratori, avere le regole dell'architettura come unico vincolo da rispettare consente ampia libertà di scelta fra le varie soluzioni possibili permettendo, per esempio, di costruire intere famiglie di elaboratori, tutti compatibili con l'architettura di riferimento, ma con diversi livelli di prestazione e di costo.

Il secondo elemento di grande novità introdotto con la famiglia S/360 è la promessa, fatta dal costruttore, in questo caso la IBM, di mantenere nel tempo la "compatibilità binaria" dei programmi eseguibili, ovvero delle applicazioni. In pratica il costruttore fornisce agli utenti la garanzia che ogni

futura estensione dell'architettura avverrà sempre in modo "compatibile (con il passato)" ovvero senza richiedere modifiche o adattamenti dei programmi esistenti per una corretta esecuzione degli stessi con la nuova architettura. Quando possibile, i vantaggi della nuova architettura verrano forniti alle applicazioni dai programmi di controllo dell'elaboratore, anche in questo caso senza richiedere rifacimenti applicativi. Questa garanzia, mai venuta meno nel corso degli anni e tuttora valida in IBM dopo numerose evoluzioni ed estensioni, anche di notevole portata, dell'architettura del Mainframe, ha di fatto favorito i massicci investimenti che hanno prodotto, nel tempo, quel grande patrimonio applicativo su cui si basano oggi i principali processi informatici delle più importanti aziende nel mondo. Tale patrimonio applicativo viene comunemente detto anche "legacy applications" proprio per indicare il ruolo insostituibile che tali applicazioni hanno. Per dare una dimensione numerica a questo fenomeno basti ricordare che oggi si valuta esistano al mondo circa 200 miliardi di linee di codice COBOL (uno dei linguaggi di programmazione più diffusi nell'industria e nel mondo del mainframe) con una crescita annuale valutata intorno ai cinque miliardi di nuove linee di codice e per un valore totale delle "legacy" applications stimato intorno ai mille miliardi di dollari.

Infine, un altro grande elemento qualificante del mainframe è la sua caratteristica di elaboratore "general purpose" ovvero di utilizzo generalizzato che lo differenzia da altri elaboratori disegnati per eseguire solo particolari tipologie di applicazioni. L'architettura, il disegno ed il programma di controllo del mainframe sono stati da sempre realizzati per permettere l'esecuzione contemporanea ed efficiente di un gran numero di programmi, siano essi di tipo transazionale, interattivo o batch, e siano essi ad alto contenuto di elaborazione o ad alto contenuto di operazioni verso dispositivi esterni. Tutto ciò nel rispetto delle priorità di esecuzione definite dall'utente e dell'esigenza di garantire sempre e comunque un alto utilizzo delle risorse fisiche in dotazione all'elaboratore, siano esse le unità di esecuzione, la memoria o i dispositivi di accesso alle unità esterne. La capacità di eseguire contemporaneamente un gran numero (anche migliaia) di programmi richiede, ovviamente, la presenza di un potente e rigoroso apparato di sicurezza, ove per sicurezza si intende sia il controllo dell'accesso alle risorse dell'elaboratore - siano esse i programmi, la memoria o le informazioni - e sia la garanzia che a nessuno, utente o

programma, interno o esterno, sia consentito accedere a quelle risorse senza una preventiva identificazione ed autorizzazione.

L'insieme delle tecnologie e dei meccanismi di controllo del Mainframe hanno nel tempo raggiunto livelli di sofisticazione e di efficienza estremamente elevati e costituiscono oggi un punto di forza e di differenziazione del Mainframe rispetto alle alternative presenti sul mercato. Uno dei motivi alla base della caratteristica "general purpose" del Mainframe è il forte impegno economico richiesto negli anni '70 per l'acquisto di un elaboratore. Si riteneva pertanto opportuno costruire un elaboratore in grado di soddisfare contemporaneamente ogni tipologia applicative, superando di fatto la necessità, economicamente non sostenibile, di acquistare elaboratori diversi per esigenze diverse. Nasce così l'informatica cosiddetta "centralizzata" ovvero centrata su un unico (grande) elaboratore (da cui anche il termine "Mainframe").

Negli anni '80 e '90 la produzione industriale di microprocessori, con la conseguente disponibilità sul mercato di elaboratori con funzionalità e costi relativamente ridotti ed i continui progressi nelle tecnologie di comunicazione, hanno modificato in modo sostanziale i tradizionali criteri di disegno delle infrastrutture informatiche orientandosi verso strutture "distribuite" ove la potenza elaborativa è fornita da un numero variabile (anche elevato) di elaboratori di varia dimensione ed architettura, distribuiti anche geograficamente e variamente interconnessi. Si parla in questo caso di infrastrutture informatiche eterogenee e distribuite.

Negli anni più recenti la crescita continua ed anche talvolta incontrollata delle dimensioni delle infrastrutture informatiche distribuite ha richiamato l'attenzione degli addetti ai lavori sugli alti costi, in prospettiva insostenibili, richiesti per la gestione di tali infrastrutture e sui problemi indotti in aree quali la sicurezza informatica e la qualità del servizio. Da qui una rinnovata attenzione verso i Mainframe e verso i molteplici elementi di valore impliciti in quella tecnologia. Appare infatti in modo sempre più evidente il ruolo insostituibile del Mainframe come elemento centrale di quelle infrastrutture informatiche tecnologicamente avanzate ed economicamente sostenibili di cui c'è oggi grande richiesta in ogni settore di mercato.

Il ciclo di lezioni universitarie da cui è stata tratta questa pubblicazione fornisce un quadro ampio e dettagliato della tecnologia del Mainframe, affrontando sia i principi ed i componenti di base hardware e software sia anche alcuni degli elementi accessori di maggiore interesse nel panorama informatico attuale, quali la virtualizzazione e l'ampio supporto degli standard informatici internazionali e di mercato.

1.0 Introduzione ai Sistemi Centrali

Da più di quarant'anni una parte significativa dei processi produttivi e dei servizi viene supportata da una infrastruttura informatica costituita dai Sistemi Centrali. Fino a metà degli anni '70 la storia stessa dell'informatica e dei sistemi operativi si identificava con tali sistemi anche detti "Mainframe". La maggior parte delle istituzioni private o pubbliche, nei più disparati settori della economia o dei servizi, basano la parte più delicata e strategica della loro attività su computer di questo tipo.

I Sistemi Centrali sono frutto di ingenti investimenti da parte di case produttrici e oggi attraversano una significativa fase di innovazione proiettata verso un loro inserimento in scenari eterogenei ed "aperti" per tecnologie e caratteristiche.

Risulta singolare che esista poca informazione su testi divulgativi ed in internet su tali argomenti ad eccezione di quella espressa dalle case produttrici.

Obiettivo di questo testo è di fornire un contributo alla descrizione di alcuni aspetti di questa materia.

Allo scopo di presentare argomenti più complessi, nei capitoli successivi introdurremo alcuni argomenti basilari per condividere una terminologia che è ampiamente in uso nell'ambito dei sistemi centrali.

1.0.0 Definizioni

Sistema Centrale

Si definisce Sistema Centrale un calcolatore dotato di elevata capacità elaborativa usato per gestire grandi volumi di dati con un grande livello di sicurezza ed affidabilità. Tale sistema è acceduto da un alto numero di utenti contemporaneamente. Esso gestisce un carico di lavoro misto eseguendo attività tra loro differenti per tipologia e impegno elaborativo.

Figura 1 Sistema centrale

Queste attività vengono eseguite in contemporanea mantenendo priorità assegnate ed armonizzando le varie necessità elaborative.

Il Sistema Centrale si contrappone a server monouso, in grado cioè di svolgere un solo tipo di workload (firewall, router, file server, print server, ecc.).
Lo scenario in cui il sistema centrale si inserisce è cambiato nel tempo. Inizialmente il sistema centrale accentrava su di sé tutte le funzioni informatiche (controllo degli accessi, elaborazioni con calcoli numerici complessi – ad esempio scientifiche – gestione delle reti ed altro). Oggi il sistema centrale è collocato in una configurazione di server in cui una serie di funzioni sono demandate a server monouso poiché questa infrastruttura è quella più efficiente dal punto di vista dei costi. Possiamo dire che nei sistemi centrali sono rimaste tre funzioni fondamentali: Data base server, Application server e Presentation server. Questi concetti verranno sviluppati in seguito. Questa cooperazione è stata resa possibile poiché i sistemi centrali hanno implementato tutti gli standard adottati e richiesti dal mercato IT, siano essi de jure o de facto (TCP/IP, Linux, POSIX, Java, XML ecc.); in questo modo l'integrazione dei sistemi centrali all'interno di una rete di elaboratori eterogenea risulta facile e completa.
Il Sistema Centrale per eccellenza e per anzianità di servizio è il Mainframe. La sua storia verrà descritta più avanti. Basti dire che oggi il termine Mainframe è sinonimo di elaborazione sicura, affidabile, autonomica e flessibile. A partire da metà degli anni '90 diversi produttori di computer hanno introdotto dei modelli di sistemi centrali esaltandone la maggiore o minore affinità con le caratteristiche del Mainframe, sebbene utilizzino tecnologie e architetture diverse.

Infrastruttura Informatica

Si definisce Infrastruttura Informatica l'insieme di hardware, software e collegamenti che implementano lo scenario in cui categorie differenti di utenti accedono a risorse elaborative quali dati e programmi. Spesso il disegno di una infrastruttura informatica rappresenta un processo continuo di adattamento di situazioni attuali e prospettive future. Tuttavia alcuni paradigmi, quali flessibilità ed affidabilità della struttura informatica, costituiscono punti fermi del progetto di una infrastruttura informatica: quest'ultima non deve mai rappresentare un ostacolo alla dinamicità dei processi di business che creano il vantaggio competitivo tra le aziende.

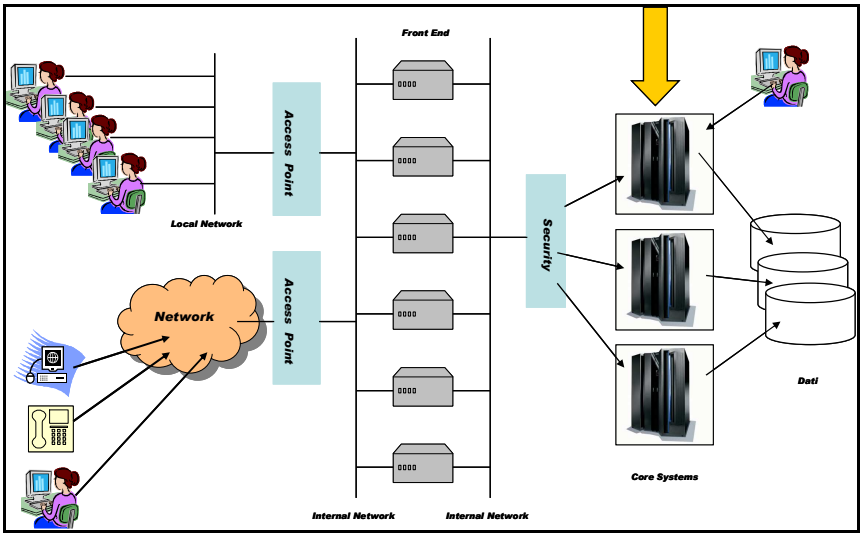


Figura 2 Infrastruttura informatica

Piattaforma informatica

Si definisce Piattaforma informatica l'insieme di un calcolatore e di un sistema operativo. Tale termine consente di definire uno scenario elaborativo identificando le caratteristiche salienti dell'hardware e del software che su di esso opera.

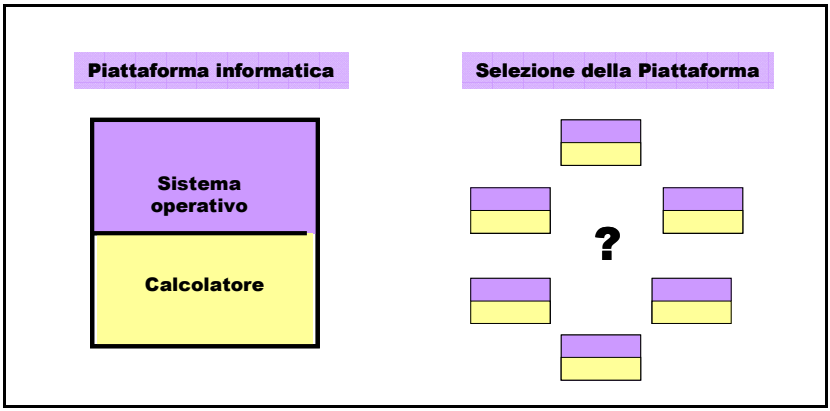


Figura 3 Piattaforma informatica

A tal riguardo esiste un processo di valutazione di una o più piattaforme all'interno di una infrastruttura informatica. Questo processo chiamato "selezione della piattaforma" si basa sull'insieme dei requisiti richiesti per una certa infrastruttura e sulle caratteristiche dei prodotti in grado di rispondere a tali esigenze.

Al compito tecnologico (Selezione della Piattaforma) si abbina la valutazione del costo di acquisizione di tali prodotti, detto TCA (Total Cost of Acqusition), e del costo di gestione, detto TCO (Total Cost of Ownership) che comprendono l'esercizio e la manutenzione dell'attrezzatura.

1.0.1 Modelli Applicativi

In questo paragrafo introduciamo i modelli infrastrutturali che maggiormente hanno caratterizzato l'evoluzione dei sistemi centrali negli anni recenti. Questi cambiamenti sono dovuti alle trasformazioni subite dalle organizzazioni aziendali e al rapporto fra queste e il mercato in cui operano.
A partire dalla fine degli anni '90 infatti, con l'esplosione del fenomeno internet e dei mercati globali, è emerso un nuovo disegno di azienda, non più monolitica e chiusa, ma "aperta" e con interazioni sempre più strette con agenti esterni (fornitori, clienti, ecc.), flessibilità sempre più spinte e in grado di riposizionare costantemente la propria attività. Tutto ciò, supportato dalle nuove tecnologie, ha influito sui modelli infrastrutturali.

I modelli infrastrutturali sono quattro:

- Modello Host Centrico
- Modello Client/Server
- Modello Web Application
- Modello Service Oriented

Modello Host Centrico

Il modello host centrico è legato alla visione del sistema centrale come "hub" di applicazioni e dati all'interno del quale vengono eseguite le principali attività in completa autonomia e con scarse interazioni dei processi con enti esterni. E' un sistema intrinsecamente chiuso che molto spesso utilizza interfacce e applicazioni fatte in casa e ha una limitata adesione agli standard informatici.
Esso ha una flessibilità molto ridotta poiché ogni innovazione e modifica deve essere costruita "ad hoc". Gli utenti che si connettono al sistema sono già ben identificati (login e password) ed in genere sono dipendenti dell'azienda. La interazione con internet è molto bassa e in genere non legata alla cooperazione di processi aziendali. La gestione del sistema è

affidata in genere a tecnici con specializzazione consolidata sui sistemi centrali ma limitata nelle nuove tecnologie.

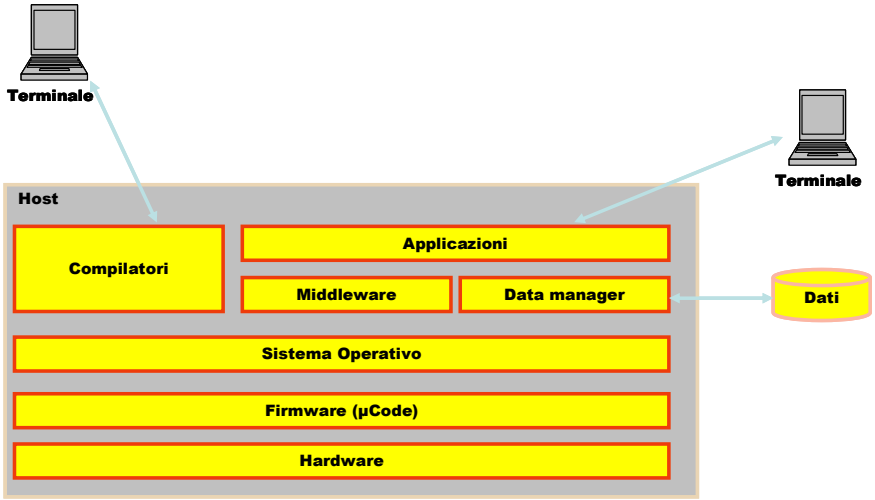


Figura 4 Modello applicativo Host Centrico

Modello Client/Server

Allo svilupparsi sempre più frenetico delle reti, del contenimento dei costi per il trasferimento di dati in rete e lo svilupparsi della cooperazione dei processi in rete è emerso (con diverse modalità) il modello Client/Server. In accordo con questo modello, un sistema monolitico viene progressivamente suddiviso in processi client (cioè richiedenti) e processi server (cioè fornitori di servizio) allocati su unità elaborative logicamente (e spesso anche fisicamente) distinte. L'elemento centrale del sistema è la rete. Tipici server sono i server dati (DBMS), i server di applicazioni, i server di sviluppo, dove ad esempio vengono eseguiti i compilatori; l'utenza, non è più necessariamente preidentificata e utilizza interfacce grafiche (GUI) per accedere alle applicazioni.

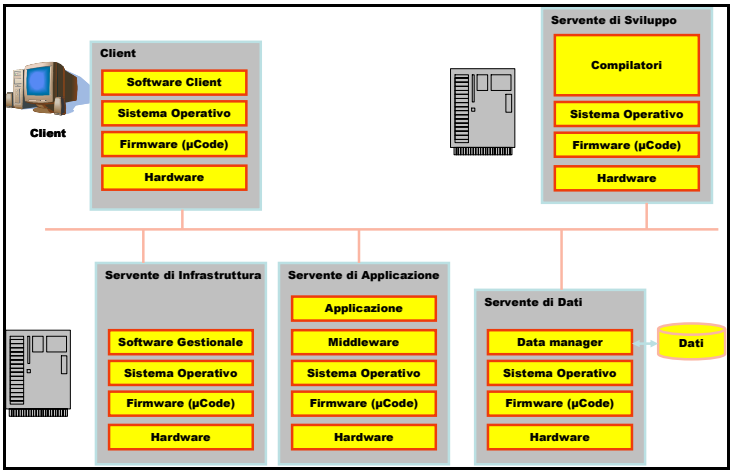


Figura 5 Modello applicativo Client/Server

In sintesi su una rete (locale o meno) si attestano diversi Personal Computer dotati di GUI, che nell'insieme fanno una parte o tutto ciò che in precedenza veniva svolto da un grande calcolatore mainframe. I programmi client e server presentano delle "rigidità", ad esempio nella non standardizzazione dell'interfaccia GUI, ma il più importante svantaggio di questa soluzione è nel non efficiente utilizzo delle risorse disponibili. Ad esempio se un server è carico di lavoro (ad esempio per la gestione dei dati) rispetto ad un altro che è libero, quest'ultimo non può mettere le proprie risorse a disposizione del primo server generando significativi colli di bottiglia.

Modello WEB Application

Un passo avanti rispetto al modello Client/Server è rappresentato dal modello Web Application. In questo modello il client utilizza in genere come interfaccia un programma "internet browser" di internet (o un applet Java) ed i servizi di connettività applicativa sono forniti da un programma di tipo

"web server". In pratica si standardizza l'accesso alle applicazioni in maniera sempre più integrata con Internet e gli standard collegati. I protocolli di comunicazione TCP/IP e http sono alla base di questo modello.

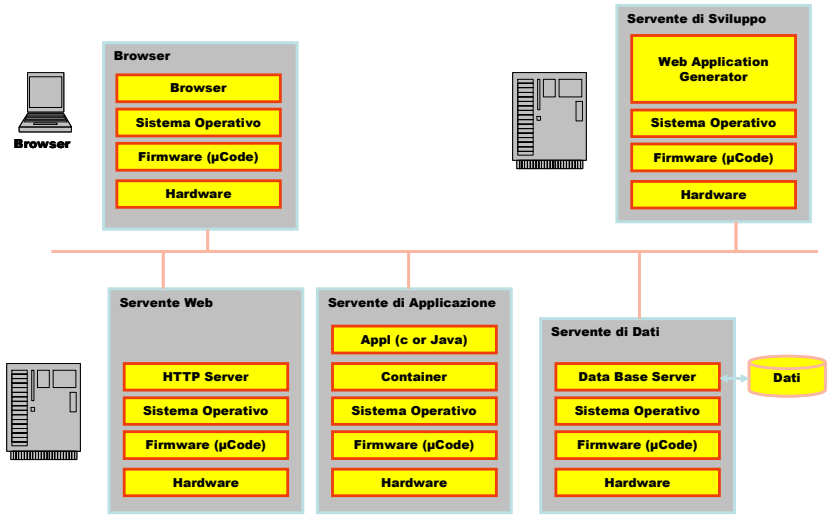


Figura 6 Modello applicativo Web Application

Nel modello, un importante ruolo è rivestito dai "contenitori" applicativi (Web Application Server) in grado di ospitare applicazioni scritte con linguaggi di programmazione ad oggetti (ad es. Java).

Modello Service Oriented

Un ulteriore passo avanti nel campo della flessibilità ed apertura è rappresentato dal modello SOA (Service Oriented Architecture), il cui scopo ultimo è il connettere una grande varietà di sistemi senza vincoli legati a software proprietario raggiungendo una vera interoperabilità.
Secondo questo modello è possibile far cooperare strettamente due programmi scritti in linguaggi differenti che sono eseguiti su sistemi

operativi differenti senza l'intervento di una qualche parte di codice proprietario e a costi accettabili.

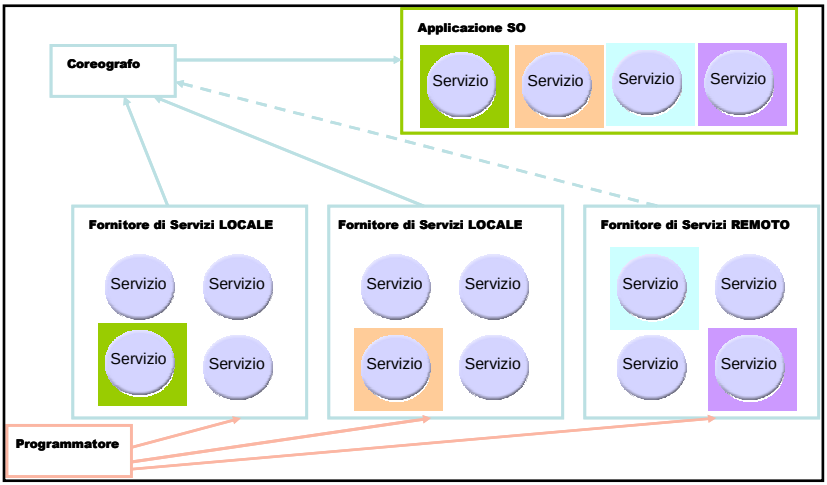


Figura 7 Coreografia SOA

Questa interoperabilità è oggi ancor più raggiungibile se la connessione operativa è realizzata attraverso il linguaggio di interscambio XML (Extended Markup Language).
Un'architettura a servizi, la SOA appunto, utilizza protocolli e linguaggi operativi standard (SOAP UDDI, WSDL, XML) e consente di realizzare un'applicazione complessa formata da servizi Web, indifferenti alla piattaforma informatica che li ospita, coordinati da un programma di "coreografia".

In altre parole questo modello gestito dal SOA Server (Server di Servizi) è un nuovo modo di implementare le applicazioni, viste come un insieme di servizi coordinati che una volta combinati offrono un servizio finale più complesso. Un vantaggio molto rilevante del modello a servizi è rappresentato dalla maggiore riusabilità dei componenti (Web Service) che rendono più veloce ed economica la realizzazione di nuove applicazioni.

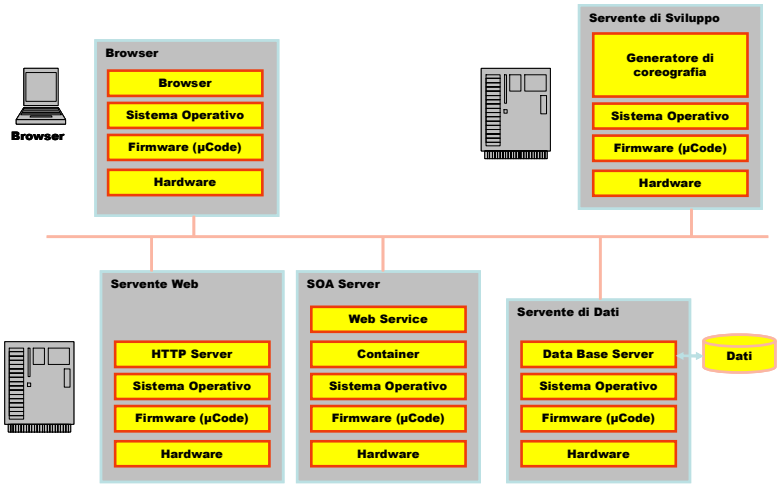


Figura 8 Modello applicativo SOA

Un esempio di utilizzo dell'architettura service oriented può essere un ente pubblico che espone sul web i risultati di applicazioni COBOL (software legacy). Le stesse applicazioni COBOL, opportunamente trattate, potrebbero essere rese disponibili alle applicazioni di altri enti mediante l'utilizzo delle interfacce standard SOA.

Il modello applicativo prevede l'esistenza di un contenitore logico di questi Web Service chiamato SOA server dotato di tutti i requisiti in termini di standard applicativi e di connessione. Esiste anche un generatore della coreografia che in ogni momento gestisce su server locali o remoti (allocati su piattaforme informatiche potenzialmente diverse) la sequenza dei processi applicativi necessaria per ottenere un dato risultato finale.

I Sistemi Centrali sono in grado di ospitare i quattro modelli applicativi descritti ovvero possono operare come un sistema Host Centrico, possono operare da Server di dati ed Applicazioni o entrambi, possono fungere da

Web server o Application o Data Server per una Web Application e possono infine coreografare e fornire servizi (ad es. Web Services) in una Architettura Orientata ai Servizi (SOA).

1.0.2 Tipi di Lavoro (Workload)

I sistemi centrali gestiscono contemporaneamente Workload profondamente diversi fra di loro. In altre parole, una singola istanza di sistema operativo controlla centinaia o migliaia di processi aventi finalità e caratteristiche diverse. Tutto ciò è possibile in quanto tali sistemi sono multiprogrammati. Una suddivisione classica dei lavori è quella che distingue i cosidetti processi o job Batch (elaborazione a lotti) da quelli delle transazioni o online. I primi sono in genere caratterizzati dal fatto che una volta immessi nel sistema hanno una bassissima interazione con l'utente. Esempi di questa tipologia sono le stampe di fatture o bollette di utenze, oppure le riorganizzazioni dei dati. La seconda tipologia di lavoro ovvero le transazioni online è invece caratterizzata da una vasta utenza che interagisce frequentemente con il sistema per richiedere e/o aggiornare dati contenuti in database attraverso programmi "ad hoc" chiamati transazioni.

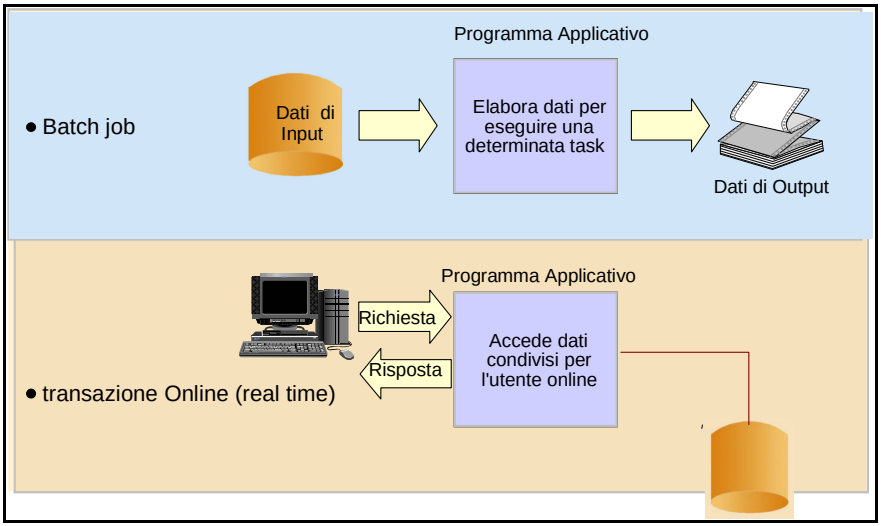


Figura 9 Tipologia di lavoro: Batch job e transazione online

Tutti questi lavori (batch e online) coesistono e usano le risorse del sistema centrale in base a politiche definite dagli specialisti che gestiscono il sistema. Queste politiche vengono poi elaborate da programmi speciali di sistema operativo, chiamati Schedulatori, che consentono di massimizzare il numero di processi per unità di tempo, garantendo i livelli di servizio definiti dai gestori del sistema..

1.0.3 Compiti e Ruoli

I sistemi centrali per la loro complessità ed importanza sono in genere gestiti da un pool di specialisti: la loro attività rende questi sistemi utili ed efficaci strumenti per il raggiungimento degli obiettivi di business. Gli investimenti in quest'area sono sempre cospicui e sono sempre confrontati con i guadagni di produttività e competitività.

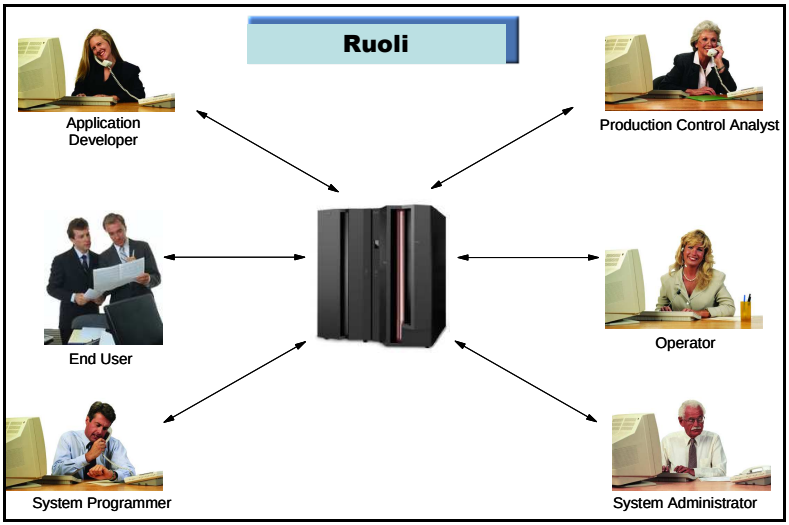


Figura 10 Compiti e Ruoli

Da questi punto di vista il personale dei sistemi informativi si può suddividere in:

Gestione e controllo del sistema

E' un gruppo di amministratori di sistema costituito da esperti di sistemi operativi, di gestione di dati, di gestione di reti, di gestione di programmi applicativi ed esperti di controllo della performance. A questi si possono aggiungere tutti coloro che si occupano del funzionamento operativo del sistema, cioè coloro che sono costantemente alla guida dei sistemi gestendo le console, rispondendo ai messaggi operativi, controllando il funzionamento dei dispositivi fisici operando in caso di errore e/o di ripartenze dei sistemi.

Sviluppo di nuove applicazioni

In questo reparto si trovano gli sviluppatori di applicazioni: coloro che analizzano le necessità applicative e disegnano le soluzioni, le implementano, le testano, le mettono in produzione e le mantengono.

Architetti di sistema

Sono le figure che disegnano le architetture informatiche, contattano consulenti indipendenti e/o esperti di aziende produttrici per valutare le possibili soluzioni in termini architetturali. In genere queste funzioni forniscono ai livelli decisionali tutte le informazioni che consentono gli investimenti in area informatica.

1.1 Architettura dei Sistemi Centrali

In questo capitolo vedremo le caratteristiche principali dei Sistemi Centrali dal punto di vista del disegno architetturale; quindi verrà descritta la z/Architecture, cioè l'architettura dei mainframe IBM: in particolare verrà descritta l'organizzazione dei Sistemi, il relativo Instruction set, la gestione della Memoria e dei dispositivi di Input/Output e la struttura dei sistemi Multiprocessori simmetrici.

1.1.0 Modelli Architetturali

Il modello architetturale secondo il quale sono progettati i Sistemi Centrali è basato sul modello della macchina di Von Neumann.

Gli elementi caratterizzanti di un calcolatore secondo Von Neumann sono:

- memoria centrale, che contiene sia i dati sia il programma da eseguire ("stored program")
- CPU, ovvero il processore che elabora le istruzioni, utilizzando un'unità aritmetico logica (ALU) e una di controllo
- dispositivi di I/O, utilizzati per inserire in memoria i dati d'ingresso di un programma e ricevere i risultati della elaborazione dalla memoria.

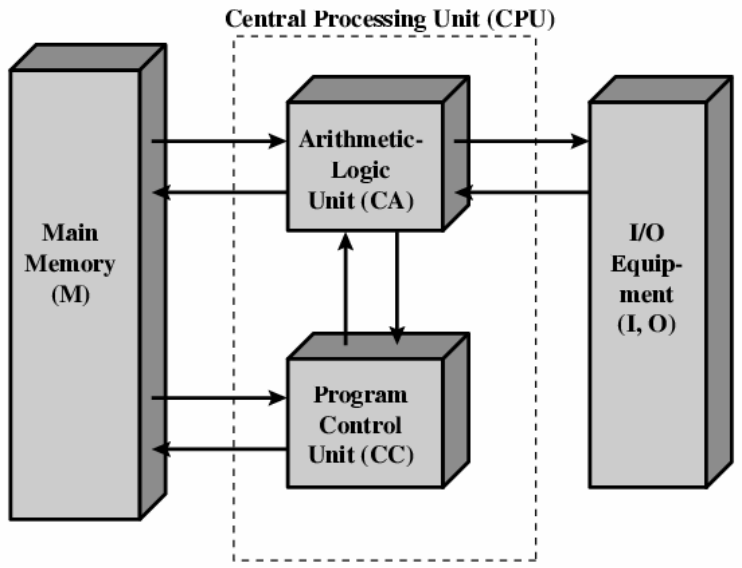


Figura 11 Struttura della macchina di Von Neumann

I modelli architetturali dei calcolatori differiscono generalmente ad alto livello per il numero di questi componenti e la modalità di interconnessione fra di essi.
Ad esempio, la struttura SMP (Symmetric Multi Processing), definita nel 1972, ha la caratteristica principale di utilizzare più di una CPU, con una sola memoria centrale ad accesso multiplo condivisa da tutti i processori.
La z/Architecture dei mainframe è basata sulla struttura SMP. I processori e la memoria sono collegati da Bus ad alta velocità. Nelle implementazioni hardware solitamente si interpongono, per ragioni di efficienza, tra le CPU e la memoria centrale memorie più veloci, chiamate "cache". È importante notare che la struttura gerarchica della memoria è una caratteristica implementativa e non architetturale.

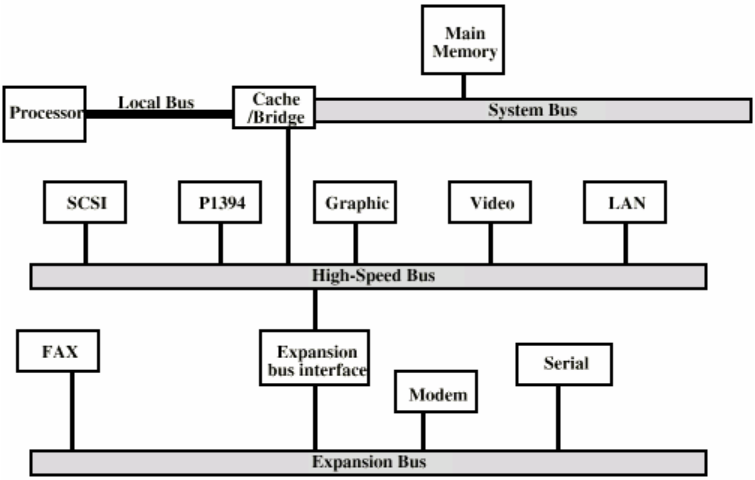


Figura 12 Schema di collegamento a Bus multiplo

Per quanto riguarda la comunicazione tra i dispositivi di I/O ("devices"), le CPU e la memoria, solitamente si utilizzano i Bus, sostanzialmente dei collegamenti elettrici con circuiteria di controllo collegata. Per questioni di performance, specie se si collegano dispositivi che operano a velocità molto diverse, spesso non si utilizza un solo Bus ma più Bus ("Struttura a Bus Multiplo").
La z/Architecture presenta un sistema di interconnessione con i dispositivi più sofisticato del sistema a Bus multiplo, chiamato "Sistema a Canali". Questo tipo di sistema sarà descritto in seguito.

Infine definiamo Cluster di Sistemi un gruppo di calcolatori completi e interconnessi che operano come una risorsa elaborativa unificata. Ciascun componente del Cluster viene detto nodo. I Cluster di Mainframe sono gestiti mediante una complessa interazione fra Hardware, Sistema Operativo e Middleware. È possibile realizzare un Cluster di Sistemi Mainframe IBM detto Parallel Sysplex. Questo tipo di cluster sarà descritto in seguito.

1.1.1 La z/Architecture

Introduzione

L'architettura di un sistema ne definisce le caratteristiche dal punto di vista del software. Essa indica la struttura concettuale e il comportamento funzionale della macchina, che è tenuta distinta dalla realizzazione fisica della stessa. In particolare l'architettura non si occupa dell'organizzazione dei flussi interni dei dati (data flow), del disegno fisico e delle prestazioni di una specifica implementazione della macchina.
Infatti, implementazioni di macchine diverse fra loro possono essere conformi ad una singola architettura. Se l'esecuzione di un insieme di programmi su implementazioni diverse produce i risultati che sono definiti da un'architettura, le implementazioni di questa sono considerate compatibili per tali programmi.

La z/Architecture è l'attuale architettura dei Sistemi Centrali ed è l'ultimo gradino nella scala evolutiva che parte dall'architettura System/360 (1964), cui seguono negli anni le architetture System/370 (1970), System/370 Extended Architecture (370-XA, 1983), Enterprise Systems Architecture/370 (ESA/370, 1988) e Enterprise Systems Architecture/390 (ESA/390, 1990).

La z/Architecture include tutte le funzioni dell'architettura ESA/390 ad eccezione di alcune funzioni utilizzate precedentemente solo dai sistemi operativi e non più necessarie. La z/Architecture fornisce inoltre estensioni significative all'architettura ESA/390, fra cui:

- l'estensione dei registri generali e di controllo a 64 bit
- la modalità di indirizzamento della memoria a 64-bit, che si aggiunge a quelle a 24 e 31 bit presenti nell'architettura ESA/390; sia gli indirizzi degli operandi sia quelli delle istruzioni possono essere a 64 bit

- l'espansione della program-status word (PSW) a 16 bytes (128 bit) al fine di contenere gli indirizzi più lunghi.
- sino a tre livelli addizionali di tabelle per la traduzione dinamica degli indirizzi (Dynamic Address Translation o DAT), chiamate *Region Tables* e utilizzate per tradurre gli indirizzi virtuali a 64 bit.

Un punto fermo nella progettazione dell'architettura è stata la compatibilità binaria all'indietro garantita ai programmi scritti dagli utilizzatori del mainframe. Infatti, il criterio che guida le scelte dei progettisti hardware e software di Sistemi Centrali pone in grande considerazione la possibilità di salvaguardare il patrimonio applicativo degli utenti finali. Nell'introdurre nuove funzioni si garantisce la **compatibilità binaria** delle applicazioni - ovvero a livello di Instruction Set Architecture e di Application Binary Interface. Gli utenti finali hanno la libertà di scegliere se sviluppare nuove applicazioni avvantaggiandosi delle innovazioni tecnologiche, se continuare ad utilizzare le applicazioni con funzioni consolidate nel tempo o se sviluppare nuove applicazioni integrandole senza particolari problemi con quelle preesistenti (caso più frequente).
Il cuore della z/Architecture è descritto in dettaglio nel libro *z/Architecture Principles of Operation*[1], pubblicato da IBM la prima volta nel 2000 e tenuto costantemente aggiornato; questo libro costituisce, insieme a qualche pubblicazione collegata, la documentazione di riferimento dell'architettura. Il libro descrive ogni funzione al livello di dettaglio necessario per preparare un programma in linguaggio assembler. Le sezioni principali contenute sono:

- *l'organizzazione del sistema*, che descrive le parti fondamentali del sistema (la memoria principale, la CPU, i sottosistemi di input/output, per citare i più importanti)
- *la memoria*, che spiega i formati delle istruzioni, l'indirizzamento e le funzionalità di protezione, la traduzione dinamica degli indirizzi (Dynamic Address Translation o DAT), funzione che, accoppiata a

[1] Il libro si può scaricare alla pagina http://publibz.boulder.ibm.com/epubs/pdf/dz9zr006.pdf

uno speciale supporto di programmazione, realizza la virtualizzazione della memoria

- *il controllo,* che descrive i servizi hardware a disposizione per gestire lo stato del sistema
- *l'esecuzione dei programmi,* che spiega il ruolo delle istruzioni nell'esecuzione di un programma, esamina in dettaglio i formati delle istruzioni, e descrive l'uso della program-status word (PSW), del *branch* e delle interruzioni
- *le interruzioni,* che descrive in dettaglio il meccanismo che permette alla CPU di cambiare il suo stato come effetto delle condizioni esterne al sistema o interne al sistema e alla CPU.
- *le istruzioni generali,* che contiene descrizioni dettagliate dei formati logici, dei dati interi binari e di tutte le istruzioni non privilegiate ad eccezione delle istruzioni decimali e in virgola mobile.
- *le istruzioni decimali e quelle in virgola mobile.*

L'organizzazione di un sistema secondo la z/Architecture

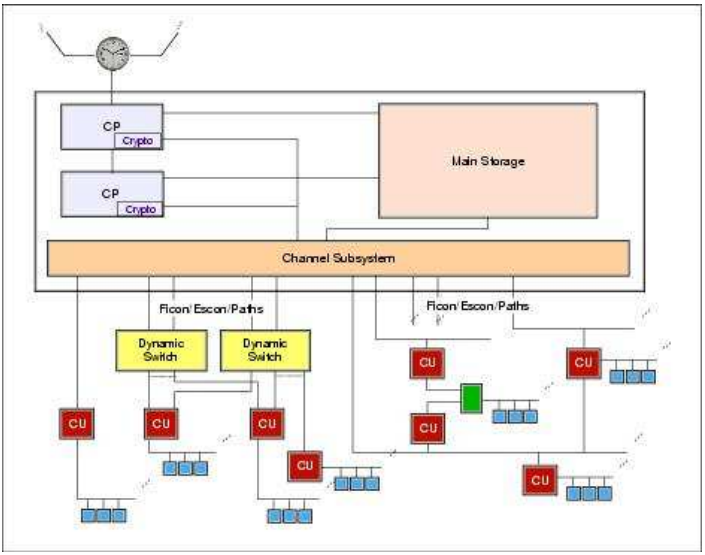


Figura 13 Organizzazione di un sistema secondo la z/Architecture

Logicamente, un sistema organizzato secondo la z/Architecture è composto dai seguenti elementi HW:

- Memoria principale ("main storage" o "main memory")
- Una o più CPU,
- Gli strumenti per la gestione (avvio/chiusura) del sistema (Operator Facilities)
- Un sottosistema a canali ("channel subsystem")
- I dispositivi di I/O

I dispositivi di I/O sono connessi al channel subsystem attraverso le unità di controllo ("control unit"). La connessione tra il channel subsystem e una unità di controllo è chiamata *channel path*. Un channel path può utilizzare o

un protocollo di trasmissione parallela (non più supportato nei dispositivi attuali) o un protocollo di trasmissione seriale: abbiamo perciò channel path paralleli o seriali.

Un Channel Path parallelo può connettere più Control Unit. Un Channel Path seriale può connettere una sola Control Unit (Topologia Point to Point). La connessione a più Control Unit si realizza utilizzando dei dispositivi chiamati switch dinamici.

Oltre alla memoria principale è prevista dall'architettura la memoria espansa, ovvero una memoria non indirizzabile direttamente da un programma applicativo ma utilizzabile dal sistema operativo a blocchi di 4KB come memoria veloce di paging. La memoria espansa è definita dalla configurazione come parte della memoria fisica di una macchina.[2] Nella CPU può essere inclusa un'unità crittografica e/o una fonte esterna di sincronizzazione (External Time Reference o ETR).

La natura fisica delle funzioni citate sopra può variare tra le implementazioni hardware, chiamate "modelli". La figura 13 la struttura logica di un sistema multiprocessore a 2 CPU con un'unità crittografica e che è connesso a un ETR. Ogni modello di mainframe ha una sua configurazione, definita dall'insieme delle funzioni disponibili e delle risorse HW quali il numero di sottocanali, dei channel path, delle unità di controllo che si possono connettere al channel subsystem, la dimensione della memoria principale ed espansa e le modalità di gestione della configurazione.

La gestione della memoria

Per indirizzare la memoria centrale l'architettura prevede vari tipi di indirizzi, fra i quali i più importanti sono quello reale e quello virtuale.

[2] La memoria espansa fu originariamente introdotta per superare alcuni limiti dovuti alla modalità di indirizzamento a 31 bit. Essendo la z/Architecture a 64 bit essa non è più necessaria e non è attualmente utilizzata dal sistema operativo z/OS. z/VM la utilizza come memoria di paginazione di primo livello per ottimizzare la gestione della memoria virtuale.

Il concetto di memoria virtuale è presente in molte architetture. I dettagli implementativi e il supporto hardware differenziano le architetture. Utilizzando la memoria virtuale ogni programma in esecuzione può assumere di avere accesso a tutta la memoria definita dallo schema d'indirizzamento dell'architettura. Il solo limite è il numero di bit in indirizzo di memoria.

Poter usare per programmi e dati più memoria di quella realmente disponibile presenta una serie di vantaggi. Il primo e più evidente è che si possono mettere sulla stessa macchina applicazioni che richiedono molta più memoria di quella realmente disponibile. L'implementazione della memoria virtuale permette di ottenere questo apparente miracolo con una perdita minima di prestazioni.

I sistemi operativi come z/OS, zLinux e z/VM supportano indirizzi virtuali e reali a 64-bit, che consentono a un programma di indirizzare fino a 16 exabyte di locazioni di memoria. Nella realtà i server oggi disponibili a 64 bit (di qualunque architettura) non si spingono oltre pochi Terabyte di memoria reale. I sistemi operativi tengono in memoria centrale le porzioni attive del programma e dei dati e, secondo un algoritmo specifico, trasferiscono quelle inattive tra la memoria centrale e quella ausiliaria su disco se richiesto dall'integrità dell'elaborazione.
La memoria virtuale che il sistema operativo assegna a ogni processo per suo uso esclusivo viene chiamata Address Space. Esso è un insieme di indirizzi virtuali contigui disponibile per eseguire istruzioni e immagazzinare i dati. Questo insieme può essere tanto grande da comprendere tutto l'intervallo indirizzabile secondo l'architettura. Quindi un Address Space in architettura a 64 bit può comprendere fino a 16 exabyte. Questa è la dimensione di ogni Address Space nel mondo z/OS, z/VM e zLinux.

L'architettura prevede che tutti gli indirizzi generati dai processori siano virtuali e quindi vengano sottoposti al processo di traduzione Virtuale-Reale. Viceversa gli indirizzi di memoria utilizzati durante l'esecuzione delle operazioni di I/O (ad esempio quelli che individuano le aree dove scrivere i dati provenienti dagli I/O device) sono reali e non sono tradotti dall'Hardware. Questa diversità di trattamento è conseguenza del fatto che i dispositivi di I/O sono più sensibili alla tempificazione delle operazioni e

quindi non possono tollerare i ritardi che la traduzione dinamica potrebbe generare.

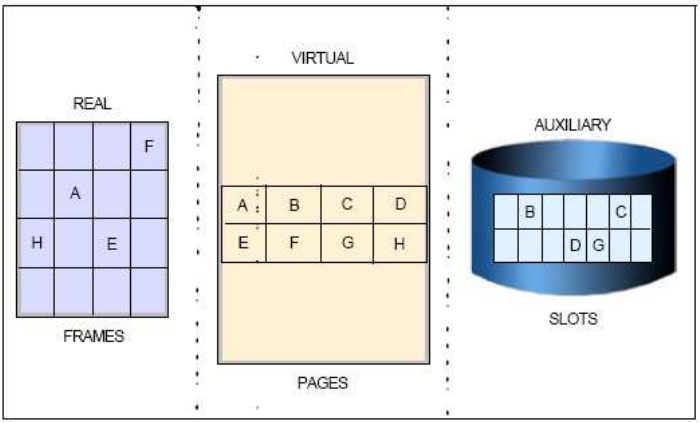


Figura 14 Frame, Page e slot

La traduzione dinamica degli indirizzi (Dynamic Address Translation o DAT) è il processo di traduzione di un indirizzo virtuale nel corrispondente indirizzo reale effettuato durante l'accesso a un riferimento di memoria. Se l'indirizzo virtuale è già in memoria centrale, l'esecuzione dell'istruzione interessata all'accesso alla memoria prosegue. Se l'indirizzo virtuale non è nella memoria centrale, si verifica un'interruzione per page fault, il sistema operativo viene notificato, carica la pagina dalla memoria ausiliaria, il processo Software interessato viene sospeso in attesa che il caricamento della pagina si compia. La DAT è implementata sia nell'hardware sia nel software attraverso l'uso di tabelle (page tables, segment tables, region tables). Poiché il processo di traduzione è abbastanza oneroso, per velocizzarlo si usano delle memorie di tipo associativo chiamate Translation Lookaside Buffer in cui vengono memorizzate le ultime "n" traduzioni Virtuale → Reale andate a buon fine; la grandezza di "n" è uno dei fattori più importanti per le prestazioni del processore.

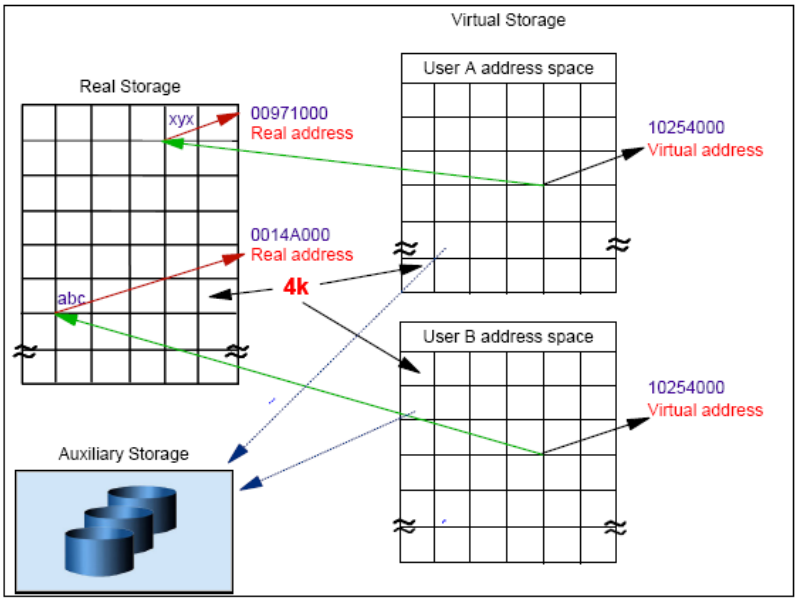


Figura 15 Real e Auxiliary storage per la memoria virtuale

Quando un programma è selezionato per l'esecuzione, il sistema lo porta in memoria virtuale, lo divide in pagine di 4KB, trasferisce le pagine nella memoria centrale per l'esecuzione.
Per il programmatore, l'intero programma sembra occupare spazio continuo di memoria in ogni istante. Effettivamente non tutte le pagine di un programma sono necessariamente nella memoria centrale o occupano spazio contiguo.
Le parti di un programma che vengono eseguite in memoria virtuale devono essere mosse tra la memoria reale e quella ausiliaria. Per permettere questo z/OS ad esempio gestisce la memoria in unità, o blocchi, di 4KB. Sono definiti i seguenti blocchi:

- Un blocco di memoria centrale è una *frame*.
- Un blocco di memoria virtuale è una *pagina.*

- Un blocco di memoria ausiliaria è uno *slot*.

Pagine, frame e slot hanno la stessa dimensione: 4KB.

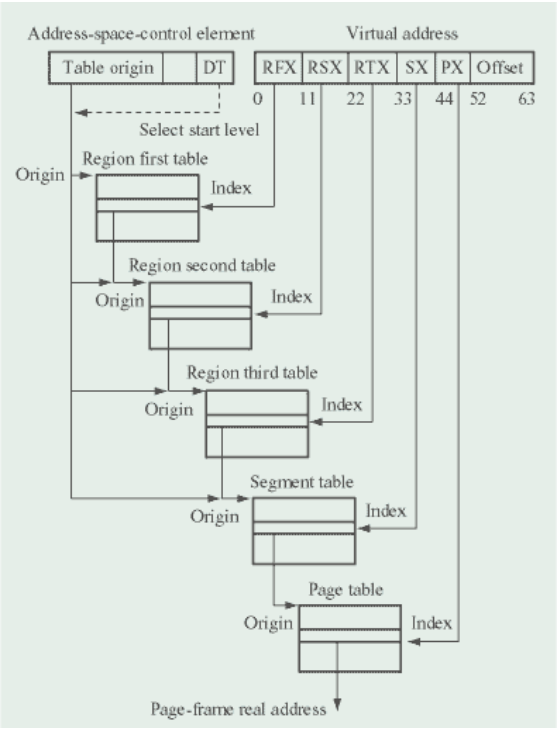


Figura 16 z/Architecture Dynamic Address Translation

I registri e le istruzioni

La z/Architecture è classificata tra le architetture CISC, poiché rende disponibili molte istruzioni complesse di lunghezza variabile. Come nelle architetture CISC, il numero di registri è inferiore a quello dei registri delle architetture RISC. Tali registri si dividono in:

- **Registri Generali (General Registers)**: sono 16 registri a 64 bit; per avere compatibilità all'indietro con le applicazioni scritte per le architetture precedenti, tali registri possono essere usati in parte come registri a 32 bit; essi possono essere utilizzati per indirizzare posizioni di memoria e come accumulatori nella aritmetica generale e nelle operazioni logiche.
- **Registri per la Virgola Mobile (Floating Point Registers)**: sono 16 registri a 64 bit; sono utilizzati per le operazioni in virgola mobile (binarie ed esadecimali); possono essere adoperati anche a coppie per contenere operandi a 128 bit.
- **Registri di Controllo (Control Registers):** sono usati dalla CPU solo per funzioni di controllo e registrazione.
- **Registri d'accesso (Access Registers):** permettono l'accesso ai Data Space (Address Space che contengono solo dati).

I formati delle istruzioni

Il **set di istruzioni CISC** nella z/Architecture è molto ricco: ci sono 21 Formati per parecchie centinaia di istruzioni diverse. Ogni istruzione è costituita da due parti:

- **Operation Code:** che specifica quale operazione deve essere eseguita.
- **Operand Designation:** può essere l'indirizzo del dato che si deve elaborare, la sua lunghezza o il dato stesso.

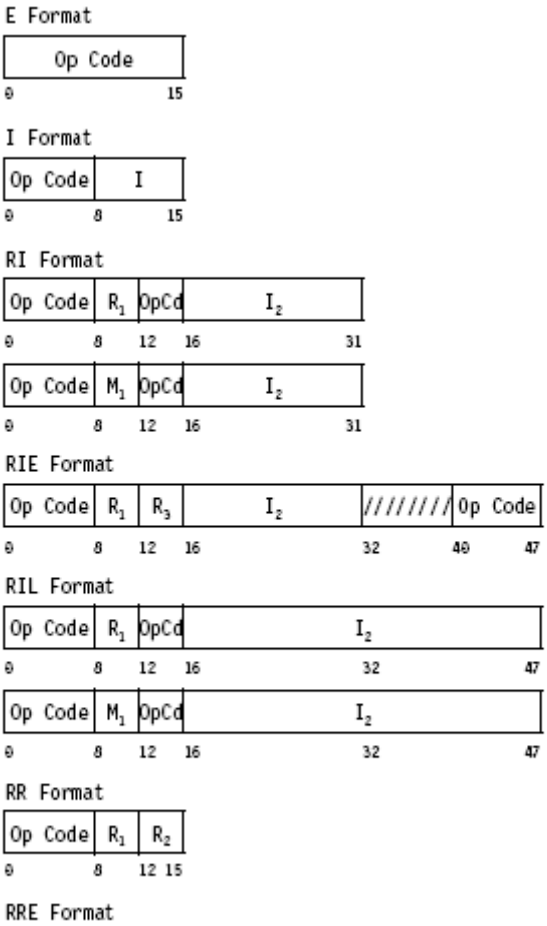


Figura 17 Basic Instruction Format

Un'istruzione può essere lunga una, due o tre halfword (ovvero 16, 32 o 48 bit) come mostrato nella figura e deve essere contenuta in memoria allineata alla halfword. Ogni istruzione è in uno dei 18 formati base E, RR, RRE, RRF, RX, RXE, RXF, RS, RSE, RSL, RSI, RI, RIE, RIL, SI, S, SSE, e SS, con 3 variazioni del RRF, due del RS e RSE, e RIL, e 4 dello SS.

L'Instruction Set

L'insieme delle istruzioni della z/Architecture (al 2008 894 di cui 668 implementate direttamente in hardware) ne costituisce l'Instruction Set. Esso è un elemento chiave di ogni architettura di Unità elaborative. Coloro che scrivono Software per una certa unità centrale dipendono dall'Instruction Set direttamente (mediante il linguaggio Assembler) o indirettamente (mediante linguaggi di alto livello come Cobol o Java): il processo di compilazione trasforma la descrizione della logica di business descritta nel linguaggio scelto in una sequenza di istruzioni Hardware appartenenti tutte all'Instruction Set; questo file binario viene comunemente chiamato eseguibile, o, per usare una terminologia z/OS, Load Module. Perciò ogni linguaggio ha tanti compilatori quante sono le piattaforme su cui può essere usato: così abbiamo un compilatore "c" per la architettura z, uno per Linux, uno per AIX ecc.

Se l'Instruction Set evolve solo aggiungendo istruzioni e mai levandole si ottiene una delle caratteristiche fondamentali della z/Architecture: la compatibilità binaria all'indietro. Infatti l'Instruction Set è sempre aumentato, comprendendo sempre le istruzioni che già esistevano; quindi programmi compilati alla nascita del Mainframe (1964) possono essere usati ancora oggi senza bisogno di essere ricompilati. L'Instruction Set è un elemento chiave dell'architettura ed è formato da tutte le istruzioni che la CPU è in grado di eseguire. Per garantire che il Software possa essere sempre usato, qualunque sia l'evoluzione dell'architettura, è necessario che il contenuto dell'Instruction Set venga solo aumentato (si aggiungono nuove istruzioni e non se ne levano mai): questo è uno dei requisiti fondamentali per garantire la compatibilità binaria del codice eseguibile (upward compatibility).

La qualità di un'architettura dipende moltissimo dalla capacità del suo Instruction Set di risolvere i vari tipi di problemi Software. La z/Architecture è una potente architettura usata per far funzionare server general-purpose e per indirizzare diversi tipi di problemi. Un Instruction Set versatile semplifica il lavoro del programmatore dell'applicazione, del compilatore, e del programmatore di sistema.

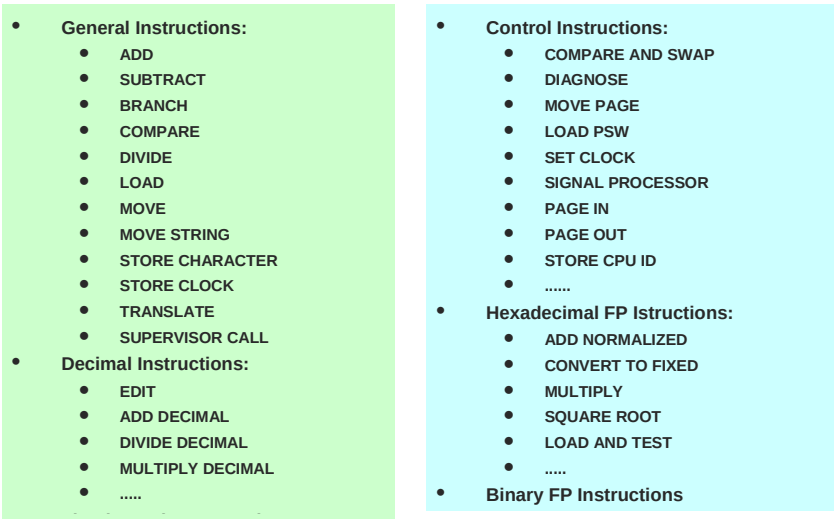


Figura 18 Principali istruzioni della z/Architecture

Per mantenere la compatibilità binaria con le architetture precedenti, è previsto l'indirizzamento trimodale (Trimodal addressing); esso si riferisce alla capacità di commutare tra i modi di indirizzamento a 24, 31 o 64 bit. La commutazione può essere fatta o con le vecchie istruzioni di branch che impostano il modo sulla base di un registro di controllo o mediante la nuova istruzione SET ADDRESSING MODE (SAM24, SAM31, e SAM64).

La Program-Status Word (PSW)

La PSW è un circuito di memoria collocato all'interno del CP (Central Processor). Essa contiene le informazioni richieste per l'esecuzione del programma attivo nel CP: contiene lo stato istantaneo (cioè ciclo per ciclo) di un CP. Ad esempio, in una configurazione con 13 CP avremo 13 PSW, una per CP. La lunghezza è di 16 byte (128 bit). La PSW include l'indirizzo della prossima istruzione da eseguire, un codice in cui viene indicato il risultato dell'istruzione che viene eseguita (Condition Code), e altre

informazioni usate per controllare la sequenza di esecuzione delle istruzioni e per determinare lo stato della CPU. La PSW viene anche indicata come Current PSW per distinguerla da due copie in memoria virtuale che vengono usate nella gestione degli Interrupt: Old PSW e New PSW. La relazione fra Current PSW, Old PSW e New PSW viene illustrata nel paragrafo seguente.

Introduciamo il concetto di Interruzione o Interrupt: un evento, codificato dall'architettura, che può essere indipendente dall'esecuzione del Software e quindi asincrono, o legato all'esecuzione del Software e perciò sincrono; l'effetto di questi eventi è di alterare la sequenza di istruzioni a seconda del tipo di evento (Interrupt) che si è verificato.

Quando avviene un'Interrupt, la CPU salva la Current PSW in una locazione di memoria assegnata (denominata appunto Old PSW); ogni tipo di Interrupt ha una sua locazione Old PSW la cui posizione in memoria virtuale è codificata dall'architettura. Contestualmente 16 byte vengono caricati nella current PSW dalla locazione indicata come New PSW. Come per le Old PSW così esiste una New PSW per ogni tipo di Interrupt. Poiché una PSW contiene l'indirizzo della prossima istruzione da eseguire, la New PSW per un certo tipo di Interrupt conterrà l'indirizzo della prima istruzione eseguibile della routine Software che dovrà gestire l'Interrupt. La Old PSW salvata servirà a riprendere l'esecuzione del processo Software che era stato interrotto dall'Interrupt partendo dall'indirizzo presente nella Old PSW che appunto indica la prima istruzione da eseguire dopo l'Interrupt.

Ci sono sei classi di interruzione; ogni classe ha una coppia di locazioni per la Old e New PSW la cui posizione in memoria virtuale è, come si è visto sopra, stabilita dall'architettura. Le classi di interruzione sono:

- External: eventi scollegati dall'esecuzione del Software e non legati alle operazioni di I/O
- Machine Check: è avvenuta una malfunzione a livello di Hardware
- I/O: una operazione di I/O si è conclusa
- Restart: un operatore richiede l'esecuzione di una certa routine Software il cui indirizzo è contenuto nella New PSW

- Supervisor Call: il programma chiede l'esecuzione di una particolare routine fornita dal Sistema Operativo, ad esempio il lancio di un'operazione di I/O – l'implementazione di questo tipo di Interrupt è dipendente dal sistema operativo
- Program Interruption: l'esecuzione del programma non può proseguire perché si sono verificate delle condizioni la cui soluzione richiede l'intervento del sistema operativo; ad esempio è necessario portare in memoria reale dai data set di page una pagina di memoria virtuale, o si è verificata una divisione per zero.

1.1.2 Architettura di I/O

Il Channel Subsystem

Uno dei principali punti di forza dei calcolatori mainframe è la capacità di gestire efficientemente un grande numero di operazioni di I/O simultanee. Il sottosistema dei canali (**Channel Subsystem** o **CSS**) gioca un ruolo fondamentale nel fornire tale capacità. Il CSS gestisce il flusso delle informazioni tra i dispositivi di I/O e la memoria centrale. Così facendo libera le CPU dal compito di comunicare direttamente con i dispositivi di I/O e permette che l'elaborazione dei dati proceda parallelamente all'esecuzione dell'operazione dell'I/O.
Il CSS svolge due tipi di attività:

- Gestisce il traffico di I/O a fronte delle richieste che nascono dalle Partizioni logiche ad esso collegate
- Reagisce agli eventi esterni legati alla configurazione di Input/Output coinvolgendo, se necessario, le partizioni logiche collegate.

Il CSS è un vero e proprio sistema operativo le cui risorse Hardware sono:

- **CP:** la natura di questo CP non viene specificato dall'architettura. In precedenti implementazioni dell'architettura mainframe erano CP costruiti ad hoc con un instruction set differente da quello dell'architettura mainframe. Nei mainframe costruiti secondo la tecnologia CMOS (dal 1994 in avanti) sono CP assolutamente

eguali ai CP che eseguono le applicazioni, circostanza che permette di risparmiare sui costi di costruzione dell'Hardware e di aumentare la robustezza della piattaforma. Prendono anche il nome di **SAP** (System Assist Processor).

- **Hardware System Area (HSA):** una porzione della memoria reale ad uso esclusivo dell'Hardware e inaccessibile alle istruzioni dell'instruction set. Il CSS usa l'HSA come propria memoria reale.
- **Canali:** questi sono dei piccoli processori che comunicano con le unità di controllo dell'I/O (Control Unit o CU). Essi gestiscono il trasferimento dei dati dalla memoria centrale al dispositivo esterno.

Dal punto di vista della connettività verso i device di I/O, il CSS implementa il concetto di Channel Path (percorso di canale). Occorre distinguere fra il Channel Path e il mezzo fisico su cui avviene fisicamente l'operazione di I/O. Il mezzo fisico è oggi tipicamente una fibra ottica o un cavo Ethernet (fino a 10 Gbit/Sec). Il Channel Path è invece una connessione logica che si stabilisce fra una partizione logica e una Control Unit. Un singolo mezzo fisico può ospitare **n** (fino a 60 con le attuali implementazioni) Channel Path. Viceversa un Channel Path esiste su un solo mezzo fisico. Il CSS provvede ad accodare le richieste di I/O che non possono essere iniziate perché una risorsa (Canale, Control Unit, Device) non è disponibile. Questo accodamento viene anche chiamato accodamento Hardware (o CSS queuing), ed è uno dei possibili accodamenti che si possono verificare durante l'esecuzione di una operazione di I/O. Ogni Device di I/O è rappresentato all'interno del Channel Subsystem da un area di memoria (allocata in HSA) chiamata Sottocanale (Subchannel). Il Subchannel costituisce l'aspetto logico di un device nei riguardi di un programma e contiene le informazioni richieste per eseguire una singola operazione di I/O. Un sottocanale viene creato per ogni I/O device definito al CSS acceduto da una partizione logica. Se un device è acceduto da "n" partizioni logiche sarà rappresentato da "n" Subchannel all'interno del CSS.

Il Channel Subsystem logico

Il concetto di Logical Channel Subsystem (LCSS) è stato introdotto con il server z990. LCSS è stato creato per permettere all'ambiente Sistem z di gestire un maggior numero di percorsi e dispositivi, disponibili fisicamente

nel server, e aumentare il numero di partizioni logiche. Ciascun LCSS può avere da 1 a 256 canali e può essere connesso ad un numero di partizioni logiche che va da 1 a 15. Si possono avere 2 LCSS nelle macchine IBM System z9 BC (Business Class) e 4 LCSS nelle macchine IBM System z9 EC (Enterprise Class) e IBM System z10 EC.

Le unità di controllo e i dispositivi di I/O

Gli ultimi due elementi presenti nella catena di I/O sono le unità di controllo (Control Unit o CU) e i dispositivi di I/O. L'unità di controllo governa il funzionamento di un dispositivo di I/O in modo che esso possa rispondere alla forma standard di controllo fornita dal CSS. Un'unità di controllo è in grado di guidare parecchie centinaia di dispositivi di I/O.

Un dispositivo di I/O può essere un disco, un nastro, un adattatore di rete o un altro dispositivo.

1.1.3 Potenze dei processori

Per quantificare la potenza dei processori è necessario definire metriche e unità di misura. Le unità di misura più comuni nell'ambito dei mainframe sono:

- **Ciclo Base (Nanosecondi):** è il tempo che impiega il sistema Hardware a passare da uno stato all'altro.
- **Frequenza (GigaHertz):** inverso del Ciclo Base.
- **MIPS (Millions Instructions per Second):** numero di istruzioni eseguite al secondo (in milioni).
- **Cicli per Istruzione:** numero medio di cicli macchina per istruzione; si ottiene dividendo la Frequenza per i MIPS.
- **MIPS UNI:** milioni di istruzioni eseguite da una macchina con un solo processore.
- **MIPS Tot:** milioni di istruzioni eseguite da una macchina con il massimo di processori attivi.

- **MSU (Millions Service Units):** unità di misura tipica del Sistema Operativo z/OS, OS/390, MVS.

Per confrontare la potenza di macchine diverse, anche della stessa architettura, la frequenza del clock è una unità di misura altamente fuorviante. Anche il numero MIPS, calcolato banalmente come numero di istruzioni eseguite per secondo, porta a percezioni sbagliate sulle performance relative fra due sistemi in quanto è fortemente dipendente dal carico di lavoro utilizzato.

IBM effettua benchmark estensivi sulle macchine mainframe e pubblica delle tabelle (LSPR - Large Systems Performance Reference) sulle performance relative. I rapporti LSPR rappresentano la valutazione di IBM delle capacità relative del processore in un ambiente non limitato da vincoli di sistema (ovvero eliminati tutti i colli di bottiglia, fatta eccezione per la potenza della CPU) per carichi specifici di benchmark definiti e stack software specificato nelle tabelle. I rapporti sono basati su misure e analisi.

1.1.4 CISC & RISC

A seconda di quanto è complesso l'instruction set, un'architettura hardware si classifica come CISC (Complex Instruction Set Computer) o RISC (Reduced Instruction Set Computer).

In questo paragrafo richiamiamo sinteticamente alcuni concetti.

La z/Architecture e l'architettura Intel sono esempi di architetture CISC. Sono caratterizzate dall'offrire al programmatore molte centinaia di istruzioni di lunghezza variabile. Solitamente il numero di registri è minore rispetto a quello che presentano le architetture RISC. Le istruzioni più

complesse possono far riferimento anche a diversi operandi in memoria e diversi registri. Ogni istruzione può richiedere anche molti cicli macchina.

Il tempo di esecuzione di una routine Software è dato dalla formula:

$$ExecutionTime = PathLength \times \frac{Cycles}{Instruction} \times CycleTime$$

Path Length: numero di istruzioni macchina che compongono la routine Software

Cycles/Instruction (CPI): Numero medio di cicli necessari per eseguire un'istruzione

Cycle Time: Tempo impiegato dal processore per passare da uno stato a quello successivo. L'implementazione dei mainframe IBM esige che questi tempi siano uguali per tutti i processori di un sistema. Il Cycle Time è l'inverso della frequenza del clock.

Le architetture CISC tendono a minimizzare la lunghezza del path, mentre quelle RISC tendono a minimizzare il tempo di ciclo e il numero di cicli per istruzione.

Negli ultimi anni, dal punto di vista microarchitetturale, le differenze fra processori CISC e processori RISC sono sempre più sfumate. Molti progetti di processori che implementano un'architettura di tipo CISC possono utilizzare tecniche in stile RISC (Es. Pentium 4).

Per la valutazione delle prestazioni globali di una macchina non è sufficiente confrontare ad esempio la velocità di esecuzione delle istruzioni. Infatti, a seconda del carico di lavoro, il tempo sulla CPU spesso non incide molto sul tempo totale, pertanto la diminuzione di quel tempo non ha un effetto sensibile sulla riduzione del tempo totale. Nella figura seguente **Z** e **A** indicano il caso di processore **Z**eta o di **A**ltro tipo di processore.

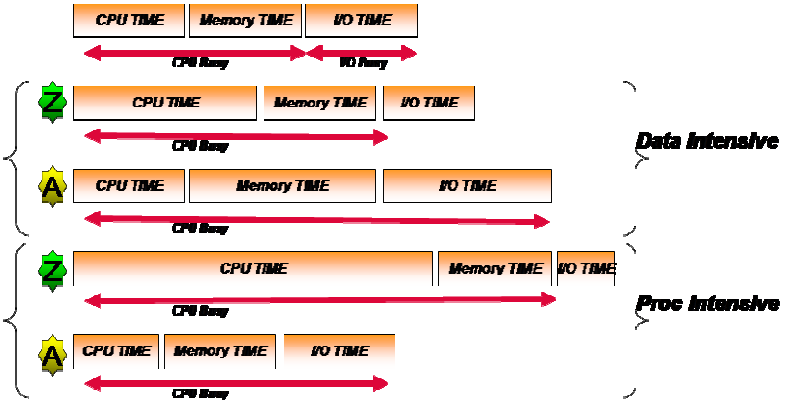


Figura 19 CISC e RISC

1.1.5 I processori z990 e z9

Nel seguito descriviamo il processore z990. Le considerazioni si applicano anche al processore z9, in quanto a livello di microarchitettura i due processori sono molto simili e differiscono principalmente per la tecnologia produttiva impiegata (SOI Silicon-On-Insulator a 130nm per lo z990 e a 90nm per lo z9). Il microprocessore dei server z990 è stato progettato per ottimizzare le performance sia delle nuove applicazioni mainframe sia di quelle tradizionali, conservando le notevoli caratteristiche RAS (*Reliability, Availability e Serviceability*) che i clienti si aspettano dai mainframe IBM.

I precedenti mainframe IBM di tecnologia CMOS avevano una microarchitettura relativamente semplice ed erano ottimizzati per le applicazioni tradizionali che giravano su di essi. Queste applicazioni

tendevano a sfruttare completamente le istruzioni complesse nell'instruction set della z/Architecture; un numero significativo dei programmi era scritto in linguaggio assembler (o almeno lo utilizzava nelle routine critiche per le prestazioni) e stressava il sottosistema di memoria. Pertanto, le pipeline delle istruzioni in questi processori CMOS precedenti non erano superscalari; in altre parole, esse potevano eseguire al massimo una sola istruzione per ciclo di clock. Invece i processori ottenevano buone prestazioni utilizzando hardware aggiuntivo per eseguire le istruzioni complesse velocemente e avevano un sistema di memoria molto performante; in particolare avevano cache L1 e Translation Lookaside Buffer (TLB) molto grandi.

I clienti di mainframe attualmente eseguono un'ampia varietà di applicazioni e middleware su questi sistemi. Le applicazioni più nuove sono tipicamente scritte in un linguaggio ad alto livello come il C, il C++ o Java. Questo codice tende ad avere caratteristiche molto diverse da quelle del codice tradizionale. In particolare esso tende a usare le istruzioni più semplici dell'instruction set e, con una frequenza più bassa, utilizza le istruzioni complesse per le quali i processori precedenti erano ottimizzati.
Quando questo microprocessore fu progettato in origine, un obiettivo progettuale era la possibilità di eseguire sia il codice tradizionale sia quello più nuovo con alte performance; pertanto il team di progetto decise di reintrodurre[3] il design superscalare.

Nel passare allo z990, un certo numero di caratteristiche presenti nei processori precedenti è stato mantenuto. Queste includono il *millicode*, che è il *microcode* verticale eseguito nel processore, e la Recovery Unit (R-unit), che mantiene lo stato completo della microarchitettura del processore salvato dopo ogni istruzione. Se viene rilevato un errore hardware, la R-unit è utilizzata per ripristinare lo stato valido ed eseguire l'algoritmo di gestione dell'errore (error-recovery). In aggiunta, il processore z990, come i suoi predecessori, duplica completamente molte delle principali unità funzionali per la rilevazione degli errori ed usa tecniche di rilevazione dell'errore (parità, duplicazione locale, controllo di congruità degli stati, ecc.) nel resto del processore per raggiungere i livelli di RAS per cui il

[3] La pipeline era stata introdotta per la prima volta con i processori in tecnologia bipolare.

mainframe è giustamente rinomato. L'architettura del microprocessore nel server z990 è relativamente non comune se confrontata con altri processori moderni, poiché, sebbene sia superscalare, esegue le istruzioni severamente nell'ordine architetturale. Tuttavia, compensa questo svantaggio con una pipeline più corta e cache e TLB più ampi rispetto agli altri processori, insieme ad altre caratteristiche che migliorano le prestazioni.

Nella scelta iniziale di una microarchitettura di alto livello, devono essere considerati molti fattori. Gli stadi della pipeline e la lunghezza complessiva devono essere scelti con molta cura per ottenere il bilanciamento ottimale tra frequenza (a parità di altre condizioni, aumentando la frequenza devono aumentare gli stadi della pipeline) e la perdita di prestazioni associata con una pipeline più lunga (branch predetto non correttamente, dati dipendenti e utilizzo degli indirizzi). La dimensione di cache e TLB devono essere bilanciate per ottenere un Hit rate (la percentuale di volte in cui per accedere ad un dato non si è dovuto accedere alla memoria reale/la percentuale di volte in cui si è evitata la traduzione degli indirizzi poiché l'indirizzo reale si è trovato nel TLB) buono senza richiedere chip eccessivamente grandi: condizione questa che rende più difficile raggiungere i target di frequenza fissati in fase di progetto del chip.

Maggiori dettagli sul processore si possono trovare nell'articolo **The IBM eServer z990 microprocessor[4].**

[4] http://www.research.ibm.com/journal/rd/483/slegel.pdf

I-unit	**D1** Decode	**AA1** Agen Superscalar grouping				Decode if prior branch mispredict			
D-cache		**C0** ALB and AAHT lookup	**C1** TLB, PAAHT, Directory, Cache access	**C2** Data sent to E-unit Store queue written		**C0ST** Store data sent to D-cache	**C1ST** Data in L1 Store data from mirror	**C2ST** ECC gen Cross-check store results	Store data check-point
E-unit			**E–1** Instruction sent to E-unit	**E0** Read GRs	**E1** Execute Branch mispredict detected	**PA** Put-away			
R-unit						Put-away data sent to R-unit	**R0** Put-away from mirror E-unit	**R1** Cross-check results	**R2** ECC gen	**R3** Register data check-point

Figura 20 Pipeline del processore z990

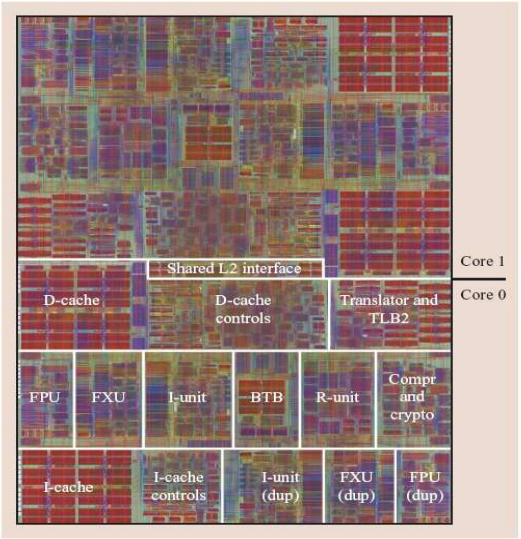


Figura 21 Chip del processore z990

Il Millicode

L"instruction set della z/Architecture diventa sempre più ricco: istruzioni e funzioni sempre più complesse sono state introdotte. Concettualmente le istruzioni più semplici (LOAD, STORE, MOVE, BRANCH) e le istruzioni logiche e aritmetiche possono essere implementate direttamente nell'hardware. Le istruzioni e le funzioni più complesse come le istruzioni di I/O, la Start Interpretive Execution (SIE), le istruzioni cross-memory, i gestori delle interruzioni e certe caratteristiche RAS, devono essere implementate con qualche tipo di codice interno. A partire dal processore S/390 G4 nel 1997 e continuando attraverso il modello G5, G6, z900, e i processori z990, il codice interno al processore è chiamato millicode.

Su molti processori precedenti al G4, che erano implementati usando chips multipli in tecnologia bipolare, il codice interno fu posto su chip del processore separati; questo è noto come microcode orizzontale. Con il sistema G4, l'intero processore fu implementato su un unico chip. A causa dei vincoli di area del chip, si impose un redesign del codice interno del processore, dal momento che esso non sarebbe potuto essere contenuto più nel chip, e, con l'aggiunta di nuovi requisiti a livello architetturale, sarebbe stato necessario una maggiore quantità di codice interno. Questo portò al progetto del millicode verticale come codice interno del processore. Il millicode è scritto in linguaggio assembler, in primis con istruzioni della z/Architecture implementate nell' Hardware, in secundis con istruzioni specializzate per il solo millicode.

1.1.6 Il processore z10

Il processore z10 è il motore dell'ultima generazione di server mainframe IBM, introdotta sul mercato a febbraio 2008. Il nuovo core del microprocessore sfrutta la più avanzata tecnologia di progettazione ad alta

frequenza, mantenendo nel contempo piena compatibilità con *l'instruction set* esistente.

Il chip del processore incorpora molte unità con funzioni speciali per accelerare specifiche operazioni e include robusti meccanismi di *fault detection* e *recovery* hardware. In cooperazione con un chip chiamato *Symmetric multiprocessing (SMP) Hub (verrà spiegato più avanti)*, il processore z10 implementa una struttura multiprocessore a cache condivisa che è ottimizzata per i carichi di lavoro di tipo *data-serving;* essi ricoprono un ruolo di grandissimo rilievo per il mercato dei mainframe.

Il chip del processore z10 (Figura 22) contiene 4 *core*, ognuno con una cache privata di secondo livello di 3 MB. I core sono disposti simmetricamente sul chip, ciascuno circondato su tre lati dalla sua cache di secondo livello. Sono presenti due unità nel ruolo di coprocessori, ognuna delle quali implementa funzioni crittografiche e di compressione dei dati, condivise da due core. Il chip include anche un *memory controller*, un *I/O bus controller* e uno *switch* che connette tutti e quattro i core a un'interfaccia condivisa con l'SMP hub chip e la sua cache condivisa. Il processore z10 è implementato nella tecnologia IBM *silicon-oninsulator* (SOI) a 65nm, la dimensione del *die* è di 454 mm^2, contiene 994 milioni di transistor e opera a 4.4 GHz in un sistema multiprocessore.

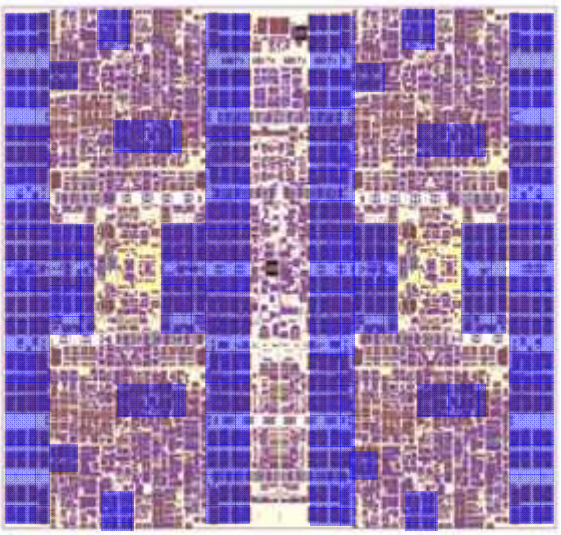


Figura 22Chip del processore z10

Il processore z10 implementa completamente la z/Architecture. Sebbene l'architettura abbia subito molte estensioni significative dal 1964, IBM ha rigorosamente mantenuto la compatibilità binaria in avanti al livello dei programmi applicativi per preservare gli investimenti dei clienti e assicurare una transizione liscia da una generazione alla successiva. Per lo z10 IBM ha aggiunto più di 50 istruzioni alla z/Architecture, principalmente per migliorare l'efficienza delle applicazioni compilate. Il processore z10 aggiunge inoltre il supporto per le *page frames* da 1MB e interfacce software-hardware per migliorare l'efficienza della cache.

Il core del microprocessore z10 è organizzato in 8 unità come mostrato nella Figura 23 La *instruction fetch unit* (IFU) contiene una cache per le istruzioni di 64-Kbyte, la logica di *branch prediction*, i controlli per *l'instruction-fetching*, e gli *instruction buffer*. Gli *instruction buffer* nella IFU alimentano la *instruction decode unit* (IDU) al centro del core. La logica analizza e decodifica i quasi 900 distinti *opcode* definiti nella z/Architecture, identifica le dipendenze tra istruzioni, forma le coppie di istruzioni per

l'esecuzione superscalare quando possibile, e immette le istruzioni nelle parti di logica di accesso agli operandi e di esecuzione.

La *load-store unit* (LSU) include una cache per i dati di 128-Kbyte e gestisce gli accessi agli operandi trattando tutte le lunghezze, modi e formati previsti nella z/Architecture, e supporta due *fetch* di blocchi di 128 bit (*quadword*) per ciclo. Inoltre essa bufferizza i risultati di operazioni *operand store* tra l'esecuzione dell'istruzione e il suo completamento, e si interfaccia con la *fabric* multiprocessore (attraverso la cache di secondo livello) per mantenere la coerenza fortemente ordinata della cache richiesta dall'architettura. La LSU è accoppiata con una *translation unit* (XU), che è costituita da un ampio *translation look-aside buffer* (TLB) di secodo livello e un'unità di traduzione hardware. Quest'ultima gestisce la traduzione dei registri di accesso e la traduzione dinamica degli indirizzi (*dynamic address translation* o DAT) per convertire indirizzi logici in indirizzi reali, inclusa la DAT innestata richiesta per i sistemi operativi in esecuzione come ospiti sotto l'ipervisore z/VM (macchine virtuali).

Tre unità gestiscono l'effettiva esecuzione delle istruzioni: la *fixed-point unit* (FXU) esegue istruzioni aritmetiche in virgola fissa, logiche e di salto. La FXU esegue la maggior parte di queste in un solo ciclo, e in coppie, con una completa rete di *forwarding* per permettere l'esecuzione di operazioni tra loro dipendenti l'una di seguito all'altra. La *binary floating-point unit* (BFU) è una *pipeline* multistadio che gestisce tutte le operazioni in virgola mobile binarie (secondo lo standard IEEE-754) ed esadecimali (eredità dell'architettura S/360). Questa unità può avviare un'operazione per ciclo, e utilizza il *forwarding* intrapipeline dei risultati per minimizzare nella pipeline le latenze tra istruzioni dipendenti. La BFU esegue inoltre le istruzioni di moltiplicazione e divisione in virgola fissa. La *decimal floatingpoint unit* (DFU) esegue le operazioni decimali sia in virgola mobile (standard IEEE-754R) sia in virgola fissa (eredità dell'architettura S/360).

In ultimo, la *recovery unit* (RU) mantiene una copia completa dello stato architetturale del processore, protetta da un codice a correzione d'errore (error-correction code o ECC). Questo stato include sia tutti i registri previsti dalla z/Architecture sia i vari registri di modo e stato usati dall'hardware e dal millicode per implementare le funzioni della

z/Architecture. La RU raccoglie tutti i segnali hardware di *fault detection* e sovraintende alle azioni hardware di recupero se questi segnali indicano un qualsivoglia malfunzionamento.

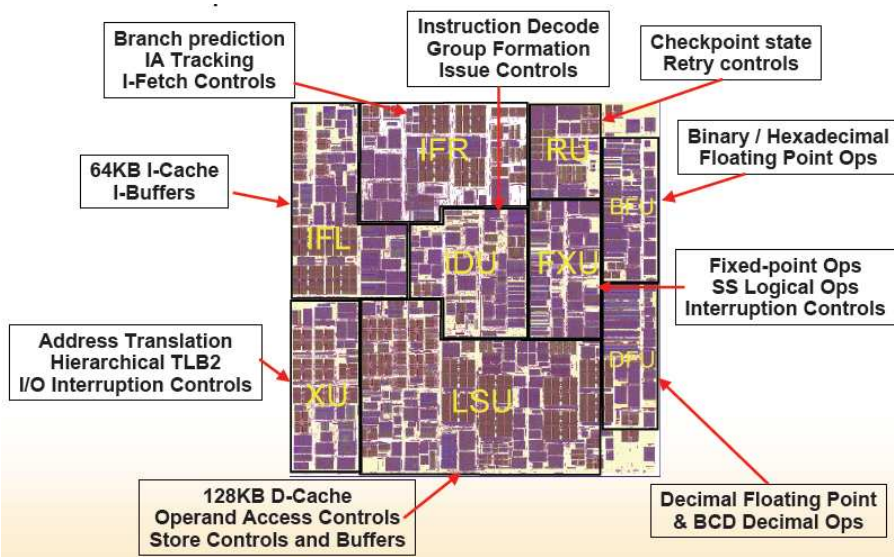


Figura 23 Le unità del core del microprocessore z10

La caratteristica che più di tutte contraddistingue il *design* del core del microprocessore z10 rispetto ai predecessori è il gigantesco salto nella frequenza di esercizio - da 1.7 GHz sul sistema z9 a 4.4 GHz sui sistemi basati su z10. A partire dal processore CMOS G4 uscito nel 1997 i core delle CPU dei mainframe IBM hanno sempre avuto un ciclo della durata da 27 a 29 FO4[5] e una pipeline lunga 6 cicli, contando dalla fase di decodifica delle istruzioni alla fase di scrittura dei registri (*put-away*). Attraverso sei generazioni di sistemi e tecnologia elettronica, i progettisti hanno

[5] Il FO4 è il ritardo introdotto da un inverter con fanout 4. L'unità è usata usata dagli specialisti del settore per confrontare tempi in maniera indipendente dalla specifica tecnologia di fabbricazione.

mantenuto costante la dimensione del ciclo e la profondità della pipeline mentre hanno aggiunto funzioni di tutto rilievo (come la capacità di esecuzione di operazioni in virgola fissa in conformità con lo standard IEEE, la *branch target prediction*, la completa estensione architetturale a 64 bit, la modalità di esecuzione superscalare e la crittografia). Il progetto del processore z10, tuttavia, è partito da un foglio bianco, con l'obiettivo di ottenere un ciclo di gran lunga più corto di 15 FO4, e di bilanciare le prestazioni, la potenza, l'area e le considerazioni di complessità progettuale. Questo cambiamento ha richiesto innovazioni nella metodologia di progetto, nella struttura della pipeline e nell'implementazione dell'architettura.

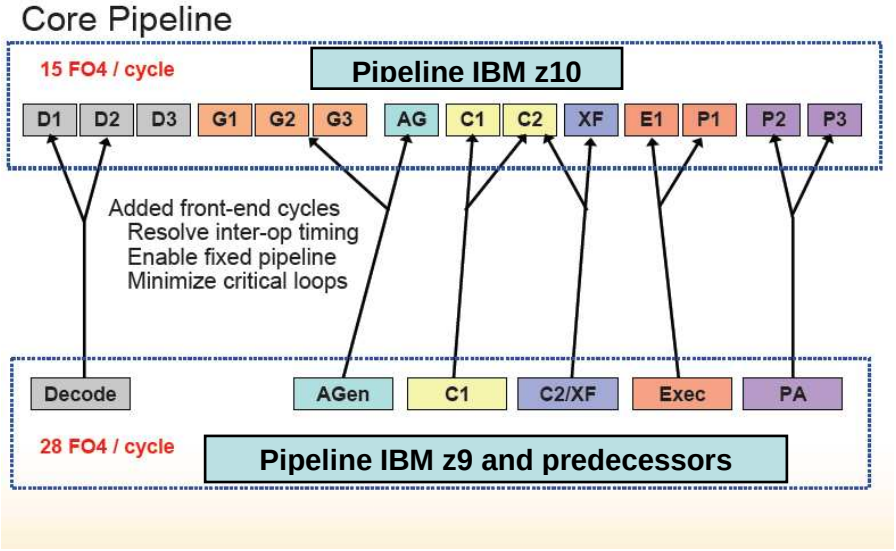


Figura 24 Confronto tra le pipeline del processore z10 e quelle dei suoi precedecessori

Esterne al core del microprocessore, il processore z10 include una coppia di unità con funzioni di coprocessore. Ognuna di queste unità implementa le funzioni di compressione dei dati e di accelerazione crittografica previste dalla z/Architecture, le quali sono fornite al software come istruzioni

sincrone convenzionali. Ogni coprocessore serve due core e contiene due motori di compressione (ognuno con una cache locale di 16 Kbyte), un motore di cifratura crittografico e un motore per il calcolo dello hash crittografico.

Sebbene non fosse un obiettivo primario nel progetto, l'efficienza energetica ricoprì un ruolo importante nel progettare lo z10 per massimizzare le prestazioni ottenibili con i vincoli di trasporto della potenza e di raffreddamento dovuti al *packaging* del sistema. Il raffreddamento a livello di sistema e l'uso selettivo di dispositivi con alta tensione di soglia (Vt) per i percorsi non critici limitano la potenza dispersa per fenomeni di *leakage*, e la tecnica di *dynamic clock gating* riduce la potenza attiva. Per ridurre ulteriormente la potenza spesa nei core di riserva (*spare*) o in stato *idle* nel sistema, l'hardware supporta lo *sleep mode* (invocato dal millicode) che può fermare la pipeline delle istruzioni fino a che non si presenti una interruzione. I mainframe raggiungono l'efficienza energetica a livello di sistema operando con prestazioni costanti a un livello di utilizzo sostenuto, permesso dalle capacità di virtualizzazione fornite dall'hardware, il sistema operativo z/OS, e gli ipervisori PR/SM (processor resource/system manager) e z/VM.

Per concludere si può osservare che il processore z10 ha un core, un chip e un disegno multiprocessore unici nel loro genere e sviluppati specificatamente per il mercato dei *data server* di fascia enterprise, anche se dal punto di vista tecnologico condivide le tecniche di progettazione ad alta frequenza e i blocchi elettronici di base con il processore Power6, utilizzato nei sistemi IBM che implementano l'architettura RISC Power.

Il core z10 a 4.4 GHz core rende fino al doppio in prestazioni rispetto al processore z9 per applicazioni che fanno un uso intensivo della CPU (*CPU-intensive*) e che costituisono una parte crescente dei carichi di elaborazione dati delle aziende. In coppia con la cache privata di 3 Mbyte private e la cache condivisa di 48 Mbyte, il design dello z10 fornisce un guadagno del 50 percento su un'ampia gamma di transazioni database tradizionali che fanno un uso intensivo del sottosistema di I/O (*Data-intensive*) e che continuano a costituire la maggior parte dei carichi di lavoro commerciali. Questo design fornisce la base per estendere la piattaforma mainframe IBM

con l'evolversi della tecnologia elettronica su silicio per molte future generazioni.

SMP Hub Chip

Un chip hub progettato specificatamente per l'utilizzo con i processori z10 permette la costruzione di sistemi SMP scalabili ad alte prestazioni ottimizzati per le applicazioni di accesso ai dati aziendali. Il chip hub connette vari chip z10, permettendo di raggiungere la banda dati di 48 Gbytes/s tra ciascuna coppia hub-processore. Il chip hub include una cache SRAM (Static Random Access Memory) da 24 Mbyte che è condivisa tra i processori collegati, e coppie di chip hub possono essere combinate per realizzare una cache condivisa unica di 48 Mbyte per l'insieme dei processori. Più coppie di chip *hub* possono essere collegate per formare sistemi SMP più grandi; la *fabric* a bassa latenza permette di scalare efficientemente in senso SMP e supporta i requisiti di coerenza forte della z/Architecture.

Questo struttura con switch centrale e cache condivisa rispetto a topologie di *fabric* più distribuite procura notevoli vantaggi in termini di prestazioni per carichi di lavoro con un alto grado di condivisione dati e di interazione tra processi, che sono comuni in molti ambienti di elaborazione di transazioni commerciali. Implementato a livello di processo di fabbricazione nella stessa tecnologia del processore z10, il chip hub SMP è costituito da 1.6 miliardi di transistors su un die di 445-mm^2.

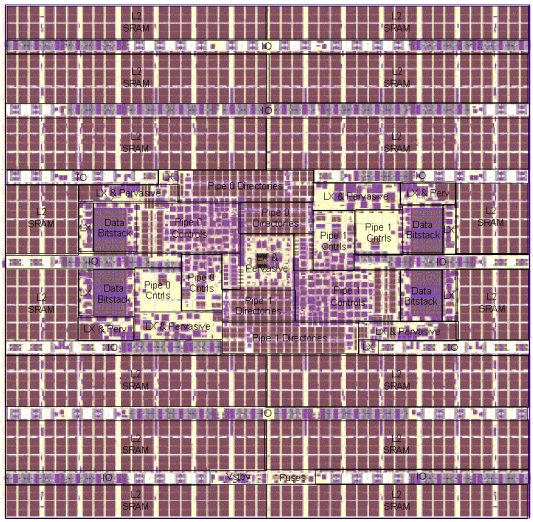


Figura 25 Il chip hub SMP

Maggiori dettagli sul processore z10 e il chip SMP hub si possono trovare nell'articolo di Charles F. Webb **IBM z10: The Next-Generation Mainframe Microprocessor**[6]

[6] L'articolo è pubblicato sul numero 2 della rivista IEEE Micro, volume 28, marzo-aprile 2008.

1.2 Hardware dei Sistemi Centrali IBM (z10)

Scopo di questo capitolo è illustrare il modo in cui sono strutturati i sistemi centrali IBM.
Dopo una breve introduzione storica, avenete lo scopo di descrivere la linea evolutiva che porta al sistema z10, la descrizione del sistema z10 verrà usata come esempio della ricchezza della struttura interna di un moderno mainframe IBM.
Vedremo quali sono i componenti principali di un sistema z10 e come essi interagiscono tra loro da un punto di vista fisico e logico. Il design dei sistemi z10 EC è l'ultimo passo evolutivo di una tecnologia, detta CMOS (Complementary Metal Oxide Semiconductor), iniziata nel 1994. In quest'arco di tempo, il design è stato adattato alle richieste ed alle circostanze mutevoli nel corso degli anni, dettate anche dall'avanzare di nuovi modelli applicativi nati dall'avvento dell'era del "e-business".

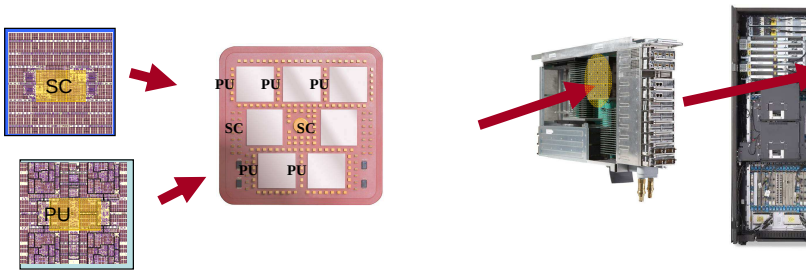


Figura 26 Building Blocks

Attualmente, lo z10 EC offre un alto livello di affidabilità, capacità di adattamento e superamento di errori (resilience), sicurezza e attendibilità (reliability) cioè "capacità di funzionamento in determinate circostanze per

lungo periodo di tempo" [7] e costituisce il punto centrale della strategia IBM nel realizzare una infrastruttura intelligente ed integrata. Secondo la concezione olistica[8] con la quale è stato progettato, non soltanto il sistema z10 è fondamentale per l'intera infrastruttura ma lo è anche tutto l'insieme costituito da sistema operativo, sottosistemi applicativi ("middleware"), storage e tecnologia di networking a supporto di standard aperti, aiutando il cliente a raggiungere i suoi obiettivi di business. Il design modulare si prefigge inoltre di ridurre i fermi di sistema, pianificati e non, offrendo manutenzione, riparazione, sostituzione e aggiornamento del sistema senza interruzione del servizio ("concurrent repair").

La tecnologia dei processori

La conoscenza della tecnologia con la quale sono realizzati i processori, di per sé non costituisce un fattore determinante nell'uso che si fa quotidianamente del sistema informatico ("computer"). Comunque aiuta a capire i singoli elementi che compongono un sistema della famiglia IBM System z.

1.2.0 Il Chip

Il componente base del design a "buiding blocks" è il Chip. All'interno del chip si trovano migliaia di circuiti molto piccoli, creati e connessi tra loro usando la tecnica Litografica. Differenti tipi di tecnologia sono stati usati, nel tempo, per la famiglia di elaboratori mainframe. Una tecnologia molto usata, fino alla metà degli anni '90, è stata quella chiamata Thermal Conduction Module (TCM), detta anche "bipolare", dove i chips erano montati in moduli di ceramica all'interno dei quali c'erano i necessari collegamenti in rame. Il problema maggiore di questa tecnologia era il

[7] da Wikipedia
[8] Termine medico che indica la visione delle persone nella loro interezza, rispetto all'ambiente interno ed esterno.

grande calore generato che necessitava di complessi sistemi di dissipazione ad acqua, che rendeva notevole la dimensione degli elaboratori. Già dal 1982, IBM stava lavorando ad una nuova tecnologia chiamata Complementary Metal Oxide Semiconductor, CMOS, che offre il grande pregio di dissipare basse quantità di calore. Inoltre, tutta la circuiteria di un singolo processore bipolare, che richiedeva circa 400 chips bipolari, con ben 4 TCM, poteva ora essere rimpiazzata con 4 chips CMOS ed un solo MCM).

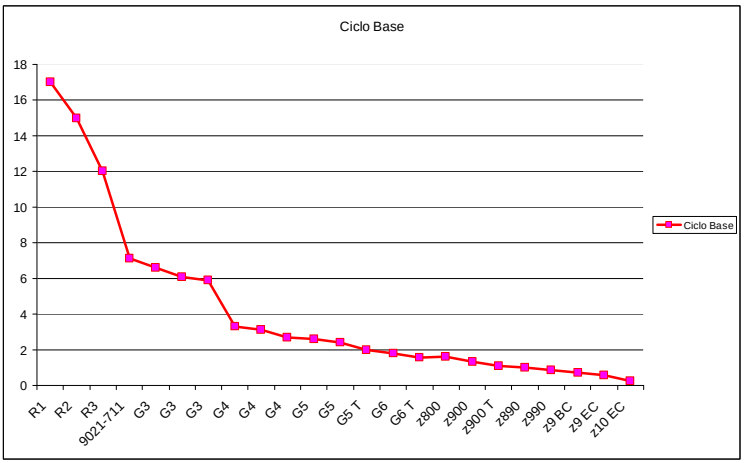


Figura 27 Ciclo base

Una delle conseguenze dell'introduzione di questa tecnologia è stata la riduzione del ciclo-base, così come evidenziato dalla Figura 27.

Nel 1994 si decide di usare esclusivamente la tecnologia CMOS per i mainframe. Il primo elaboratore della nuova famiglia (il modello 9672-R1) aveva un ciclo base di 17ns mentre, come si vede dal grafico, l'elaboratore 9021-711 a tecnologia bipolare aveva già un ciclo-base di 7,1ns. La successiva evoluzione della tecnologia CMOS è arrivata con il sistema IBM System z10, ad un ciclo-base di 0,23ns.

1.2.1 La famiglia dei Mainframe

Per famiglia si intende un insieme di calcolatori con la stessa architettura, quindi compatibili dal punto di vista del software sviluppato, ma con caratteristiche di prestazioni e di prezzo differenti.

Figura 28 La Famiglia del MainFrame nel 2008

La famiglia di mainframe si è evoluta nel tempo dal punto di vista delle caratteristiche hardware; tutte le famiglie sono contraddistinte da una sigla commerciale ed ogni famiglia comprende un gran numero di modelli di diversa potenza e configurazione. Ogni nuova famiglia ha conservato una

caratteristica molto importante: garantire la compatibilità binaria con la precedente.
La compatibilità binaria permette ad ogni programma scritto in base alle regole della z/Architecture di essere eseguito su qualunque elaboratore compatibile senza modifiche né al source né al codice eseguibile. Ciò implica che tutti gli elaboratori "compatibili" devono eseguire tutte le istruzioni previste dalla z/Architecture con gli stessi risultati, indipendentemente dalla realizzazione tecnologica del processore (TCM, CMOS…).
La conseguenza più importante di questo è che la quasi totalità degli utenti mainframe è in grado ancor oggi di eseguire ed usare programmi compilati nel 1964 senza alcuna modifica.

Qui di seguito si riporta una breve sintesi delle principali caratteristiche delle più importanti famiglie di elaboratori IBM mainframe.

System/360

Inizialmente IBM aveva due linee separate di computer, una per il calcolo scientifico ed un'altra per quello commerciale. Si pensò di unificarle e nel 1964 uscì il primo elaboratore della linea IBM System/360 (Figura 29), che aveva un addressing-mode a 24bit con una capacità totale pari a 16MB di memoria indirizzabile, ed usava la rappresentazione EBCDIC invece che ASCII. La fortuna di questa generazione di elaboratori fu la facilità di conversione da un modello all'altro, cioè l'introduzione di quel concetto di "compatibilità" espresso prima. IBM System/360 non usava la memoria virtuale.

Il design team del Sistema IBM 360 ha ricevuto dal Presidente R. Reagan il premio "National Medal of Technology".


Figura 29 S/360

System/370

Nel 1970 fu introdotta la generazione /370 (vedi figura 30), evoluzione della precedente, con raffreddamento ad acqua. Con questi elaboratori venne introdotta una nuova modalità di calcolo, il multiprocessing, cioè la possibilità di condividere il carico di lavoro su più processori. Venne anche introdotto il concetto di "virtual storage", cioè ogni processo indirizza la propria area virtuale di memoria che viene opportunamente mappata sulla memoria reale. Alla fine degli anni '70, la serie /370 viene suddivisa in "large" 30XX series (3031, 3033, 3081, 3084) e "midrange" 43XX series (4321, 4331, 4341…) in concorrenza con i calcolatori DEC VAX ed altri simili minicomputers.

Figura 30 S/360

System/370 XA

Nel 1983 la serie "large" 30XX fu migliorata con l'introduzione dell'Architettura XA, la cui principale innovazione era la possibilità di indirizzare fino a 2GB di memoria grazie all'addressing-mode a 31bit. Nel 1985 nacque il concetto di Logical Partitioning, LPAR, cioè la possibilità di suddividere i sistemi in due o più entità logiche separate ma in grado di condividere le risorse di sistema.

System/390

Introdotta nel 1990, aveva a disposizione la nuova Architettura ESA/390, con un indirizzamento misto a 24 o 31bit, canali più veloci (ESCON) e più numerosi, maggior numero di processori e un set di istruzioni molto ampio. In questa generazione fu introdotto per la prima volta un modello CMOS, il 9221. A seguire, nel 1994, arrivarono modelli esclusivamente CMOS, i 9672

R1, raffreddati ad aria. I primi modelli erano sicuramente meno potenti dei precedenti modelli 90XX e per ovviare a questo limite venne data la possibilità di accoppiare più elaboratori e costituire un'entità logica comune, un cluster, nella quale la potenza a disposizione fosse la sommatoria delle varie entità fisiche. Questo è il concetto alla base del Parallel Sysplex. Negli ultimi modelli del S/390 era presente, per la prima volta, il formato delle istruzioni IEEE floating point.

zSeries & System z

L'inizio del nuovo secolo ha visto l'avvento di una nuova generazione di elaboratori, definiti zSeries, dove la z è l'acronimo di "zero downtime" e mette in evidenza la principale qualità dei mainframe, che è appunto la loro altissima affidabilità, sia hardware che software.

Viene introdotta anche una nuova architettura, la z/Architecture, la cui novità più importante è l'indirizzamento a 64bit, pur conservando la possibilità di indirizzare a 31 o 24bit, in nome della ormai famosa compatibilità.

Altra grande innovazione è la possibilità di ospitare il sistema operativo Linux, in modalità nativa all'interno di una partizione logica (LPAR), o come ospite del software di virtualizzazione z/VM.

1.2.2 z10 Multiple Chip Module (MCM)

Uno degli obiettivi dei sistemi IBM z10 è fornire il doppio delle prestazioni della precedente famiglia IBM z9. Per raggiungere questo obiettivo, la tecnologia usata per la costruzione dei chip doveva fare uso di una frequenza del 40% più alta senza aumentare il consumo e la relativa dissipazione di calore. La generazione precedente usava la tecnologia chip 130nm Silicon-On-Insulator, SOI, con 8 strati di interconnessione. Per raggiungere l'obiettivo prefissato è stata usata la tecnologia 65nm SOI con 10 strati di interconnessione in rame, che fornisce un aumento

prestazionale circa 1,4 volte superiore. L'elemento base hardware degli elaboratori z10 è il Multi Chip Module (MCM) sul quale sono alloggiati fino a 20 chip. Questi chip assolvono a tutte le funzioni necessarie, e cioè processori (PU), le memorie cache, system data cache (SD) e storage control (SC), funzioni di controllo d'accesso alla memoria, Memory Subsystem Control (MSC) ed il clock.
Il chip è un "quad-core" e, in ogni MCM, ce ne sono 7 di cui 5 compongono il pool di processori e gli altri due sono dedicati alle funzioni di Storage Control.

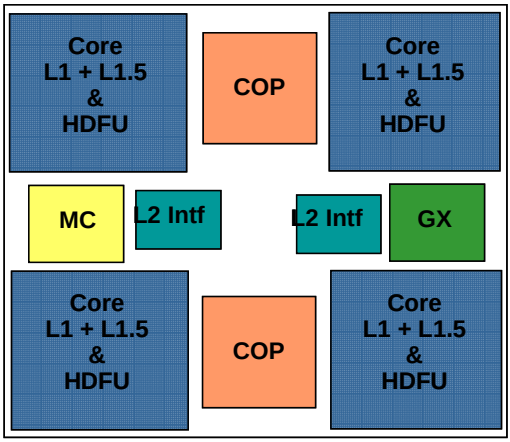


Figura 31 z10 Chip Quad Core

Le dimensioni del modulo MCM sono paragonabili a quelle di un comune floppy disk, 95x95mm, ed è fatto in ceramica sulla quale sono montati i chip composti da circa 8 miliardi di transistors.

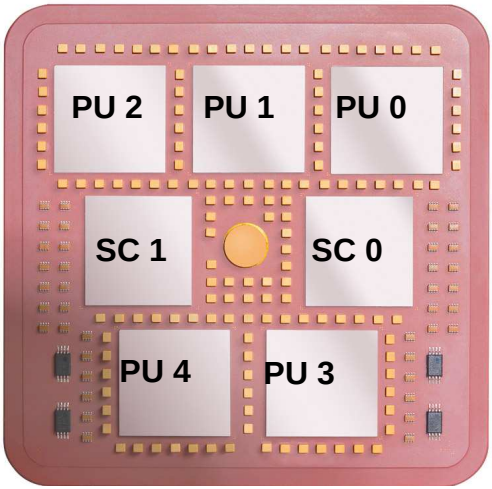


Figura 32 z10 MCM

La tecnologia utilizzata da tutti i chip è la CMOS 11S, basata su 10 strati di interconnessione in rame ed isolamento in Silicon. Lo spessore litografico è di 0,065 micron. Come detto, i 5 chip "quad-core" forniscono 17 o 20 chip processori utilizzabili. Nella versione a "17", 3 chip forniscono 3 soli processori (3x3 = 9), e gli altri 2 ne forniscono 4 (2x4 = 8) per un totale di 17. Nella versione a "20" tutti e 5 i chip forniscono 4 processori (5x4 = 20). Il ciclo base è per tutti i chip è di 0,23ns.

In un MCM a "17" , la seguente tabella illustra lo scopo a cui vengono destinati i processori contenuti nei chip "quad-core":

Modello	PU applicativi	SAP	Spare	Totale
E12	12	3	2	17
E26	26	6	2	34
E40	40	9	2	51
E56	56	10	2	68

Il modello di fascia alta, z10 EC E64, usa un MCM da "20" i cui processori vengono usati in questo modo:

Modello	PU applicativi	SAP	Spare	Totale
E64	64	11	2	77

I processori System Assist Processor (SAP) sono assegnati al Channel Subsystem per svolgere operazioni di I/O e i processori Spare svolgono la funzione di processori di "scorta". Analogamente all'elaboratore z9, anche lo z10 EC, qualunque sia il numero di MCM, ha al massimo solo 2 processori di scorta.

Ogni processore ha a disposizione, sul chip, 256KB di memoria cache di Livello 1, L1, suddivisa in 64KB di cache per le istruzioni e 128KB per i dati. Inoltre è stata aggiunta una cache intermedia, L1,5, di 3MB per processore.

Inportante novità dello z10, è la presenza di un processore per l'Hardware Decimal Floating Point, HDFU, in ogni singolo core. Questo processore svolge tutte le funzioni di calcolo che implicano la presenza di operazioni decimali. In precedenza, fino allo z9, queste funzioni venivano svolte in millicode o dal software. Con l'aggiunta del HDFU, le operazioni svolte

direttamente dal core migliorano le prestazioni fino ad un massimo di 10 volte. Inoltre vengono evitati tutti gli arrotondamenti ed altri problemi dovuti alla conversione da binario a decimale. Questa novità è estremamente importante per tutte le applicazioni finanziarie e commerciali.

Lo Storage Control chip (SC) controlla e gestisce il traffico tra PU e L2, tra L2 e le altre L2, tra L2 gli altri componenti, MSC e MBA.

La cache L2 è alloggiata sui 2 chip System Data (SD), ognuno con una capacità di 24MB.

L'algoritmo di scrittura è Store Through fra L1 e L2; ogni scrittura su L1 viene replicata in modo sincrono - rispetto all'esecuzione dell'istruzione - su L2. E' Store In fra L2 e la memoria reale (le scritture su L2 non sono replicate in modo sincrono sulla memoria reale).

1.2.3 z10 Book

L'altro elemento fondamentale della struttura dei sistemi z10 è il Book (libro), dove viene ospitato il MultiChip Module (MCM). I sistemi z10 usano la stessa tecnica di packaging usato per la famiglia precedente, z9, basato sul concetto di Book. Un Book contiene i processori, alloggiati all'interno del MCM, la memoria centrale, una combinazione di Memory Bus Adapter (MBA) e Host Channel Adapter per la connessione verso i dispositivi di I/O.

I sistemi z10 EC hanno almeno un singolo Book (libro) e possono ospitarne fino ad un massimo di quattro. Tutti i Book risiedono all'interno della CEC (Central Electronic Complex) cage dell'elaboratore.

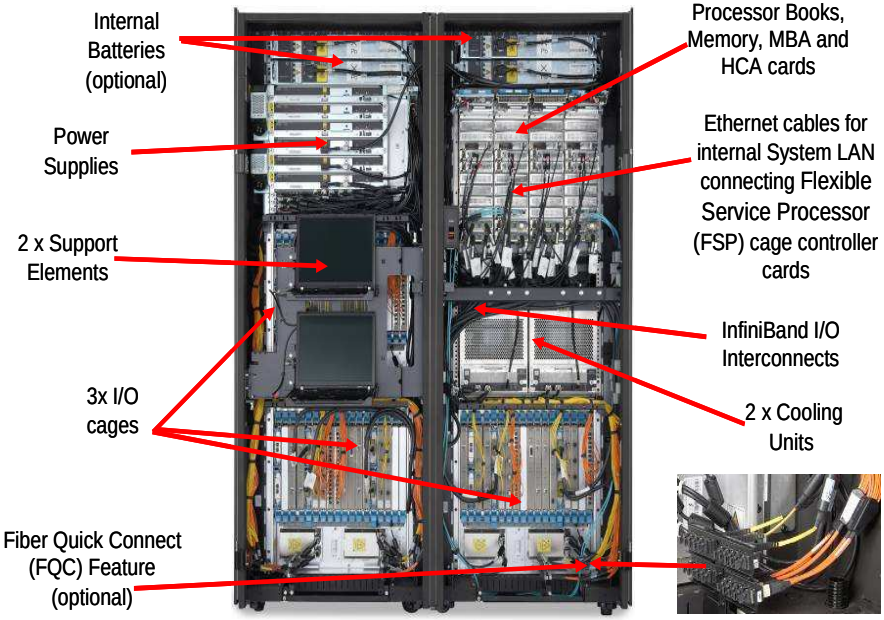


Figura 33 Il sistema z10

Figura 34 Struttura di un Book

In dettaglio, gli elementi contenuti all'interno di un singolo Book sono i seguenti:

- un MCM da 17 o 20 processori (PU)
- memoria da 64 a 384GB
- fino a 8 Fanout Card, HCA o MBA, per la connessione verso le I/O
- tre Distributed Converter Assemblies (DCA), che forniscono "power" al Book

Il raffreddamento dell'elaboratore z10 EC è di tipo ibrido, in quanto ci sono due sistemi indipendenti, uno ad aria, "air-cooled system", ed uno ad acqua, ottenuta grazie ad un sistema interno a liquido refrigerante

I Book di cui è composto un sistema z10 sono interconnessi tra loro per permettere la comunicazione e la condivisione delle risorse di memoria e di I/O tra tutti i processori presenti nell'elaboratore. Dal punto di vista della gestione Hardware esiste un unico Address Space per la memoria Hardware, che comprende tutta la memoria installata su tutti i libri presenti nella configurazione. Logicamente la memoria, che dal punto di vista costruttivo è allocata su Book diversi, è vista come un unico spazio indirizzabile che può arrivare fino a 1,5TB.

Un punto essenziale della struttura a libri è che le cache di L2 memorizzano i dati acceduti dai CP presenti sul libro stesso. Non sono legate alla memoria installata sul Book. Una conseguenza di questo disegno è che una certa posizione di memoria può essere presente in più di una cache L2. Questo principio costruttivo è essenziale per riuscire a mantenere un Cache Hit elevato.

E' abbastanza evidente che per mantenere un elevato livello di prestazioni è necessario che ogni CP acceda il più possibile a dati che sono presenti sul suo Book o perché sono presenti sulla memoria del Book, o perché si trovano sulla cache di L2 del proprio Book. Per ottenere questo si è resa necessaria una profonda revisione degli algoritmi che regolano il funzionamento del Partizionamento logico. Per un approfondimento su questo tema si rimanda al seguente articolo di "IBM Journal of Research and Development"
http://www.research.ibm.com/journal/rd/483/siegel.html

Diversamente dalla precedente famiglia di elaboratori z9, nello z10 EC la connessione tra i diversi Books è di tipo point-to-point, che consente ad ogni Book di comunicare con tutti gli altri in maniera diretta. In questo modo il data-tansfer non passa mai attraverso un altro Book.

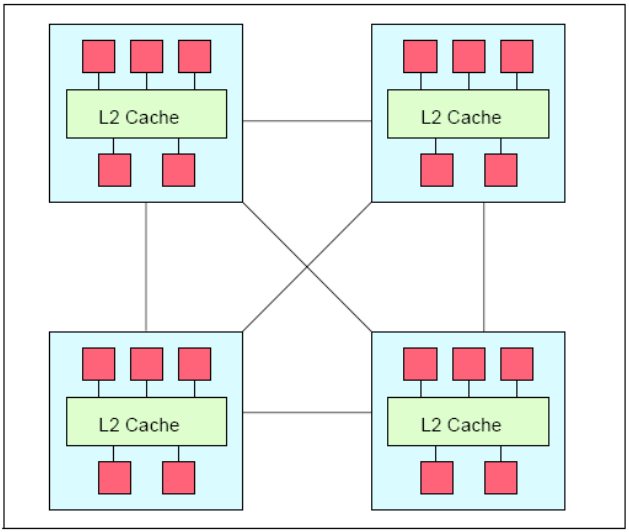


Figure 2-5 Point-to-point topology for book-to-book communication

Figura 35 Comunicazione tra i Book

La connettività verso i dispositivi di I/O è assicurata da una combinazione di otto adapters fanouts di tipo MBA o HCA, localizzati nella parte frontale del Book. Ogni fanouts ha due connessioni di tipo InfiniBand. Ci sono 3 diversi tipi di fanouts:

- HCA2-O: con connessioni ottiche InfiniBand di tipo Coupling Links PSIFB ad un max data rate di 6GB/sec
- HCA2-C: con connessioni in rame InfiniBand per tutti gli altri dispositivi di I/O , ISC-3 e Crypto cards
- MBA: solo per le connessioni ICB-4

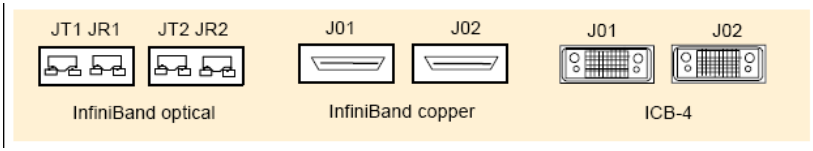


Figura 36 Fan out

1.2.4 La specializzazione dei processori

Un MCM contiene 17 oppure 20 processori quad-core. A questi processori viene assegnata una "caratterizzazione", ovvero un ruolo, mediante un opportuno firmware [9] caricato all'accensione della macchina. Una volta caratterizzati, i processori possono essere di due tipi, Utente e Servizio.

I processori Utente sono:

- **Central Processor (CP):** possono essere utilizzati da qualunque sistema operativo; sono anche chiamati Processori Standard o General Purpose.
- **Integrated Facility for Linux (IFL):** usati esclusivamente da sistemi per Linux e z/VM.
- **zSeries Application Assist Processor (zAAP):** dedicati all'esecuzione di applicazioni Java che girano sotto z/OS.
- **zSeries Information Assist Processor (zIIP):** dedicati all'esecuzione di attività legate al DB2 e alla Crittografia (IPSEC) sotto z/OS.
- **Integrated Coupling Facility (ICF):** usati nella realizzazione della struttura di Cluster di Sistemi z/OS denominata Parallel Sysplex.

I processori di Servizio sono:

- **Service Access Processor (SAP):** legati alla gestione delle Operazioni di I/O (Vedi 1.2.6); ??? in ogni sistema di ultima generazione è sempre attivo almeno uno di essi.
- **Spare Processor (Spare):** presenti sull'MCM ma non attivi, essi vengono attivati automaticamente in caso di guasto di uno degli altri sia di Servizio che Utente; in un Sistema z10 EC sono sempre presenti almeno due processori Spare.

[9] programma nativo del computer e non modificabile dall'utente, punto di incontro fra componenti logiche e fisiche, ossia fra hardware e software

I processori Utente denominati zAAP e zIIP sono stati introdotti per ottimizzare l'infrastruttura IT che ospita l'Application Server e il Database Server. Fino a qualche anno fa, una delle soluzioni più adottate era quella di realizzare reti di serventi inteconnesse tra loro, distribuendo le funzioni da svolgere tra i vari componenti. Ad esempio, alcuni serventi eseguono i programmi applicativi ("application server"), altri accedono ai dati ("database server"). Una rappresentazione schematica è quella illustrata nella configurazione di sinistra di Figura 37.

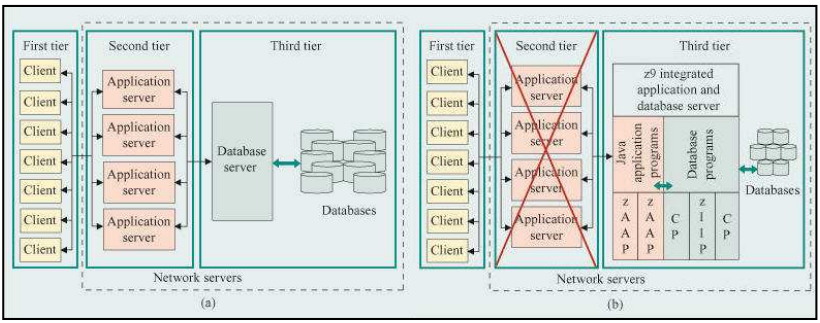


Figura 37 Multitiered networks (a) distributed (b) simplified with zIIP+zAAP

Con il tempo, il numero di serventi applicativi è cresciuto in maniera significativa, specialmente per quelle applicazioni disegnate per supportare i nuovi modelli architetturali internet-oriented o web-based. All'interno di questi modelli emergenti le nuove tecnologie di programmazione Java e XML[10] giocano un ruolo fondamentale, in quanto offrono dei vantaggi dal punto di vista della produttività nello sviluppo dei programmi, riducendone i tempi ed i costi associati.
Conseguenze non volute di questa proliferazione dei serventi distribuiti ("server farm") sono la riduzione della affidabilità totale dell'infrastruttura informatica, l'aumento della complessità di gestione e l'inefficienza

[10] Extensible Markup Language

nell'utilizzo dei serventi. Per risolvere questi problemi si è provveduto ad ospitare all'interno dei sistemi z/OS sia gli Application Server che i Database Server. I processori zAAP sono delegati all'esecuzione degli Application Server basati su Java, i processori zIIP supportano i Database Server basati su DB2. Il processore zAAP esegue istruzioni di codice applicativo Java; zIIP è delegato ad eseguire parte del carico generato dalle funzioni di gestione del database relazionale DB2 per z/OS ovvero operazioni relative alla sicurezza IP.

Insieme, questi processori riescono ad eseguire una considerevole quantità del carico di lavoro richiesto dalle suddette funzioni. Di conseguenza, liberano della capacità di calcolo a carico dei processori cosiddetti "general purpose" che la rendono disponibile ad altri utenti. In questo modo, la piattaforma IBM System z10 è in grado di offrire ottime prestazioni, funzioni avanzate ed un costo estremamente competitivo in relazione a soluzioni distribuite più complesse e meno efficienti. La configurazione di destra della Figura 37 illustra una soluzione che usa motori Standard, zAAP e zIIP con una evidente semplificazione di tutta l'infrastruttura.

Entrambi i processori operano in maniera completamente trasparente ai programmi Java che usano lo zAAP e alle funzioni di database DB2 che usano lo zIIP. Il sistema operativo z/OS e il meccanismo di Partizionamento logico coordinano l'attività di motori Standard, zAAP e zIIP. Tutto questo viene svolto in modo trasparente agli Application Server e Database Server. A differenza dei processori standard (CP), zAAP e zIIP hanno delle caratteristiche particolari quali, ad esempio, l'impossibiltà di eseguire la funzione di IPL[11], cioè il caricamento iniziale del sistema operativo o, eventualmente, di un programma di utilità particolare, o di eseguire istruzioni legate alle operazioni di I/O.

[11] Initial Program Load

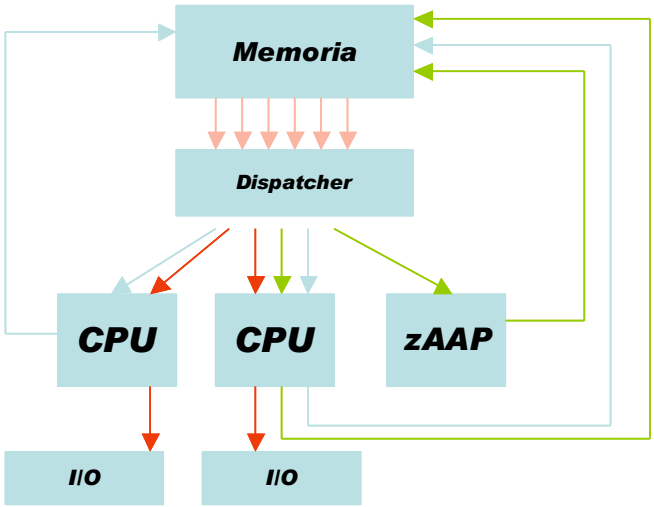


Figura 38 Motori specializzati e motori tradizionali

1.2.5 Struttura e Principali dispositivi di I/O

I processori dei sistemi z10 eseguono esclusivamente parte di programmi che non prevedono operazioni di I/O. Quando c'è la necessità di eseguire questo tipo di operazioni, viene investito di questo compito un componente chiamato Channel Subsystem che usa particolari processori specializzati di Servizio detti Service Access Processors (SAP). Figura 39.

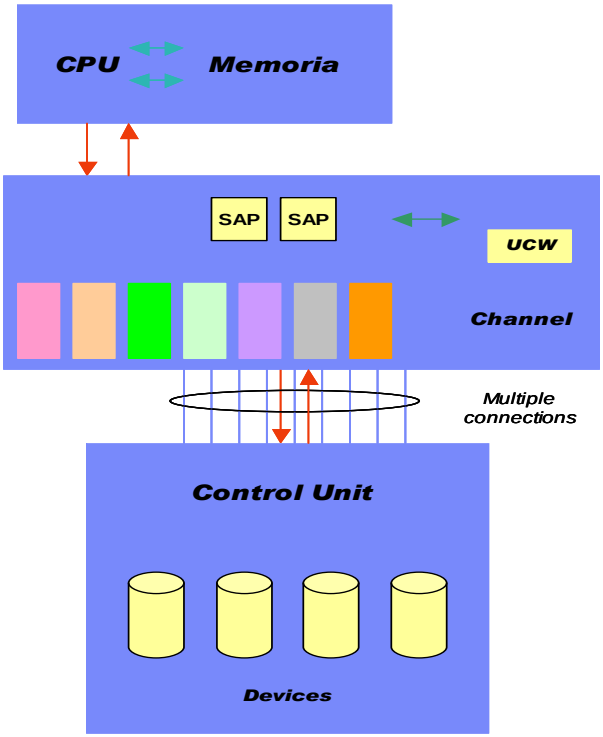


Figura 39 Il Channel Subsystem

Il Channel Subsystem è formato da alcuni processori SAP, in dipendenza del modello e del numero di Book presenti, 1024 canali (in configurazione massima) per sistema, un numero massimo di blocchi di controllo, detti subchannels o UCW, di 64k che rappresentano i devices collegati, i devices collegati attraverso le unità di controllo.

Il Channel Subsystem ha due strutture logiche caratteristiche:

- **Channel:** rappresenta una linea di connessione con il device di I/O
- **UCW:** elemento descrittivo del device

Queste due strutture permettono il collegamento ad ogni dispositivo esterno (device) con un numero massimo di connessioni simultanee di 8. Il protocollo di I/O permette di fare quella che viene detta "Dynamic Path Reconnection" cioè interrogare il device con un canale e ricevere la risposta su un altro dei canali ad esso collegati, in modo da utilizzare in maniera paritaria e bilanciata tutte le connessioni. Il vantaggio, oltre ad un uso bilanciato di tutte le connessioni, è l'alta affidabilità perché se un canale si rompe posso sempre ottenere la risposta da un altro.

Una richiesta di I/O viene fatta da un task al componente del Sistema Operativo denominato Input/Output Supervisor (IOS) attraverso una SVC 0 instruction (Supervisor Call). Come stabilito dall'architettura, IOS lancia l'istruzione START SUBCHANNEL (SSCH) e il processore (CP) delega ad un canale l'esecuzione dell'operazione.

L'operazione è un dialogo tra il canale e l'unità di controllo per muovere i dati tra la central storage ed il device controllato da questo controller. Dopo l'esecuzione della SSCH, IOS restituisce il controllo al task il quale rimane in WAIT, fino alla fine della operazione di I/O.

Come già detto, l'operazione di I/O non è gestita dal processore centrale, CP, ma dal processore SAP, il quale libera il processore CP che potrà dedicarsi ad altri lavori. Il SAP seleziona un canale disponibile e si assicura che l'operazione "parta".

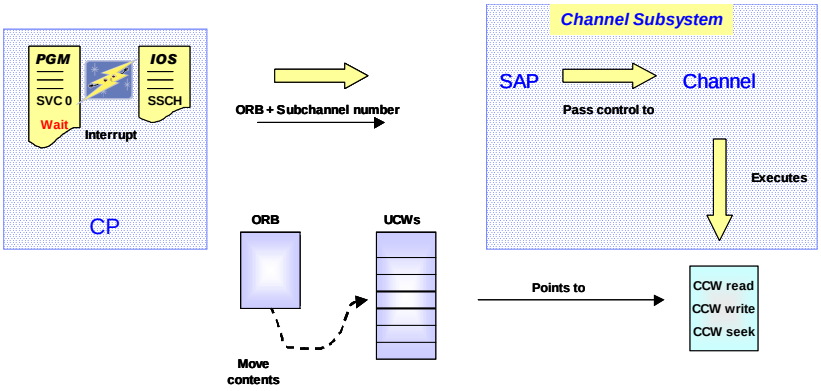


Figura 40 Logica delle operazioni di I/O

A valle del Channel Subsystem ci sono le schede di I/O che permettono l'accesso, attraverso i canali ad esse collegate, ai device.

Le principali schede di I/O sono:

- **ESCON:** connessione a fibra ottica con protocollo IBM proprietario, usata per collegare dischi, nastri, terminali di vecchio tipo e stampanti. Velocita max 17MB/sec.
- **Ficon/FC:** connessione a fibra ottica con protocollo IBM proprietario nel caso di Ficon o protocollo standard Fiber Channel nel caso FC. Usata per collegare dischi, nastri e Storage Area Network (SAN). Velocità 1, 2, 4Gbit/sec con connessioni led shortwave (SX) o laser longwave (LX).
- **OSA:** connessione standard per LAN Ethernet, supporta collegamenti in fibra ottica, SX o LX, e in rame RJ45. Usata dal protocollo di rete standard TCP/IP. Velocità da 10 a 1000 Mbit/sec, in rame, e da 1 a 10Gbit/sec in fibra.

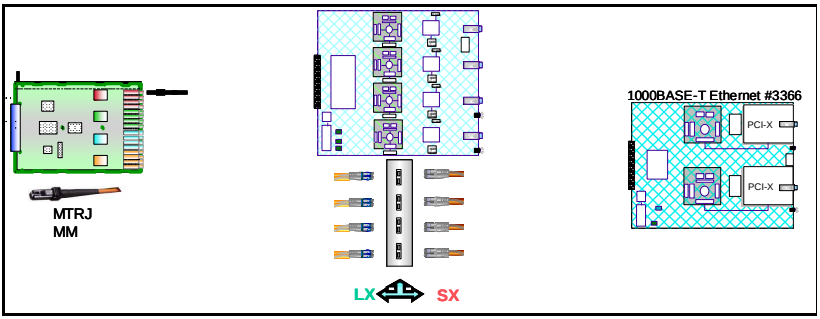


Figura 41 ESCON, Ficon, OSA

Oltre queste schede, ne esistono altre utilizzate per il Parallel Sysplex per i collegamenti interni e per la crittografia. Inoltre ricordiamo alcuni tipi particolari di connessioni interne molto usate dai dispositivi di virtualizzazione:

- **Internal Coupling Links:** connessioni per il clustering all'interno dello stesso sistema con protocollo proprietario
- **Coupling Links (ISC):** connessioni per il clustering con apparecchiature Coupling Facility all'esterno del sistema. Velocità da 100 a 200 MB/sec.
- **Crypto Processor:** schede per il supporto della crittografia
- **HiperSocket**: connessioni virtuali interne al sistema per collegare le partizioni logiche tra loro. Supportano i protocolli SNA e TCP/IP. Velocità maggiore di 300Mbytes/sec.

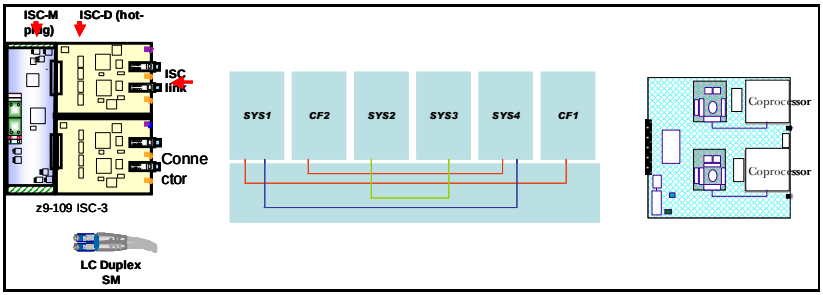


Figura 42 ISC, HiperSocket, Crypto

1.2.7 Struttura di Clustering di Sistemi (Parallel Sysplex)

L'insieme di due o più immagini di sistema operativo z/OS, o di due o più sistemi di z/Architecture, fino ad un massimo di 32 immagini, quando operano come una unica entità (cluster), viene definita Parallel Sysplex. Gli scopi per i quali i sistemi co-operano tra loro sono:

- controllo del sistema unico e centralizzato (console unificata)
- controllo degli accessi e della sicurezza centralizzato
- possibilità di condividere e distribuire il carico di lavoro tra i vari sistemi co-operanti
- capacità di condividere dati (Data Sharing) e mantenerne l'integrità in lettura e, soprattutto, in scrittura

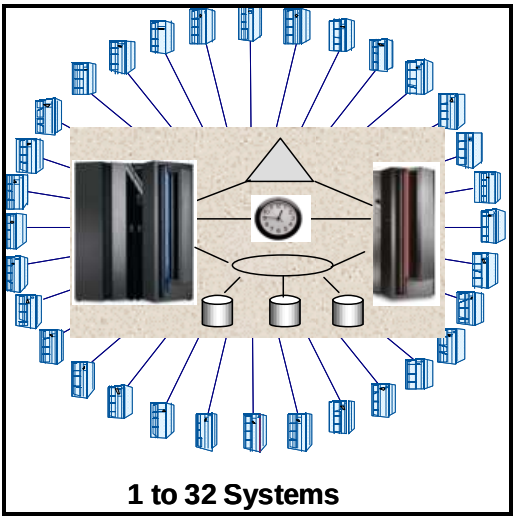


Figura 43 Parallel Sysplex

Per realizzare la condivisione dei dati e il controllo centralizzato è necessario disporre di un terzo componente, in genere duplicato, denominato **Coupling Facility**, il quale può essere esterno o interno ad uno dei sistemi del cluster. Coupling Facility è un elaboratore nel quale gira un sistema operativo specializzato detto Coupling Facility Control Code (CFCC). In esso la memoria interna è dinamicamente partizionata in entità dette "strutture", che hanno funzioni specifiche e posso essere di 3 tipi:

- **Cache:** usata come un buffer ad alta velocità per memorizzare dati acceduti in read e/o write da tutti i componenti del cluster. Viene usato un meccanismo di "buffer invalidation" per assicurare l'integrità dei dati in cache
- **List:** usato per funzioni di condivisione di aree di lavoro e per memorizzare lo status riguardo a funzioni e/o componenti di sistema
- **Lock:** per serializzare l'accesso alle strutture condivise.

Per connettere tra loro le componenti di un Parallel Sysplex si usano connessioni ad alta velocità dette Coupling Links e per far sì che tutti i timer di tutti i Sistemi componenti il Cluster siano sincronizzati tra loro viene utilizzata un'apparecchiatura esterna chiamata Sysplex Timer, collegata a ciascun sistema fino ad una distanza massima di 40km, che fornisce loro lo stesso orario, "time". Di recente è stata introdotta una nuova tecnologia denominata Server Time Protocol basata sull'uso di Sorgenti e protocolli di Tempo standard.

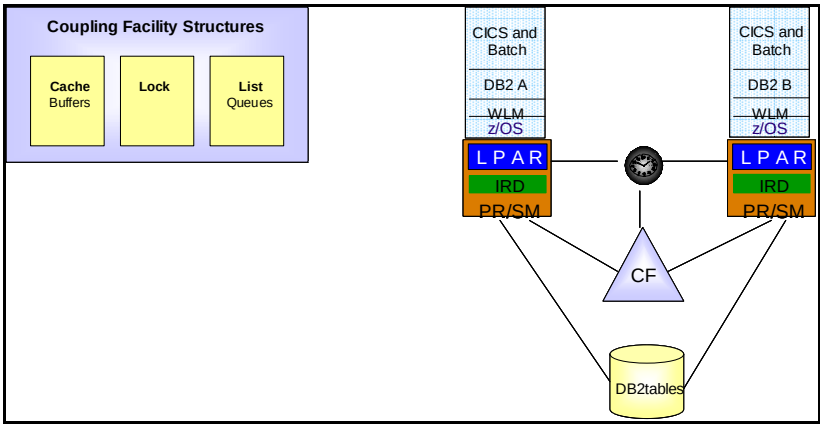


Figura 44 Coupling Facility

Le caratteristiche funzionali del Parallel Sysplex sono le seguenti:

- **Data Sharing:** condivisione di dati gestiti da sottosistemi quali DB2, IMS, VSAM, con garanzia della integrità in read e write. L'accesso ai dati può avvenire anche da sistemi posti in locazioni fisicamente lontane.
- **Operazioni centralizzate:** da uno qualsiasi dei componenti (ad esempio console unica) con la possibilità di attivazione e disattivazione senza interruzione di servizio.
- **Scalabilità orizzontale:** i sistemi possono essere aggiunti e attivati senza interruzione del servizio e in maniera trasparente alle applicazioni.
- **Bilanciamento Automatico del carico:** ciascun carico di lavoro sarà eseguito con un obiettivo di prestazioni pre-assegnato in base alle politiche del Workoad Manager e potrà essere dirottato in maniera automatica e trasparente sul sistema meglio posizionato degli altri per quanto riguarda l'occupazione di carico ed utilizzo di risorse, cioè in grado di ricevere altro lavoro.
- **Disponibilità Continua**: il cluster continua a fornire servizio nel caso una delle sue componenti dovesse fermarsi o cadere. Per

questa ragione si presenta come un unico sistema in grado di assicurare la disponibilità continua di dati e applicazioni.

1.3 Il Sistema Operativo z/OS

Questo capitolo descrive le più importanti caratteristiche del Sistema Operativo IBM denominato z/OS, si tratta del Sistema Operativo maggiormente diffuso sui Sistemi Centrali di z/Architecture.

1.3.0 Introduzione

Il Sistema Operativo z/OS è il sistema che meglio utilizza le caratteristiche offerte dalla z/Architecture. Esso rappresenta l'ultimo gradino di una evoluzione le cui tappe fondamentali sono OS/360 (1964), MVS - Multiple Virtual Storage (1974) e OS/390 (1996). La versione corrente è la 1.8, la versione 1.9 sarà disponibile a breve.
z/OS è sviluppato nei laboratori di Poughkeepsye (NY) da IBM Corporation e viene concesso in licenza d'uso agli utenti. Esso rappresenta il Sistema Operativo maggiormente usato dagli utenti di mainframe nel mondo. A tutt'oggi è il sistema che gestisce la massima parte delle applicazioni "Mission Critical" di molte grandi organizzazioni, specialmente di grandi Banche e grandi Enti Governativi.
In Italia è utilizzato da tutte le maggiori Banche e da molti Enti Centrali presso i quali gestisce le applicazioni Core Business.
Le ragioni del successo di un tale Sistema Operativo e la sua lunga permanenza sul mercato (le prime versioni risalgono al 1964) vanno ricercate in due importanti caratteristiche:

1. il fatto che esso sia stato "pensato" e quindi progettato "ab initio" come gestore di grandi volumi di dati e transazioni con molti utenti concorrenti con alte capacità di integrità e sicurezza;
2. la caratteristica di compatibilità verso l'alto delle applicazioni che sotto il suo controllo vengono eseguite: è molto frequente che utenti di z/OS eseguano ancora oggi senza modifiche applicazioni

progettate negli anni ’60 ed ancora valide dal punto di vista applicativo.

Tuttavia negli ultimi anni z/OS è profondamente mutato per adeguarsi alla crescita dell’hardware, alle nuove implementazioni dell’Architettura e per aderire ai nuovi standard dell’informatica quali ad esempio il linguaggio C prima e Java dopo, i protocolli TCP/IP, gli standard web html/http. Tutte queste tecnologie emergenti ed i relativi “standard” fanno parte oggi del Sistema Operativo z/OS, che comunque rimane compatibile con gli ambienti tradizionali.
Tra le sue caratteristiche è molto importante la capacità di realizzare Cluster di Sistemi anche distanti fra di loro parecchie decine di chilometri (denominati Parallel Sysplex o Geographically Dispersed Parallel Sysplex per le soluzioni di Disaster Recovery), che consentono continuità operativa dell’infrastruttura anche a fronte di interruzioni di uno dei suoi nodi (Business Continuity). La continuità viene realizzata attraverso una completa condivisione dei dati e senza perdita di alcuna transazione. Una continuità di questo tipo richiede la capacità da parte della piattaforma di ridistribuire il carico fra i nodi superstiti ed eventualmente attivare la capacità di backup (“on demand”).
Tale versatilità permette di aumentare o diminuire dinamicamente e senza interruzione di servizio la capacità di calcolo dei Sistemi, il numero e la velocità dei processori che lo compongono, il numero ed il tipo di risorse di I/O collegate. Questa caratteristica, presente nell’hardware, viene utilizzata dai Sistemi Operativi della famiglia “mainframe” da sempre e solo nel recente passato ha iniziato a comparire sul mercato in altri sistemi basati su altre architetture e sistemi operativi.
Di seguito vengono riportate le caratteristiche principali del Sistema e le sue componenti:

- z/OS è un Sistema Operativo a 64 bit
- è stato progettato per trattare grandi volumi di lavoro
- è concepito per servire contemporaneamente molti utenti
- è ottimizzato per gestire grandi volumi di dati
- è progettato per servire applicazioni “critiche” per il business in maniera sicura

Un componente fondamentale è il WorkLoad Manager (WLM). La sua funzione principale è classificare ogni lavoro in base ad un "obiettivo di prestazione" (GOAL) basato su criteri di business e non tecnici: un esempio di goal è il fatto che il sistema di immissione delle prenotazioni di una aerolinea risponda agli operatori di terminale in meno di un secondo dopo la richiesta. Il WLM traduce questi goal in priorità di esecuzione per le transazioni legate a quegli obiettivi in maniera trasparente all'utente ed in modo tale che tutti i goal previsti per i lavori presenti nel Sistema possano essere raggiunti tenendo conto delle risorse disponibili.

Il WLM è anche in grado di comunicare all'Amministratore del Sistema quali obiettivi vengono raggiunti e per questi ultimi è in grado di indicare perché l'obiettivo non viene raggiunto. Il WLM permette di controllare la qualità del servizio (Quality of Service - QoS) erogato agli utenti della piattaforma e consente di attivare su un sistema z/OS lavori di diversa natura garantendo per ognuno di questi il raggiungimento del proprio goal, ottimizzando le risorse per il raggiumento dei requisiti applicativi.
Per quanto concerne la struttura del Sistema Operativo osserviamo che:
z/OS si basa sull'uso della Memoria Virtuale. In particolare in z/OS viene implementata la funzione delle memorie virtuali multiple. L'idea base è che ad ogni processo venga assegnato per proprio uso esclusivo uno spazio virtuale grande quanto l'indirizzamento dell'architettura permette. Questi spazi virtuali vengono chiamati **Address Space (AS).**
All'interno di ogni Address Space l'esecuzione vera e propria è identificata da due diverse strutture, denominate anche **Unit of Work (UoW): Task Control Block (TCB)** e **Service Request Block (SRB)**; qualunque processo in esecuzione sul Sistema, è sempre riconducibile ad un una serie di TCB o SRB i quali a loro volta indicano i programmi che debbono essere eseguiti per quel processo.

Le componenti eseguibili (programmi) di z/OS si distinguono in:

- Programmi (moduli) di **Sistema**, che sono le componenti del Sistema Operativo prodotte da IBM
- Programmi (moduli) dell' **Utente**

z/OS è progettato per essere eseguito su Sistemi a Processori Multipli Simmetrici (SMP). Non esistono "a priori" criteri di affinità tra un certo processore ed i TCB ed SRB attivi nel sistema. Infatti in linea di principio

ogni UoW è eseguibile su qualunque processore a qualunque Address Space appartenga.L'introduzione dei processori specializzati (vedi capitolo 1.2.5) ha creato alcune "affinità" nell'esecuzione:

- Ogni TCB, nel momento in cui esegue codice Java, può essere eseguito sui processori zAAP (zSeries Application Assist Processor).
- Alcuni SRB relativi alla gestione di dati relazionali (DB2) possono essere eseguiti sui processori zIIP (zSeries Information Integration Processors).

1.3.1 L'Address Space

Vediamo adesso la struttura degli Address Spaces.

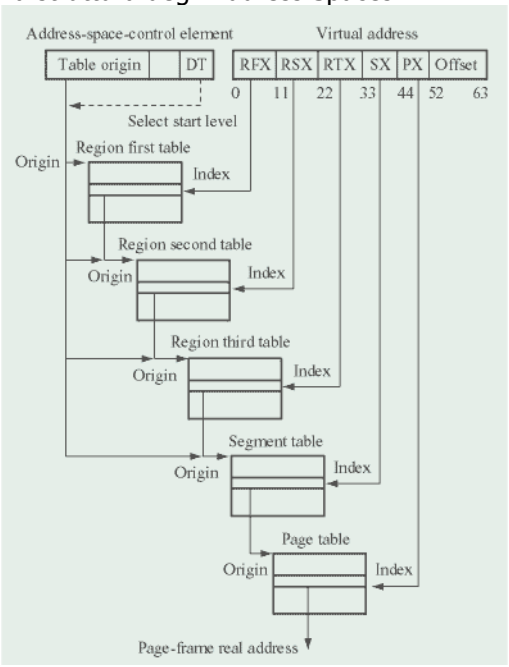


Figura 45 z/Architecture Dynamic Address Translation

Lo z/OS ha scelto per l'implementazione della memoria virtuale una dimensione della pagina virtuale di 4KB (4096 Byte). Ogni Address Space è suddiviso in Pagine, Segmenti, Regioni secondo lo schema di Figura 39. La dimensione della pagina virtuale determina le dimensione della Frame reale e dello Slot ausiliario: tutti avranno una dimensione uguale. Per eseguire un programma esso deve innanzitutto essere caricato in memoria virtuale dalla libreria su disco che lo contiene. Poiché caricare un programma da una libreria di programmi eseguibili equivale a trasferire mediante una serie di operazioni di I/O il testo del programma in memoria centrale, alla fine del caricamento il programma si trova caricato in memoria virtuale e in memoria centrale per cui è pronto per essere eseguito.

Il processo di Paging viene assistito dal dispositivo hardware denominato DAT (Dynamic Address Translator) descritto nel capitolo 1.1.1, il DAT si occupa di tradurre gli indirizzi "virtuali" della memoria virtuale in indirizzi assoluti della memoria reale del Calcolatore.

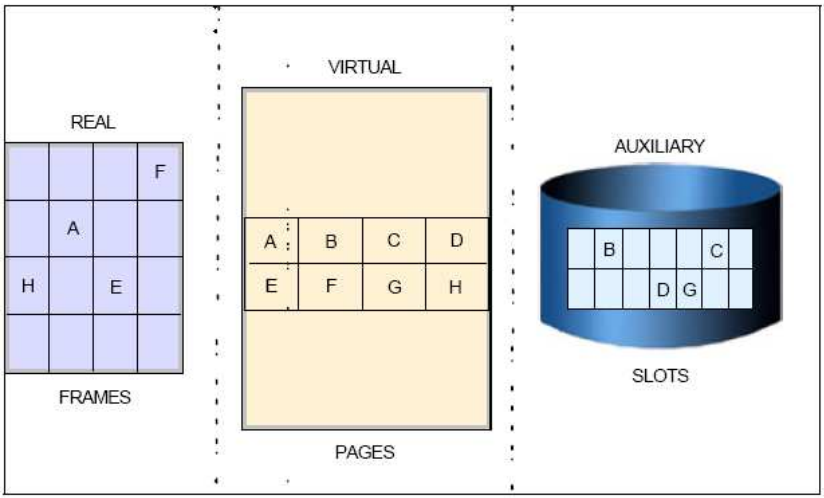


Figura 46 Strutture per la memoria virtuale

Le Pagine della memoria centrale prendono il nome di Frames, quelle della memoria ausiliaria prendono il nome di Slot. Nella figura 40 si vede la relazione fra Memoria Virtuale, Memoria Reale e Memoria Ausiliaria.

Per supportare questa funzione un componente dello z/OS, il Real Storage Manager (RSM) provvede a costruire Region Table, Segment Table e Page Table. In z/OS viene assegnato un Address Space (AS) ad ogni processo. Ogni utente che si connette ad un sistema z/OS via TCP/IP è un processo autonomo che risiede in un proprio AS. Per ogni AS esiste un set di Region/Segment/Page Table che serve alla traduzione degli indirizzi di quell'AS.

Cominciamo col definirne la dimensione, infatti poiché un AS è una struttura "logica", esso avrà le dimensioni massime indirizzabili dall'Architettura. Essendo la z/Architecture un'architettura a 64 bit, la dimensione dell'AS sarà di 16 Exabyte (2**64). Poiché la dimensione dell'Address Space è aumentata continuamente, l'architettura supporta tre modalità di indirizzamento (24, 31 e 64 bit).

Perciò all'interno di un AS si individuano due confini ben precisi indicati nella figura seguente.

- La linea dei 16 Megabytes (indicata anche come ***The "Line"***) per programmi che funzionano con indirizzamento a 24 bit. Questi programmi dovranno risiedere nella zona compresa 0 e 16 MB (cioè risiedere ***below the line***).

- La linea dei 2GB (indicata anche come ***The "Bar"***) entro la quale devono risiedere i programmi a 31 bit.

- La zona al di sopra dei 4GB (e fino a 16 Exabyte) a disposizione per tutte le applicazioni in grado di sfruttare l'indirizzamento a 64 bit.

I programmi che supportano solamente indirizzi a 31 bit dovranno risiedere nell'area da 0 a 2GB. La spiegazione del contenuto fra 2GB e 4GB è riservata e quindi può essere argomento trattato in un corso di approfondimento dell'architettura.

Ad oggi lo z/OS supporta l'indirizzamento a 64 bit solo per Operand Fetch e non per Instruction Fetch. In altre parole potranno risiedere *above the bar* solo dati; i programmi in esecuzione dovranno risiedere sotto i 2GB (l'instruction fetch resta a 31 bit).

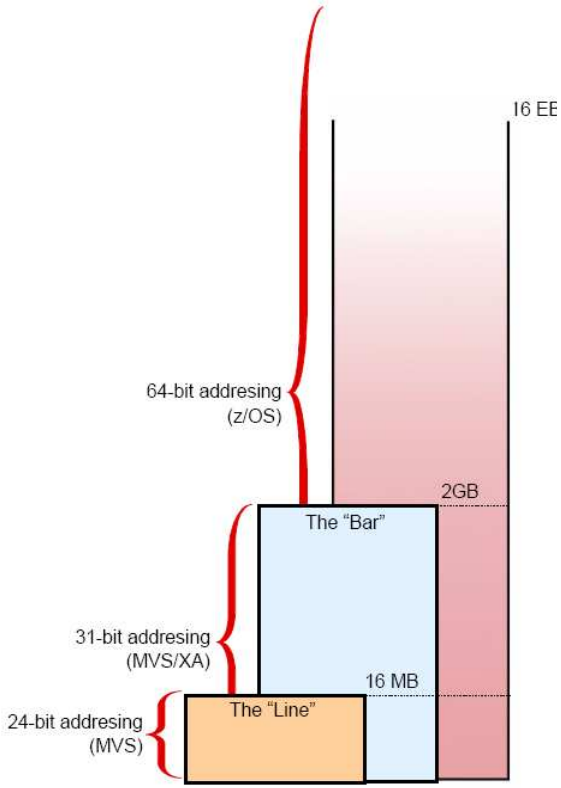


Figura 47 Differenti memory addressing

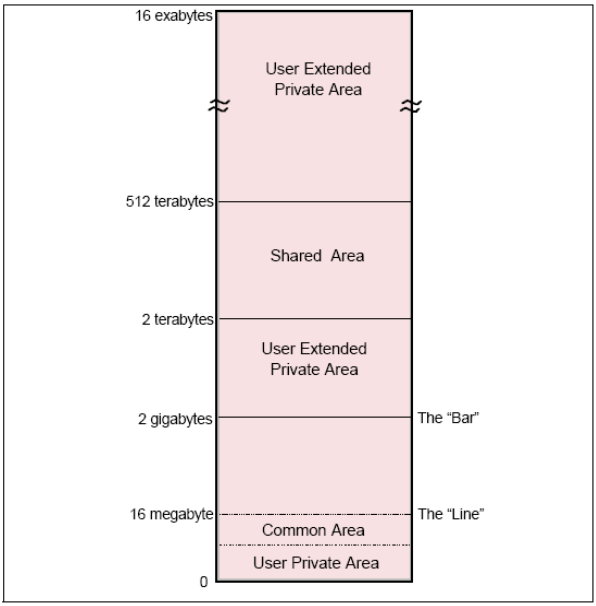


Figura 48 Struttura di un Address Space

La Figura 48 indica la struttura di un Address Spaces per un Sistema a 64 bit . All'interno dell'AS rileviamo la presenza di parti "comuni", "shared" e "private". Le parti "**comuni**" sono quelle relative a moduli del Sistema Operativo "vitali"; essi vengono utilizzati da tutti i processi e pertanto fanno idealmente parte di tutti questi. Si osservi che dire che tutti gli AS hanno lo stesso modulo "comune" non vuole dire che esso venga copiato tante volte quante sono gli AS, bensì che esso viene "usato" da tutti e quindi fa idealmente parte di tutti gli AS.

Le parti "**private**" sono invece tipiche di un certo processo e solo di quello: esse contengono ad esempio i programmi ed i dati utilizzati da un certo utente del Sistema Operativo in un certo momento. Per definizione esse fanno parte di un solo AS per volta.

Si osservi che se lo stesso programma viene utilizzato da due utenti in due momenti diversi, esso sarà in generale considerato come un processo differente.

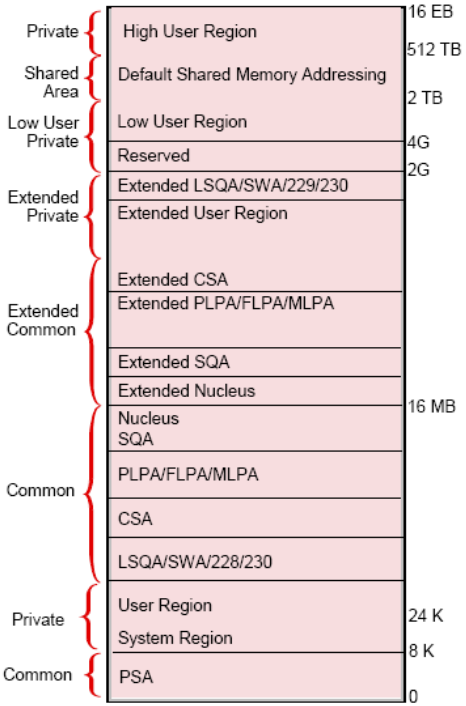


Figura 49 Struttura di un Address Space (dettagliato)

Le parti "**shared**" ovvero condivise, possono appartenere ad AS differenti allo stesso momento e quindi condivise.

La Figura 43 mostra il dettaglio della struttura di un AS a 64bit e di questo evidenzia le parti comuni (Common), Private e Condivise (Shared). Per una descrizione dettagliata dei contenuti si rimanda a successivi approfondimenti.

Poiché la logica degli AS si estende a tutti i processi (programmi) presenti nel Sistema, cioé anche ai moduli del Sistema Operativo stesso, distinguiamo diversi tipi di AS:

- **System Address Spaces**: vengono creati all'avvio del Sistema Operativo (IPL - Initial Program Load) ed eseguono tutte le funzioni necessarie al funzionamento del Sistema e degli altri AS.
- **SubSystems Address Spaces**: sono relativi ai cosidetti sottosistemi; i Sottosistemi sono particolari programmi che svolgono funzioni per più utenti contemporaneamente; sono sottosistemi ad esempio i Gestori di Basi di Dati (DBMS) o i Middleware o il JES2 ed il TSO/E descritti nel seguito.
- **TSO/E Address Spaces:** Il TSO/E è un sottosistema che consente l'accesso al Sistema z/OS da parte di un terminale ad un utente singolo: per ciascun utente che accede al Sistema il TSO/E crea un singolo Address Spaces di questo tipo.
- **User Address Spaces**: creati per ogni programma che viene eseguito nel Sistema al di fuori da tutti i casi precedenti (per esempio un programma Batch, cioé senza interventi umani continuativi).

Il numero di AS attivi in un Sistema z/OS è quindi grande, mediamente dell'ordine delle centinaia; è bene osservare che il numero di AS attivi non è direttamente correlato al numero di utenti che stanno interagendo col Sistema in un certo momento: se è vero infatti che per ogni utente che accede tramite TSO/E viene creato un address space, è anche vero che esistono dei Sottosistemi transazionali (vedi dopo) che a fronte di un solo o di un numero contenuto di AS sono in grado di servire decine di migliaia di utenti.
Poiché un AS è associato ad un utente, l'AS diventa anche il contenitore per tutti i processi che la presenza dell'utente innesca nel sistema.

Il meccanismo degli AS assicura tra l'altro la capacità di utilizzo di Sistemi a molte CPU (con le regole indicate in precedenza riguardo i processori specializzati) e l'isolamento tra i vari utenti (Processi) mediante l'utilizzo di

apposite chiavi di protezione, autorizzazioni specifiche e con un sistema apposito di code.

1.3.2 La gestione della Memoria Reale

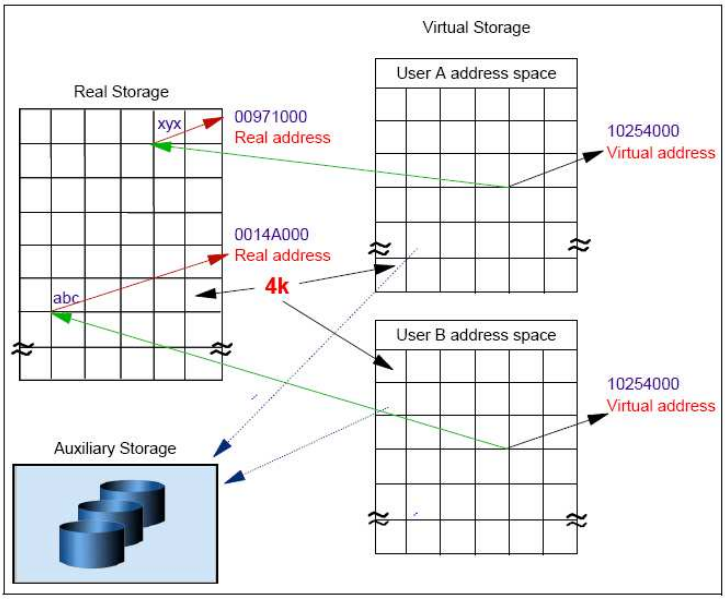


Figura 50 Memoria virtuale

Le risorse Hardware su cui insiste l'implementazione della memoria virtuale dello z/OS sono:

- Memoria Centrale – ovvero la RAM a disposizione dei processi ed accessibile dai Processori;
- Memoria Ausiliaria – ovvero qualsiasi altro dispositivo in grado di ospitare temporaneamente i contenuti della memoria Centrale quando essa non riesce più a contenerli.

Perché un programma possa essere eseguito è necessario che le pagine virtuali che contengono le istruzioni in corso di esecuzione e gli operandi che esse richiedono si trovino in memoria reale. Tutte le altre pagine virtuali relative al programma possono trovarsi sulla memoria ausiliaria. Quindi la Memoria Reale necessaria all'esecuzione di un programma può essere minore della dimensione del programma stesso.

Cosa avviene quando l'esecuzione di una istruzione non trova in memoria reale i campi di cui ha bisogno? o, in termini più aderenti all'architettura, cosa avviene quando una o più pagine contenenti i campi necessari all'esecuzione dell'istruzione non si trovano in memoria reale? Questo evento viene indicato dall'architettura con il termine **Page Fault**.
Ogni sistema operativo costruito sulla z/Architecture deve provvedere a portare in memoria reale da disco la pagina mancante: deve cioé effettuare l'operazione di I/O che legge 4096 byte corrispondenti alla pagina mancante memorizzati in apposite strutture chiamate **Page Data Set** (su disco) in Frame in memoria reale. Questa operazione si chiama **Page-In** e ha la durata che ogni operazione di I/O su disco ha: tipicamente 1 msec; l'esecuzione di un Page-In è sincrona rispetto all'esecuzione dell'istruzione che l'ha causato.
E' facile fare il confronto fra l'esecuzione vera e propria dell'istruzione, pochi nanosecondi, e l'esecuzione dell'operazione di I/O che risolve un Page Fault: il rapporto è vicino al milione di volte. E' evidente come sia indispensabile per garantire un'efficiente esecuzione di un programma che gli eventi di Page Fault siano tenuti al minimo, senza ricorrere all'ovvia soluzione che tutto il programma stia in memoria reale perché così si avrebbe bisogno di una quantità di memoria reale molto più grande di quella necessaria per avere delle performance accettabili.
Introduciamo il concetto di Working Set di un programma: è l'insieme delle Frame Reali che ospitano le pagine virtuali relative al programma. Non è detto che tutte le sue pagine virtuali siano residenti su Frame reali; anzi, la condizione normale è che il Working set sia inferiore alla dimensione virtuale del programma.
Per quanto detto precedentemente, poiché per eseguire un programma non è necessario che esso debba risiedere tutto in memoria reale, esiste un Working Set ottimale, inferiore alla dimensione del programma, che

consente di eseguire il programma stesso con una penalizzazione accettabile (causata da qualche Page-In necessario per risolvere i Page Fault). Un certo numero di Page-In durante l'esecuzione di un programma è da considerarsi fisiologico. E' perciò essenziale avere dei meccanismi di controllo che garantiscano ad ogni programma di avere il proprio Working Set ottimale. Una delle funzioni essenziali del Workload Manager visto prima è appunto di garantire questo stato per ogni processo presente nel sistema.
La corretta gestione della Memoria Centrale dovrà perciò garantire a tutti i processi attivi nel sistema un working set ottimale.
Nello z/OS questo viene ottenuto da due componenti:

- Real Storage Manager (RSM).
- System resources manager (SRM)

Questi due componenti cooperano per:

- Garantire che sia sempre disponibile un pool di Frame libere (Available Frame Queue - AFQ) per soddisfare i Page-In. Queste frame sono libere in quanto non sono associate a nessuna pagina.
- Determinare quali Frame possano essere liberate del loro contenuto virtuale per poter essere assegnate al pool AFQ. Questa attività è estremamente critica. Infatti liberare una frame implica che il working set di qualche programma diminuisce: bisogna adottare una strategia che mantenga AFQ ad un livello accettabile; lo z/OS monitorizza costantemente questo livello e quando questo scende sotto una certa soglia innesca il processo di ripristino del livello accettabile attraverso l'RSM – questa attività si chiama AFQ replenishment nella terminologia z/OS - senza penalizzare eccessivamente i working set dei processi attivi nel sistema. Un effetto collaterale del liberare una frame dal suo contenuto virtuale è che questo contenuto non deve essere perso; perciò esiste il processo di **Page-Out** che copia il contenuto della frame su un Page Data Set. Quando l'operazione di I/O collegata è conclusa, la frame sarà considerata libera e mossa nell'AFQ.

Ogni volta che il processo AFQ replenishment deve essere eseguito, il componente System Resource Manager (SRM) dello z/OS viene innescato. L'SRM interfaccia con RSM per portare a termine l'attività.

Le Frame da liberare sono scelte secondo questi criteri:

- Least Recently Used (LRU) - le Frame inattive da più tempo vengono liberate
- Parere del Workload Manager - per garantire i livelli di servizio si deve garantire alle applicazioni un Working Set adeguato
- Frame modificata o no - si tende ad evitare il Page Out.

L'attività che sottrae Frame al Working Set di un Address Space è il "**Page Stealing**". Il meccanismo LRU viene usato perché si assume che le Frame inattive da più tempo siano quelle che hanno minore probabilità di essere referenziate in futuro.
Per la corretta gestione del meccanismo LRU lo z/OS mantiene un contatore per ogni Frame che indica da quanto tempo la Frame non viene acceduta (Unreferenced Interval Count – UIC); per gestire questo contatore lo z/OS sfrutta un bit Hardware associato ad ogni Frame di 4K chiamato il Reference bit (definito dall'architettura). Questo bit viene messo ON ogni volta che la Frame viene referenziata. L'analisi dei Reference bit viene fatta dal System Resource Manager (SRM) secondo intervalli prefissati. Un intervallo tipico di quest'analisi è di qualche secondo.
Gli UIC (uno per Frame rale) si trovano in una struttura chiamata Page Frame Table (PFT) che contiene un'entrata (Page Frame Table Entry – PFTE) per ogni Frame Reale disponibile allo z/OS.
Poiché liberare delle Frame implica eseguire operazioni di I/O (per il Page-Out), è possibile evitare questa attività sapendo se la Frame è stata modificata. L'architettura supporta questa funzione assegnando ad ogni frame un bit chiamato Change Bit: questo bit diventa ON ogni volta che la Frame viene modificata. Se si deve liberare una Frame il cui Change bit è OFF la Frame è semplicemente mossa nell'AFQ senza Page-Out (non è stata modificata dall'ultimo Page-Out); altrimenti ne viene eseguito il Page-Out e, terminata l'operazione di I/O relativa, il corispondente Change bit viene messo OFF.

Il complesso di Page-In + Page-Out viene indicato come Paginazione. Con l'aumentare delle dimensioni delle Memorie Centrali e con il loro minore costo, l'importanza della Memoria Virtuale si è notevolmente ridimensionata, va detto però che il processo di Paginazione per il Sistema z/OS deve

essere ritenuto "fisiologico" in quanto facente parte della struttura stessa del Sistema e pertanto esso è sempre presente anche se in piccola parte.

Per riassumere diremo che i Page-In sono sincroni all'esecuzione del programma, quindi il tempo speso sarà parte integrante dell'Elapsed Time del programma coinvolto. Viceversa l'attività di Page Stealing e i possibili Page-Out collegati sono asincroni rispetto all'esecuzione dei programmi e l'unico effetto su di loro sarà la conseguenza della riduzione del Working Set; sarà cura del WLM guidare il page stealing in modo da riuscire a soddisfare sempre i goal stabiliti dall'installazione.

Un meccanismo analogo alla Paginazione, ma di portata diversa, prende il nome di "Swapping".
Si ha lo **swapping** quando tutte le pagine che costituiscono un Address Space vengono portate fuori dalla memoria centrale (Swap-Out). Si effettua lo Swap-Out di un Address Space quando si prevede che l'AS resterà inattivo per parecchio tempo (cioè almeno secondi): questa condizione viene indicata come Long wait.
La condizione di Long wait è tipica di un utente interattivo (terminale) che, fra un'interazione e la successiva non opera con il Server poiché "sta pensando" (questo intervallo viene spesso indicato con Think Time). I Think Time hanno dimensioni "umane" e sono tipicamente dell'ordine di decine di secondi. Perciò quando un'interazione è terminata (è terminata una transazione) l'Address Space relativo all'utente viene Swapped-Out in quanto il sistema immagina che l'utente non genererà transazioni per parecchi secondi.

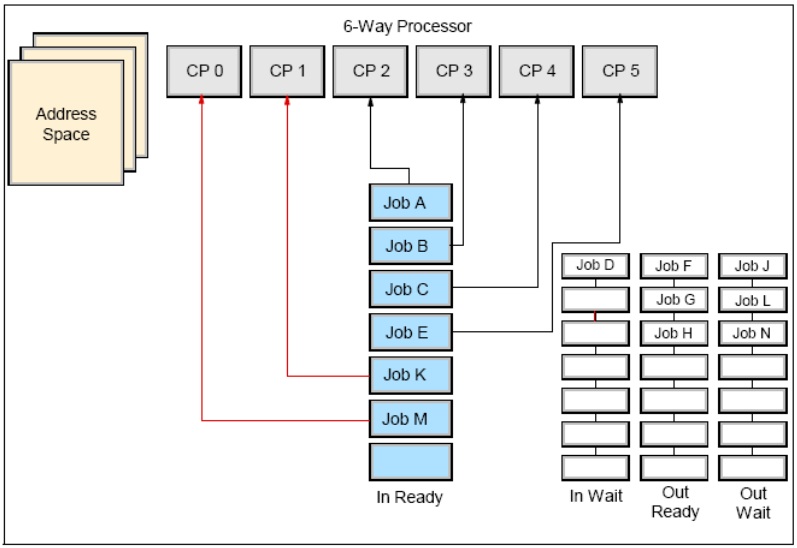


Figura 51 Address Spaces in un sistema multiprocessore

Oggi lo Swap-Out di un utente è prevalentemente solo logico. Il suo Working Set viene comunque passato tutto intero al RSM perché ne inserisca le Frame nell'AFQ (però solo se lo z/OS decide che si deve iniziare l'AFQ replenishment). Se lo Swap passa da Logico a Fisico allora il Working Set dell'Address Space verrà trasferito nell'AFQ.
L'unica conseguenza pratica di uno Swap-Out logico è che tutte le UoW attive nell'Address Space saranno rese non-Dispatchable (cioè non eleggibili per essere eseguite).
Quando la condizione di Long wait cesserà (cioè l'utente interattivo avrà inviato un input al Server) l'AS sarà Swapped-In. Nel caso lo Swap-Out precedente fosse stato fisico avremo un'attività di Swap-In fisico (una serie di operazioni di I/O dai Page Data Set in Memoria Reale). Altrimenti lo Swap-In solo Logico si limiterà a ripristinare le UoW dell'AS come Dispatchable (possono nuovamente essere mandate in esecuzione su un Processore).

1.3.3 Componenti dello z/OS

z/OS fornisce una serie di funzioni di servizio quali ad esempio:

- funzioni di sicurezza ed autenticazione, attraverso meccanismi specifici o standard "di fatto" come LDAP;
- programmi per le comunicazioni secondo gli standard più diffusi (ad esempio TCP/IP o SNA);
- programmi per gestire i dispositivi di I/O;
- programmi per il controllo del Sistema;
- programmi di Monitoring del Sistema;
- meccanismi automatici per la ripartenza;
- programmi di gestione del Cluster (Parallel Sysplex);
- programmi di Utilità e Servizio.

Il Sistema Operativo è integrato da una serie di prodotti quali:

- Compilatori per linguaggi Evoluti;
- Data Base Managers:
- Security Manager and Auditing;
- Web Servers;
- Java Application Servers;
- Java Virtual Machine;
- XML Processing functions;
- Message Queuing and Routing Functions;
- Transaction Managers;
- Vendor Product (ISV).

1.3.4 Interfacce Utente (User Interfaces)

Questo capitolo descrive le interfacce Utente verso il Sistema Operativo z/OS.

Un'interfaccia utente è lo strumento che consente di interagire col Sistema al fine di fornire comandi e verificarne il risultato. La richiesta di esecuzione di un programma può essere data attraverso un comando, come pure la richiesta di apertura o di cancellazione di un archivio.

Le interfacce utente alle quali siamo abituati nella pratica quotidiana sono ormai tutte di tipo grafico, tali interfacce sono presenti anche in z/OS, ma in effetti molto poco usate. Le interfacce presentate di seguito sono invece quelle più comunemente usate dai Programmatori di Sistema e dagli Operatori.

Tali interfacce non vanno confuse con le interfacce applicative, ovvero quelle attraverso cui gli utenti utilizzano le applicazioni eseguite sotto il controllo di z/OS. Ad esempio, la schermata attraverso la quale viene interrogato un conto bancario è un'interfaccia applicativa ed, in generale, essa non dipende dal Sistema Operativo del Server Centrale.

Occorre distinguere inanzitutto due tipi di dispositivo di comunicazione:

1. La Console di Sistema
2. Il Terminale video

La Console di Sistema (**System Console)** è costituita da uno o più dispositivi di I/O (generalmente video con tastiera) che rappresentano lo strumento di comunicazione primaria del Sistema. Attraverso la System Console, il Sistema fornisce già dall'avvio i messaggi informativi o di errore provenienti da tutti gli Address Spaces attivi e richiede le risposte ad eventuali situazioni nelle quali sia necessario assumere decisioni. Attraverso la System Console è possibile fornire direttamente comandi al Sistema Operativo o ai sottosistemi.

La console di Sistema è il primo dispositivo che viene attivato alla partenza del Sistema (IPL) e l'ultimo ad essere spento.

La System Console non è solo un dispositivo fisico, ma è anche una funzione logica. Per questa ragione essa può essere interrogata anche da un Terminale video sotto il controllo del TSO/E con appositi prodotti o può essere interfacciata da un programma automatico che funge da operatore.
In passato, in effetti, la Console era ordinariamente predisidiata da un addetto, l'operatore di console, il quale aveva il compito di leggere i messaggi e prendere le relative decisioni. Oggi l'operatore di console viene sovente sostituito da un Software che opportunamente programmato è in grado di assumere le decisioni di routine o di segnalare ad un responsabile eventuali anomalie non gestite. Uno dei motivi di questa evoluzione è che il flusso di messaggi è talmente elevato che un'operatore "umano" difficilmente riuscirebbe a reagire tempestivamente. Perciò questi Software provvedono a prendere le decisioni più "automatizzabili" lasciando all'operatore le decisioni di più lungo respiro (ad esempio il far partire una sequenza di Attività Batch).
In un sistema z/OS si possono avere contemporaneamente molte console di Sistema, si può operare in modo che certi messaggi di errore o di segnalazione (ad esempio quelli provenienti da un certo sottosistema) siano inviati solo a determinate console. I sistemi in Cluster (Parallel Sysplex) possono condividere lo stesso gruppo di console.

Le console accettano comandi su due linee ed hanno uno scorrimento verso l'alto dei messaggi. I messaggi e le risposte sono memorizzati su un file (detto Syslog) e pertanto possono essere rivisualizzati a pagine (funzione di re-display), anche con opportuni filtri.

Il **Terminale Video** ha una funzione decisamente diversa essendo soggetto a maggiori controlli, ma consente funzionalità molto più avanzate nella gestione del Sistema.

Attraverso il terminale video si può accedere con tre modalità diverse, ricordiamo ancora una volta che si sta qui discutendo di Operazioni di Amministrazione del Sistema e non di Applicazioni:

1. Modalità TSO (Time Sharing Option)
2. Modalità ISPF/PDF (Interactive System Programming Facility / Program Development Facility)
3. z/OS UNIX Shell

Per meglio comprendere la differenza tra le tre opzioni, è opportuno ricordare che la codifica usata nativamente dal Sistema Operativo z/OS è quella EBCDIC e che tale codifica prevede l'uso di terminali video denominati IBM 3270 (o semplicemente **3270**). Tali terminali prevedono un protocollo di trasmissione SDLC (VTAM) ed una modalità di gestione "a pagine". Una pagina è ordinariamente costituita da una schermata di 1920 caratteri. La caratteristica principale della modalità e del protocollo 3270 è la possibilità di muoversi con il "cursore" all'interno della "pagina" e poi inviarne tutto il contenuto in una sola volta mediante la pressione di un tasto detto "invio" (o ENTER), tale funzione è particolarmente utile quando si fa l'Editing di testi o di codice sorgente di programmi.
L'interazione tra il Terminale ed il Sistema Operativo avviene solo alla pressione del tasto ENTER.

Negli ultimi dieci anni la modalità 3270, che rimane quella nativa e quindi più usata, è stata affiancata in z/OS ad un'altra proveniente dagli ambienti UNIX (e quindi dalla codifica ASCII) denominata Telnet (o VT100). Tali terminali sono caratterizzati dalla codifica ASCII e sono spesso usati con modalità a linea o a carattere, essendo in questo caso l'interazione col Sistema Operativo molto più frequente. Le schermate sono caratterizzate da un riempimento dall'alto verso il basso con scorrimento verso l'alto.

Contemporaneamente a questa caratteristica è stata introdotta una modalità operativa denominata z/OS UNIX System Services (USS) che consente all'utente che si interfaccia col Sistema di comportarsi come se stesse colloquiando con una "Shell" UNIX. In questo caso le modalità di accesso, le organizzazioni degli archivi e le utilities utilizzate sono quelle definite dalle specifiche POSIX relative all'ambiente UNIX.

Nonostante la codifica intrinseca dei dati e delle istruzioni rimanga EBCDIC, l'interfaccia potrà utilizzare direttamente terminali con codifica ASCII (VT100 o Telnet).

Di seguito vengono descritte le tre interfacce:

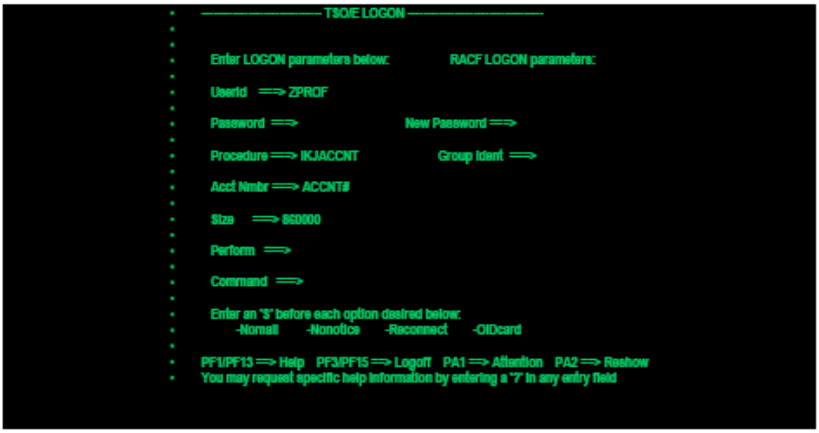


Figura 52Logon al TSO

Interfaccia TSO/E: rappresenta la più semplice ed antica interfaccia verso z/OS, quando un utente collega un terminale 3270 al Sistema Attivo, viene visualizzata una schermata detta di LOGON.

Una volta che l'utente è stato identificato mediante una password, gli viene proposta una "linea di comandi" attraverso la quale egli può dialogare col Sistema.

La presentazione di una linea comandi in effetti corrisponde alla creazione di un Address Space dedicato a quell'utente. Le caratteristiche di accesso e le modalità di creazione del relativo Address Space possono essere modificate all'installazione del Sistema.

La modalità TSO/E richiede una conoscenza approfondita dei comandi e per tale ragione è utilizzata solo da utenti molto esperti. Più frequente è l'uso

combinato delle capacità di creazione di un Address Space offerte dal TSO con l'ambiente ISPF/PDF che è invece di più facile utilizzo.
La figura seguente mostra l'utilizzo della linea comandi TSO/E per richiedere attraverso il comando "time" l'ora al Sistema e la relativa risposta.

```
READY
time
TIME-11:42:11 AM. CPU-00:00:00 SERVICE-788 SESSION-00:25:25 JULY 21,2003
READY
```

Figura 53 Linea Comandi TSO

Interfaccia ISPF/PDF: questa interfaccia si fonda ancora sul TSO/E ma è caratterizzata da videate con scelte multiple nelle quali è possibile l'uso del "mouse" e, a richiesta, possono avere una grafica avanzata.

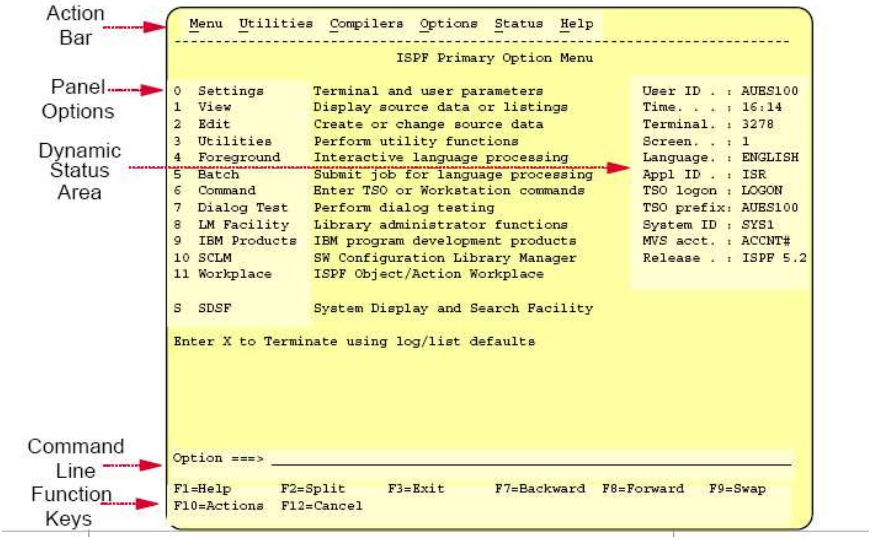


Figura 54 Interfaccia ISPF

L'ambiente PDF fornisce un potente EDITOR di testi e di programmi a pagine ed una interfaccia avanzata verso il TSO per poter dare comandi al sistema; tali comandi sono filtrati sulla base delle capability dell'utente controllate dal Security Manager del Sistema Operativo.

La Figura 55 mostra le caratteristiche di una schermata tipica di ISPF/PDF che rappresenta il Menu standard di un Sistema z/OS. Le schermate ISPF/PDF sono dotate di una funzione di aiuto in linea.

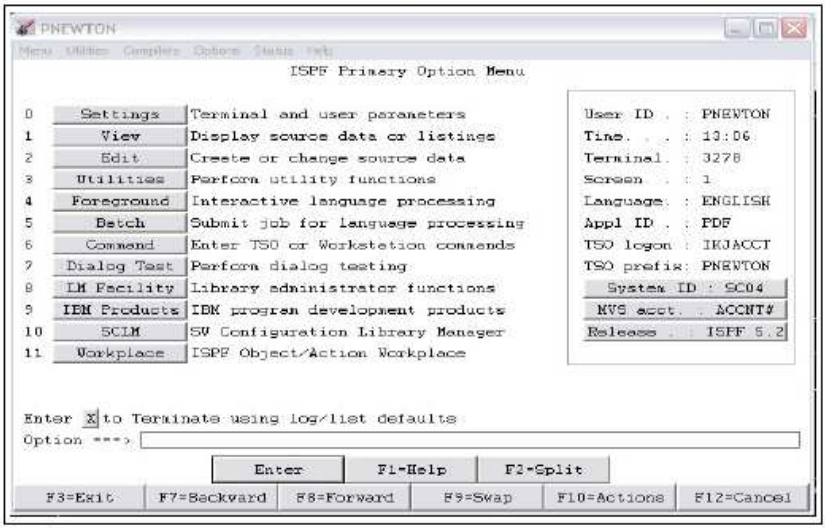


Figura 55 Interfaccia ISPF

Le schermate ISPF/PDF se visualizzate attraverso un Client con capacità grafiche possono essere visualizzate in formato grafico avanzato.

Usualmente l'interfaccia ISPF è utilizzata da tutti gli utenti del TSO/E. ISPF fornisce anche limitate funzioni applicative ed è programmabile attraverso un linguaggio procedurale denominato REXX: per tale ragione molte funzioni applicative di base, legate alla gestione ed amministrazione del

Sistema, vengono realizzate sotto il controllo di ISPF. Esempi sono la creazione degli utenti, l'amministrazione dei dati, la gestione dei nastri magnetici, la gestione dei lavori Batch, la gestione di stampanti e delle linee di comunicazione.

Di fatto ISPF rappresenta il "cuore" dell'amministrazione del Sistema ed il metodo di accesso principale a tutte le relative funzioni di gestione.

Interfaccia UNIX (UNIX Shell): l'interfaccia UNIX venne introdotta in z/OS a metà degli anni '90 (a quel tempo il Sistema Operativo si chiamava MVS/ESA). Nello stesso periodo il Sistema Operativo OS/390 Rel. 1.2 ottenne la certificazione UNIX95 secondo gli Standard internazionali.
Alla intefaccia UNIX si può accedere in due modalità diverse:

1. Nativamente, cioè attraverso Terminali ASCII su TCP/IP
2. In modo emulato attraverso TSO/ISPF/PDF da Terminali 3270 mediante i comandi:

- OMVS – che fornisce accesso in modalità UNIX emulata su 3270
- ISHELL – che fornisce una interfaccia a pagina

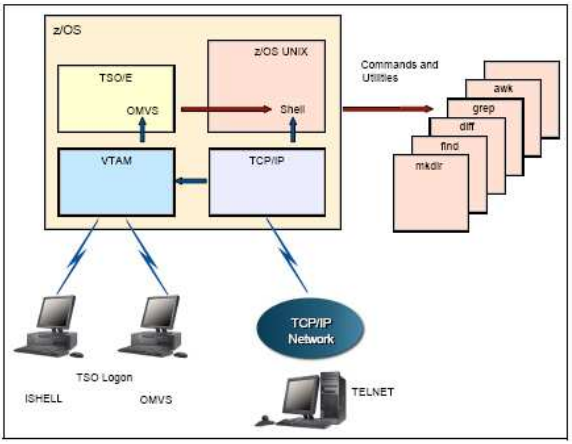


Figura 56UNIX System Services

```
# Improve shell performance
if [ -z "$STEPLIB" ] && tty -s;
then
    export STEPLIB=none
    exec sh -L
fi
# Set the time zone as appropriate.
TZ=EST5EDT
# Set a default command path, including current working dir (CDW)
PATH=/bin:.
# Sets the path environment variables
LIBPATH=/lib:/usr/lib:.
MANPATH=/usr/man/%L
NLSPATH=/usr/lib/nls/msg/%L/%N
# Sets the language
LANG=C
# Sets the name of the system mail file and enables mail notification.
MAIL=/usr/mail/$LOGNAME
# Export the values so the system will have access to them.
export TZ PATH LIBPATH MANPATH NLSPATH MAIL LANG
# Set the default file creation mask - umask
umask 022
# Set the LOGNAME variable readonly so it is not accidentally modified.
readonly LOGNAME
```

Figura 57 Esempio di script UNIX

La Figura 56 mostra i due diversi modi e le tre diverse caratteristiche dell'accesso alla Shell UNIX di z/OS. Va ricordato che le modalità USS sono sensibilmente differenti da quelle TSO/E; infatti l'utente della Shell UNIX non ha un Address Space dedicato come gli utenti TSO, a meno che non stia accedendo da TSO/ISPF, ma esegue i suoi lavori su un Address Space comune denominato OMVS, almeno fino a quando esegue comandi di Shell.

La Figura 57 mostra l'aspetto della Shell UNIX richiamata da un terminale ASCII di tipo VT100. La Shell UNIX può essere usata per richiamare script o per eseguire programmi (se scritti in maniera compatibile con l'ambiente UNIX). E' disponibile solo su terminali ASCII l'editor tradizionale degli ambienti UNIX denominato VI.

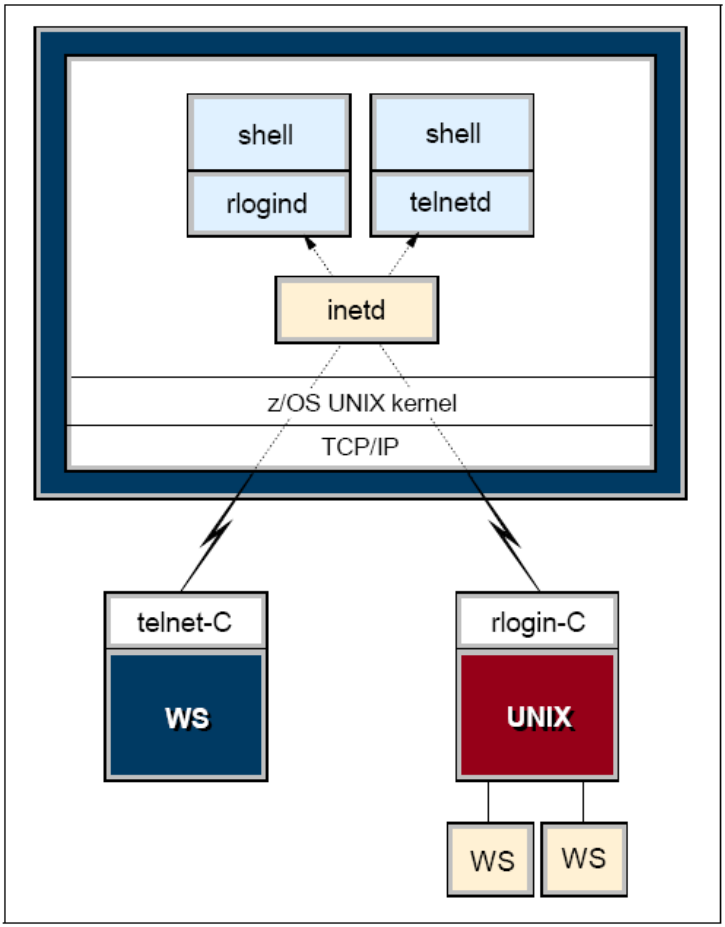


Figura 58 Modi per accedere alla shell UNIX

E' possibile connettersi all'ambiente USS di z/OS anche da altri Sistemi UNIX mediante la modalità di Remote Login o di Remote Telnet .

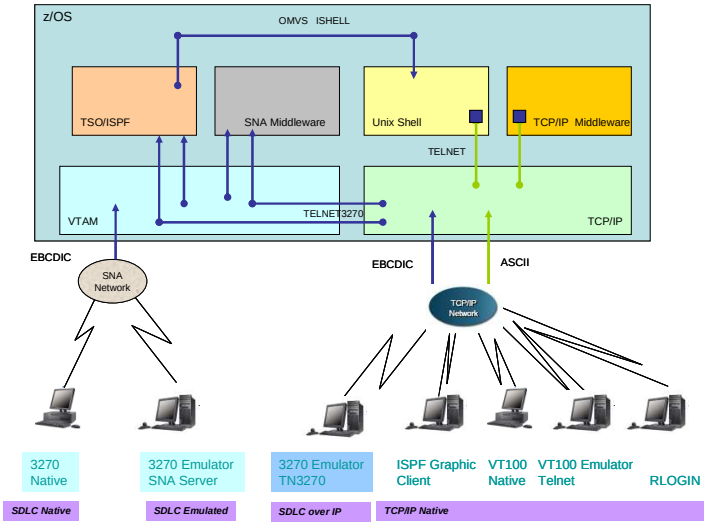


Figura 59 Le differenti interfacce verso z/OS

La Figura 59 sintetizza e riepiloga tutte le interfacce Utente disponibili per collegarsi ad un Sistema z/OS; anche se nel testo non è stato esplicitato, le interfacce sono funzionali anche all'ambiente applicativo (indicato come Middleware).

1.3.5 Il JES2 (Job Entry Subsystem)

L'esecuzione di lavori "batch" ovvero "a blocchi" rappresenta ancora oggi una delle attività più importanti del Sistema Operativo z/OS.

Un lavoro batch è costituito da uno o più programmi eseguiti secondo un ordine prefissato e, in generale, con una scarsa interazione utente. In linea di massima un lavoro "batch" non richiede la presenza o l'interazione di un Terminale Video: il lavoro viene usualmente "sottomesso" al sistema

attraverso diversi possibili meccanismi di input (al limite attraverso un Terminale Video) e al termine dell'esecuzione vengono controllati i risultati prodotti (con un Terminale Video o attraverso una Stampa di controllo).

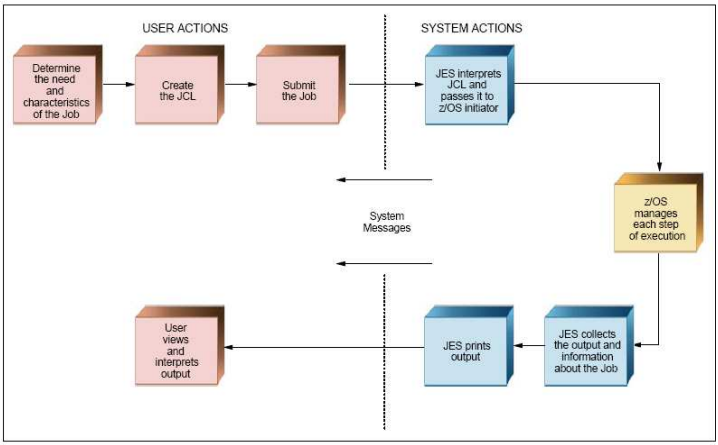


Figura 60 Elaborazione batch con JES

Per lungo tempo e nei primi anni dell'informatica, le eleborazioni "batch" hanno costituito l'unica modalità per eseguire lavori su un Calcolatore. Ancora oggi i lavori batch rimangono il modo di operare di molti processi applicativi: un lavoro batch è, ad esempio, la produzione dei cedolini dello stipendio dei dipendenti, la chiusura trimestrale dei conti correnti bancari o la stampa della lista dei passeggeri di un volo. E' chiaro che tutti questi lavori possono essere svolti anche con l'interazione diretta di un terminale e, in generale, il loro modo di operare dipende dalle scelte operate dal programmatore dell'applicazione. Nel testo si intendono lavori "batch" quei lavori nei quali l'interazione con l'uomo è minima, ovvero quei lavori che vengono "sottomessi" al Sistema e poi "controllati" al termine della loro esecuzione.

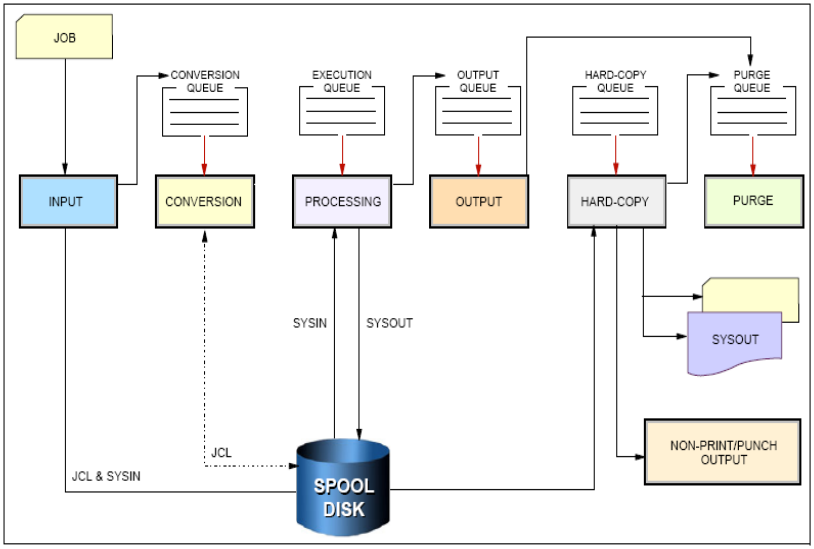


Figura 61 SYSIN e SYSOUT

La Figura 60 indica il flusso logico di un lavoro "batch" all'interno del Sistema Operativo z/OS. Vista la particolare importanza assunta storicamente dai lavori di tipo batch all'interno del Sistema Operativo, sono state specializzate delle componenti specifiche per la loro gestione: i Job Entry Subsystem (JES): la versione più usata di questi componenti si chiama Job Entry Subsystem 2 (JES2).
Il JES2 fornisce il supporto per la sottomissione, l'esecuzione e il controllo dei lavori di tipo batch. Esso consente, inoltre, la verifica ed il controllo dei risultati di tali lavori. JES2 fornisce anche un supporto generico per l'automazione dell'ambiente, ovvero per la creazione di appositi software in grado di automatizzare la gestione dei lavori batch; tali software prendono il nome di schedulatori (job scheduler).
Nel seguito indicheremo l'insieme dei programmi che costituiscono un lavoro batch col termine di "Batch Job" o semplicemente di "Job". Un job

può contenere oltre ai programmi da eseguire anche alcuni dati di controllo e le istruzioni da sottomettere al JES2 affinché possa operare: per tali istruzioni è stato predisposto un apposito linguaggio procedurale denominato Job Control Language (JCL) che descriveremo in maggiore dettaglio nel prossimo capitolo.

Il JES2 deve essere attivato obbligatoriamente all'avvio del Sistema Operativo z/OS in quanto fornisce anche funzioni di base per l'esecuzione degli altri lavori, anche se non ti tipo batch; esso fornisce, ad esempio, il supporto per l'attivazione di altri sottosistemi o Middleware che in generale non sono correlati ai lavori di tipo batch.

La Figura 61 descrive il flusso operativo e le componenti del JES2, oltre al già citato Job Control Language che costituisce la via primaria per fornire dati al JES2; il diagramma riporta alcune fasi essenziali e le relative componenti proprie del Sistema Operativo che vengono invocate per gestire il Job:

- **Input:** Rappresenta la fase iniziale di sottomissione del JOB a partire dalle istruzioni in JCL. La sottomissione può avvenire attraverso un Terminale Video connesso a TSO/ISPF, attraverso un programma (Schedulatore Automatico), da un altro Job in esecuzione o con una particolare modalità definita "procedura", descritta nel seguito. In passato la fase di INPUT poteva avvenire attraverso la lettura di un "pacchetto" di schede perforate in cartoncino, tale pacchetto veniva chiamato "file di schede": una struttura simile può essere creata oggi attraverso un archivio memorizzato su dischi magnetici con la stessa struttura (in particolare lunghezza di ogni "riga" o record di 80 caratteri). In generale un Job è sempre rappresentato o riconducibile ad un file su schede. Il componente usato è l'Input reader.
- **Conversion:** E' una fase molto importante del processo. Un apposito modulo del JES2 invoca una funzione chiamato Converter/Interpreter (C/I) il quale legge le "schede" JCL che descrivono il Job, le interpreta, le traduce in strutture dati che serviranno alla funzione seguente (Initiator) per lanciare l'esecuzione del Job, e accoda il Job nella Execution Queue; in questa fase il C/I verifica anche che nella stesura degli statement

non ci siano errori. Un Job Batch ha bisogno di un ambiente di esecuzione: ad esempio, se si vuole stampare la lista dei passeggeri di un volo occorre disporre di un archivio (File di dati) contenente i dati dei passeggeri, di un programma di stampa e di una stampante. I riferimenti all'ambiente necessario al programma batch, ad esempio, il nome del File di dati, il nome del programma, il nome della stampante sono definti nelle JCL e vengono controllati nella fase di conversione. Nel caso in cui uno degli elementi richiesti non venga trovato viene emesso un errore (JCL error) ed il processo batch viene interrotto.

- **Processing:** E' la vera fase di elaborazione del processo. I Job vengono messi in esecuzione a cura di un particolare processo di z/OS chiamato Initiator/Terminator (I/T). L'I/T manda in esecuzione i Job prendendoli dalla Execution Queue. E' importante notare che per il mondo Batch l'AS è creato per una istanza di Initiator/Terminator. L'I/T provvede a caricare nel proprio AS il job batch. Da questo momento il processo attivo nell'AS sarà il Job stesso. Finita l'esecuzione del Job, l'I/T tornerà a prendere il controllo del proprio AS e, se c'è un altro Job nella Execution Queue, provvederà a iniziare di nuovo il processo. Se non c'è nulla in attesa di esecuzione, l'I/T resterà in attesa nel proprio AS di nuovi Job da caricare dalla Execution Queue. I dati vengono elaborati e viene prodotto un risultato. Il risultato di un processo batch non è necessariamente una stampa, esso può essere un nuovo archivio, o una serie di modifiche ad un archivio esistente, il riordinamento di un archivio secondo una chiave di ordinamento (Sort) oppure una combinazioone di tutti questi.
- **Output:** La fase di Output pone in un'apposita coda tutti i file di stampa prodotti dall'esecuzione: ad esempio, se il processo prevede la produzione di una stampa essa viene "parcheggiata" su un archivio temporaneo su disco in attesa di successive decisioni, in tale fase può essere visualizzata, effettivamente inviata ad una stampante per una fase di hardcopy o cancellata mediante una fase di purge.
- **Hardcopy:** E' la fase che produce fisicamente la stampa, cioè invia il file di stampa sulla stampante fisica. Oggi tale fase è relativa quasi sempre a stampe, in passato essa serviva anche a creare un

altro tipo di Output di tipo "punch": si perforavano delle schede di cartone da riutilizzare per altre elaborazioni. Il componente di z/OS usato è il SYSOUT Writer.

- **Purge:** E' la fase di pulizia e cancellazione degli archivi già stampati o non più necessari al processo.

Tutte le elaborazioni del JES2 si basano su "**code"**, in particolare notiamo code di "input" che contengono i processi in attesa di esecuzione, code di "output" e code di "stampa" o di "punch". I processi interni del JES2 che fanno passare i Job da una fase alla successiva si chiamano JES2 Processor.

L'insieme delle code del JES, tutto il loro contenuto, le stampe e il JCL in attesa di elaborazione sono contenuti in uno spazio disco predefinito che prende il nome di "**File di Spool"** o semplicemente spool.

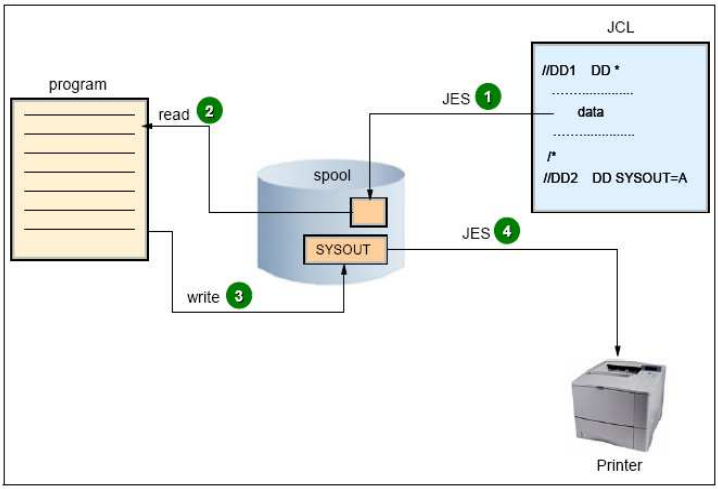


Figura 62 Funzionamento di un programma Batch

1.3.6 Job Control Language (JCL)

Il Job Control Language (JCL) è il linguaggio di controllo interpretato dal JES2; esso deve fornire al JES:

- le indicazioni circa l'ambiente di esecuzione di ogni lavoro batch (JOB)
- la sequenza di esecuzione delle fasi di uno stesso lavoro (Job Steps)
- le indicazioni di controllo per la esecuzione del JOB
- il nome dei programmi da eseguire e le caratteristiche di esecuzione.

Il JCL è un linguaggio procedurale basato su alcune semplici regole con una sintassi estremamente semplice. Le righe o statements sono sempre costituiti da "schede" di ottanta caratteri maiuscoli dell'alfabeto inglese più caratteri speciali e numeri; è possibile continuare una "scheda" su più righe, esiste la possibilità di indicare variabili simboliche che vengono istanziate al momento della conversione.

Ogni statement (frase) del JCL inizia col carattere "//" seguito da una label e da uno dei verbi:

- JOB - Definisce il nome del processo e le caratteristiche di esecuzione
- EXEC - Definisce il nome del programma da eseguire
- DD - Definisce l'ambiente di esecuzione, il nome degli archivi e le caratteristiche degli outputs da produrre. La label di uno statement DD (altrimenti nota come DDName) è decisa da chi ha scritto il programma indicato alla scheda EXEC. Questa label è di fatto l'aggancio fra il programma ed il particolare file che si vuole usare per questa esecuzione del programma. Una delle voci fondamentali della descrizione di un programma è quali label debbono essere usate per accedere a quale set di informazioni.

La figura seguente illustra questa relazione:

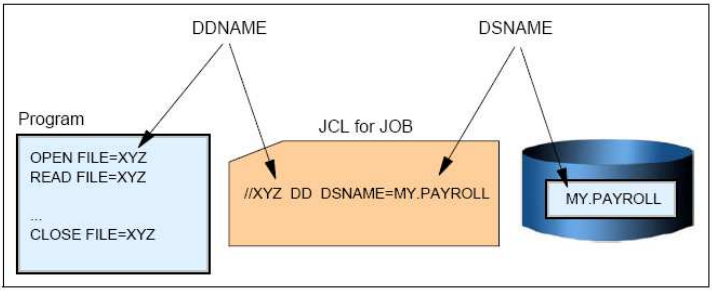


Figura 63 Nomi simbolici e reali in JCL

Ciò esprime l'importante caratteristica dei Job batch ed in generale di tutti i programmi in esecuzione su z/OS di referenziare al loro interno sempre nomi generici (simbolici) che non hanno riferimento con i nomi degli archivi sui dischi. Ciò consente di usare la stessa copia di programma con archivi diversi in diversi momenti usando diverse JCL.

Analizziamo come esempio il semplice JCL:

```
//MYJOB   JOB 1
//MYSTEP  EXEC PGM=SORT
//SORTIN  DD DISP=SHR,DSN=ZPROF.AREA.CODES
//SORTOUT DD DISP=SHR,DSN=ZPROF.AREA.NEWCODES
//SYSOUT  DD SYSOUT=*
//SYSIN   DD *
   SORT FIELDS=(1,3,CH,A)
/*
```

Il nome che il JES2 assegnerà al JOB è "MYJOB"; esso è costituito da un unico "passo" o step denominato MYSTEP, scopo dello step è di eseguire un programma di nome sort su un archivio di INPUT che il programma indica con SORTIN e che sui dischi magnetici ha il nome "ZPROF.AREA.CODES". Questo archivio di INPUT viene riordinato secondo le specifiche indicate nel file puntato dalla scheda DD di nome SYSIN. Per

semplicità queste specifiche sono incluse nel file JCL stesso; nell'esempio si richiede che il contenuto del File di Input venga riordinato in chiave ascendente considerando i campi nelle prime 3 posizioni. Il File riordinato viene ricopiato sull'archivio già esistente (DISP=SHR) che ha per nome: "ZPROF.AREA.NEWCODES". Tutti gli eventuali messaggi prodotti dal programma "SORT" vengono inviati al JES che li porrà su una coda di output (SYSOUT=*).

La sequenza di caratteri "/*" indica la fine degli statement che costituiscono il file SYSIN.

Esistono alcuni DDName riservati per usi specifici come i seguenti:

```
//JOBLIB
//STEPLIB
//JOBCAT
//STEPCAT
//SYSUDUMP
.....
```

Essi non sono in effetti nomi interni di nessun programma ma indicazioni particolari relative a "cataloghi" o "librerie" dove sono posti programmi e dati riservati a particolari funzioni diagnostiche (Dump).

Un particolare tipo di JCL è quello che costituisce una "procedura" detta PROC, una PROC è un particolare file JCL che viene memorizzato in maniera permanente in una area di disco denominata Libreria delle procedure (PROCLIB). La procedura può essere richiamata direttamente dalla console di sistema mediante il comando "Start *nome della procedura"*. Un Job eseguito con questa modalità prende il nome di "Started Task". Lo z/OS assegna un AS ad ogni Started Task.
Le procedure PROC vengono usate per eseguire lavori ripetitivi nei quali solo uno o più elementi sono variabili oppure sono usate per avviare funzioni di sistema.

Ad esempio analizziamo la PROC:

```
//MYPROC  PROC
//MYSORT  EXEC PGM=SORT
//SORTIN  DD DISP=SHR,DSN=&SORTDSN
//SORTOUT DD DISP=SHR,DSN=&SORTNEW
//SYSOUT  DD SYSOUT=*
//SYSIN   DD DISP=SHR,DSN=&PARM
//        PEND
```

Questa PROC esegue le stesse funzioni dell'esempio precedente salvo il fatto che essa può essere invocata in modo parametrico attraverso le variabili simboliche:

&SORTDSN che rappresenta il nome dell'archivio di input
&SORTNEW che rappresenta il nome dell'archivio di Output
&PARM che rappresenta un archivio che contiene le direttive di controllo del programma (in particolare la stringa `SORT FIELDS=(1,3,CH,A))`. Supposto che queste ultime siano contenute in un archivio di nome ZPROF.SORT.PARAMETERS, si ottiene lo stesso effetto dell'esempio precedente con il comando:

```
START MYPROC,SORTDSN=ZPROF.AREA.CODES,
PARM=ZPROF.SORT.PARAMETERS,
SORTNEW=ZPROF.AREA.NEWCODES
```

Spesso la sottomissione di un JCL al JES2 o l'esecuzione di una PROC possono dare lo stesso risultato, come nel nostro esempio, tuttavia ricordiamo che il comando "submit" da TSO/ISPF viene utilizzato prevalentemente per lavori tipicamente "batch" come stampe ed elaborazioni periodiche, manutenzione e copie, mentre il comando START e le PROC vengono usate per i sottosistemi, i programmi di servizio o i middleware che generalmente vengono avviati una sola volta e mai interrotti.

Ad esempio all'avvio del Sistema (IPL) i comandi:

```
START TSO
START VTAM,LIST=AA
START TCPIP
START RMF,PARM=00
```

servono ad avviare rispettivamente il TSO (Gestore di Terminali), il VTAM (gestore del protocollo SNA), il gestore del TCP/IP e uno strumento software di controllo del Sistema denominato Resource Monitor Facility. A ciascuno di essi corrisponde una PROC con i nomi TSO, VTAM, TCPIP, RMF ed alcune di esse hanno la possibilità di inserire parametri come la lista di partenza del VTAM e un file di parametri per RMF.

1.3.7 Spool Display & Search Facility (SDSF)

La componente del Sistema z/OS denominata Spool Display and Search Facility (SDSF) contiene l'interfaccia utente a panelli o grafica per interfacciarsi con le altre componenti di sistema e con il JES2: analogo interfacciamento può anche avvenire attraverso la System Console o un terminale TSO.

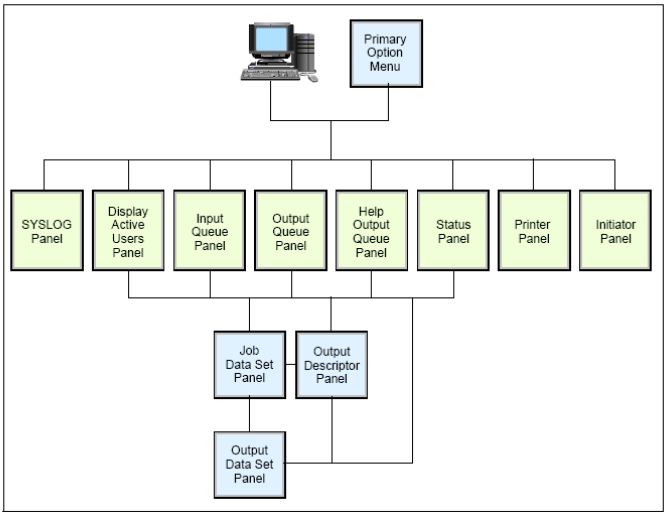


Figura 64 Scelte possibili di SDSF

SDSF è una applicazione eseguita sotto TSO/ISPF e pertanto richiede un terminale di tipo 3270, reale o emulato. Si tratta di una interfaccia a pannelli (Videate) concatenate che consentono di accedere a tutte le funzioni principali del Sistema e del JES2, fornendo anche supporto per il controllo delle autorizzazioni o delle abilitazioni alle quali l'utente è sottoposto.
La Figura 64 illustra le scelte possibili di SDSF.
Il look del pannello principale (schermata) di SDSF è riportato nella Figura 65 seguente.

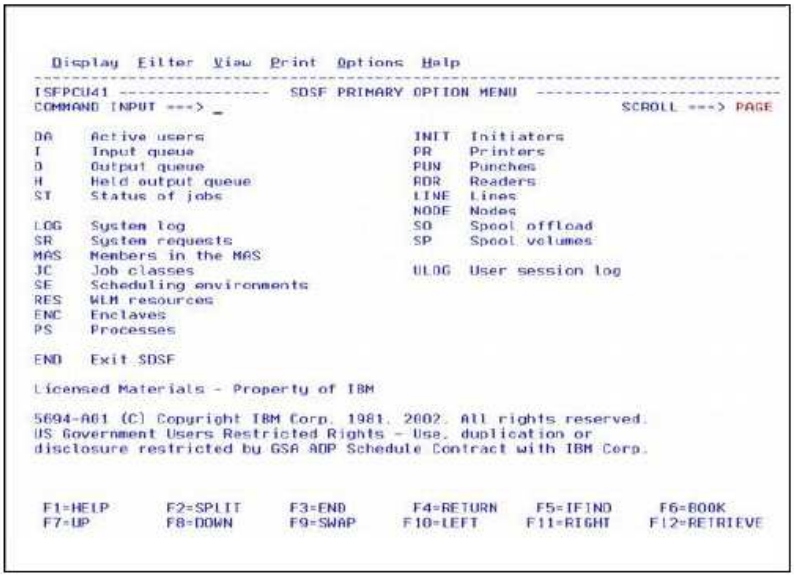


Figura 65 Pannello principale SDSF

Tra le funzioni principali di SDSF ricordiamo:

- L'accesso alla System Console (Syslog) in grado di fornire comandi al sistema come se si fosse fisicamente collegati alla Console. In questo caso i comandi debbono essere preceduti dal carattere "/"
- L'accesso allo SPOOL per visualizzare le code dei lavori sottomessi al JES2 ed in attesa di essere elaborati (Comando INPUT)
- L'accesso ai lavori già eseguiti ed alle stampe prodotte (comando OUTPUT)
- L'interrogazione di tutti i lavori attivi (Comando DA)
- L'interrogazione sullo stato di un lavoro (Comando ST)

Dalle schermate (pannelli di SDSF) è possibile fornire comandi di linea che consentono di filtrare i contenuti (Comando PREFIX), o di visualizzarli direttamente sul terminale video dopo averli selezionati (Comando SELECT).

Mediante SDSF è anche possibile visualizzare tutti gli address spaces attivi nel Sistema, anche quelli non iniziati dal JES2 e non relativi a lavori Batch o a started task; su di essi è possibile intervenire variandone alcune caratteristiche o cancellarli. Se un address space viene cancellato tutti i lavori in esso eseguiti vengono terminati immediatamente, ciò potrebbe arrecare danni significativi al Sistema e quindi la cancellazione di un address space è un'attività sottoposta a controlli. Alcuni address spaces non sono cancellabili.

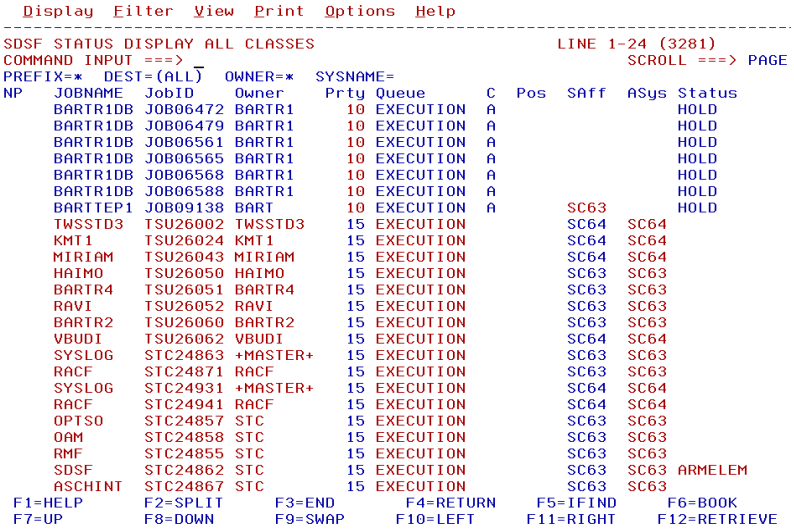


Figura 66 Schermata SDSF

La figura 66 mostra il risultato del comando di visualizzazione di tutti i lavori attivi (DA). Esso consente di visualizzare tutti gli address spaces presenti nel Sistema.

```
  Display  Filter  View  Print  Options  Help
--------------------------------------------------------------------------------
SDSF DA SC67  SC67     PAG    0 SIO     7 CPU   6/  7  LINE 1-25 (64)
COMMAND INPUT ===>                                          SCROLL ===> PAG
PREFIX=*  DEST=LOCAL  OWNER=*  SORT=JOBNAME/A
NP   JOBNAME  STEPNAME PROCSTEP JOBID    OWNER    C POS DP REAL PAGING    SIO
     *MASTER*                   STC06373 +MASTER+   NS  FF 1369   0.00   0.00
     ALLOCAS  ALLOCAS                               NS  FF  190   0.00   0.00
     ANTAS000 ANTAS000 IEFPROC                      NS  FE 1216   0.00   0.00
     ANTMAIN  ANTMAIN  IEFPROC                      NS  FF 4541   0.00   0.00
     APPC     APPC     APPC                         NS  FE 2653   0.00   0.00
     ASCH     ASCH     ASCH                         NS  FE  267   0.00   0.00
     BPXOINIT BPXOINIT BPXOINIT                     LO  FF  315   0.00   0.00
     CATALOG  CATALOG  IEFPROC                      NS  FF 1246   0.00   0.00
     CICSPAAY CICSPAAY CICS520  STC06504 STC        NS  FE 4330   0.00   0.00
     CONSOLE  CONSOLE                               NS  FF  597   0.00   0.00
     DFRMM    DFRMM    IEFPROC  STC06363 STC        NS  FE  510   0.00   0.00
     DFSMSHSM HSMSC67  DFSMSHSM STC13178 STC        NS  FE 6199   0.00   0.00
     DUMPSRV  DUMPSRV  DUMPSRV                      NS  FF  160   0.00   0.00
     FTPDMVS1 STEP1              STC06477 STC       LO  FF  470   0.00   0.00
     FTPDOE1  STEP1              STC06475 FTPDOE    LO  FF  469   0.00   0.00
     GRS      GRS                                   NS  FF  894   0.00   0.00
     IEFSCHAS IEFSCHAS                              NS  FF   25   0.00   0.00
     IMWEBSUF IMWEBSUF WEBSRV   STC15245 WEBSRV     IN  FE  15T   0.00   0.00
```

Figura 67 Schermata SDSF

Il Comando "Status", (ST) Figura 67, consente invece la visualizzazione dello stato dei lavori sottomessi compresi quelli già conclusi; questi ultimi sono visualizzati con un diverso colore. Selezionando una riga si possono vedere tutte le stampe prodotte da un lavoro con vari livelli di dettaglio.

Screen 1

```
 Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF HELD OUTPUT DISPLAY ALL CLASSES  LINES 44          LINE 1-1 (1)
COMMAND INPUT ===>                                        SCROLL ===> PAGE
PREFIX=*  DEST=(ALL)  OWNER=*  SYSNAME=
NP   JOBNAME  JobID    Owner    Prty C ODisp Dest                 Tot-Rec  Tot-
?_   MIRIAM2  JOB26044 MIRIAM    144 T HOLD  LOCAL                    44
```

Screen 2

```
 Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF JOB DATA SET DISPLAY - JOB MIRIAM2  (JOB26044)    LINE 1-3 (3)
COMMAND INPUT ===>                                        SCROLL ===> PAGE
PREFIX=*  DEST=(ALL)  OWNER=*  SYSNAME=
NP   DDNAME   StepName ProcStep DSID Owner    C Dest              Rec-Cnt Page
     JESMSGLG JES2                 2 MIRIAM   T LOCAL                20
     JESJCL   JES2                 3 MIRIAM   T LOCAL                12
     JESYSMSG JES2                 4 MIRIAM   T LOCAL                12
```

Figura 68 Altre schermate SDSF

1.3.8 Il Workload Manager (WLM)

IL Workload Manager (abbreviato in WLM) è il componente di z/OS responsabile della gestione delle risorse disponibili (CPU, Memoria, I/O) in funzione del raggiungimento di obiettivi prestazionali (Goal).

Il suo compito principale quindi è quello di ottimizzare l'utilizzo delle risorse disponibili in modo da consentire il raggiungimento di un obiettivo prefissato dall'utente per quel Sistema

Il WLM si pone tre obiettivi principali:

1. Raggiungere gli obiettivi che sono stati definiti dall'utente assegnando o disassegnando automaticamente le risorse di cui

dispone all'interno del singolo sistema o del Cluster di Sistemi (Parallel Sysplex).
2. Raggiungere un uso ottimale delle risorse disponibili.
3. Ottimizzare le risorse all'interno del singolo Address Space bilanciando i tempi di risposta con le operazioni di paginazione. (questa parte si esprime con i termini "Response Time and Turnaround Time balance").

Il primo di questi obiettivi (Goal Achievement) è di gran lunga il più importante ed è quello che il WLM tende a privilegiare soprattuttto in quanto gli altri due potrebero contrastare con esso: infatti il tentativo di minimizzare il TurnAround tra gli Address Spaces potrebbe contrastare con le necessità di fornire determinati tempi di risposta ad uno di essi rispetto agli altri.

Per poter bilanciare il Tempo di Risposta ed il tempo di Turnaround il WLM opera come segue:

- Controlla costantemente (Monitoring) l'uso delle risorse da parte di ogni Address Space.
- Controlla il Sistema nel suo complesso per vedere se tutte le risorse sono completamente utilizzate
- Determina quali Address Spaces deve essere swapped out e quando.
- Limita la creazione di nuovi address spaces o avvia processi di page stealing nel caso vi sia carenza di memoria
- Cambia le priorità di esecuzione degli Address Spaces (dispatching priority) in base a quante risorse ciascun address space sta consumando.
- Seleziona i dispositivi di I/O se è possibile farlo in modo da ottimizzarne l'utilizzo.

Per raggiungere i suoi scopi il WLM scambia informazioni con tutte le altre componenti del Sistema Operativo e dei sottosistemi , ad esempio viene "informato" quando:

- Una parte della memoria viene rimossa dal Sistema
- Un nuovo address space è stato creato

- Un Address Space è stato distrutto
- Una operazione di swap out è stata completata
- Un dispositivo di I/O è stato allocato

Tali operazioni sono eseguite sia nel caso in cui il Sistema Operativo z/OS operi singolarmente, sia nel caso in cui operi all'interno di un Cluster di Sistemi (Parallel Sysplex). In questo ultimo caso ad esempio il WLM può decidere su quale delle immagini del Cluster eseguire un certo lavoro (ad esempio Batch) in modo da bilanciare il carico complessivo e fornire i tempi di risposta richiesti.

Il WLM è un componente di Sistema obbligatorio e non può essere eliminato.

Per decidere le politiche di esecuzione dei lavori è necessario fornire al WLM alcune informazioni e dargli degli obiettivi. Questa modalità operativa si definisce "Goal Mode".

Prima della introduzione del WLM le prestazioni dei singoli Job (Programmi) potevano essere caratterizzate da un elenco di parametri tecnici : ad esempio i cicli di CPU da assegnare, la priorità di esecuzione, il tempo di permanenza in memoria ecc., tale definizione risultava in generale complessa e poco simile al modo di pensare comune.

Con l'introduzione del WLM è stata fornita una semplice interfaccia sotto TSO/ISPF mediante la quale l'utente del Sistema Operatvo (System Programmer) può definire le politiche di esecuzione usando concetti "intuitivi" come ad esempio il tempo di risposta, o la durata di un Job batch: sarà poi compito del WLM tradurre le richieste in parametri tecnici e controllare che essi siano rispettati.

In questo modo l'utente del sistema operativo potrà tradurre facilmente i requisiti operativi in parametri tecnici evitando lo spreco di risorse e controllandone il funzionamento in base a parametri intuitivi.

Sotto il controllo del WLM un Sistema z/OS risulterà sempre utilizzare al meglio le risorse disponibili e si accorgerà subito di quanto esse siano adeguate o inadeguate agli obiettivi prefissati e richiesti.

La funzione di controllo sull'ottenimento dei risultati è importante anche per stabilire se gli obiettivi prefissati sono consoni alle risorse disponibili, ovvero quale è la risorsa che necessita di esssere aumentata.

Nel futuro immediato il WLM potrà anche essere usato in congiunzione con la possibilità di allocare o disallocare potenza elaborativa da una parte all'altra di un Sistema Fisico per comunicare con le funzioni esterne al sistema Operativo e più vicine all'hardware richiedendo nuove CPU o cedendone alcune non usate (Capacity Upgrade on demand).

1.3.9 Gestione di dati e programmi su z/OS

In questo capitolo ci occuperemo sommariamente della gestione dei dati sul Sistema z/OS e dei principali dati che costituiscono la configurazione del Sistema stesso.

Cominciamo col dire che i dati ed i programmi gestiti dal Sistema Operativo z/OS risiedono di norma su dischi magnetici che prendono il nome di Direct Access Storage Device (DASD) o di nastri magnetici noti come TAPE Units.

I dati contenuti su dischi e nastri sono organizzati in "Volumi", mentre il dispositivo hardware che ne permette la lettura o la scrittura è il DASD o l'unità a nastro. Dal punto di vista della architettura trattata nel capitolo 1.1.2 , DASD e unità a nastro sono DEVICES e pertanto sono collegati a "Control Unit" e a "CANALI". Un device viene indicato con un numero esadecimale di quattro digit (Esempio FF00, 0110, 0540, ecc.). Di norma ad ogni device su disco corrisponde un solo "VOLUME" quindi ad esempio al device numero "0100" corrisponde il volume denominato "MVSRES". Il nome dei volumi è una label di 6 caratteri (max); questa label viene chiamata Volume Serial Number (VOLSER), ogni volume deve avere un

VOLSER. Un volume DASD è non smontabile dal suo supporto; un volume Nastro è invece per sua natura smontabile: perciò su un certo device Nastro si potranno avere montati volumi di volta in volta differenti. L'operazione di montaggio di un volume nastro prende il nome di MOUNT, l'operazione di scarico del nastro e suo spostamento prende il nome di UNMOUNT.

Una differenza fondamentale fra DASD e nastri è che i dati su DASD possono essere acceduti in modo casuale, mentre i dati su nastro debbono essere acceduti in modo sequenziale. Un'altra differenza fondamentale è che un volume DASD può essere allocato (assegnato in uso) a più processi contemporaneamente mentre un Device nastro può essere allocato ad un solo processo alla volta. Lo z/OS verifica autonomamente che questo requisito sia rispettato.
I VOLSER acceduti da una singola istanza di z/OS debbono essere univoci; ovvero un'istanza di z/OS non accetta di vedere due volumi con lo stesso VOLSER.
Su ogni volume, sia su disco che su nastro, possono essere contenuti "archivi" o files di natura e lunghezza differenti. I files sotto z/OS si chiamano "Dataset" ed il loro nome DSNAME. Il DSNAME ha un formato molto preciso:

- Non può avere più di 44 caratteri
- E' sempre maiuscolo
- Si possono usare caratteri Alfanumerici e alcuni caratteri speciali
- Il DSNAME è suddiviso in parti lunghe fino a 8 caratteri chiamate Qualifier o Qualificatori
- Queste parti sono separate dal punto "**.**"
- Particolare importanza riveste il Qualifier più a sinistra che viene chiamato High Level Qualifier (HLQ): esso è il fondamento su cui si basa la ricerca del DSNAME nel catalogo (vedi oltre). Ad esempio OPER.STIPENDI.MAGGIO è costituito da tre qualificatori. L'HLQ è OPER.

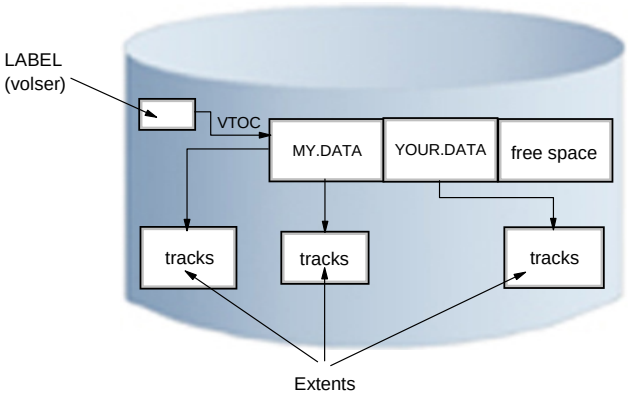


Figura 69 VTOC

Ogni volume DASD contiene un elenco dei DSNAME presenti in esso detto "Volume Table of Contents" comunemente abbreviato con l'acronimo VTOC; all'interno di un volume il DSNAME deve essere UNIVOCO in quanto esso rappresenta il modo per reperire i DATA SET all'interno del disco.

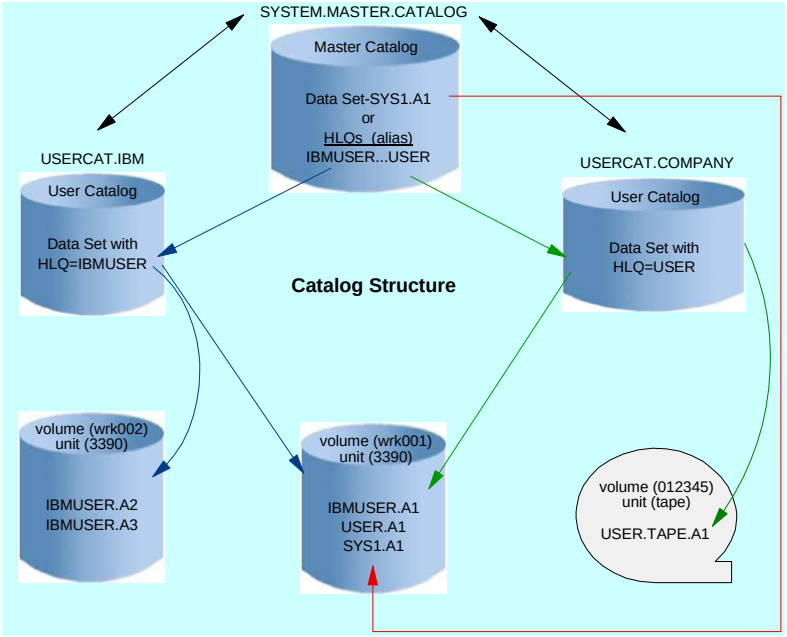


Figura 70 Struttura a cataloghi

Dal punto di vista fisico invece il "Volume" viene suddiviso in parti concentriche dette "Cilindri" costituiti a loro volta da tracce, le quali sono suddivise in settori circolari detti "Blocchi". Ogni blocco è diviso in porzioni contenenti le informazioni cui il Software accede (Data) ed altre informazioni di controllo per permettere il posizionamento delle testine di Lettura/Scrittura sul blocco stesso (Count e opzionalmente Key). Da questa divisione la tecnica di registrazione su disco propria del mainframe prende il nome: Count-Key-Data (CKD), e più recentemente Extended Count-Key-Data (ECKD).

Ogni Data Set esistente su un Volume DASD è rappresentato in VTOC da un'entrata chiamata Data Set Control Block (DSCB). Le informazioni contenute in un DSCB sono, fra le altre, il DSName e l'indicazione della posizione sul volume dei dati che compongono il Data Set. Lo spazio che un

data set occupa su un volume è costituito da un numero variabile di porzioni (fino a 123) di spazio contiguo chiamate Extent. Un Data Set può avere i suoi Extent residenti su più di un volume (se le dimensioni lo richiedono). Un extent è sempre contenuto in un solo Volume e la sua dimensione è variabile da un Data Set ad un altro.
Un'altra informazione fondamentale scritta nella VTOC è la posizione degli Extent liberi, cioè non occupati da alcun Data Set, che quindi sono disponibili per ospitare nuovi Data Set o nuovi Extent di Data Set già esistenti. Perciò la VTOC di un Volume è la struttura base su cui insiste il componente di z/OS Direct Access Device Space Management (DADSM): la funzione che gestisce lo spazio storage per conto delle applicazioni.

Una maniera alternativa, ma complementare e senza dubbio più funzionale di reperire le informazioni e la posizione di un Data Set si basa su un insieme di Data Set chiamati Cataloghi.

Questo sistema, (che chiameremo per brevità Catalogo) gestito da un apposito Address Space (Catalog Address Space), costituisce una struttura complessa in grado di reperire velocemente le informazioni sul dataset ed in particolare i suoi extent a partire dal nome, senza esaminare la VTOC di tutti i dischi collegati: infatti poiché i "VOLUMI" gestiti da z/OS sono relativamente "piccoli" in un Sistema produttivo se ne possono trovare migliaia e sarebbe estremamente oneroso esaminarli tutti per ricercare un archivio.

Nel Catalogo l'informazione fondamentale è il DSName e il/i volume/volumi su cui risiede. In questo modo la ricerca di un Data Set è:

1. Sul Catalogo
2. Sul volume su cui il Catalogo indica trovarsi il Data Set.

Il sistema di cataloghi si basa su una serie di Data Set (Cluster VSAM – vedi oltre) raggruppati su due livelli, come illustrato in Figura 70:

- Master Catalog: E' il punto di partenza per tutte le ricerche. Per ogni z/OS ci può essere un solo Master Catalog. L'indicazione del Master Catalog (DSName e volume su cui si trova) deve essere fornita al sistema in fase di IPL. Se la

ricerca del Data set non si conclude sul Master, si procede al secondo livello (se possibile)
- User Catalog: Rappresentano il secondo e ultimo livello. La ricerca non può proseguire a valle di uno User Catalog.

Un Catalogo può essere Master o User secondo cosa l'installazione decide. Però in un Master Catalog debbono essere presenti almeno queste informazioni:
- I Data Set di sistema
- I Connettori agli User catalog che costituiscono il secondo livello di ricerca
- I Data Set di paginazione

Sui cataloghi potranno essere inseriti anche nomi di Data Set su nastro. L'operazione di inserimento di un archivio in un catalogo prende il nome di catalogazione (Catalog), il suo disinserimento prende il nome di scatalogazione (Uncatalog). Eliminare un Data Set da un catalogo non vuole dire cancellarlo dal disco ma semplicemente renderlo "invisibile" dalla funzione di Ricerca tramite catalogo: si potranno avere quindi files su disco non catalogati (e pertanto accessibili solo attraverso la VTOC) oppure entrate di catalogo che non corrispondono a nessun Data Set. Di fatto se un Data Set è assente dal catalogo è perso visto l'elevatissimo numero di Volumi (migliaia e migliaia) oggi presenti in un sistema z/OS.

L'aggancio fra un Master Catalog e i suoi User catalog avviene mediante i Connettori. Facendo riferimento alla Figura 70 vediamo il funzionamento dei connettori.

Prendiamo ad esempio il Data Set IBMUSER.A1. La ricerca per questo data set seguirà questa trafila:
1. Si cerca nel Master Catalog (SYSTEM.MASTER.CATALOG) se esista un connettore ad uno User Catalog per l'High Level Qualifier IBMUSER
2. Poiché questo connettore esiste e punta allo User Catalog USERCAT.IBM la ricerca proseguirà all'interno di USERCAT.IBM
3. Trovata, in USERCAT.IBM, la descrizione di IBMUSER.A1 si andrà sul volume indicato (WRK001) per verificare se effettivamente il Data Set è lì. Se non c'è, viene riportata la condizione di Data Set

not found al processo che aveva chiesto di accedere ad IBMUSER.A1

4. Se il connettore IBMUSER non si fosse trovato, la ricerca per IBMUSER.A1 si sarebbe conclusa nel Master Catalog.

L'insieme delle funzioni di gestione e reperimento dati attraverso cataloghi prende il nome di Integrated Catalog Facility o ICF.
I Data Set presenti sui dischi gestiti da z/OS possono avere delle strutture differenti in base all'uso che di essi viene fatto. I dati all'interno di un Data Set sono acceduti dalle applicazioni non byte per byte ma aggregati in unità logiche chiamate Record. Tutte le funzioni z/OS che leggono e scrivono dati per conto dei programmi muovono dei record interi. Questi record sono scritti all'interno di un Data Set aggregati in blocchi di record. Il blocco è la quantità di byte che viene trasferita da un volume con una singola operazione di I/O. Non si trasferiscono quantità inferiori al blocco. La dimensione del blocco è variabile e ottimizzata dallo z/OS. Le possibili aggregazioni Record Blocchi possono essere (v. Figura 71):

- FIXED (F): tutti i Record hanno lunghezza eguale. Un blocco contiene un solo Record
- FIXED BLOCKED (FB): tutti i Record hanno lunghezza eguale. Un blocco contiene più di un record

  In questi due casi la lunghezza di Blocchi e Record è registrata nel DSCB relativo al Data Set in VTOC

- VARIABLE (V): Ogni Record può avere una lunghezza variabile. Un campo di 4 byte (Record Descriptor Word – RDW) all'inizio del Record indica la lunghezza del Record. Un Blocco contiene un solo record.
- VARIABLE BLOCKED (VB): Ogni Record può avere una lunghezza variabile. Un campo di 4 byte all'inizio del Record indica la lunghezza del Record. Un Blocco contiene più di un record. La lunghezza del blocco è descritta da un campo di 4 byte all'inizio del blocco (Block Descriptor Word – BDW)

- UNDEFINED (U): Non esiste alcuna delimitazione di record all'interno del blocco. La logica per estrarre le informazioni dal blocco fisico è scritta all'interno dell'applicazione che legge e scrive. E' il formato usato per scrivere i programmi eseguibili in una libreria di programmi.

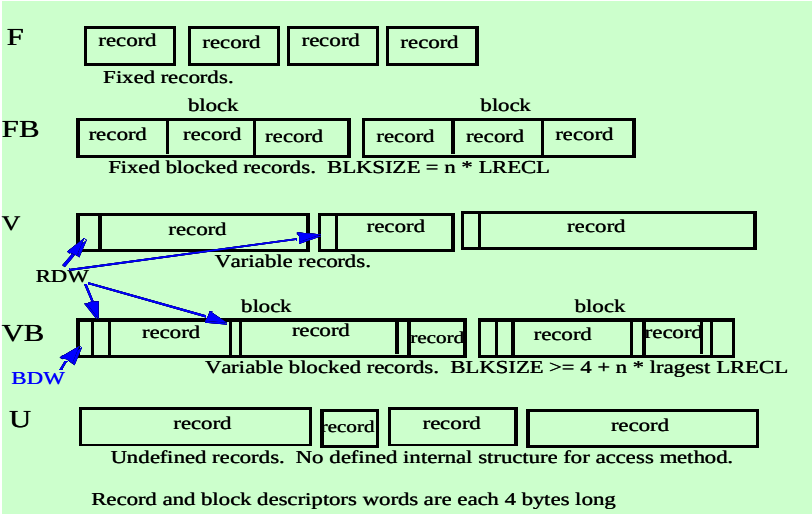


Figura 71 Record e Blocchi all'interno di un Data Set in z/OS

Tra le organizzazioni più comuni di Data Set ricordiamo:

- I **File Sequenziali:** Usualmente F, FB, V, VB, si tratta di semplici raccolte di dati con organizzazione a "RECORD", sono accessibili e visibili direttamente.
- Le **Librerie:** Dette Partitioned Data SET (PDS) o Partitioned data set Extended (PDSE) rappresentano dataset con un ulteriore livello di indice detto "Directory" contenente i nomi di tutti i componenti della libreria detti "membri": essi possono essere programmi eseguibili (LOAD) o programmi sorgenti (SOURCE) o sequenze di JCL (JOB) o procedure (PROC), o files di testo contenenti qualunque informazione o informazioni di controllo o input di altri programmi. I membri delle

librerie sono normalmente editabili cioè modificabili attraverso l'editor di ISPF/PDF (questo non si applica per le librerie che contengono programmi eseguibili).

- I file detti **VSAM:** Virtual Sequential Access Method (VSAM) è un metodo di accesso che consente l'organizzazione dei dati in forma sequenziale con l'introduzione di una o più "chiavi" di accesso dette "indici". Ad esempio il metodo di accesso VSAM consente di archiviare il Nome, l'Indirizzo ed il Numero di Telefono di una serie di persone potendo accedere all'archivio per "NOME", per Indirizzo o per Numero di Telefono senza avviare alcuna operazione di ordinamento dell'archivio stesso. Ogni file VSAM deve essere obbligatoriamente catalogato in un Catalogo. Esso è costituito da un insieme di componenti detto "cluster VSAM". Questi componenti possono essere:
    - Data Component (obbligatorio – almeno il data Component deve sempre essere presente) che contiene i dati veri e propri
    - Index Component (facoltativo – costituisce un indice per accedere in modo diretto alle informazioni contenute nel data Component).

Alcuni esempi di Cluster VSAM sono:

- I Data Set di Page. Si tratta di Cluster VSAM di cui esiste solo la parte dati. L'accesso diretto da parte dell'Auxiliary Storage Manager avviene mediante strutture che esistono in Memoria Virtuale.
- I Cataloghi. Sono Cluster VSAM composti dal Data Component (l'insieme delle informazioni sui Data Set) e dall'Index Component per consentire l'accesso diretto alle informazioni. Un accesso al catalogo richiede nella massima parte dei casi non più di due/tre operazioni di I/O, qualunque sia la dimensione del catalogo stesso.

z/OS UNIX File System (ZFS): Si tratta di un particolare dataset di tipo PDSE in grado di ospitare al suo interno una struttura di File System tipica del mondo UNIX. ZFS è stato introdotto in z/OS quando è stata attivata l'interfaccia di tipo UNIX Shell della quale ZFS rappresenta il "File System". Ciascun File System è contenuto in un Data Set e può essere "montato" o

"UnMounted" da una directory. Esistono utilities che permettono di copiare dati da un file system ad un altro o dall'ambiente UNIX a quello tradizionale e viceversa. Una struttura così organizzata prende il nome di HFS (Hierarchical File System).

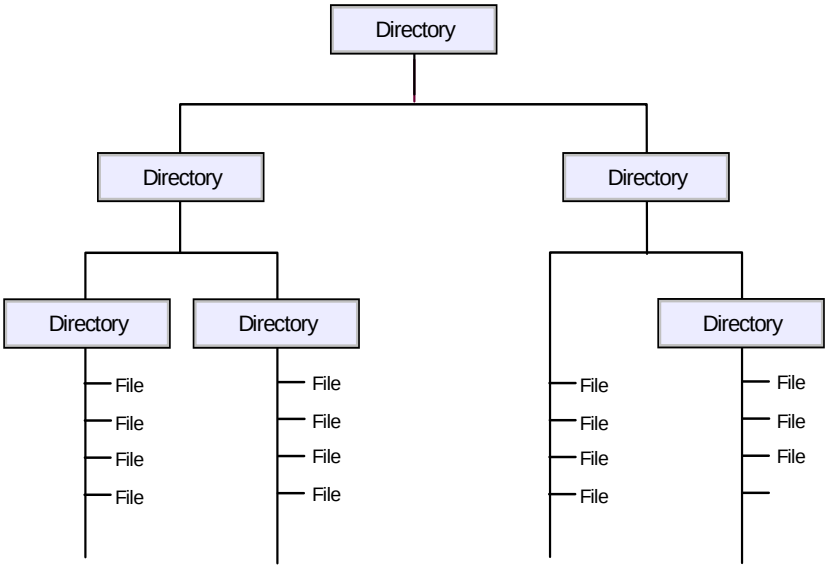


Figura 72 Struttura di un File System

Concludiamo la trattazione dell'argomento ricordando alcuni tipi "speciali" di files e librerie presenti nel sistema, in particolare:

- I Files di Configurazione, sono usualmente preceduti dal prefisso SYSx e contengono le informazioni sui devices (SYS1.IODF), sulla configurazione della rete (SYS1.VTAMLIST), o su tutto il Sistema (SYS1.PARMLIB), o le informazioni di avvio dello z/OS (SYS1.IPLPARM)
- Le Librerie di Sistema che contengono le parti principali del Sistema Operativo (SYS1.NUCLEUS) o le procedure principali relative agli

"Started task" di Sistema (SYS1.PROCLIB), oltre ai più importanti prodotti software e Middleware, i moduli caricati sempre in memoria all'avvio (SYS1.LPALIB) e alcuni moduli base per la gestione del Sistema (SYS1.LINKLIB).

Si possono copiare dati e membri dalle librerie o agire sui Data Set o sui Files VSAM usando alcuni programmi di Utilità. I più comuni tra questi sono:

- IEBGENER – Copia DataSet Sequenziali
- IEBCOPY – Copia Membri di Libreria
- IDCAMS – Gestisce tutto l'ambiente VSAM, con l'uso di parole chiave come sotto specificato.

```
DEFINE CLUSTER -
(NAME(VWX.MYDATA) -
VOLUMES(VSER02) -
RECORDS(1000 500)) -
 DATA -
(NAME(VWX.KSDATA) -
 KEYS(15 0) -
RECORDSIZE(250 250) -
BUFFERSPACE(25000) ) -
INDEX -
(NAME(VWX.KSINDEX) -
CATALOG (UCAT1)
```

Figura 73 Comandi VSAM

1.3.10 Gestori di basi dati (DBMS)

Un DBMS (Data Base Management System) è un sistema software progettato allo scopo di garantire la condivisione dei dati, in lettura ed aggiornamento, salvaguardandone l'integrità.

Un database è un insieme integrato di dai, controllato centralmente da un DBMS.

I vari DBMS presenti sul mercato differiscono per le caratteristiche del modello dati supportato.

I primi DBMS, sviluppati a partire dalla metà degli anni '60, supportavano il modello Gerarchico oppure il modello Reticolare. I DBMS oggi più diffusi sono, invece, quelli basati sul Modello Relazionale, sviluppato sulla base di un'idea originale di Ted Codd, un ricercatore IBM, che la pubblicò nel 1969. Dopo un decennio speso nella ricerca e nello sviluppo di prototipi, all'inizio degli anni '80 cominciarono a diffondersi i primi RDBMS (Relational DBMS).

Non tutti i DBMS precedenti, tuttavia, sono stati soppiantati dal nuovo arrivato: alcuni sono sopravvissuti. Nel contesto dell'ambiente z/OS, il DBMS Gerarchico continua ad essere utilizzato da molti clienti soprattutto a supporto di applicazioni che, seppur datate, svolgono ancora una valida funzione di business. Per questo motivo, nel seguito della trattazione, prenderemo in considerazione anche i DBMS Gerarchici oltre a quelli Relazionali.

I Data Base Management System (prodotti da IBM) utilizzati in ambiente z/OS sono:

- DB2 – Data Base Relazionale caratterizzato dal linguaggio di consultazione Structured Query Language (SQL)
- IMS/DB – Data Base Gerarchico caratterizzato dal linguaggio di consultazione Data Language One (DL/1)

Le pagine che seguono descrivono le caratteristiche principali del modello Relazionale e di quello Gerarchico, e dei DBMS appena menzionati.

I DBMS Relazionali

I DBMS Relazionali sono largamente utilizzati in aziende di ogni tipo, come banche, assicurazioni, aziende di trasporto, di telecomunicazioni, ecc. per elaborazioni di tipo transazionali (ambienti OnLine Transaction Processing – OLTP), dove più utenti ovvero più processi richiedono un accesso rapido e, soprattutto, integro a dati condivisi sia in lettura che aggiornamento.

Gli stessi RDBMS usate per l'OLTP sono spesso utilizzati anche per soluzioni di Data Warehousing e Business Intelligence, grazie alla loro capacità di trarre vantaggio, in modo trasparente per l'utente, dalla potenza di calcolo e il parallelismo delle macchine multi-processori e in cluster, indispensabile per rispondere in tempi più brevi o molto più brevi ad interrogazioni che richiedono l'elaborazione di grandi moli di dati.

Il Modello Relazionale è caratterizzato da

1. una struttura formata da tabelle (supportata da definizioni molto rigorose)
2. un insieme di regole atte a salvaguardare l'integrità dei dati a vari livelli (integrità di dominio, di tabella, di correlazione, ecc.)
3. un insieme di operatori per l'accesso e l'aggiornamento dei dati, basati sulla teoria degli insiemi.

CUST_NBR	CUST_NAME	CUST_REGION	CUST_CREDIT
239	TOTALLY WIRED	NE	A
376	BRIKS	NW	AAA
921	BALLOONS, ETC.	SW	AAA
631	GREAT WORKS	SW	A
227	GRATE DANES	SE	AA
338	TODAY'S WORLD	NW	AA

Figura 74 Esempio di Tabella di database relazionale

Uno dei punti cardini dell'idea di Ted Codd, che sta alla base del Modello Relazionale, è l'**Information Principle**[12] secondo il quale "The entire information content of a relational database is represented in one and only one way: namely, as attribute values within tuples within relations".

Diversi sono i vantaggi del Modello Relazionale dei dati rispetto ai Modelli precedenti. Tra i più significativi, che hanno contribuito maggiormente al suo successo, vale la pena di ricordare la facilità d'uso, la flessibilità al cambiamento e, soprattutto, il supporto della cosiddetta "data independence"; ovvero, la separazione molto più netta che nel passato tra il modello logico (la Vista Utente) e il modello fisico (il modo in cui i dati vengono memorizzati) che garantisce la possibilità di variare una serie di elementi di entrambi i modelli (come ad esempio, la possibilità di aggiungere nuove entità o nuovi attributi ad entità già esistenti, a supporto di nuove esigenze di business, o quella di aggiungere, modificare e togliere indici per migliorare, ad esempio, le prestazioni) senza incidere sulla funzionalità applicativa e quindi senza bisogno di modificare le applicazioni esistenti.

Un ruolo importante e, per molti versi, rivoluzionario rispetto ai sistemi pre-esistenti è stata l'elaborazione per insiemi (e non più una riga alla volta), caratteristica intrinseca del Modello Relazionale, fatta propria anche dallo Structured Query Language (SQL), il nuovo linguaggio che i sistemi Relazionali hanno messo a disposizione per la consultazione e l'aggiornamento dei dati. È proprio questa capacità intrinseca di elaborare per insiemi che ha reso possibile l'elaborazione parallela anche di una singola query, sfruttando in modo trasparente all'utente, tutti i processori disponibili per garantire un tempo di risposta ottimale.

Il linguaggio SQL, originariamente sviluppato nei laboratori IBM, è oggi uno standard di fatto e di diritto.

I DBMS Relazionali, originariamente nati per la gestione di dati strutturati, oggi sono usati anche per gestire tipi di dati semistrutturati (come, ad esempio, documenti) e non strutturati (come immagini, filmati, ecc.).

[12] Si veda, ad esempio, http://www.sigmod.org/codd-tribute.html

Recentemente, gli RDBMS sono stati estesi in modo da supportare anche il formato XML, diventato popolare soprattutto per lo scambio di informazioni in rete, per la maggiore flessibilità di gestione di dati di struttura molto variabile e per il già citato supporto di dati semi-strutturati.

Il Linguaggio SQL

Il linguaggio SQL, uno standard universale per i DBMS Relazionali, è composto di tre componenti, che servono ai seguenti scopi specifici:

- la definizione delle strutture dei dati (**Data Definition Language - DDL**)
- la "manipolazione" dei dati (**Data Manipulation Language - DML**), ovvero, il reperimento, l'inserimento, la cancellazione e l'aggiornamento dei dati.
- il controllo degli accessi ai dati tramite concessione e revoca di privilegi (**Data Control Language - DCL**).

Il linguaggio offre istruzioni specifiche per ciascuna delle tre categorie appena menzionate:

- SELECT, UPDATE, INSERT, DELETE per il DML
- CREATE, ALTER, DROP per il DDL
- GRANT, REVOKE per il DCL.

In particolare, le caratteristiche della componente DML permettono di individuare le informazioni da consultare e/o aggiornare senza la necessità di conoscere l'organizzazione fisica del database che le contiene. La scelta della strategia di accesso ai dati per ogni richiesta è a carico dell'Ottimizzatore, una componente peculiare del DBMS Relazionale.

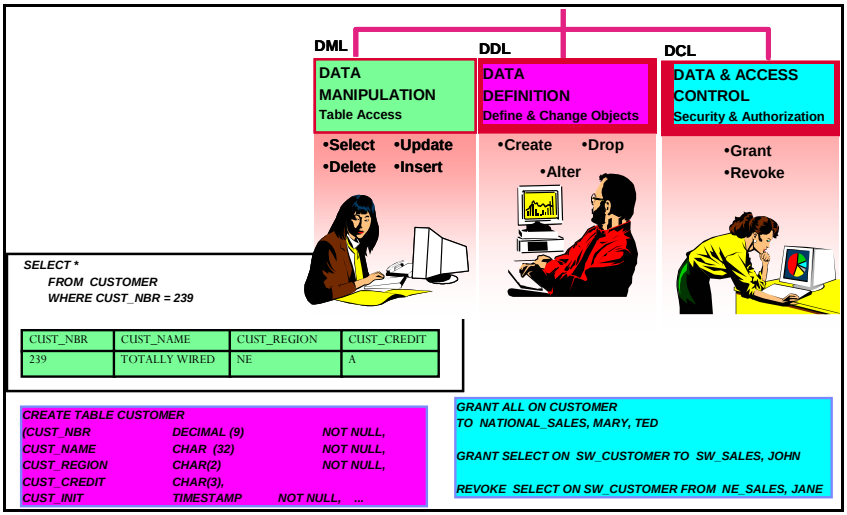


Figura 75 Le componenti del Linguaggio SQL

Le Viste, un altro concetto innovativo introdotto dal Modello relazionale e supportato dal linguaggio SQL, sono lo strumento che permette di raggiungere un livello di Data Independence ancora più elevato rispetto a quello fornito nativamente dall'uso delle Tabelle.

Le Viste possono essere considerate come una finestra logica sui dati: usando gli stessi operatori utilizzati per interrogare la base dei dati, le Viste permettono di selezionare un sottoinsieme di dati (righe / colonne) di una tabella o di più tabelle tra loro variamente e dinamicamente correlate.

Per l'utente, le Viste sono Tabelle a tutti gli effetti[13] grazie ad un'altra caratteristica del Modello relazionale, nota come "Relational Closure", per la quale il risultato dell'uso degli operatori del linguaggio è sempre e comunque una struttura relazionale valida. Questa caratteristica permette un uso *ricorsivo* del linguaggio, che ne aumenta in modo notevolissimo la potenza espressiva.

[13] A parte alcuni limiti al loro uso per aggiornare i dati. Limiti più o meno severi a seconda dell'implementazione

Nel tempo, il linguaggio SQL standard ISO/ANSI si è arricchito di molte nuove funzioni, tra le quali merita di essere menzionata l'estensione procedurale per lo sviluppo di Stored Procedure (SQL/PL), e quella per l'accesso a documenti in formato XML (SQL/XML).

Le Stored Procedure altro non sono che degli eseguibili, sviluppabili appunto in SQL/PL, ma anche in Java oppure in linguaggi tradizionali (Assembler, Cobol, C Language, ecc.). Esse possono essere richiamate tramite una estensione standard del linguaggio SQL da altri programmi, locali o remoti, eseguiti in qualunque degli ambienti (es. IMS, CICS, TSO, batch) e su qualunque piattaforma supportata.

L'uso appropriato delle Stored Procedure permette un più elevato riutilizzo di routine che svolgono funzioni utili condivise da più applicazioni, e un minor traffico di rete rispetto all'esecuzione remota di comandi SQL.

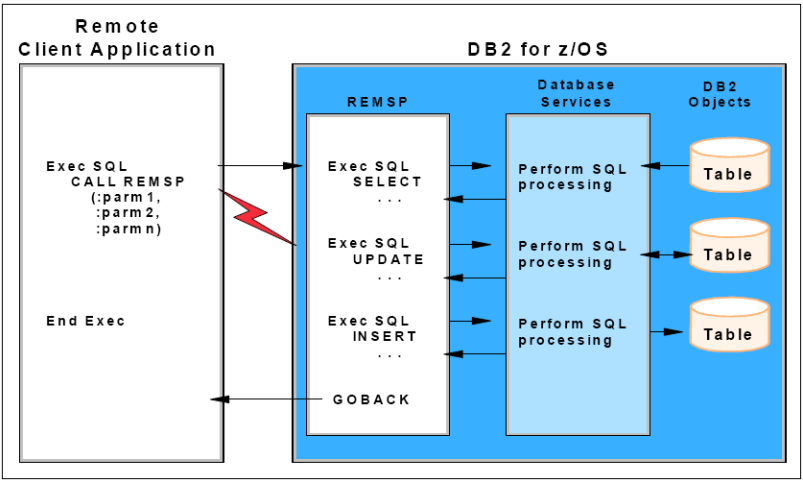


Figura 76: Stored Procedure e Traffico di Rete

IBM DB2 for z/OS

Il DB2 (IBM Database 2), la cui versione per la piattaforma mainframe fu annunciata nel Giugno 1983, è il più importante esponente di una famiglia che opera su tutte le più diffuse piattaforme informatiche.

Il prodotto contiene al suo interno tutte le funzionalità più evolute ed è corredato da una suite di strumenti che ne migliorano la gestione e il controllo.

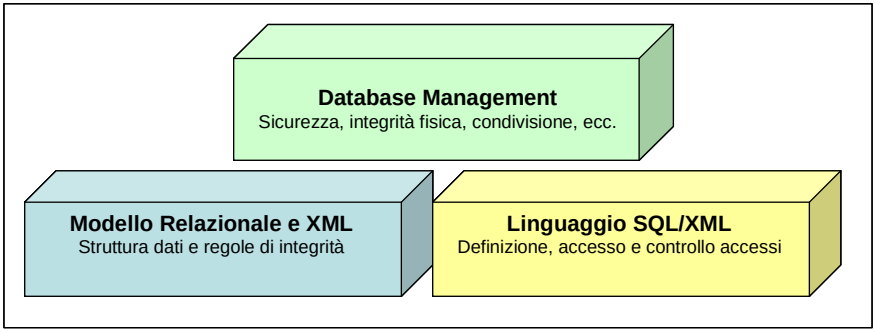


Figura 77: DB2 for z/OS: un RDBMS Ibrido (Relazionale / XML)

Il DB2 for z/OS è dotato di componenti specifiche (Attachement Facilities) per la connessione da parte di applicazioni che operano in ambiente quali CICS, IMS, TSO e batch.

L'integrità della transazione (e, di conseguenza, l'integrità dei dati), è garantita anche nel caso in cui la transazione effettui aggiornamento a più basi di dati locali o remote, grazie al supporto del **two-phase commit**.

La funzione Data Sharing, basata sull'architettura Parallel Sysplex, consente la condivisione integra di un'unica copia dei dati e del relativo Catalogo del metadati da parte di più sottosistemi DB2 che fanno parte di uno stesso Data Sharing Group e operano sulla stessa copia di sistema operativo o su sistemi z/OS distinti, garantendo elevata scalabilità e una continuità di servizio difficilmente eguagliabile su altre piattaforme.

Il DB2 supporta l'accesso a basi di dati distribuite secondo le specifiche della Distributed Relational Data Base Architecture (DRDA). Questa architettura definisce un insieme di protocolli per l'interoperabilità di tipo "client" / "server" tra gestori di dati e client di fornitori diversi. Il DB2 può svolgere sia il ruolo di server nei confronti di client DRDA locali o remoti che il ruolo di client verso altri server DRDA, che operano sia sulla piattaforma z/OS che su altre piattaforme (come AIX, Sun Solaris, HP-UX, Linux, z/Linux, Windows Server).

Il DB2 for z/OS è un sottosistema costituito da diversi componenti attivi in più regioni (Address Spaces):

- **System Services Address Space** (SSAS – procedura xxxxMSTR) che si fa carico delle attività di sistema, come il logging, il controllo delle prestazioni, la gestione dei comandi operativi, ecc.
- **Database Services Address Space** (DSAS – procedura xxxxDBM1) che si occupa più propriamente della gestione dei dati a cura delle funzioni RDS (Relational Data System), Data Manager (DM) e Buffer Manager (BM).
- **Distributed Data Facility Address Space** (DDF – procedura xxxxDIST), che integra sia le funzioni di Client che quelle di Server DRDA.
- **Internal Resource Lock Manager** (IRLM – procedura IRLMPROC) per la serializzazione degli accessi alle risorse condivise.
- **WLM-managed stored procedure address spaces**, utilizzati in numero variabile in funzione del carico di lavoro, per l'esecuzione delle "Stored Procedures"

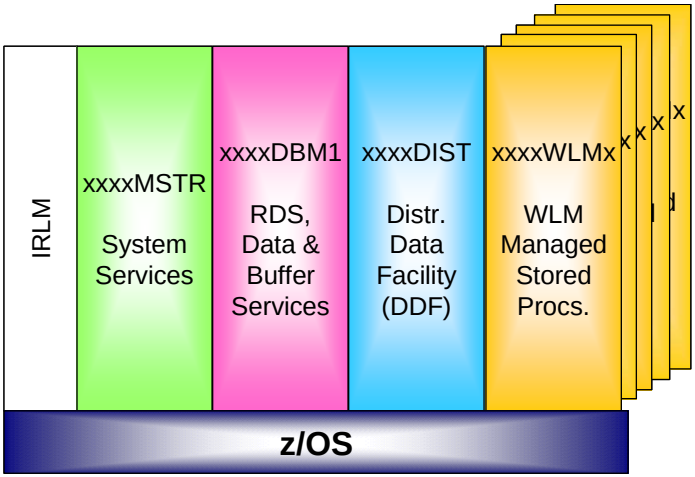


Figura 78: DB2 Address Space

Le tabelle, nelle quali sono memorizzati i dati, si appoggiano su uno o più spazi fisici di tipo VSAM Linear Data Set (LDS) denominati Tablespace.

Gli indici, strutture di tipo B-tree il cui scopo principale è quello di garantire un accesso più veloce ai dati di interesse, sono memorizzati in strutture fisiche separate, costituite sempre da VSAM Linear Dataset.

L'entità Database, invece, a differenza di altri ambienti relazionali, nell'ambiente DB2 for z/OS rappresenta semplicemente un insieme di Tabelle, Indici e Tablespace senza nessun ulteriore estensione a livello fisico (a parte alcune informazioni e blocchi di controllo nelle strutture interne e nel Catalogo dei metadati), con valenza operativa (ad esempio, è possibile attivare e disattivare un Database estendendo automaticamente l'effetto a tutti gli oggetti fisici associati).

Recentemente, il DB2 si è arricchito della capacità di gestire dati in formato XML, permettendone l'accesso tramite il supporto dell'estensione già citata del linguaggio SQL (SQL/XML).

Il DBMS Gerarchico

Il database gerarchico ha caratteristiche fondamentalmente diverse da quello relazionale. Infatti, si adatta a strutture logiche prevalentemente statiche e presuppone che l'utente conosca e navighi nella struttura per raggiungere il record desiderato.

In un database gerarchico le informazioni sono strutturate come in un albero a più livelli, ciascuno caratterizzato dalla presenza di entità dette segmenti. Il segmento al livello più alto della gerarchia prende il nome di radice (o root). Sia la radice che i segmenti dipendenti sono usualmente suddivisi in campi.

La struttura rappresenta intrinsecamente delle relazioni di tipo padre-figlio (ovvero, uno-a-molti, come ad esempio la relazione tra Ordine e Righe dell'Ordine e quella tra Reparti e Impiegati), nelle quali un padre può avere zero, uno o più figli, ma ogni figlio ha un solo padre.

Ogni albero è formato da un unico segmento radice (detto anche segmento padre o genitore) e da un insieme di segmenti (uno o più) dipendenti da esso, in modo ripetitivo (ogni segmento a sua volta può essere padre di altri segmenti, e così via).

Ciascuna struttura ad albero costituisce, quindi, un insieme organizzato di segmenti o rami strutturati del database.

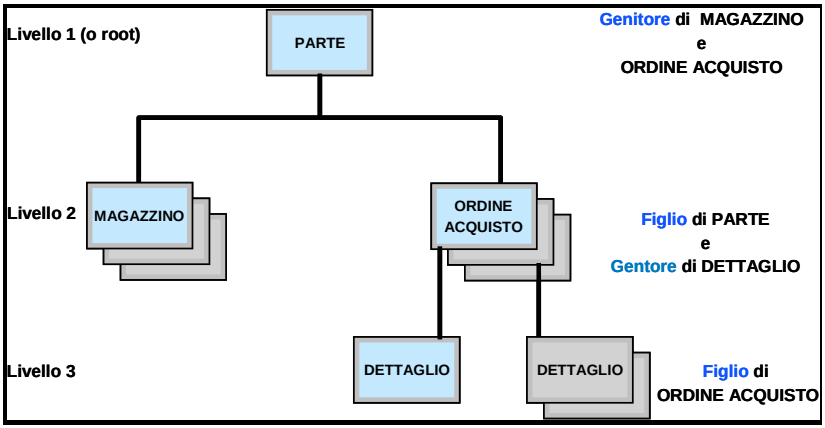


Figura 79: Esempio di database gerarchico

Il Modello Gerarchico si presta bene, ovviamente, alla modellazione di porzioni del mondo di interesse che si presentano, intrinsecamente, come un insieme gerarchico; ma presenta notevoli difficoltà quando l'universo di interesse presenta anche relazioni di tipo molti-a-molti (come ad esempio nel caso, ad esempio, nella relazione Clienti – Ordini – Righe dell'Ordine – Prodotti).

Information Management System/DB (IMS/DB)

L' Information Management System (IMS) è stato disegnato dall'IBM, insieme a Rockwell e Caterpillar, a partire dal 1966, a supporto del programma Apollo.

Nato come sistema esclusivamente batch, arricchitosi successivamente delle componenti OLTP, è ancora in uso in molte delle installazione basate su piattaforma z/OS.

Le prestazioni sono senz'altro uno dei valori specifici di questo sottosistema, che enfatizza ulteriormente questa caratteristica con la funzione "Fast Path", che utilizza strutture dati semplificate e database mantenuti costantemente in memoria per garantire prestazioni ancora più elevate.

L'accesso ai dati avviene tramite un linguaggio proprietario, il Data Language/One (DL/1), che consente di inserire, cancellare, modificare e interrogare le informazioni del database.

L'IMS supporta tre tipi di database a struttura gerarchica:

"Full Function" DB database con indici primari e/o secondari, supporto di relazioni logiche tra database diversi.

"Fast Path" DB - questa categoria di database non dispone di funzioni di indicizzazione. L'accesso avviene tramite uso di tecniche di randomizzazione della chiave di ricerca. Supporta database costantemente mantenuti in memoria.

High Availability Large Databases (HALDB) - sono una estensione della tipologia "Full Function" cui si aggiungono funzioni di alta affidabilità e migliore gestione di grandi volumi di dati.

1.3.11 Disegnare Applicazioni per z/OS

La realizzazione dell'applicazione a livello industriale comprende un insieme di problematiche legate alla complessità dello sviluppo e della gestione del progetto. Sappiamo infatti che una applicazione è costituita da uno o più programmi in grado di soddisfare una specifica richiesta o di risolvere uno specifico problema. Tale soluzione può richiedere risorse di vario tipo allocate su sistemi e piattaforme diverse.

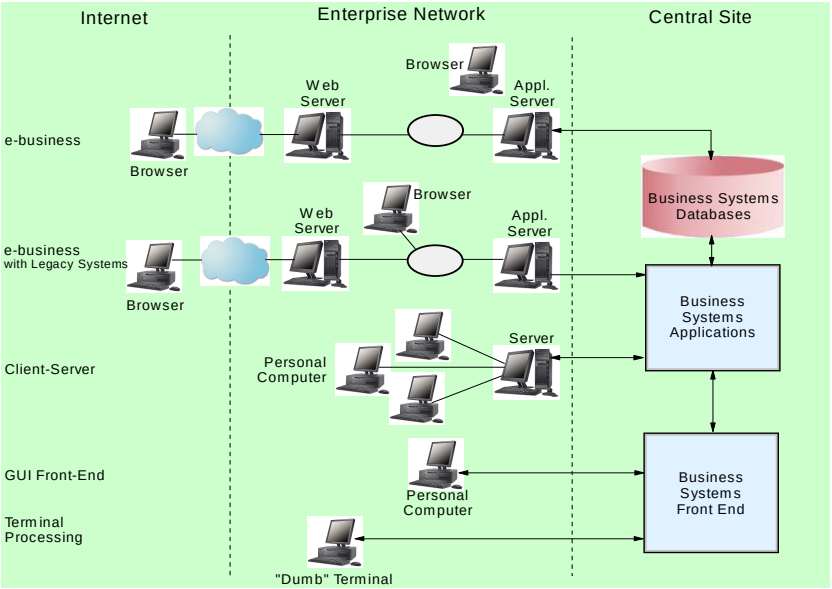


Figura 80 Infrastruttura delle applicazioni

La realizzazione di un progetto informatico industriale di sviluppo applicativo è quindi spesso una operazione che presenta un elevato grado di complessità sia dal punto di vista tecnico che da quello organizzativo. Considerando solo questo ultimo aspetto possiamo individuare due figure professionali:

- **Application Designer -** il cui compito è quello di operare scelte al fine di raggiungere la soluzione ottimale, ovvero, a partire dai requisiti dell'applicazione e da una visione globale del sistema, selezionare la piattaforma e le risorse hardware e software necessarie alla realizzazione.

- **Application Programmer -** il cui ambito di competenza è quello di costruire, provare e rilasciare l'applicazione (o parte di essa) in base alle specifiche del disegno applicativo, utilizzando tutti i tools a disposizione.

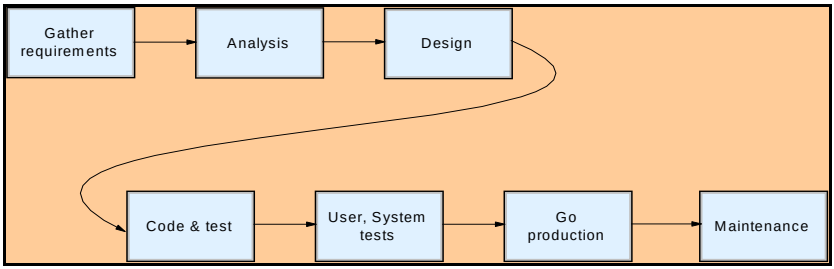


Figura 81 Fasi dello sviluppo di un applicazione

Sempre dal punto di vista organizzativo la programmazione applicativa è costituita da una serie di iterazioni in cui l'applicazione viene suddivisa in componenti omogenee. Quindi a partire dai requisiti richiesti si percorrono le fasi di disegno ad alto e basso livello, scrittura del codice e compilazione delle componenti e loro test. Quindi le componenti vengono integrate e il sistema viene testato in maniera globale. Seguono le fasi di rilascio in produzione.

Per ciò che concerne invece i requisiti di una applicazione, essi in estrema sintesi saranno di tipo Funzionale e Non Funzionale.

I **requisiti funzionali** provengono da una esigenza ben definita di business come ad esempio la gestione di un conto corrente di una banca. Essi descrivono un problema o una funzione ben definita relativa alle funzioni che l'applicazione deve compiere.

I **requisiti non funzionali,** egualmente importanti, sono quelli che individuano le caratteristiche di ambiente in cui l'applicazione deve poi operare. Tra queste identifichiamo: affidabilità, interoperabilità, performance, usabilità, portabilità, connettività, ecc.

Le scelte che in genere L'Application Designer è chiamato a fare sono legate all'ambiente infrastrutturale e alle modalità di esecuzione dell'applicazione. Esempi tipici di alternative da valutare sono legate al tipo di processo (batch o online), tipo di infrastruttura dati (database, tape, flat

file, ecc.), tipo di compilatore (COBOL, PL/I, Java, Assembler, ecc.), sistema operativo (z/OS, UNIX, Linux, Windows), potenza e caratteristiche dei server, utilizzo anche parziale di soluzioni a pacchetto, ecc.

I tool di programmazione sono molteplici e legati alle diverse necessità: esistono editor (ad esempio TSO/ISPF), vari gestori del codice sorgente che tra l'altro consentono di mantenere versioni diverse dello stesso codice (PDS, SCLM ed altri), strumenti di debugging e strumenti di monitoring software per controllare l'esito delle attività nelle varie fasi dello sviluppo.

1.3.12 Linguaggi di Programmazione

Il Sistema Operativo z/OS è in grado di supportare diversi linguaggi di programmazione.

- ***First and Second generation*** come:
  - Linguaggio **Assembler** – non usato o raramente usato per lo sviluppo di applicazioni; comprende una traduzione abbastanza fedele delle istruzioni macchina specifiche della z/Architecture. Esso viene usato per:
    - accedere direttamente a bit o byte;
    - accedere ai system control blocks;
    - avere tempi di risposta molto ridotti o efficienza molto elevati;
    - realizzare pezzi di codice riusabili richiamati all'interno di programmi scritti in linguaggi ad alto livello.
- ***Third generation*** (High Level Languages) come:

- **COBOL** - utilizzato per applicazioni business-oriented; è stato integrato nel sistema z/OS con la capacità di interagire con codice Java, gestire documenti XLM e formati Unicode;
- **PL/I** ("Programming Language/One) - è un linguaggio disegnato per applicazioni di business o scientifiche. Esso è uno dei più ricchi di funzioni. Supporta la recursività e la programmazione strutturata. Una sua variante chiamata **PL/S**, non commercializzata, viene usata per la scrittura del sistema operativo z/OS;
- **C language -** usato per applicazioni Scientifiche o Gestionali;
- **Fortran -** utilizzato prevalentemente per Applicazioni Tecnico-Scientifiche.

- ***Fourth Generation*** (4GL) di natura non procedurale - sono linguaggi che realizzano il paradigma dell'Object Oriented programming; come:

  - **C++** derivato dal linguaggio C;

  - **Java** - caratterizzato da un'alta portabilità tra differenti piattaforme informatiche. Tale portabilità è realizzata attraverso uno strato interpretativo di Software denominato Java Virtual Machine (JVM) il quale realizza il disaccoppiamento tra l'architettura ed il linguaggio. Il Sistema Operativo z/OS è dotato di una JVM integrata ed appositamente ottimizzata. Sotto z/OS Java può essere eseguito direttamente o attraverso appositi Middleware come ad esempio Websphere AS (vedi seguito).

Particolare importanza ai fini dell'Automazione del Sistema assumono particolari **Linguaggi Procedurali** di Servizio - si tratta di linguaggi orientati a raggruppare comandi di sistema ed altre funzioni in procedure; tra questi troviamo ad esempio:

- **CLIST** Command List - usato come linguaggio di procedure di comandi per la programmazione delle routine di accesso al TSO/E e per alcune applicazioni ISPF come la creazione di Menu per l'utente o il controllo del flusso applicativo dello stesso;
- **REXX** - è un linguaggio che può essere compilato o interpretato, con maggiori funzioni rispetto al linguaggio CLIST; anch'esso è usato per la programmazione di routine di accesso del TSO/E e per alcune applicazioni ISPF.

Ricordiamo infine un componente del Sistema z/OS denominato **Language Environment**: si tratta della realizzazione di un ambiente di esecuzione comune a tutti i linguaggi di programmazione; esso in particolare combina tutti i servizi essenziali per l'esecuzione dei programmi e la gestione degli errori oltre la gestione della memoria. Crea interfacce comuni consentendo tra l'altro il richiamo annidato di altri programmi scritti in linguaggi diversi da quello del chiamante.

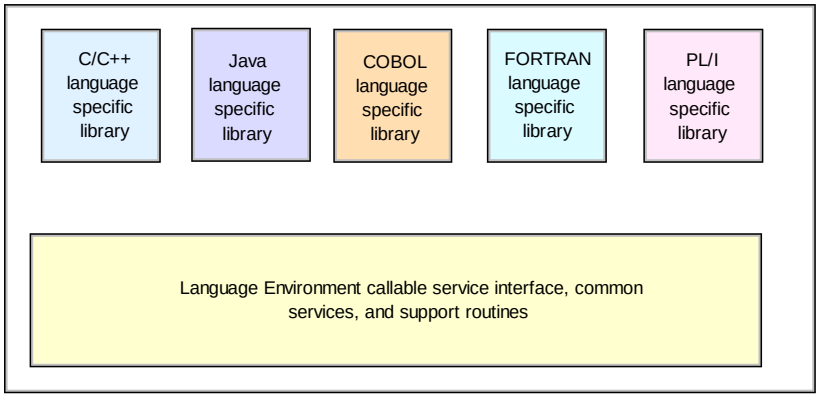


Figura 82 Language Environment

Come accade in generale i processi di compilazione in un sistema z/OS di un codice sorgente scritto in uno dei linguaggi appena descritti vengono suddivisi in moduli specifici.

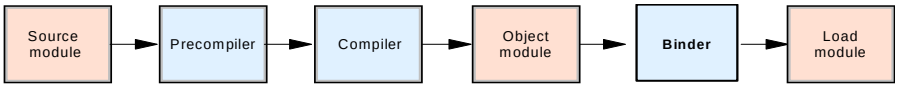


Figura 83 Flusso di compilazione e linkage editor

Ogni modulo viene assemblato usando un assemblatore o compilatore da cui otteniamo un "Source Module" che diventa a sua volta input di un Language translator che attraverso la precompilazione e compilazione genera un "Object Module" che viene infine mandato al BINDER per generare il "Load Module".

1.3.13 Sottosistemi Transazionali

I sottosistemi transazionali sono particolari programmi che permettono l'esecuzione simultanea di **transazioni** da parte di **molti utenti collegati** al sistema attraverso terminali detti ***workstations***. In effetti non è strettamente richiesto che una workstation sia un terminale video con tastiera, esso potrebbe essere anche un processo automatico, un altro programma eseguito su una differente piattaforma informatica o un dispositivo speciale (ad esempio un ATM, un lettore di tessera magnetica, un rivelatore di RFID, una macchina contapezzi, un telefono cellulare ,ecc.).

Una transazione è un'unità di lavoro indivisibile che comprende più operazioni.

Ad esempio una transazione è l'insieme di operazioni che vengono svolte nell'atto di prelevare una somma di denaro contante da un dispositivo "cash dispenser" (ATM), esse sono ad esempio la verifica della disponibilità sul conto corrente, la verifica della disponibilità delle banconote nell'ATM, la verifica del PIN della carta magnetica, l'erogazione del denaro, l'addebito del conto, la registrazione della operazione contabile sulla contabilità dell'agenzia, l'aggiornamento della carta magnetica, ecc.

Caratteristica fondamentale della transazione è l'*Atomicità*: o tutte le operazioni che la transazione prevede vanno a buon fine, o nessuna operazione deve essere eseguita.
Le proprietà della transazione sono:

Atomicità, ovvero l'indivisibilità dell'operazione - o è totalmente completata o non è effettuata.
Consistenza, ovvero la transizione da uno stato all'altro indotto dalla transazione deve permettere il ritorno alla situazione precedente nel caso di recovery.
Isolamento, ovvero ogni transazione non deve interessare altre transazioni concorrenti.
Durevolezza, ovvero una volta che la transazione è completata con successo i cambiamenti di stato effettuati sono duraturi e persistenti.

Il Sottosistema Transazionale soprassiede alla sequenza di eventi che fanno parte di una transazione e si assicura che le caratteristiche della transazione siano mantenute (ACID).
Esso include le funzioni di interfacciamento ai database e il coordinamento del ***commit/rollback*** delle transazioni: cioè è in grado di stabilire quando la transazione è stata completata definitavemente o di "ritornare indietro" nel caso ciò non sia avvenuto per un qualsiasi motivo.

Esso fornisce inoltre una interfaccia comune per i programmi applicativi in modo che le applicazioni possano usare al meglio i servizi di transaction monitor indipendentemente dal linguaggio in cui sono state scritte.

Ciascun Sottosistema Transazionale può supportare un certo numero di linguaggi di programmazione e l'interfaccia con alcuni Gestori di Dati.

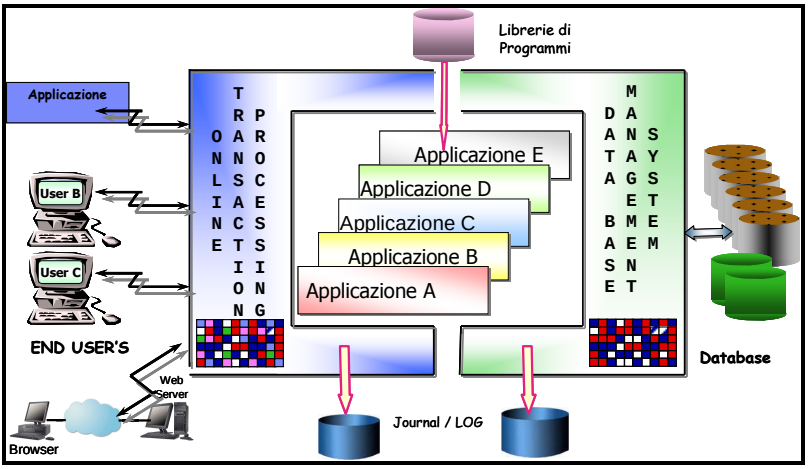


Figura 84 Sottosistemi Transazionali

I principali Sottosistemi Transazionali in z/OS sono:

- CICS/TM - Customer Information Control System/Transaction Manager
- IMS/TM - Information Management System/Transaction Manager
- WebSphere Application Server

CICS/TM Customer Information Control System / Transaction Manager

CICS è un sottosistema di gestione dell'elaborazione delle transazioni online (Online Transaction Processing - OLTP) che fornisce funzioni di middleware per la gestione del sistema e l'accesso ai database.

CICS è utilizzabile da programmi scritti in Cobol, PL/1,C, Java e supporta dati VSAM, Data Base gerarchici (DL/I) e relazionali (DB/2).

Il CICS è un software che si posiziona tra sistema operativo e applicazione. Il programmatore scrive un'applicazione utilizzando le API fornite dal CICS

che realizzano le interazioni necessarie con il sistema operativo. Ad esempio, le schermate relative a transazioni in ambito CICS sono inviate e ricevute tramite un emulatore 3270 in forma di mappe. Quando un programma prevede un input o un output di una transazione esso utilizza le mappe richiamandole tramite chiamate al CICS (Command level EXEC).

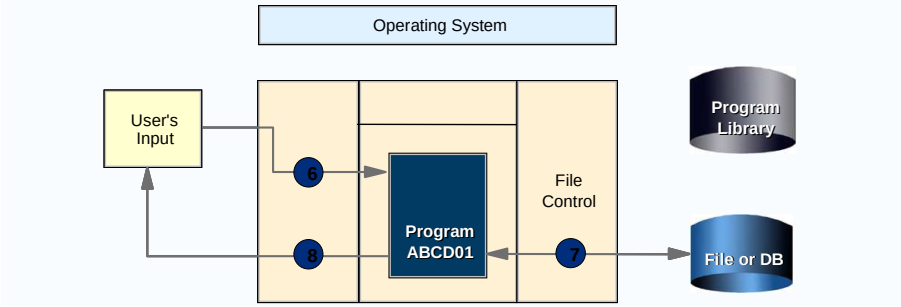


Figura 85 CICS

Una copia di CICS in uno z/OS può gestire le transazioni di molti utenti e molti programmi applicativi.

Inoltre più copie di CICS in vari serventi possono comunicare tra loro e condividere dati.

Il CICS supporta il Parallel Sysplex utilizzando il datasharing in sinergia con DB2, IMS/DB e i VSAM (Virtual Storage Acces Method) data set.

Sotto la stessa istanza di Sistema Operativo z/OS possono essere attivate molte istanze di CICS (dette anche regions), ciascuna istanza crea un apposito Address Space.

Grazie a questa caratteristica, all'interno di un sistema z/OS spesso viene utilizzata una configurazione con regioni CICS multiple. Queste regioni

gestiscono uno stesso ambiente transazionale ma sono specializzate secondo le tre funzioni che di solito sono gestite da un singolo CICS ovvero:

- Presentation (TOR) - gestione delle comunicazioni e dei terminali
- Application (AOR) - gestione delle transazioni e dei programmi
- Data Management (DOR) - gestione dei dati

In più il CICS fornisce il framework per lo sviluppo, messa in produzione e gestione operatività per semplificare la creazione di grandi applicazioni transazionali.

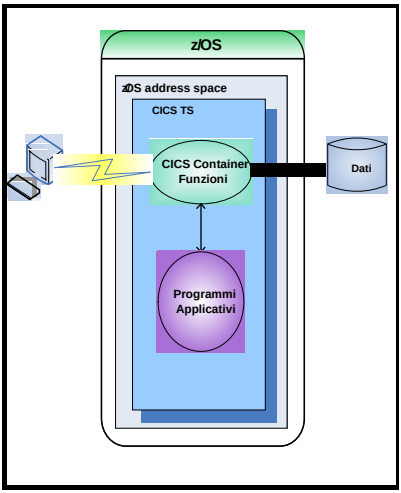


Figura 86 CICS/TM

IMS/TM Information Management / Transaction Manager

L' IMS rappresenta un sistema completo disegnato per alte prestazioni che, oltre ai servizi di base comuni, racchiude al suo interno due componenti fondamentali: una componente di gestione dei dati denominata **IMS/DB**, già analizzata nei paragrafi precedenti e una componente di gestione delle transazioni chiamata **IMS/TM,** di cui andiamo ora ad analizzare alcuni aspetti.
L'IMS/TM è un sottosistema che fornisce l'elaborazione di programmi applicativi che accedono a grandi volumi di dati su Database IMS (DL/I), DB2 o code di messaggi (Message Queue).
Le principali funzioni sono la gestione dei programmi applicativi, il dispatching dei lavori, il caricamento dei programmi applicativi, la gestione del coordinamento e la serializzazione delle risorse interessate. Esso fornisce anche la gestione dei messaggi provenienti dalla rete (3270s, APPC, TCP/IP, WebSphere MQ, ecc.) o da programmi applicativi "batch oriented" non connessi in rete.

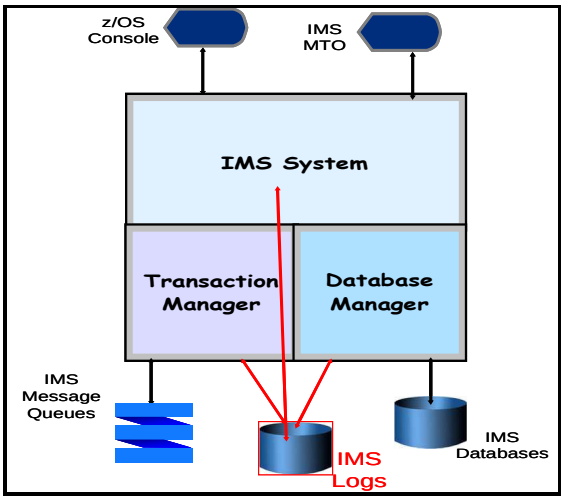


Figura 87IMS

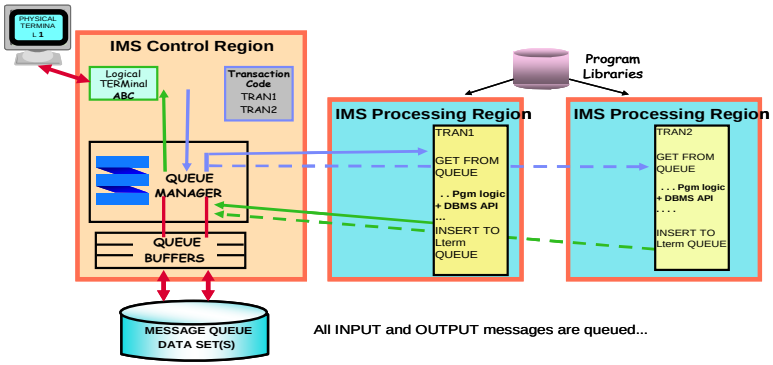


Figura 88 IMS Regions

WAS WebSphere Application Server

Il processo di modernizzazione che sta investendo le imprese muove molte applicazioni verso il Web, cosicché nelle attuali realtà ove sono presenti i Sistemi Centrali troviamo lo sviluppo di nuove applicazioni (o la revisione di alcune di quelle vecchie) con tecnologie legate al web e al contempo i workload tradizionali come ad esempio il lavori batch. In tale ottica la base dei nuovi sottosistemi transazionali è il **WebSphere Application Server** (WAS).

Il WAS è un gestore di applicazioni aderenti allo standard Java 2 Enterprise Edition (J2EE).
Esso fornisce inoltre un ambiente di implentazione e runtime costruito su tecnologia standard "open" che supporta tutte le principali funzioni come Servlets, Java Server Pages (JSP) e Enterprise Java Beans (EJB) includendo inoltre le più recenti tecnologie di integrazione di servizi e interfacce.

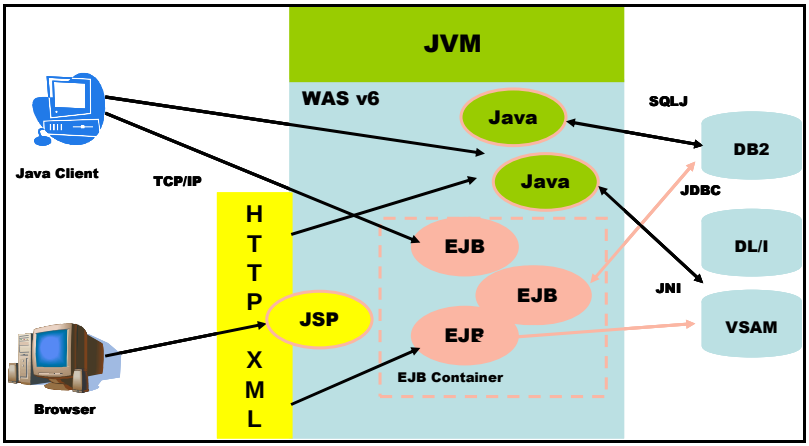


Figura 89 WAS in z/OS

L'ambiente runtime di WAS è altamente integrato con tutte le funzioni e i servizi di z/OS. Infatti esso opera con tutti i maggiori sottosistemi quali DB2, IMS e CICS.
Il WAS in ambiente z/OS enfatizza le caratteristiche di sicurezza, scalabilità e recovery tipiche dei Sistemi Centrali.
La struttura dei processi (Address Spaces) del WAS è costituito da un processo di controllo detto Control Region (CR) e uno o più processi dipendenti chiamati Servant Region (SR).

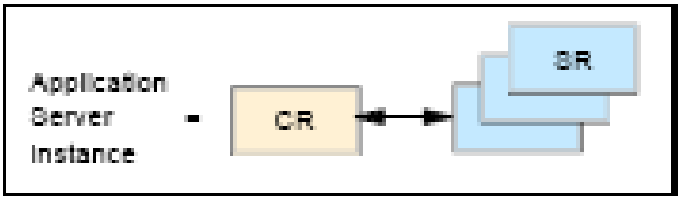


Figura 90 WAS CR & SR

L’application server in z/OS fornisce due possibili configurazioni mantenendo in entrambe le stesse gerarchie architetturali.
Esistono infatti una opzione base (**Base Server Node**) basata su un'unica istanza di application server e di un daemon server e una opzione estesa chiamata **Network Deployment Server** che consente di gestire da un'unica locazione centralizzata più istanze di application server.

2.0 La Virtualizzazione

In questo capitolo vengono introdotti i concetti generali sulla virtualizzazione e i motivi che ne hanno giustificato la necessità nei sistemi informatici in genere.

Premettiamo che oggi la virtualizzazione delle risorse informatiche rappresenta una caratteristica essenziale dei sistemi informatici dalla quale è molto difficile sottrarsi per varie ragioni che analizzeremo in dettaglio in questo capitolo e nel successivo caoito 4.

Prima tra queste ragioni è la necessità economica di massimizzare l'utilizzo dei Sistemi e seconda quella di renderli "elastici" alle variazioni anche rapide del Carico Applicativo.

Quest'ultima caratteristica detta comunemente "resilienza" (resiliency) si rende necessaria a causa della imprevedibilità del carico dovuta alla estrema estensione delle applicazioni accessibili in rete da un grande numero di "potenziali" utenti, a priori non noto.

2.0.0 Concetti Generali

Partiamo dalla definizione del concetto di Virtualizzazione, intendendo con tale termine la capacità di creare, attraverso una componente hardware o software, immagini di risorse informatiche virtuali sulla base delle risorse fisiche realmente disponibili delle quali rappresentano un sottoinsieme logico.

Un'altra definizione di Virtualizzazione comunemente usata è la seguente: "Virtualizzazione è il processo di presentazione delle risorse di un sistema di calcolo in maniera tale che gli utenti e le applicazioni possono facilmente

averne un beneficio, invece di rappresentarle in base alla loro implementazione, dislocazione geografica o composizione fisica. In altre parole, viene fornita una visione logica invece che fisica dei dati, potenza di calcolo, capacità di immagazzinamento ed altre risorse".

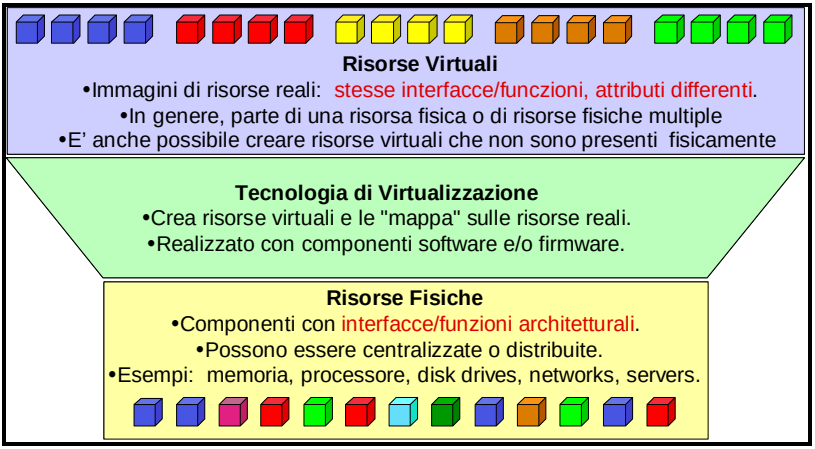


Figura 91 Virtualizzazione

2.0.1 Caratteristiche della Virtualizzazione

- Ogni macchina virtuale, o partizione logica, si presenta nei confronti delle altre come se fosse fisicamente separata.

- Indipendenza - ogni evento che si verifica su una macchina virtuale, o partizione logica, non deve ripercuotersi né interferire sulle attività delle altre.

- V=R - all'interno di un sistema a più macchine virtuali, o partizioni logiche, deve essere sempre permesso riservare risorse fisiche ad

una o più di esse. Tali risorse vengono identificate come Risorse Dedicate.

- Unico Punto di Gestione - un sistema con più macchine virtuali, o partizioni logiche, ha un unico punto di accesso in grado di gestire e controllare le macchine virtuali e la loro configurazione.

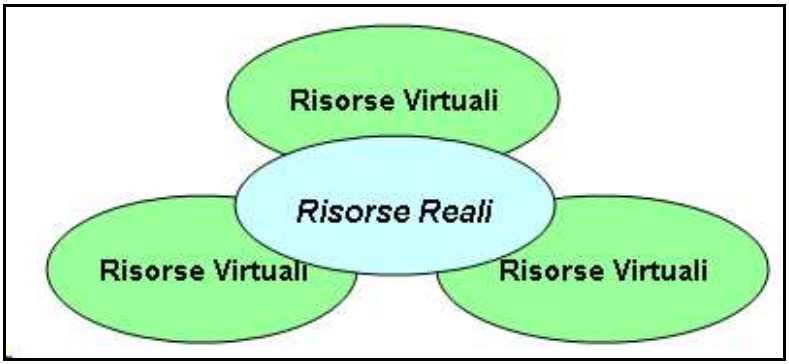


Figura 92 Risorse reali e virtuali

2.0.2 Le ragioni della Virtualizzazione

I motivi che hanno portato all'adozione di un sistema in grado di virtualizzare le risorse fisiche finite e renderle disponibili alle macchine virtuali come se fossero reali sono essenzialmente legati al tentativo di ridurre ed ottimizzare le infrastrutture informatiche, viste le sempre più pressanti esigenze di ordine economico connesse alla riduzione del personale, del consumo di energia, di riduzione dei costi gestione.

Altro beneficio della virtualizzazione è la possibilità di creare all'occorrenza nuovi sistemi senza dover necessariamente acquistare nuovi dispositivi o nuovi server. Ad esempio, in questa maniera sono soddisfatte le esigenze di avere nuovi sistemi che ospiteranno ambienti per un periodo limitato di

tempo, sistemi di sviluppo legati ad un determinato progetto alla conclusione del quale sarà possibile "distruggere" le macchine virtuali senza dover "distruggere" di conseguenza alcun dispositivo hardware. Dal punto di vista economico, questo è sicuramente un grosso beneficio. Riassumendo:

- Ottimizzazione - per definizione, la somma delle risorse virtuali definite può essere maggiore delle risorse effettivamente presenti e di conseguenza è possibile utilizzare risorse virtuali in eccesso. Questo fenomeno viene definito "over committment" oppure "over booking".

- Separazione - con la Virtualizzazione si possono suddividere le risorse reali in ambienti logicamente separati gestiti in maniera indipendente.

Ecco un esempio di come la virtualizzazione possa essere un efficiente metodo per ottimizzare ed ottenere un beneficio sia tecnico che economico.

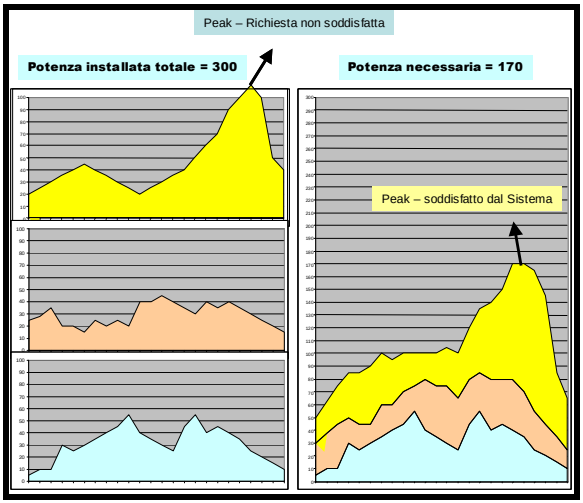


Figura 93 Condivisione = Risparmio

Nella parte sinistra della Figura 93 sono riportati i grafici che mostrano l'attività giornaliera di 3 sistemi diversi che ospitano altrettanti workloads. Prescindendo dall'unità di misura, assumiamo che ogni sistema abbia una capacità installata pari a 100, per cui la potenza totale installata è pari a 300. Come si può notare, il primo sistema (giallo) non riesce ad onorare il picco di carico.
Sulla destra, è stato rappresentato il grafico, stessa scala, di un sistema virtualizzato che ospita tutti e 3 i workloads precedenti. Come si può notare, la potenza necessaria richiesta è notevolmente inferiore, 170/300, e soprattutto il sistema dimostra una notevole flessibilità nell'assorbire i picchi di carico.

L'esempio si basa comunque sull'importante concetto che tutte le risorse siano state "**condivise"** tra i tre carichi di lavoro nel Sistema virtualizzato.
Il concetto di condivisione delle risorse tra carichi di lavoro diversi (in questo caso Sistemi Operativi diversi) non è intuitivo: per definizione e per caratteristiche costruttive ogni risorsa informatica infatti può essere utilizzata da un solo "utente" per volta nello stesso intervallo "elementare" di tempo (frequenza propria).
Per spiegare il concetto di condivisione ricorreremo ad un esempio: immaginiamo di avere due calcolatori dotati ciascuno di una CPU (processore) che eseguano entrambe lo stesso "workload" consistente nel "contare" per uno (istruzione A=A+1) scrivendo ogni volta il risultato su un dispositivo esterno qualunque, rileggendo il dato scritto prima di ogni iterazione.
Il programma che i nostri due calcolatori stanno eseguendo in estrema semplificazione si compone di poche istruzioni, assumiamo che esse siano:

```
A0 READ A
   A = A+1
   WRITE A
   GOTO A0
```

Chiaramente il nostro programma realizza un "loop" infinito e peraltro di scarsa utilità pratica, ma ciò non è importante ai fini del nostro esempio.

Se assumiamo che entrambe le CPU siano capaci di eseguire quattro milioni di istruzioni per secondo ci aspettiamo che dopo un secondo i due dispositivi esterni su cui abbiamo scritto il numero A contengano il valore di "un milione" in entrambi i calcolatori, in effetti ciò probabilmente non accadrà perché mentre le istruzioni di somma (+) e salto (GOTO) saranno state eseguite alla velocità massima, le istruzioni di I/O (WRITE e READ) risentiranno di una certa "latenza" dovuta ai tempi di accesso al dispositivo esterno; se assumiamo che tale latenza sia pari ad un ciclo "perso" per ogni accesso, in pratica i nostri due calcolatori eseguiranno le quattro istruzioni in sei milionesimi di secondo invece che in quattro come potrebbero, ciò non è un errore ma un comportamento "fisiologico" che possiamo aspettarci da qualunque dispositivo "reale".
Da ciò segue che ognuno dei nostri due calcolatori rimarrà potenzialmente inattivo per due cicli ogni quattro cioè per un ciclo ogni due e quindi alla fine del nostro periodo di osservazione avrà "contato" circa un terzo di meno di quello che ci aspettavamo.

Torniamo alla condivisione, se fossimo in grado di "unificare" i due calcolatori in uno solo con due CPU (CPU1 e CPU2) la potenza sarebbe doppia (cioè in grado di eseguire otto milioni di istruzioni al secondo). A questo punto se "assegnamo" fisicamente una CPU ad ognuno dei due workloads avremmo lo stesso risultato del caso precedente: questo realizza un partizionamento "fisico".

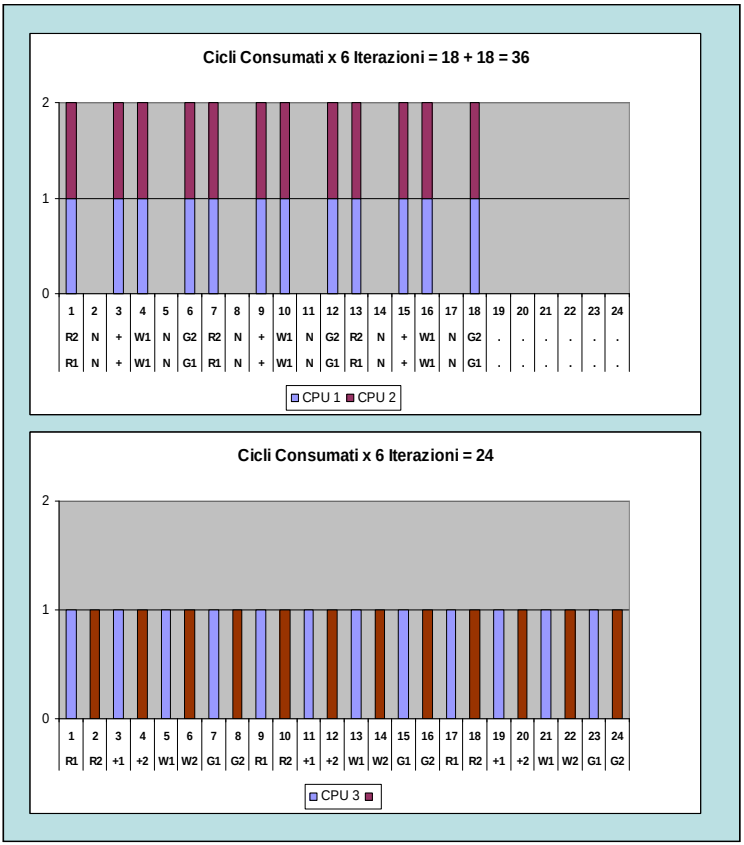


Figura 94 Condivisione della CPU

Se invece introduciamo un dispositivo in grado di condividere una sola CPU di potenza doppia (CPU3) tra i due workloads ma di eseguirli nello stesso intervallo di tempo, potremo ottenere il risultato che il secondo "workload" potrà utilizzare tutti i cicli "persi" dal primo e quindi almeno in teoria la capacità di "contare" dei due nostri programmi risulterà complessivamente maggiore di quella misurata prima (circa il 50% di più): ciò perché siamo riusciti ad utilizzare tutti i cicli precedentemente persi dalle nostre CPU in

attesa del trasferimento dei dati sul dispositivo esterno. Abbiamo così realizzato un "partizionamento logico" o "Virtualizzatore".

In effetti ancora "*una istruzione per volta"* sarà stata eseguita dalla nostra CPU ma in modo tale che nessuna sua potenzialità esecutiva sia andata perduta.

Questo esempio in effetti non è realistico per due ragioni, la prima è che il "Virtualizzatore" ha comunque un peso non "nullo" nel processo, la seconda è che abbiamo ipotizzato un caso "ideale" molto diverso dalla realtà, per il quale si è assunta la capacità di perfetto "sincronismo" tra una operazione "lenta" di un programma con una "veloce" dell'altro, ciò in pratica non è sempre realizzabile.
L'esempio è comunque efficace per chiarire il concetto di "condivisione": diremo quindi che una risorsa informatica è "condivisa" (Shared) tra due o più processi quando *osservandola per un periodo di tempo prolungato* (di almeno un ordine di grandezza superiore rispetto ai tempi determinati dalle frequenze proprie della risorsa) *osserveremo che essa è utilizzata da più di un processo .* E' ovvio che ognuno dei processi la utilizzerà da solo per uno o più cicli della frequenza propria della risorsa stessa.
In complesso la risorsa "condivisa" sarà stata utilizzata "meglio" di come lo sarebbe stata se non fosse stata condivisa: questo principio non è nuovo in quanto tipico di concetti come la multiprogrammazione già in uso da decenni: stavolta però lo stiamo estendendo non solo ai singoli utenti di un Sistema Operativo, ma a diversi Sistemi Operativi, utenti di un unico Calcolatore fisico.

2.0.3 Virtualizzazione e Server Consolidation

La Virtualizzazione offre l'opportunità di effettuare quello che viene comunemente chiamato "Server Consolidation", cioè la possibilità di ospitare in uno o più elaboratori o server le funzioni e i carichi applicativi presenti su un numero superiore di server (di tale argomenti ci occuperemo più diffusamente nel Capitolo 4).

Il Server Consolidation può essere di tipi diversi:

- **Centralizzazione -** la capacità di concentrare in uno o più server dislocati in diverse locazioni anche geograficamente differenti. Questa esigenza nasce dalla necessità di razionalizzare un tipo di infrastruttura molto diffusa negli anni '90, detta Client/Server che prevedeva la distribuzione dei server in maniera capillare su zone di territorio molto vaste. Questo modello creava problemi di tipo gestionale e una eccessiva proliferazione dei server.

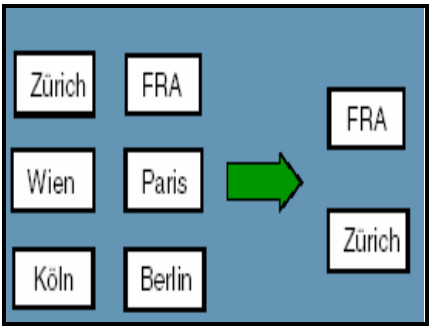


Figura 95 Centralizzazione

- **Consolidamento Fisico -** la capacità di concentrare in un unico server le istanze di una stessa applicazione o funzione, ad esempio la funzione di database server, presenti su più server distribuiti.

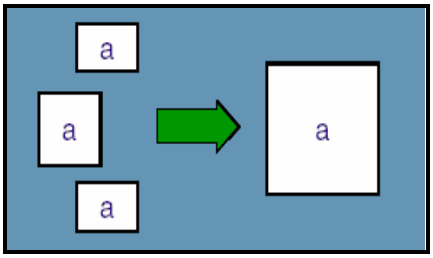


Figura 96 Consolidamento Fisico

- **Consolidamento Logico -** la capacità di concentrare in un unico server applicazioni e/o funzioni diverse presenti su più server. Condizione essenziale è che, dal punto di vista applicativo, non devono essere richieste modifiche di alcun genere.

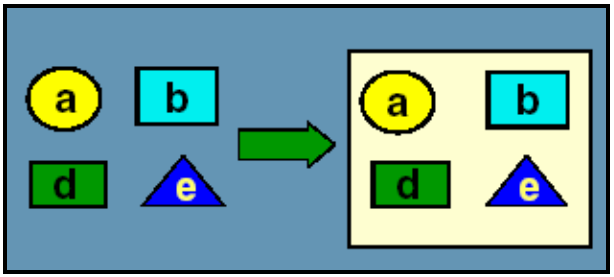


Figura 97 Consolidamento Logico

- **Consolidamento Virtuale -** la capacità di concentrare in un unico server funzioni e/o applicazioni diverse tra loro, intervenendo sulla struttura architetturale delle stesse. Può essere di tipo Applicativo o Virtuale.

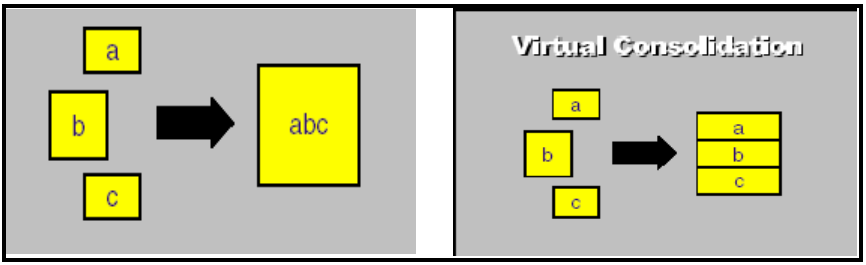


Figura 98 Consolidamento Applicativo e Virtuale

2.0.4 Le risorse virtualizzabili

Le risorse virtualizzabili sono le seguenti:

- **CPU** - Più macchine virtuali possono condividere lo stesso processore nelle implementazioni software z/VM ed ESX con granularità teoricamente infinita, praticamente limitata dalla potenza del processore. Tutte le implementazioni IBM ed ESX consentono dinamicità di assegnazione della risorsa CPU anche durante il funzionamento in base alle richieste dei Sistemi Ospiti;

- **Memoria** - In tutte le implementazioni Hardware IBM e non IBM la memoria è fisicamente assegnata ad una Partizione Logica o Fisica e può essere riconfigurata sia pure a Sistemi attivi, ma solo con un intervento esterno; essa pertanto non viene virtualizzata. Nelle implementazioni Software z/VM, ESX, vPars, la memoria è virtualizzata in quanto risorsa gestita dal Virtualizzatore che possiede di suo meccanismi di Memoria Virtuale, di paginazione e swapping. Se necessario, con z/VM è possibile assegnare della memoria reale direttamente ad un Sistema Ospite (V=R): in questo caso la memoria assegnata è gestita direttamente dal Sistema

Ospite e viene esclusa dai meccanismi di paginazione del Virtualizzatore;

- **Dischi** - Nelle implementazioni Software sono sempre Virtualizzati mentre nelle implementazioni Hardware possono essere virtualizzati con l'uso di appositi Sistemi di servizio (vedi dopo) denominati Virtual I/O server; nell'architettura zSeries questo non è richiesto;

- **Dispositivi di I/O** - E' un argomento di cui il mercato si occupa poco anche se molto importante. Immaginiamo di avere una scheda di rete a 6GBit che sia dedicata ad una partizione logica che la utilizza solo al 50%. La partizione adiacente, cui è stata assegnata una scheda di rete da 2Gbit, non riesce a smaltire tutto il traffico con gli utenti. Se si condivide la scheda di rete a 6Gbit fra le due partizioni si ha banda passante sufficiente a entrambe le partizioni e si risparmia la scheda da 2Gbit. La virtualizzazione dei dispositivi di I/O avviene direttamente solo nelle implementazioni Software della Virtualizzazione (z/VM, ESX), in questo caso la funzione di Gateway viene eseguita dal Supervisore del virtualizzatore (CP); nelle implementazioni via Hardware microcodificate essa viene usualmente gestita da una istanza di Servizio (Virtual I/O Server). Anche qui la virtualizzazione del mainframe evita il ricorso ad un Virtual I/O Server.

2.1 Virtualizzazione Hardware

La virtualizzazione può essere operata più facilmente a mezzo di dispositivi Software (detti Software di virtualizzazione) ed in effetti questo è stato il primo modo in cui essa è stata realizzata (negli anni '60 da IBM col software denominato CP e poi VM).

Relativamente recente (1987) è la prima introduzione sul mercato di sistemi in grado di "virtualizzare" usando solo l'hardware (appositi Firmware con caratteristiche più vicine all'Hardware che al Software). Tali dispositivi sono anche detti "Partizionatori" in quanto suddividono i Sistemi Fisici (calcolatori) in parti.

Osserviamo che "partizionamento" e "virtualizzazione" non esprimono lo stesso concetto: in generale un partizionatore non è necessariamente un virtualizzatore (mentre non è vero il viceversa) .

I partizionatori tendono a suddividere i Calcolatori Reali in parti (spesso legate addirittura alle caratteristiche fisiche delle macchine) per le quali la somma delle risorse (parti) equivale alle risorse reali disponibili, e quindi tendono a non realizzare la condivisione delle risorse come descritto all'inizio del capitolo.

Viceversa i virtualizzatori si pongono l'obiettivo di condividere le risorse tra le parti (logiche) definite: in questo caso la somma della risorse logiche può essere anche maggiore delle risorse fisiche disponibili.

L'obiettivo della condivisione totale non è sempre raggiunto, ad esempio la condivisione della "memoria centrale" tra le partizioni è un obiettivo non ancora completamente raggiunto dai virtualizzatori hardware.

2.1.0 Le diverse opzioni disponibili

Esistono diverse possibilità di effettuare il partizionamento ed in questa sezione prendiamo in considerazione le tecniche più utilizzate.

La prima, detta Partizionamento Fisico o Hardware Partitioning, consiste nel suddividere una macchina e le sue risorse in frazioni, dette generalmente domini, nei quali gira un sistema operativo. Tali domini sono rigidi, nel senso che una risorsa appartenente ad un dominio non può né interagire con le risorse di un altro dominio né essere assegnata ad un altro dominio a meno di un fermo del Sistema Operativo con conseguente ridistribuzione delle risorse. La rigidità di questa soluzione ne costituisce il limite più importante per una larga diffusione totale nel mercato informatico.

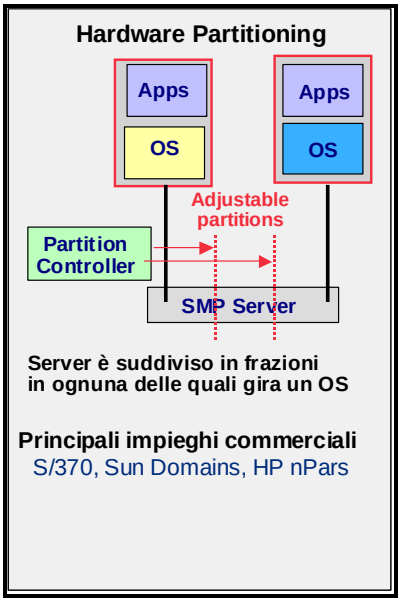


Figura 99 Partizionamento fisico

L'altra tecnica di partizionamento consiste nel suddividere logicamente l'elaboratore senza le restrizioni dovute ai domini fisici del caso precedente. In questo caso non c'è corrispondenza, ad esempio, tra le partizioni e le posizioni fisiche delle risorse ad essa assegnate. Il partizionamento logico può essere realizzato con il firmware, o hypervisor, Type1, o con firmware e sistema operativo, Type2.
Il Type1 è diventata l'opzione dominante nel mercato dei server per l'alta efficienza ed affidabilità.
La soluzione Type2 si rivolge essenzialmente al mercato delle soluzioni workstation e clients, dove è richiesta un'alta integrazione con il sistema operativo.

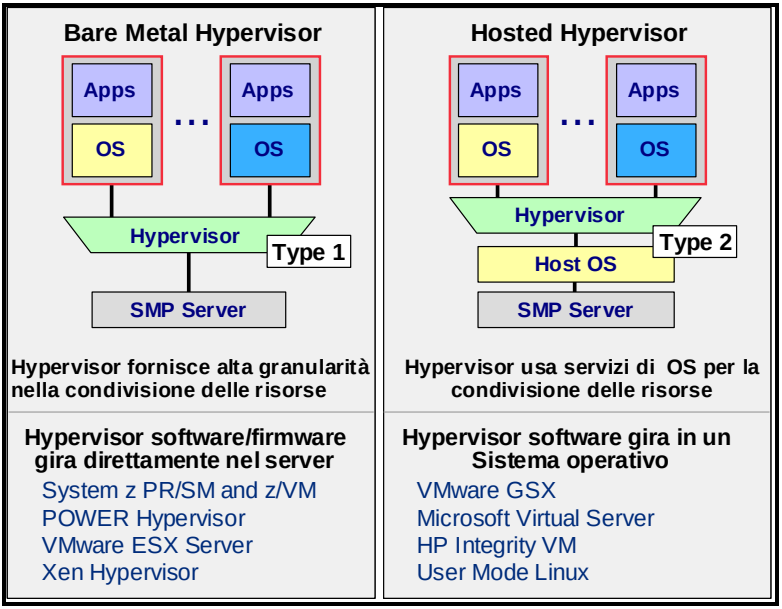


Figura 100 Partizionamento logico

Altro elemento di differenziazione all'interno dei virtualizzatori basati su Firmware o su Hypervisors (Type1) è rappresentato dalla presenza o meno di particolari specifiche istruzioni (o set di istruzioni) legate alla

virtualizzaione all'interno della architettura della CPU. Ad esempio le CPU basate sulla z/Architecture contengono una istruzione denominata SIE (Start Interpretive Execution) in grado di facilitare la creazioni di Partizioni Logiche e Sistemi Virtuali basandosi anche sull'uso di particolari dispositivi Hardware presenti nelle CPU stesse (Special Register). Tale tecnica è in via di estensione anche ad altre architetture.

Nel caso di presenza di dispositivi Hardware specifici e delle relative istruzioni il processo di virtualizzazione risulterà più efficiente (Hardware Virtualization) rispetto al caso contrario (Paravirtualization).

2.1.1 Il Partition Resource & System Management (PR/SM)

Il partizionamento è la possibilità di suddivisione delle risorse di un singolo calcolatore (server) in più sistemi indipendenti ed isolati in grado di eseguire il proprio sistema operativo.

Un breve accenno storico:

prima del'avvento del PR/SM, un calcolatore (server) poteva essere partizionato fisicamente al massimo in due immagini, nelle quali veniva eseguito un sistema operativo. Le risorse fisiche erano assegnate rigidamente, non condivise e, tranne qualche eccezione, non trasferibili da un immagine all'altra.
Nel 1984 IBM propose una funzione chiamata Conversion Assist Facility per facilitare la migrazione dal sistema operativo MVS/370 al sistema operativo MVS/XA sullo stesso elaboratore e senza cambiare architettura. Il successo di questo dispositivo costituì la base del prodotto detto Multiple Domain Facility (MDF). Successivamente, nel 1988, IBM rilasciò la prima versione del PR/SM seguita da Hitachi che propose il suo Multiple Logical Partition Facility.

Figura 101 Un CED di qualche anno fa

Il Partition Resource & System Management (PR/SM) è un firmware microcodificato e non modificabile dall'utente e, in quest'accezione, può essere considerato un dispositivo Hardware. La funzione principale è di considerare tutte le risorse di macchina come un unico shared-pool e permette a tutte le Partizioni Logiche (LPAR) definite di considerare le risorse logiche ad esse assegnate come reali e di uso esclusivo.

La definizione migliore per il PR/SM è la seguente: "un dispositivo standard, hardware e LIC[14], disponibile sugli elaboratori IBM System z che permette ad una macchina fisica l'esecuzione contemporanea di più sistemi operativi".

[14] Licensed Internal Code

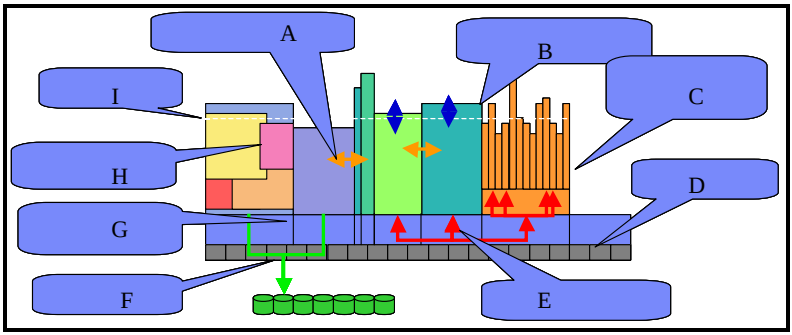


Figura 102 PR/SM Highlights

Principali caratteristiche del PR/SM:

- Condivisione completa delle risorse
- Supporto simultaneo di sistemi operativi diversi
- TOD[15] diversi per le diverse Partizioni Logiche
- Allocazione dinamica delle risorse
- Risorse dedicate - se necessario
- Capping di risorse
- Carichi di lavoro misti in ogni LPAR
- Ri-definizione delle risorse assegnate in base alle esigenze
- Granularità delle risorse

La seguente tabella riassume il numero massimo di partizioni logiche, per ogni famiglia di elaboratori IBM, che il PR/SM permette di ospitare.

Uno degli aspetti più qualificanti del PR/SM è la granularità offerta in termini di suddivisione della risorsa processore. Infatti, su un elaboratore con un solo processore e con 60 partizioni logiche definite, ogni partizione avrà a disposizione 1/60 del singolo processore reale. Inoltre, se consideriamo la possibilità di assegnare ad ogni partizione un "peso" relativo rispetto alle altre, ed il peso è espresso in millesimi, ecco che una partizione con peso di 1/1000 avrà a disposizione 1/1000 di processore.

[15] Time of Day

Famiglia	Max Lpar
z900 , z800	15
z990 , z890	30
System z9	60

La granularità di un sistema z9 è di 1/60 di un singolo processore

Figura 103 Numero Max LPAR

2.1.2 PR/SM - Caratteristiche delle partizioni logiche

Una partizione logica si presenta come un sistema isolato costituito da processore, memoria ed I/O. Ad ogni partizione è assegnato un nome, un numero di processori logici, peso, memoria centrale (ed espansa), canali, TOD e specifiche per l'accesso alle interfacce di I/O (canali, path e devices). Per ogni tipo di risorsa è possibile assegnare diversi parametri e definizioni.

Prendiamo, ad esempio, la risorsa processore. I processori logici possono essere:

- Dedicati
  - in rapporto 1/1 con un processore fisico
  - appartengono in maniera esclusiva a quella partizione
- Shared
  - in rapporto 1/n con un processore fisico
  - la condivisione con le altre partizioni è in funzione del "peso" assegnato alla partizione logica
  - è possibile superare il peso assegnato solo se le altre partizioni non richiedono l'uso del processore

- il Capping Hardware non permette di superare il "peso" assegnato
- il Capping Software non permette di superare la capacità assegnata

Analizziamo in dettaglio alcuni aspetti, quali il peso ed il capping. Ad ogni partizione è possibile assegnare un valore di peso da 1 a 999, e tale valore viene preso in considerazione dalla componente di dispatching del PR/SM in caso di contesa della risorsa processore.

Vediamo un esempio:

3 partizioni
6 processori fisici
8 processori logici

Partizioni	Proc logici	Peso	LP/PP	Peso Relativo	%
LparA	2	300	2/6	300/1300	23,1%
LparB	1	100	1/6	100/1300	7,7%
LparC	5	900	5/6	900/1300	69,2%
Totale	8	1300			100%

Figura 104 Esempio di definizioni

In questo caso, su un sistema con 6 processori fisici ed 8 processori logici definiti, la partizione **LparC** avrà garantito l'uso del 69,2% della potenza totale disponibile.

Come detto in precedenza, la potenza non utilizzata o non richiesta da una partizione viene lasciata libera per le altre le quali possono usare più potenza superando lo "share" loro assegnato. Il Capping, hardware o software, serve a contenere il consumo di potenza di una partizione nel rispetto del peso ad essa assegnata.

2.1.3 PR/SM - Gestione I/O - Multi Image Facility (MIF)

Le prime versioni del PR/SM consentivano la condivisione della risorsa processore ma non prevedevano ancora la piena condivisione delle interfacce di I/O, i canali. Era necessario quindi, per permettere l'accesso ad un dispositivo esterno, device, alle N partizioni logiche, prevedere N accessi diversi. Tutto ciò portava ad un sovradimensionamento e ad una ridondanza che non consentiva un efficiente gestione dei sistemi informatici. Nel **1992** fu introdotto il Multiple Image Facility (MIF) per risolvere il problema dei canali replicati in un ambiente partizionato, consentendo la condivisione dei canali tra le partizioni logiche.

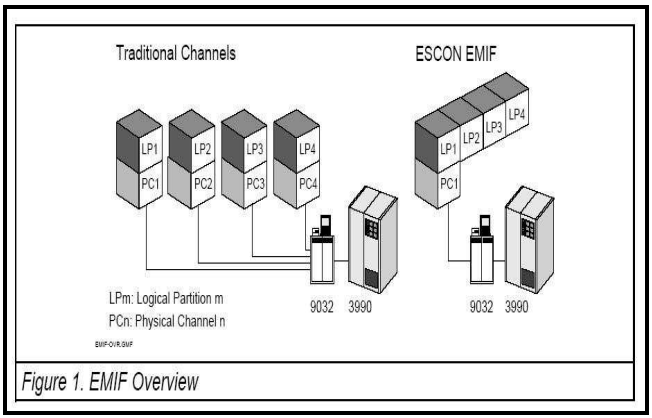


Figura 105 MIF

Il benefici del MIF sono riduzione del numero dei canali necessari, maggiore efficienza nell'utilizzo dei canali e facilità nella gestione delle configurazioni. La Multi Image Facility è realizzabile grazie alle caratteristiche architetturali della gestione delle operazioni di I/O sui sistemi z/Series basati sul concetto di Channel Subsystem con processori dedicati.

2.1.4 Risorse di I/O virtuali (HiperSocket)

Il PR/SM consente la virtualizzazione delle connessioni LAN tra le partizioni logiche. Questa modalità viene chiamata HiperSockets. La velocità di questo tipo di connessione, regolato dal protocollo TCP/IP, è estremamente elevata in quanto lo scambio di messaggi avviene direttamente in memoria, per cui la velocità è in relazione al tempo d'accesso alla memoria stessa. Con questa modalità, a parte la notevole differenza di esecuzione, vengono eliminate tutte le connessioni fisiche esterne (ad esempio cavi, ecc.), i relativi delay e le possibilità di guasto associate.

2.1.5 Riflessioni sulla condivisione delle CPU

Uno dei fenomeni tipici dei sistemi ad architettura SMP [16] è quello denominato "effetto MP", cioè la potenza di ogni processore diminuisce ad ogni aggiunta di un processore alla configurazione. Questa diminuzione, o "effetto MP", è dovuta all'overhead associato alla comunicazione necessaria tra i processori e la relativa gestione a carico del firmware PR/SM. Osserviamo i seguenti grafici:

[16] Simmetric Multi Processor

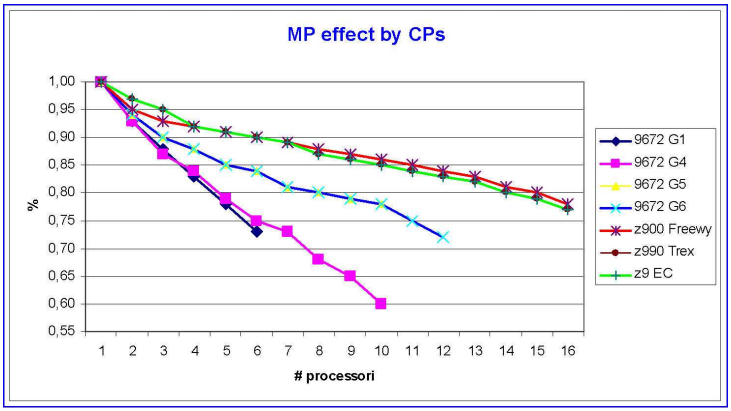


Figura 106 Attenuazione dell'effetto MP

In Figura 106 notiamo come questo effetto si sia attenuato nel corso dell'evoluzione tecnologica delle famiglie di elaboratori IBM nel corso degli anni. L'aggiunta del "processore N", il sesto in questo caso, nella prima generazione di processori IBM CMOS, 9672 G1, forniva solo il 72% della potenza del processore; invece, l'aggiunta del "processore N", il sedicesimo in questo caso, nell'ultima generazione di processori IBM CMOS, System z9, riesce a fornire il 78% della potenza del processore. Le ragioni dell'attenuazione sono dovute ai miglioramenti del codice del firmware PR/SM, del sistema operativo z/OS e dell'hardware, necessari per supportare un numero sempre maggiore di processori, partizioni logiche e memoria.
In Figura 107 abbiamo posto in ascissa il numero di CPU e sulle ordinate ai MIPS che la macchina può erogare. Si nota che con l'aumentare delle CPU l'aumento della potenza erogata non è lineare, ma tende a stabilizzarsi (cioè si raggiunge un punto in cui aggiungere motori non fa guadagnare potenza); le cause principali sono:

- Hardware - man mano che aggiungo processori, le varie CPU si devono sincronizzare tra loro, devono comunicare tra loro, hanno degli elementi comuni (le Cache di livello 2); per garantire

l'integrità dell'elaborazione debbono usare dei protocolli che consumano cicli utili.

- Software - il sistema operativo z/OS, all'aumentare del numero di CPU da gestire, crea i relativi blocchi di controllo e impiega più tempo per la loro gestione ed il dispatcher deve gestire code più lunghe ed aumenta l'overhead.

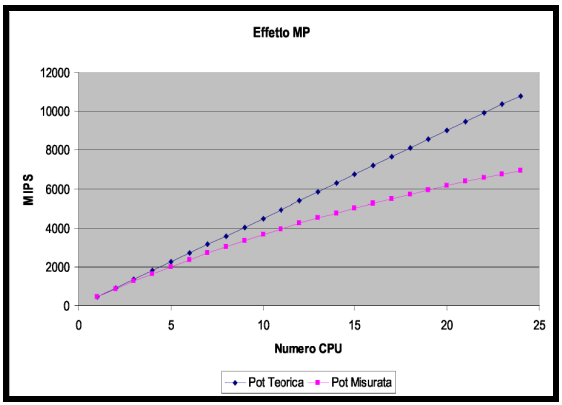


Figura 107 MIPS vs Numero CPU

A causa di questo "effetto MP" all'aumentare delle CPU la potenza totale diminuisce rispetto alla potenza teorica.
Analizziamo però questo fenomeno riportato nel grafico successivo. Un elaboratore IBM 2084-320, con 20 processori, eroga una potenza pari a 6000 MIPS (barra azzurra a sinistra), con una potenza per processore pari a 303 MIPS (6060/20).

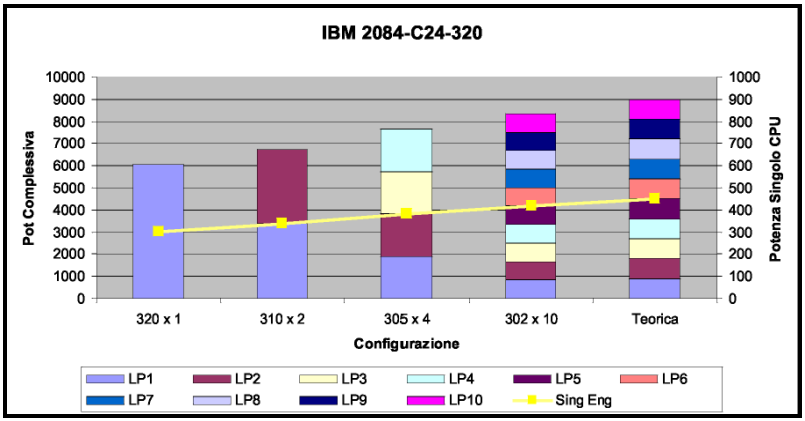


Figura 108 le LPAR riducono l'effetto MP

Supponiamo di dividere l'elaboratore in due ed avremo 2 IBM 2084-310, ognuno di potenza pari a 3370 MIPS , totale 6740 MIPS, potenza per processore di 337 MIPS e così via fino all'ultimo caso dove l'elaboratore è stato diviso in 10 entità da 2 processori ciascuno la cui potenza erogata, 8350 MIPS, è molto vicina alla potenza teorica di 9000 MIPS, 20 x 450 [potenza singolo processore].

	LP1	LP2	LP3	LP4	LP5	LP6	LP7	LP8	LP9	LP10	Sing Eng
320 x 1	6066										303
310 x 2	3370	3370									337
305 x 4	1915	1915	1915	1915							383
302 x 10	835	835	835	835	835	835	835	835	835	835	418
Teorica	900	900	900	900	900	900	900	900	900	900	450

Figura 109 le LPAR riducono l'effetto MP

Come si può evincere dal grafico, la potenza ricavabile dalla somma delle singole LPAR è maggiore della potenza complessiva che si potrebbe ricavare dal sistema se esso venisse usato come un unico SMP.
La realtà dell'IT è però tale per cui si preferisce comunque crescere con il numero di motori sulla singola partizione logica; infatti per bilanciare un

carico reale fra più partizioni logiche in modo che tutte riescano a erogare lavoro come nell'esempio citato, sono necessarie delle tecniche di clustering estrememente sofisticate.

Attualmente solo il Parallel SYSPLEX dello z/OS si avvicina a questi requisiti. Senza contare che aumentando il numero di Partizioni logiche aumenta il costo di gestione della configurazione.

2.2 La Virtualizzazione Software

Questo capitolo descrive il sistema operativo z/VM, esempio di virtualizzatore Software basato sulla z/Architecture. Di esso descriveremo la gestione delle risorse fisiche e le modalità di attivazione delle "Macchine Virtuali".

Rimandiamo al capitolo 3 l'implementazione pratica di un ambiente virtualizzato z/VM con macchine virtuali zLinux.

2.2.0 Il Sistema Operativo z/VM - Caratteristiche

In tema di virtualizzazione non si può dimenticare il primo Virtualizzatore Software sviluppato da IBM, denominato Virtual Machine (VM). La sua versione attuale aggiornata alla z/Architecture è denominata z/VM.

Questo sistema operativo, nato alla fine degli anni 60, prima dei virtualizzatori hardware, si basa sui concetti di memoria virtuale per dividere le risorse di un singolo computer in una molteplicità di computer virtuali indipendenti e isolati, ognuno dei quali dotato di potenza elaborativa, memoria centrale, memoria secondaria (dischi, nastri) e altri dispositivi I/O ed ognuno con il suo proprio sistema operativo. Le risorse, dal loro canto, sono gestite da uno strato software e aggregate in insiemi condivisi. Esse sono attribuite agli utenti come risorse virtuali, separando la presentazione delle risorse dalle entità fisiche reali. Su questi computer virtuali operano sistemi operativi di vario tipo (CMS, z/OS, z/Linux , z/VM ed altri) aderenti anch'essi alle varie architetture supportate (es XA, ESA/390,...), compresa la z/Architecture.

Ogni sistema operativo ospite ha a disposizione un ambiente elaborativo completo, del tutto analogo ad un ambiente fisico. I servizi di virtualizzazione di z/VM fanno credere al sistema operativo ospite di avere il completo controllo dei processori e delle altre risorse loro assegnate.

Dal suo ruolo di controllo z/VM inoltre può far condividere le risorse fisiche delle varie CPU, devices e network alle varie immagini di Sistemi ospite come nella figura seguente:

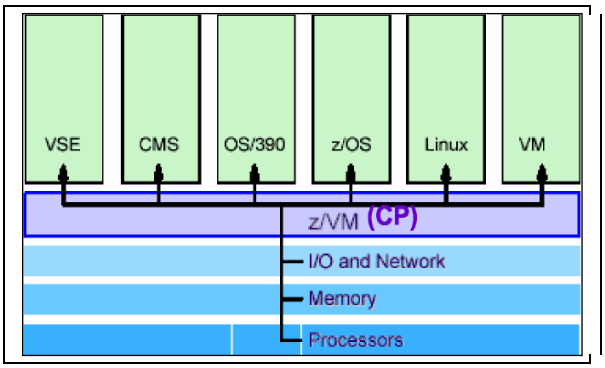


Figura 110 Ambiente z/VM

Gruppi di macchine virtuali possono essere collegate insieme e formare network virtuali. Le funzionalità fondamentali che un virtualizzatore e quindi anche z/VM deve assicurare sono:

- **Multitasking:** ovvero supportare l'esecuzione contemporanea di più sistemi operativi ospiti (guest OS).
- **Isolamento:** cioè garantire che il malfunzionamento di una macchina virtuale e del sistema operativo su essa operante non si ripercuota sulle altre macchine virtuali.
- **Gestione delle risorse e del carico elaborativo:** ovvero non deve accadere che un sistema operativo si accaparri tutta la potenza di calcolo della CPU rallentando quasi completamente gli

altri, né che possa allocare tutta la memoria disponibile privandone gli altri.

- **Amministrazione:** L'amministratore di sistema deve poter avviare, interrompere, riconfigurare o clonare le macchine virtuali senza dover arrestare quelle non interessate dall'operazione.

z/VM fornisce una gestione centralizzata dei dati e del loro backup e inoltre ha funzioni di monitoraggio del carico e delle performance. Il vantaggio legato alla virtualizzazione e alla condivisione delle CPU virtuali è il cosiddetto "over commitment" vale a dire la possibilità di definire risorse virtuali che sono sovrabbondanti rispetto a quelle fisiche. Infatti, se una macchina non può utilizzare tutta una sua risorsa, la parte da essa non utilizzata può essere disponibile ad altre risorse.

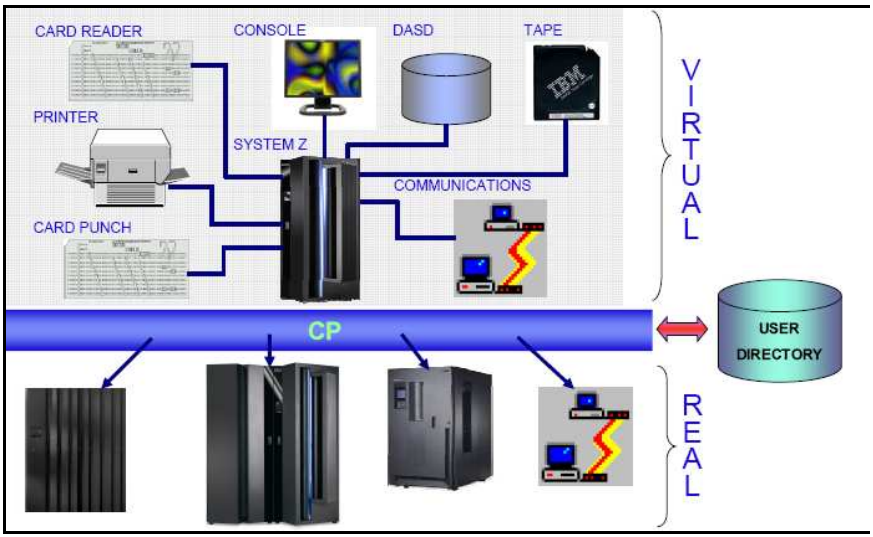


Figura 111 Ambiente z/VM

Le peculiarità di z/VM in termini di gestione delle risorse (ovvero di suddivisione di un sistema singolo in sistemi virtuali multipli ognuno dotato di risorse proprie) lo rende particolarmente utile nel workload interattivo,

nelle applicazioni client/server e nel supporto ai sistemi operativi. Infatti nel caso di elaborazione interattiva il sistema operativo z/VM fornisce una base per un dialogo flessibile tra utenti e programmi applicativi; nel caso di client/server è molto comune la configurazione in cui una macchina ospita un programma che fornisce servizi (server) a utenti di altre macchine virtuali (clients); le efficienze generate attraverso z/VM lo rendono adatto a creare, testare ed eseguire applicazioni per i vari ambienti operativi supportati.

Il modello elaborativo di z/VM utilizza un programma per gestire le risorse hardware di un computer. Tale programma è chiamato Control Program (CP). Le attività degli utenti (macchine virtuali) sono gestite da un sistema operativo ospite. Un particolare sistema operativo fornito insieme a z/VM si chiama Conversational Monitor System (CMS).

2.2.1 Le Macchine Virtuali

Il termine Macchina Virtuale, in questo contesto, viene usato per indicare un ambiente di esecuzione che si comporta come se fosse un computer a se stante ma che in realtà condivide con altre macchine virtuali la stessa macchina fisica.

La "macchina virtuale " è completamente indipendente e lavora in contemporanea con le altre macchine virtuali sotto il controllo di z/VM. Questo è in grado di creare "macchine virtuali" tra loro indipendenti che possono usare le risorse interne del computer (memoria, dischi, porte di comunicazione) senza andare in conflitto tra loro.
La tecnologia di Virtualizzazione è quindi trasparente al Sistema Ospite. La gestione delle macchine virtuali coinvolge vari strati dell'architettura, a partire dai vari meccanismi con cui viene gestita la Memoria Virtuale di z/VM, alla gestione di risorse condivise (CPU, memoria, dispositivi di I/O), alla gestione delle operazioni di I/O ecc.

Un Sistema Operativo che opera su una macchina virtuale ha la stessa operatività di quello che gira su una macchina reale. Ogni dispositivo assegnato ad una macchina virtuale, come la memoria, il processore o un qualsiasi dispositivo di I/O, si deve comportare come se la macchina virtuale fosse reale. Nuovi sistemi ospiti posso essere aggiunti velocemente senza richiedere risorse dedicate.
Ci sono vari sistemi operativi ospiti supportati come ad esempio z/OS, zLinux, z/VSE, z/TPF ed anche z/VM di secondo livello.
Come accennato precedentemente, z/VM permette di far funzionare copie multiple e tipi differenti di Sistemi Operativi sullo stesso Sistema Centrale.

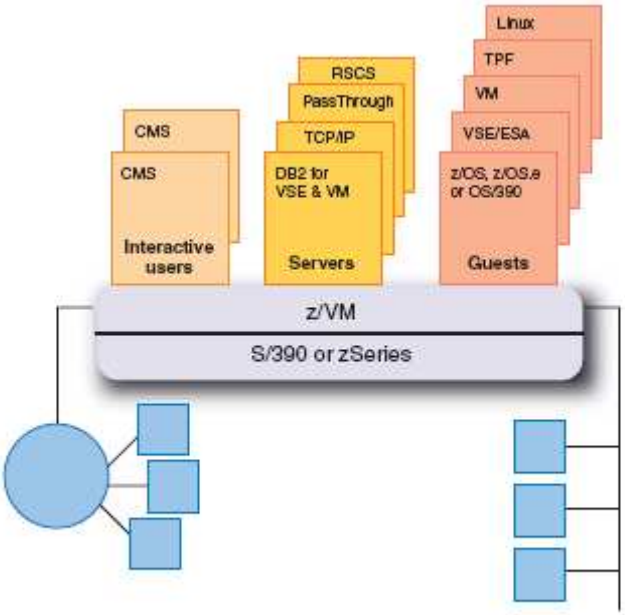


Figura 112 Ambiente z/VM

z/VM tramite uno dei suoi componenti fondamentali, il CP, ha un file di configurazione fondamentale denominato "SYSTEM CONFIG" che definisce, tra le altre cose, le risorse reali a sua disposizione che poi dovrà gestire. In particolare le informazioni sui dischi su cui risiederà il sistema operativo z/VM, la lista dei dischi che utilizzerà il CP, i dischi a disposizone delle macchine virtuali, la configurazione di memoria reale e la configurazione di tutti gli altri dispositivi reali.
D'altro canto z/VM mantiene la descrizione di tutte le informazioni delle macchine virtuali ospitate nel sistema anche nel file USER DIRECT chiamato z/VM User Directory; tale file risiede in memoria alla attivazione di z/VM. All'interno di un sistema z/VM si possono considerare vari tipi di Macchine Virtuali:

- **Macchine virtuali mono-utente con Sistema Operativo CMS:** Macchine Virtuali che ospitano un particolare Sistema Operativo mono-utente, denominato Conversational Monitor System (CMS) in grado di eseguire un ambiente di produttività personale o di sviluppo delle applicazioni o della istallazione e manutenzione di z/VM

- **Macchine Virtuali con Sistemi Ospiti:** Macchine Virtuali che ospitano un qualsiasi Sistema Operativo a 31-bit e 64-bit che usa una architettura supportata tra cui la z/Architecture; tra queste possiamo annoverare macchine che ospitano Sistemi z/VM di Secondo livello ovvero Macchine Virtuali che contengono una nuova istanza di z/VM, utilizzate a vari scopi (manutenzione, nuove generazioni, ecc.)

- **Macchine Virtuali di Servizio:** Macchine virtuali CMS con applicazioni di servizio per tutto z/VM cui si rivolgono gli utenti delle altre macchine virtuali e del CP per usufruire di particolari servizi. Esempi di Macchine Virtuali di Servizio sono:

    - **TCP/IP** – generato per gestire le connessioni in rete delle varie immagini z/VM;

- **Operator** – macchina virtuale con comandi privilegiati per l'interazione con il CP a disposizione degli Operatori di Sistema;
- **MAINT o DirMaint (Directory Maintenance)** - per gestione delle "directory" di macchine virtuali;
- **GCS** - Gestore di Risorse e controllore degli accessi simultanei sui files;
- **VMRDM** - Performance Toolkit;
- **RSCS** - Remote Spooling Communication Subsystem.

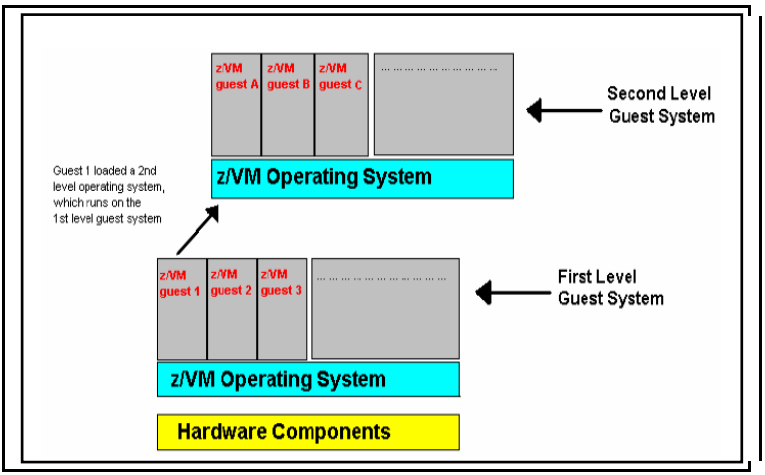


Figura 113 z/VM di secondo livello

2.2.2 Componenti di Base

I componenti di base di z/VM sono il Control Program (CP) e il CMS (Conversational Monitor System)

2.2.2.1 Control Program (CP)

Il Control Program (CP) è un sistema operativo che sta alla base di z/VM. Esso è responsabile della virtualizzazione dell'hardware reale del Sistema Centrale e permette a molte macchine virtuali di condividere risorse hardware contemporaneamente. Il CP si occupa anche della creazione delle Macchine Virtuali, garantendone la sicurezza, l'integrità e l'isolamento. Il CP però non è un sistema a operativo "classico", nel senso che non ha funzioni di gestione dei file e non fornisce un metodo conveniente di caricamento ed esecuzione di programmi differenti e non consente agli utenti di eseguire task produttivi, Web serving, text editing, o data processing.

La sua principale funzione è quella di "Resource Manager". Le risorse che gestisce sono le macchine virtuali. E' poi compito del sistema ospite, sia esso CMS o zLinux o z/OS o altro, usare le risorse fornite dal CP per eseguire i vari Task produttivi.

Come "Resource Manager" il CP permette ad un insieme di Macchine Virtuali di eseguire vari sistemi operativi in un'unica partizione logica (LPAR). Quando un il sistema operativo ospite sarà in IDLE un altro sistema può usare quelle risorse hardware.

La Digura 114 illustra un sistema reale di 4 CPU e 16 GB di RAM in cui le macchine virtuali ospiti hanno tutte insieme un totale di 12 CPU virtuali e 22 GB di RAM.

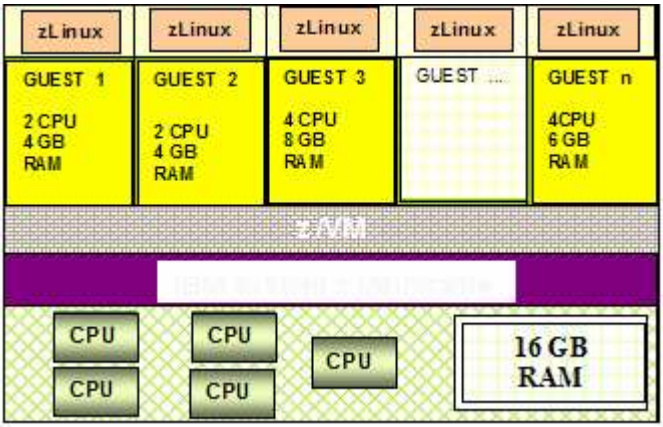


Figura 114 Esempio di ambiente z/VM

Il CP è dotato di un interprete di Comandi accessibile da:

- Interfaccia Line-mode (semplice)
- API programmabile
- Un comando speciale

Questi comandi sono raggruppati in classi di privilegi a seconda dell'utilizzatore (ad esempio System Operator, System Analyst, System Programmer, ecc.).

Il CP è anche fornito di:

- Spazio su disco (DASD) riservato per paging spool e dischi temporanei (TDISK)
- Funzioni estese di monitoring, debugging e tuning
- Funzione di error recovery

z/VM fornisce un supporto ai dispositivi di I/O con vari gradi di coinvolgimento del Control Program (CP):

- *Device dedicati o Attached* - Gli ospiti hanno uso esclusivo del device reale e il coinvolgimento del CP è minimo;
- *Device Virtualizzati* - CP presenta un'immagine di un device reale a più macchine virtuali o esso suddivide un device reale cosicché una porzione appare reale ad un sistema ospite (per esempio DASD, cripto devices, minidisk). In questo caso il CP è più coinvolto ed inoltre offre anche dei servizi aggiuntivi come ad esempio il minidisk caching;
- *Device emulati/simulati* - CP emula un pezzo di device reale al sistema e ai suoi ospiti come un tipo diverso di hardware oppure il CP simula completamente un device per un ospite senza l'esistenza di quell'hardware reale. Sono esempi di device simulati i Virtual Disk (VDISK), i LANs ospiti, i Virtual Unit Record devices (reader, punch, printer, virtual console). Il coinvolgimento del CP in questo caso è molto alto.

I comandi CP

La via principale per comunicare con il CP è una interfaccia a linea di comandi dalla Console. z/VM usa i comandi CP per configurare e manipolare le risorse, in particolare per controllare processori, dispositivi di I/O, e inoltre per controllare la configurazione delle Macchine Virtuali e il loro ambiente di esecuzione.

I Comandi CP hanno parametri posizionali. I nomi dei comandi sono Alfanumerici e minori di 12 caratteri, caratterizzati da verbi che descrivono la funzione; gli operandi sono keywords e simboli minori di 8 caratteri. Essi non sono sempre presenti nei comandi.

Figura 115 Esempio di comandi CP (QUERY VIRTUAL ALL)

Le macchine virtuali comunicano con il CP in una delle due seguenti maniere:

- Un utente (o un tool di automazione al suo posto) può sottomettere comandi CP dalla console virtuale della macchina ospite;
- I programmi che girano nella macchina virtuale possono comunicare direttamente col CP tramite un'istruzione chiamata DIAGNOSE, i cui parametri forniscono al CP tutti i dettagli per ottenere input e fornire una risposta.

L'insieme dei comandi CP e delle numerose funzioni della DIAGNOSE sono suddivisi in gruppi funzionali chiamati classi di privilegi. L'insieme dei comandi e funzioni dell'utente a disposizione su ogni macchina virtuale, quali abilità di fare un IPL, di connettersi ai Minidisk ammessi, ecc. è confinato ad un'unica classe di privilegi detta classe G. Ulteriori classi sono specializzate per i System administrator e per gli Operatori.

2.2.2.2 Il CMS

Il Conversational Monitor System è un sistema operativo mono-utente virtuale. Esso, infatti, può operare solamente all'interno di una macchina virtuale creata dal CP. Il CMS è utilizzato per facilitare l'amministrazione delle machine virtuali fornendo all'utente un ambiente con una funzionalità operativa superiore a quella del CP. Il CMS può eseguire una grande varietà di compiti come scrivere, testare e correggere applicazioni sviluppate nel CMS o su Sistemi ospiti, creare ed editare file di dati, condividere dati tra sistemi operativi ospiti e comunicare con vari utenti di sistema.

Una caratteristica interessante è quella che vede la possibilità del CMS di operare da caricatore di un altro sistema operativo (zLinux, z/OS, ecc.) dal suo interno. In effetti, fatto un boot o IPL di un sistema operativo, il CMS non è più visibile ed il controllo passa completamente al sistema operativo caricato. Ogni macchina CMS è definita in System Directory (IPL CMS).

Il CMS è dotato di:

- Un suo file system che generalmente non può essere letto o scritto da altri sistemi operativi;
- Ambiente di sviluppo e operativo per applicazioni, compilatori, ecc.;
- Funzioni caratteristiche:
    - Pipelines - ovvero dei comandi che consentono a più programmi (stages) di passarsi dei records. Le sorgenti dei dati sono varie (comandi, xedit, dischi, storage ecc.)
    - REXX – un potente linguaggio procedurale di programmazione e script
    - XEDIT – un editor molto versatile
    - Interfacce line-mode, full screen e GUI (XPG4)
- Comandi CMS per le sue operazioni.

I files all'interno del file system CMS sono denominati usando un identificatore (file ID) composto da tre campi:

- File name (FN) definito con lettere a A a Z che identificano i minidisk o le directory dove risiede il file;
- File type (FT) è una estensione del nome che ne descrive le caratteristiche (es di filetipe sono TEXT, COBOL , LISTING ,..);
- File mode (FM) o Directory name (dirname) definito con un numero da 0 a 6 che è assegnato quando il file è creato o rinominato (default=1). E' usato per identificare ed operare su un subset di files.

I comandi CMS

La macchina Virtuale CMS è guidata attraverso un considerevole insieme di comandi da una console, come per il CP. Il linguaggio di comandi CMS consente di creare, modificare, fare debug ed in generale maneggiare files. Inoltre i comandi CMS consentono di leggere cards dal card reader virtuale, perforare schede su un card punch virtuale e stampare records su una stampante virtuale. Ci sono anche comandi che aiutano a maneggiare i dischi virtuali, le directory nei file pools e i files sui minidisk.

Un comando CMS consiste in nome di comando, seguito di solito da uno o più operandi posizionali e in molti casi da un lista di opzioni. Il separatore comune in tutti i comandi è il carattere "blank":

Nome comando operandi (opzioni….

Un esempio di comando CMS è il seguente:

COPY GIORNO1 TXT A ARCHIVIO DATA D

che permette di fare una copia del file GIORNO1 TXT sul disco A memorizzandolo sul disco D come ARCHIVIO DATA.

File di configurazione

La maggior parte delle informazioni sulla configurazione del sistema sono contenute in due file ovvero il System Configuration file (SYSTEM CONFIG) e lo z/VM User Directory System file (USER DIRECT).

System Configuration file

Il System Configuration file (SYSTEM CONFIG on MAINT minidisk CFn) definisce le caratteristiche di base del sistema e consente di definire le opzioni principali quali:

- I dispositivi che il CP mette online all'IPL
- I dischi volumi ad uso esclusivo del CP (CP_OWNED)
- I dischi destinati alle macchine virtuali (USER_VOLUME_LIST)
- Il time zone che il CP seleziona tra quelli disponibili
- Le macchine virtuali che vengono attivate automaticamente (AUTOLOG)
- La locazione delle aree di Checkpoint e di ripartenza veloce (warm start del sistema)

IL file SYSTEM CONFIG contiene definizioni che identificano i volumi che sono per uso specifico del CP denominati CP_OWNED. Tali volumi contengono oltre alle aree di spool, di paging, di checkpoint e warmstart anche l'area dove è caricata la SYSTEM CONFIG che viene letta dal CP in fase di inizializzazione di z/VM. Tali volumi devono avere una formattazione/allocazione a blocchi di 4 Kbytes con una "Byte Map" di allocazione sul cilindro (0) del disco. La formattazione e allocazione dei volumi CP_OWNED è effettuata con il programma di utilità CPFMTXA. La Byte Map è costituita da una stringa di Bytes, uno per ogni cilindro di disco. Ogni byte definisce se quel cilindro può essere usato per formattare ed allocare volumi CP_OWNED. La figura seguente illustra le varie aree allocate nei dischi CP_OWNED.

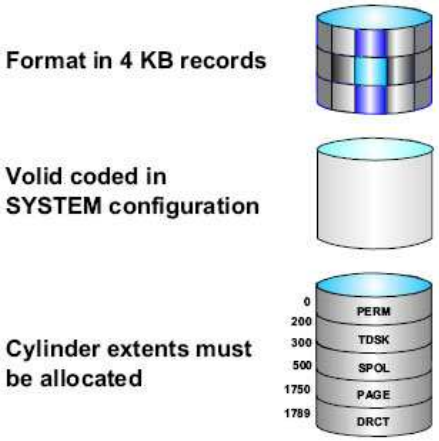


Figura 116 I Dischi CP_OWNED

DRCT Area allocata per il file USER DIRECT
SPOL Area allocata per il files di SPOOL (System Spool Space)
PAGE Area allocata per il file di PAGE
TDISK Area allocata per per il file temporanei (tdisks)
PARM Area allocata per i minidisks di PARM
PERM Area allocata come permanente (a disposizione del CP)

Nella PARM Area sono allocati tre minidischi che sono letti dal CP al tempo di IPL e contengono i parametri di inizializzazione di z/VM. Nel minidisco CF1 è memorizzata la SYSTEM CONFIG.

z/VM User Directory

La z/VM User Directory (USER DIRECT) è un file che è usato per gestire le definizioni di ogni utente. Tenere traccia di queste definizioni può rapidamente diventare complesso all'aumentare del numero di macchine virtuali. Per tale motivo per gestire le user directories in genere viene usato un prodotto denominato IBM Directory Maintenance Facility (DIRMAINT).

La z/VM User Directory descrive al CP la configurazione e le caratteristiche operative di ogni macchina virtuale. La User Directory contiene i nomi e le caratteristiche iniziali di tutte le macchine virtuali attivabili sotto uno z/VM. Questo file può essere editato usando l'editor XEDIT.

Essa è contenuta in forma testuale nel file system del CP. Per ciascuna Macchina Virtuale sono indicate:

- Nome della macchina (un massimo di 8 Caratteri Alfanumerici);
- Password (se non presenti altri prodotti di Security);
- Architettura Hardware della macchina ospite (24, 31, 64 Bit);
- Numero di CPU Virtuali;
- Memoria Virtuale Minima e Massima;
- Minidischi;
- Altri Devices come consoles, spool devices (reader,punch,printer), special devices;
- Opzioni di priorità;
- Il Disco di bootstrap (IPL).

La Directory è composta da una serie di definizioni delle Macchine Virtuali. Ecco un esempio di definizione di Macchina Virtuale in USER DIRECT.

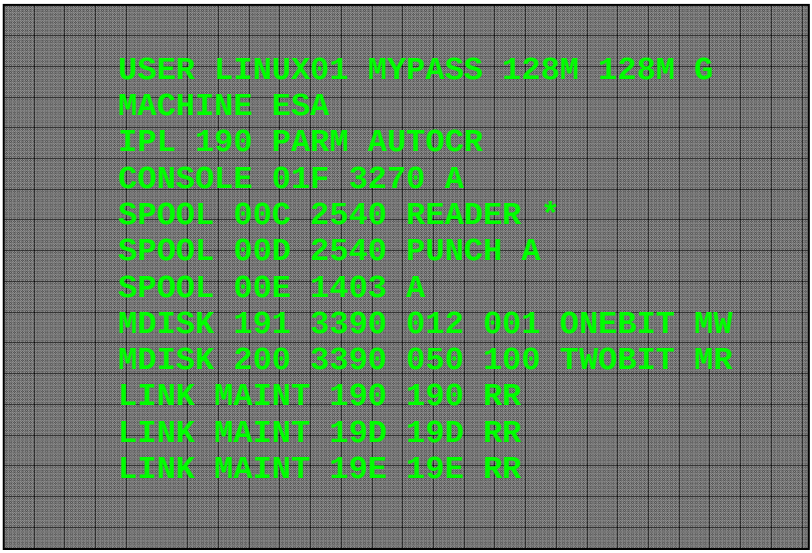


Figura 117 Esempio di definizione in z/VM User Directory

Per poter essere utilizzata dal CP la z/VM User Directory deve essere "compilata" con il programma di utilità DIRECTXA. Una volta compilata da DIRECTXA il file ottenuto da USER DIRECT viene posto nell'area allocata come DRCT in uno dei volumi CP_OWNED (SYSRES).

Il formato del comando è **DIRECTXA fn ft** dove **fn** è il file name della directory (USER è il DEFAULT fn) e **ft** è il file type della directory (DIRECT è il DEFAULT ft).

Le macchine virtuali definite nella USER DIRECT sono "dormienti" fino a che un utente si collega ad esse (LOGON). Esse possono anche essere attivate automaticamnete alla partenza di z/VM tramite il Parametro AUTOLOGON nella definizione della Macchina Virtuale in USER DIRECT.

Accesso ai dischi tramite comando di LINK

Il CP fornisce una via per accedere ai dischi "posseduti" da un altro utente del sistema. Ciò avviene con il comando di LINK. Fare il LINK ad un disco è molto utile per condividere i files con un altro ospite del sistema.

Ad esempio tutti gli z/VM networking tools (come telnet, netstat e ftp) sono locati di solito sul disco 592 della macchina TCPMAINT. Per avere accesso al disco 592 della macchina TCPMAINT si usa il comando LINK.

Il comando LINK ha quattro parametri di base:

- il nome della macchina virtuale che possiede il disco richiesto
- il numero di device virtuale di quel disco (relativo alla macchina virtuale che lo possiede)
- il numero di device virtuale con cui la macchina richiedente definisce il disco "linkato"
- un codice di due caratteri per descrivere il livello di accesso Read/ Write al disco.

Un esempio di questo comando è:

LINK MAINT 190 194 RR

in cui la macchina da cui si da il comando richiede un accesso in read only del minidisco 190 della macchina virtuale MAINT. La macchina richiedente al suo interno accederà questo disco con l'indirizzo 194.

La sessione z/VM

Quando un utente entra nel sistema (ovvero fa "logon") il CP crea una macchina virtuale per quell'utente e questa costituirà l'ambiente operativo dove verrà caricato il sistema operativo ospite (CMS, z/OS, ecc.). Questo verrà caricato con un comando di IPL (Initial Program Load) direttamente dal disco dove si trova il Kernel (o Nucleo) del sistema operativo ospite. E' possibile anche caricare un sistema operativo direttamente da una copia di

un kernel di sistema operativo salvata precedentemente in memoria utilizzando la funzionalità detta Named Saved Segment (NSS). Poi in tale ambiente vengono caricate le varie applicazioni.

I modi di esecuzione del CP

Ad ogni istante una macchina virtuale si trova in uno dei due seguenti stati di esecuzione:

- GUEST MODE - ovvero eseguendo istruzioni del sistema operativo ospite
- CP MODE - ovvero non eseguendo istruzioni del sistema operativo ospite

Quando un sistema operativo sta eseguendo in GUEST MODE esso controlla i device e le risorse della macchina virtuale. Tutti i comandi dati alla macchina virtuale saranno interpretati dal sistema operativo ospite e non dal CP.

Però quando la macchina virtuale non sta eseguendo il suo sistema operativo, è il CP ad avere il controllo. Processori e device virtuali della macchina virtuale sono generalmente in IDLE e non consumano risorse reali.

Quando si fa il Logon di una macchina virtuale si è in CP MODE dopo di che quando si fa il boot o IPL del sistema operativo si va in GUEST MODE. Allo Shutdown ovvero alla chiusura ordinata del sistema operativo si ritorna in CP MODE.

Tutto ciò accade da un terminale locale o remoto in emulazione 3270.

2.2.3 Dispositivi di I/O in ambiente z/VM

All'interno di z/VM occorre far chiarezza sul tema dei dispositivi reali e virtuali su:

- Unità a Disco
- Unità a Nastro
- Spool devices
- Unità di collegamento alla rete (Network Devices)

2.2.3.1 Unità a Disco

Il Control Program (CP) erige una barriera tra le macchine virtuali e i devices al quale il programma ha accesso. Una funzione primaria del CP è quella di mediare l'accesso ai device reali in varie modi, a seconda che il device debba essere condiviso tra due o più macchine virtuali simultaneamente, come ad esempio un DASD, o se il device debba essere disponibile per l'uso esclusivo di una singola macchina virtuale, come ad esempio una unità a nastro.

Quando una macchina virtuale fa una richiesta di I/O, questa è intercettata dal CP cosicché gli indirizzi sono tradotti nei corrispondenti indirizzi reali. Il CP esamina che la richiesta di I/O non sia impedita da altre I/O prioritarie effettuate da altre macchine virtuali autorizzate e, in caso negativo, inizia l'operazione di I/O al posto della macchina virtuale. Il principale device su cui si memorizzano i dati per z/VM e i dati delle macchine virtuali è il disco (DASD). Esso può essere utilizzato dal sistema operativo z/VM oppure utilizzato dal CP (CP_OWNED) o dagli utenti (USER_VOLUME_LIST) per:

- Checkpoint data, Warmstart data, Paging, spooling data
- Minidisks dei sistemi ospiti (tramite MDISK z/VM System Directory Statement)
- Dischi temporanei dei sistemi ospiti (tramite comando CP DEFINE T3390).

I dischi virtuali (virtual DASD) usati dalle macchine virtuali sono chiamati "minidisk". Essi sono creati dal partizionamento di dischi fisici reali (Real DASD) in intervalli di cilindri che appaiono come dischi al server virtuale. Un minidisk può anche essere un intero disco reale.

Esistono tre tipi di minidisco:

- **Permanente** o residente ovvero memorizzato su una porzione di disco reale che rimane memorizzato indipendentemente dalle sessioni. Esso è assegnato ad una o più macchine virtuali con una definizione in z/VM System Directory;

- **Temporaneo** (TDISK) ovvero minidisco virtuale temporaneo usato per appoggiare dei dati all'interno di una sessione VM (LOGON). Tali dati si perdono alla fine della sessione (LOGOFF). Si possono definire TDISK di varie grandezze senza preoccuparsi della locazione su disco (non dobbiamo perciò preoccuparci della contiguità dei cilindri allocati);

- **Disco virtuale** (VDISK) **in memoria** ovvero un dispositivo di I/O ad alta velocità in grado di compiere le stesse operazioni dei dischi fisici anche se allocati in Memoria Virtuale e quindi paginati sulla memoria reale di z/VM. I Dischi virtuali sono volatili. Se accade un errore grave nel CP o uno shutdown i dischi virtuali sono persi. Esso ha le stesse caratteristiche fisiche del disco fisico (velocità, geometria, ecc.). Mentre è in uso, i blocchi di VDISK sono in memoria centrale. Ciò è molto vantaggioso in termini di performance. Quando non è in uso esso, se necessario, può essere ospitato sui paging dataset. Poiché il dato del VDISK è volatile esso dovrebbe essere usato per file temporanei (ad esempio assembly, VSE lock file, or sort applications). Un VDISK può essere creato tramite comandi CP o all'interno di una definizione di MDISK nella z/VM System Directory. Virtual Disk (VDISK) creati all'interno della definizione MDISK sono condivisibili da più macchine virtuali. Il vantaggio dei VDISK è che l'operazione di I/O è in effetti trasferita dal device fisico per i files residenti al paging I/O con vantaggi di performance. (nel caso ci sia sufficiente memoria reale addirittura

non si effettua neanche il paging I/O). L'area di paging VM dedicata ai Virtual disk occupa circa un quarto del totale.

2.2.3.2 Unità a Nastro Virtuale

In maniera del tutto analoga ai dischi abbiamo la possibilità di condividere le unità a nastro fisiche tra varie macchine virtuali utilizzando le tecniche di virtualizzazione. Il nastro virtuale, come quello reale, può essere montato, letto, scritto, riavvolto e scaricato.

2.2.3.3 Spool devices

Nei tempi in cui fu creato il VM la maggior parte delle installazioni comprendevano stampanti, lettori e perforatori di schede reali chiamati Spool Devices. L'uso di tali dispositivi virtuali mantiene la sua fondamentale importanza nella operatività abituale di z/VM anche se i dispositivi reali che li denominano sono obsoleti. I dati coinvolti in input o in output su tali dispositivi sono memorizzati sul System Spool Space.

Lo z/VM riesce a emulare questi tre tipi di device per gli utenti:

- **Reader** o RDR cioè lettore di schede - è un dispositivo di input delle macchine virtuali. Per ciascuna macchina virtuale è riservata un'area di Spool detta RDR in cui è possibile accodare file in input. Il formato del file deriva da quello originario delle schede perforate e cioè un insieme di record di 80 caratteri. L'uso principale che viene fatto attualmente del reader è la funzione di mailbox dei files ad esempio con li comando CMS SENDFILE si può trasferire un file da una macchina virtuale ad un'altra indirizzandolo alla coda di RDR della macchina ricevente. Rimane anche la funzionalità orginaria del Card Reader RDR di area di input per sottomettere i JOB con un linguaggio che viene interpretato dallo spool;

- **Punch** o PCH ovvero perforatore di schede - indica il dispositivo virtuale di output della macchina virtuale. Per ciascuna macchina

virtuale è riservata un'area di Spool detta PCH in cui è possibile accodare file in output. Nell' esempio precedente la macchina che esegue il comando CMS SENDFILE spedisce il file attraverso il PCH che lo indirizza nella coda di RDR della macchina ricevente;

- **Printer** o PTR cioè stampante - ovvero il dispositivo di stampa di un output. Per ciascuna macchina virtuale è riservata un'area di Spool detta PTR in cui è possibile accodare file in output in formato di stampa.

2.2.4 Gestione dispositivi di rete

Le tecniche di virtualizzazione di z/VM consentono implementazioni molto interessanti nell'ambito delle configurazioni di rete. Normalmente la connettività hardware in rete è garantita da una scheda OSA (Open System Adapter). Quindi sia z/VM che le macchine virtuali ospiti si possono presentare in rete utilizzando questo canale.

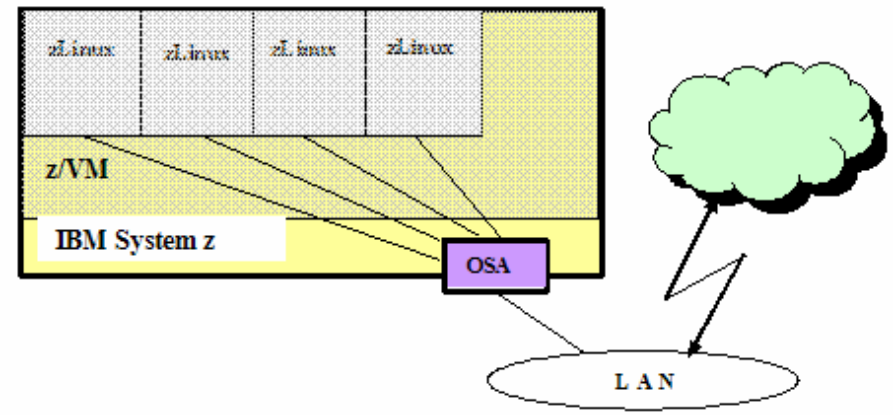


Figura 118 Ambiente z/VM e rete

La comunicazione tra macchine virtuali è fornita da vari dispositivi simulati o da funzionalità proprie di z/VM. Esistono percorsi di comunicazione tradizionali di z/VM quali InterUser Communication Vehicle (IUCV) o Virtual Channel to Channel. Oltre ad essi, z/VM utilizza la funzionalità QDIO Hardware facility e il QDIO CP subchannel per fornire due tipi di reti virtuali per la comunicazione tra macchine virtuali all'interno di z/VM: z/VM guest LAN e z/VM Virtual Switch.

2.2.4.1 z/VM Guest LAN

z/VM Guest LAN consente di creare delle reti virtuali di sistemi ospiti all'interno del sistema z/VM. Tramite questa funzionalità è possibile creare un numero indefinito di reti virtuali mediante definizioni contenute nel file di configurazione SYSTEM CONFIG o con il comando CP DEFINE LAN. Ogni z/VM Guest LAN è isolata dalle altre a meno che non vi sia un TCP/IP Router che le connetta. Ciò è vero anche nel caso che le z/VM Guest LAN debbano comunicare con reti esterne a z/VM. Il TCP/IP router è una macchina Virtuale CMS specializzata nell'indirizzamento del traffico TCP/IP. Gli Adapter utilizzati da zVM Guest LAN sono di tipo HiperSockets (nel caso di comunicazioni interne al Sistema Centrale) e OSA Express .Tali Adapter sono all'interno dello Stack TCP/IP delle Network Interface Card Virtuali. Ogni machina virtuale possiede una Network Interface Card Virtuale che è connessa alla Guest LAN. Essa fornisce l'accesso alle z/VM Guest LAN o ai Virtual Switch.

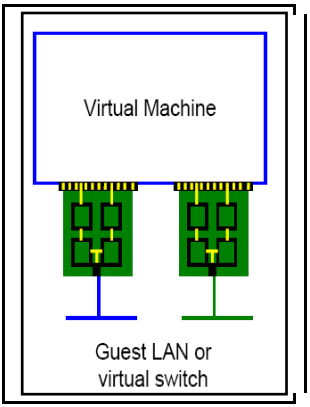


Figura 119 Virtual Network Interface Card (NIC)

Sebbene connessioni dedicate possono essere più performanti in presenza di un carico di rete intenso, si ha bisogno di una banda di rete garantita o quando è richiesto un accesso diretto alla rete fisica. Le z/VM Guest LAN risultano utili quando si hanno risorse hardware limitate, quando molti nodi sono all'interno dello stesso z/VM o quando occorre isolare un certo carico di rete (ad esempio ambienti di test) dalla rete primaria. Le Guest LAN possono utilizzare i protocolli Ethernet IPv4 e IPv6.

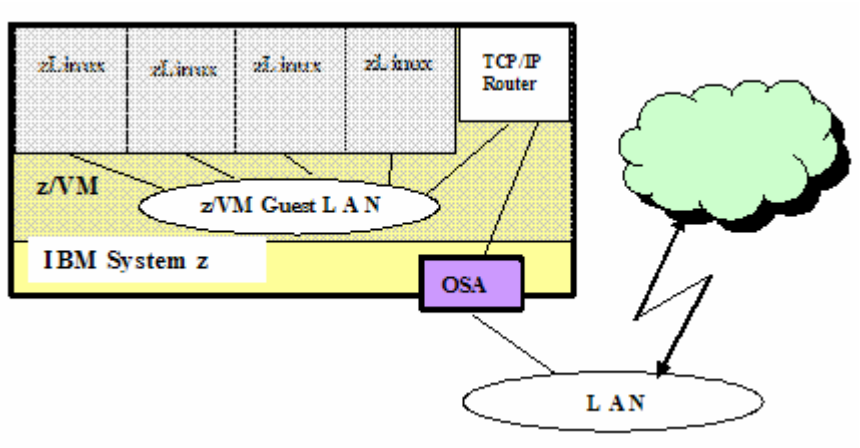


Figura 120 z/VM Guest LAN

2.2.4.2 z/VM Virtual Switch

Le Guest LAN interne a z/VM possono connettersi a reti esterne al sistema che ospita z/VM direttamente via adapter di tipo OSA-Express. In alternativa in ambiente z/VM si possono utilizzare gli z/VM Virtual Switch.
z/VM Virtual Switch fornisce una connettività tra un'ambiente costituito da LAN Esterna e Guest LAN. Simile ad una Guest LAN basata su interfaccia virtuale di rete di tipo OSA-Express, z/VM Virtual Switch opera solamente con adapter di tipo OSA-Express. Esso supporta il trasporto di pacchetti IP e di frame Ethernet. Gli z/VM Virtual Switch non richiedono un router per connettersi alla rete esterna.

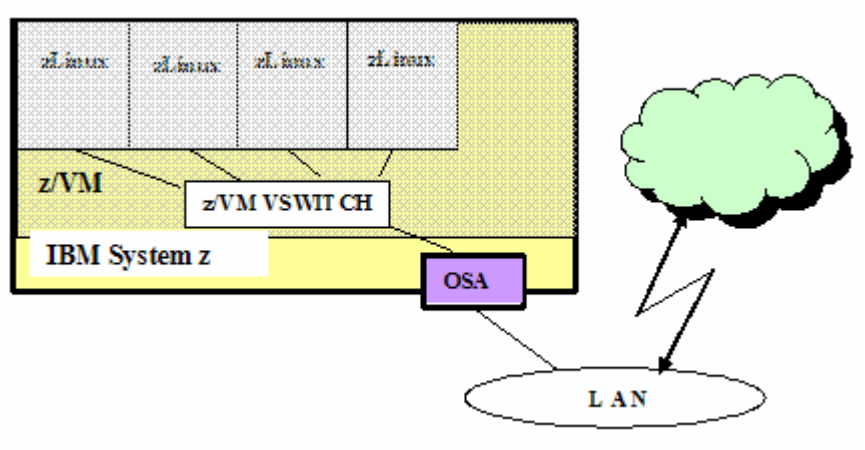


Figura 121 z/VM Virtual Switch

Si può a questo punto richiamare il concetto di Virtual LAN (VLAN) che è complementare alle configurazioni di z/VM Guest LAN e z/VM Virtual Switch. Lo standard IEEE 802.3 definisce una Virtual LAN (VLAN) ovvero la funzionalità che consente a reti fisiche di essere divise amministrativamente in reti logiche separate. Dal punto di vista operativo queste reti sono fisicamente indipendenti (separazione dei domini di collisione).
Esse non vanno confuse con le z/VM Guest LAN sebbene le z/VM Guest LAN e i Virtual Switch possono far parte di una VLAN. Tra i dispositivi simulati che supportano le VLAN c'è il Virtual Switch.
Relativamente al Virtual Switch lo standard IEE 802.1Q regola la gestione dell'assegnazione di utenti z/VM a specifiche VLAN e assicura che essi riceveranno solo i pacchetti appartenenti alle VLAN cui gli utenti sono autorizzati.
Nel caso di Guest LAN sarà necessario un router (una macchina virtuale specializzata) che connetta la rete interna ad una subnet esterna differente. Nel caso di Virtual Switch ciò non sarà necessario in quanto la subnet esterna sarà la stessa (con un supporto di transparent bridge).

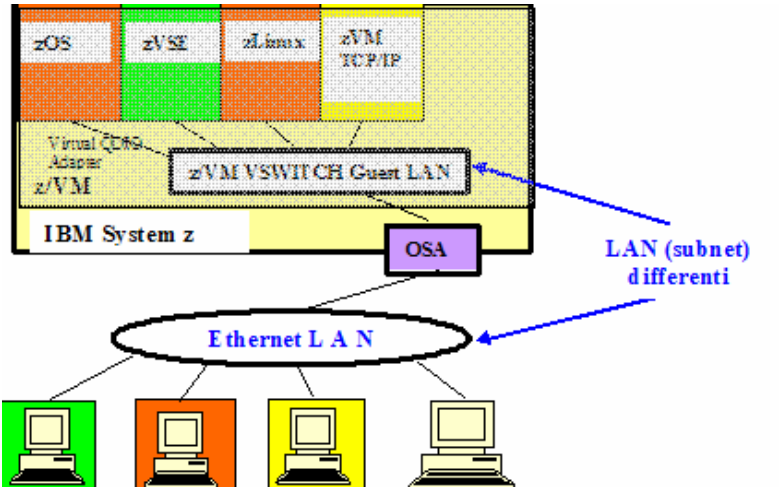


Figura 114 Configurazione Virtual Switch con VLA

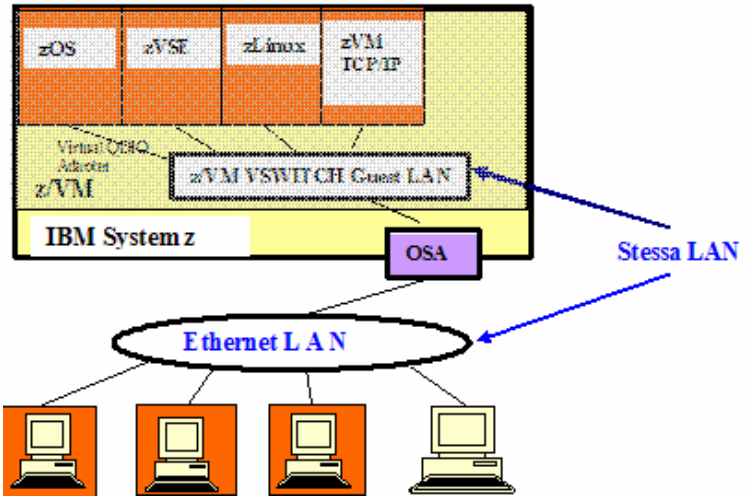


Figura 122 Configurazione Virtual Switch senza VLAN

2.2.5 Virtualizzazione di CPU e Memoria

La virtualizzazione del processore viene affrontata da z/VM con l'obiettivo di dare ad un sistema ospite "la convinzione" di avere il controllo esclusivo del processore. In effetti i processori sono condivisi tra i vari sistemi operativi ospiti.

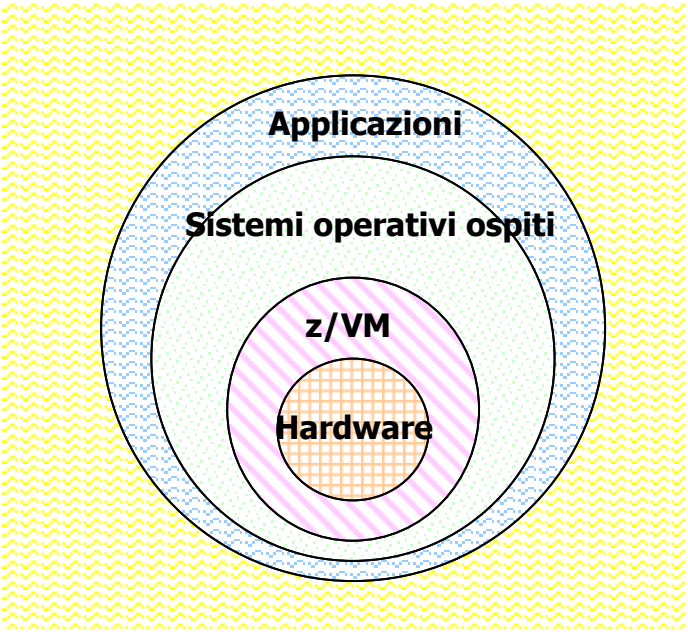


Figura 123 Livelli di Virtualizzazione

Andiamo a considerare dunque i virtual processor ovvero i CPU che la Virtual Machine ha a disposizione. Essi possono essere CPU *dedicati o condivisi*. Ovvero una macchina virtuale può ottenere la risorsa CPU in modo esclusivo (dedicato) nel tempo oppure può condividere questa risorsa con altre macchine virtuali. Ciò avviene in maniera dinamica nel tempo, all'interno del concetto di time slice (che rimane la base di

coesistenza delle macchine virtuali in z/VM). In tal senso anche i CPU virtuali dedicati sono utilizzati dinamicamente.

La condivisione di CPU può essere assoluta o relativa, vale a dire che i vari livelli di virtualizzazione consentono varie tecniche (per esempio hard e soft capping) che assegnano parti di risorsa CPU a seconda delle necessità. I vantaggi della condivisione della risorsa CPU possono consentire ad una macchina virtuale l'utilizzo di tutta la potenza elaborativa in mancanza di contesa con una o più macchine virtuali potenzialmente abilitate a quella stessa CPU.

Un altro vantaggio legato alla virtualizzazione e alla condivisione delle CPU virtuali è il cosiddetto "over commitment" ovvero la definizione di risorse virtuali che sono sovrabbondanti rispetto a quelle fisiche. Infatti se una macchina non riesce ad utilizzare tutta una sua risorsa, la parte da essa non utilizzata può essere disponibile ad altre risorse.

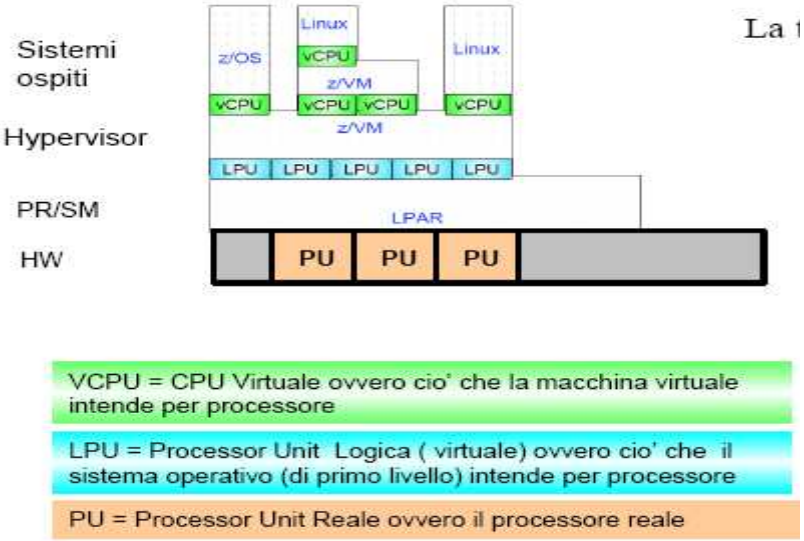


Figura 124 Virtualizzazione della CPU

La gestione della memoria virtuale di z/VM è quella legata alla architettura tipica dei Sistemi Centrali già descritta precedentemente. Tramite una funzione di traduzione degli indirizzi DAT (Dynamic Address Translation) z/VM può creare, gestire e isolare spazi di indirizzamento (address space). Ogni address space è costituito di un insieme di tabelle (region, segment, page tables) associate che contengono informazioni precise sulle locazioni di memoria reale che vengono usate dai processi. Queste tabelle sono usate dalla DAT per convertire indirizzi di memoria virtuale in indirizzi di memoria reale. Poiché queste tabelle sono matenute dal sistema operativo e non sono accessibili dai processi stessi, non è possibile per loro leggere e/o scrivere in memoria ciò che è usato dal sistema operativo o da un altro processo.

z/VM va oltre e utilizza questa capacità attraverso due livelli di traduzione di indirizzi. Un sistema operativo che opera in una macchina virtuale sotto

z/VM costruisce le sue tabelle di traduzione degli indirizzi come di consueto ai fini di isolare e contenere i suoi processi. L'intera memoria di questa macchina virtuale, sebbene vista da un sistema operativo ospite come memoria reale, è nei fatti una memoria virtuale definita da un insieme di tabelle gestite dal Control Program (CP) di z/VM.

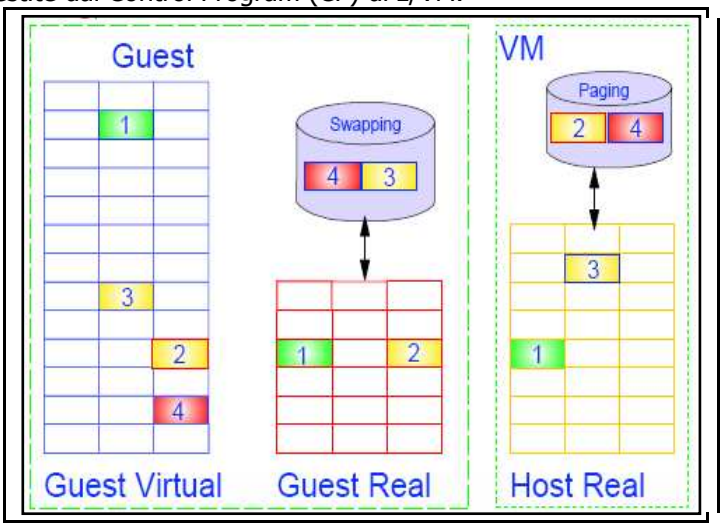


Figura 125 z/VM virtual memory

Esiste una funzionalità specifica di z/VM che pemette a più macchine virtuali di condividere memoria virtuale. Questa funzionalità è chiamata Shared Segments permette di ridurre drasticamente la quantità di pagine duplicate di codice e di dati.
Gli Shared Segments possono essere blocchi di memoria read-only o read-write. Ad esempio molte macchine virtuali Linux possono condividere il codice read-only del kernel. Accade che più di una segment table di macchine virtuali punterà alle stesse page frames in memoria reale.
z/VM esercita una protezione della memoria per non far alterare senza il suo assenso i segmenti salvati dichiarati in read-only. Esiste anche la possibilità di condividere segmenti salvati in read-write mode. z/VM permette ad una sola macchina virtuale di definire e salvare gli shared-segments.

2.2.6 Scheduling e Dispatching

z/VM gestisce le Macchine Virtuali e assegna loro la risorsa elaborativa utilizzando i componenti del sistema operativo denominati Dispatcher e Scheduler. Le Macchine Virtuali che divengono attive (al Logon) e che concorrono per ottenere la CPU vengono inserite e accodate in tre liste *dormant, eligible e dispatch list*. Quando si fa Logon allo z/VM, il CP crea un Blocco di controllo (control block1) che è un descrittore di un grande numero di informazioni della Macchina Virtuale. Tale blocco è denominato Virtual Machine Descriptor Block (VMDBK).

Lo scheduler si occupa di priorità all'interno delle liste e degli spostamenti dei VMDBK tra queste. Il dispatcher è quello che toglie gli utenti dalla dispatch list, permette loro di eseguire e di ritornare eventualmente nella Eligible list.

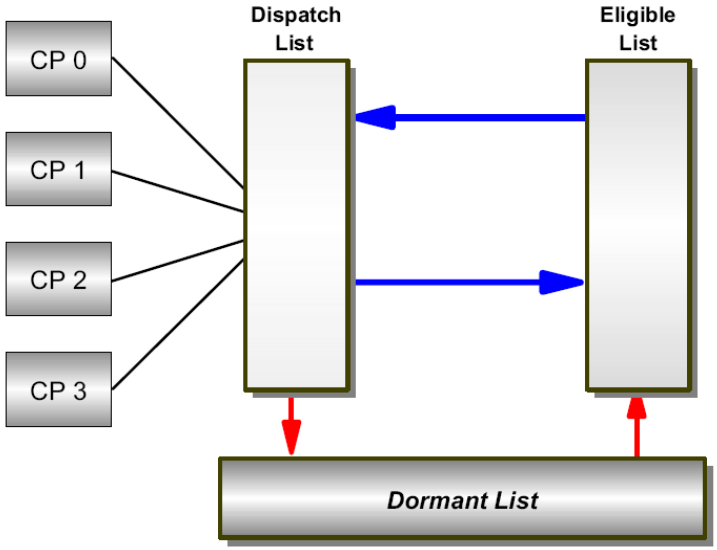


Figura 126 Liste di Dispatcher

Questo blocco sarà puntato dalle varie liste di scheduler/dispatcher che sono mostrate in figura. Se la macchina virtuale non sta facendo lavoro produttivo il CP la pone nella dormant list che indica che essa è "dormiente". Se per tale macchina si inizia a lavorare, ad esempio premendo un Enter Key dalla console, essa è posta in una lista che indica che essa riceverà delle risorse di sistema (eligible list).

Prima che sia data qualsiasi risorsa, però, la macchina virtuale è esaminata dallo scheduler. Lo scheduler guarda nel VMDBK della macchina virtuale per vedere quante risorse elaborative (Processori, memoria e paging) ha usato in passato.

Se si è alla fase di Logon non ci saranno informazioni accumulate nel VMDBK perciò lo scheduler assumerà o meglio caratterizzerà la macchina virtuale come se fosse un Utente interattivo anche detto utente di Classe 1.
Un utente di Classe 1 è normalmente un utente interattivo CMS o un sistema operativo ospite che esegue transazioni veloci.
Quando lo scheduler determina che ci sono risorse sufficienti a soddisfare la richiesta della Macchina Virtuale senza porre altri utenti in difficoltà, questa viene spostata in dispatch list dove attenderà il suo turno per accedere alla risorsa CPU per un certo intervallo di tempo (time slice).
Quando arriva il suo turno, la Macchina Virtuale verrà fatta eseguire su un processore disponibile. Il tempo riservato a tale macchina, conosciuto come dispatcher minor time slice, ad un certo punto si esaurirà.
Se la Macchina Virtuale in oggetto ha ancora lavoro da svolgere, essa viene mantenuta in dispatch list e al suo turno otterrà un altro time slice fino a che o avrà finito di lavorare (nel qual caso sara spostata in dormant list) o lo scheduler di nuovo stimerà la richiesta di risorse prima di rischedulare tale macchina virtuale.
Nel caso in cui la macchina ritorni molte volte in dispatch list, lo scheduler può decidere che essa non si caratterizza più come un utente interattivo e che ha bisogno di un quantità maggiore di risorsa elaborativa. Tale Macchina Virtuale viene allora innalzata a utente di Classe 2 che gli permetterà di rimanere in dispatch list per intervalli più lunghi che consentano di finire il lavoro .

Questo ha un vantaggio ulteriore perché riduce il numero di decisioni che il dispatcher deve prendere per gestire gli utenti della elegible list. Un utente di Classe 2 può essere tipicamente un programma di compilazione di un utente CMS o un Web Server zLinux.
Dopo che la Macchina Virtuale è ospitata in dispatch list come utente di Classe 2 molte volte e non ha ancora finito il suo lavoro, lo scheduler la innalzerà alla Classe 3 che le assegnerà un intervallo più lungo per finire il lavoro. Di solito un utente di Classe 3 sarà un sistema operativo (ospite) di produzione.
Ci sono degli svantaggi a entrare in Classi di dispatching più alte. Per esempio se le risorse di sistema (come paging o Memoria) diventano critiche, allora lo scheduler metterà gli utenti delle Classi più alte in eligible list e lascerà in dispatch list quelli delle classi più basse.
Però c'è un tipo di utente conosciuto come utente di Classe 0 che non sarà mai in eligible list (questo avrà l'opzione OPTION QUICKDSP nella sua entrata in USER DIRECT). Tipicamente questo tipo di utente è un utente speciale o un server di primaria importanza. La tabella che segue descrive le varie classi di dispatch list.

Classe di utente	Spiegazione
Classe 0	Classe di utenti che furono aggiunti in dispatch list senza ritardi in elegible list, senza tener conto della lunghezza del lavoro da svolgere
Classe 1	Classe di utenti che hanno appena iniziato una transazione e perciò si assume che stiano processando transazioni corte
Classe 2	Classe di utenti che non hanno terminato la transazione corrente nella loro prima pemanenza in dispatch list e perciò si assume che stiamo processando transazioni medie
Classe 3	Classe di utenti che non hanno terminato la transazione corrente nella loro seconda pemanenza in dispatch list e perciò si assume che stiamo processando transazioni di lunghe

Figura 127 Classi di utenti

2.2.7 z/VM Sicurezza ed Integrità (la SIE)

La virtualizzazione in z/VM è supportata da una funzione chiamata Interpretive Execution che consente una interazione del sistema operativo con l'hardware sottostante tale da consentire al contempo performance e integrità.

La z/Architecture fornisce la base della capacità del CP di mantenere l'integrità del sistema. Il CP si avvale di una funzionalità denominata Interpretive Execution Facility che permette ad un flusso di istruzioni di una Macchina Virtuale di essere eseguite su di un processore usando una singola istruzione chiamata Start Executive Execution (SIE) eseguita dal CP. Una Macchina Virtuale è attivata e le sue operazioni iniziano quando il dispatcher del CP esegue tale istruzione. La SIE ha come unico operando un puntamento ad un blocco di controllo in memoria chiamato State Description il quale specifica lo stato e l'architettura della Macchina Virtuale ove un programma chiamato Programma Ospite (guest program) deve essere eseguito (il programma ospitante che esegue la SIE è chiamato host program).

Il blocco di controllo State Description contiene anche informazioni come PSW (Program Status Word), registri CPU timer TOD clock, Interrupts pendenti, ecc. del programma da eseguire.

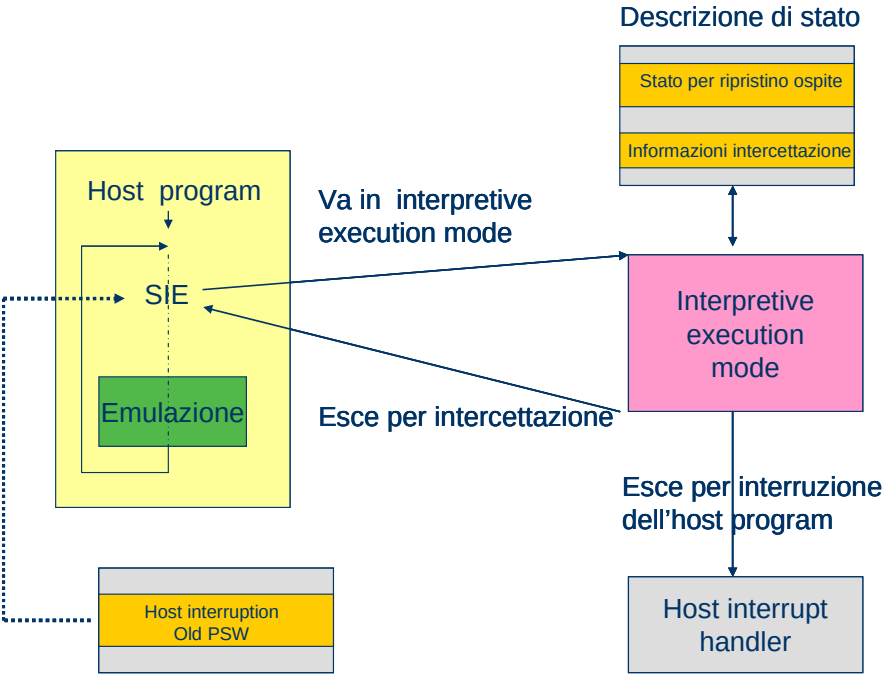


Figura 128 Start Interpretive Mode (SIE)

La maggior parte delle funzioni del Programma Ospite sono eseguite dalla Macchina Virtuale. Quando avviene una condizione che richiede l'intervento del CP si ha una intercettazione. La macchina lascia l'interpretive mode e trasferisce il controllo al CP.

Questa include alcune condizioni come l'esecuzione di alcune istruzioni privilegiate, in particolare istruzioni di I/O, gestione di certe interruzioni dei sistemi ospiti e del wait state della PSW ospite. La macchina lascia l'interpretive mode anche quando un external interrrupt o I/O interrupt causa l'interruzione del programma ospitante.
Quando la macchina lascia l'interpretive mode, il control block che descrive lo stato è aggiornato per riflettere lo stato corrente del Programma Ospite

cosicché in seguito l'esecuzione del Programma Ospite può essere ripristinata senza ulteriori cambiamenti alla descrizione dello stato.

Quando accade una intercettazione, l'informazione è inserita nella descrizione dello stato per facilitare l'identificazione e l'analisi delle condizioni dell'ospite da parte del programma ospitante.

Quindi le macchine virtuali possono operare in privileged mode ovvero eseguire istruzioni privilegiate senza che vengano intercettate dal CP, ad eccezione di istruzioni di I/O che vengono simulate/emulate dal CP stesso.

3.0 Linux nei Sistemi Centrali IBM

Linux è un sistema operativo il cui kernel è mantenuto da Linus Torvalds e il primo rilascio risale al 1991. Il codice del kernel è open source e, nel corso degli anni, Linux è evoluto diventando un sistema operativo ampliamente supportato da un crescente parco software. Oggi le distribuzioni Linux affiancano al kernel una varietà di tool e programmi di utilità forniti dalla comunità Open Source (ad esempio dal progetto GNU) fornendo un sistema completo.

Linux è caratterizzato da una forte indipendenza dalla piattaforma e viene eseguito su molte architetture, incluse Intel, Alpha, or Sparc. Linux per S/390 (a 31 bit) e il System z (a 64 bit) è un port di Linux nell'architettura S/390 e System z. Linux per S/390 e System z è un Linux "puro" dal punto di vista utente. Esso supporta l'architettura dei processori e i dispositivi che sono specifici degli ambienti mainframe.

IBM ha introdotto Linux sui Grandi Sistemi Centrali alla fine del 2000.

Si è trattato inizialmente di un esercizio accademico di una Università americana (Marist College) dove alcuni ricercatori hanno modificato il Kernel per supportare l'hardware dei grandi Sistemi scrivendo, ad esempio, i driver per gestire le interfacce di I/O, i dischi, i nastri magnetici e le schede di rete.
Col contributo di altri laboratori IBM Europei e della comunità open source l'esercizio è diventato presto un codice affidabile incluso nelle più comuni distribuzioni. Linux on System z è un'implementazione ASCII nativa, che pertanto entra a fare parte delle opzioni possibili per Linux al pari delle altre. IBM rilascia regolarmente sul sito di Developerworks http://www-128.ibm.com/developerworks/linux/linux390/ le patch da applicare ai sorgenti del kernel.

L'interesse del mercato per questa implementazione è dovuto alla capacità dei Grandi Sistemi di ospitare un numero elevato di sistemi Linux (in cui

generalmente viene ospitata un'applicazione o componente strutturale di un'applicazione, ad esempio il database) attraverso le tecniche di Partizionamento e Virtualizzazione disponibili su questi Sistemi. Tali tecniche permettono uno sfruttamente efficiente della capacità installata riducendo, ad esempio, in maniera sensibile i costi legati alle licenze software. Per i clienti che hanno significativi carichi tradizionali (CICS, IMS,..), il Linux on System z rappresenta un importante abilitatore tecnologico per i progetti di modernization in quanto permette un'efficace "esposizione" di tali asset verso gli altri sistemi (ad esempio in ottica Service Oriented Architecture).

3.1 Utilizzi di Linux in ambienti enterprise

In ambito aziendale, gli usi più frequenti di Linux sono:

- realizzare applicazioni distribuite aziendali
- realizzare soluzioni infrastrutturali
- consolidare i workload aziendali
- creare cluster ad alta capacità di calcolo

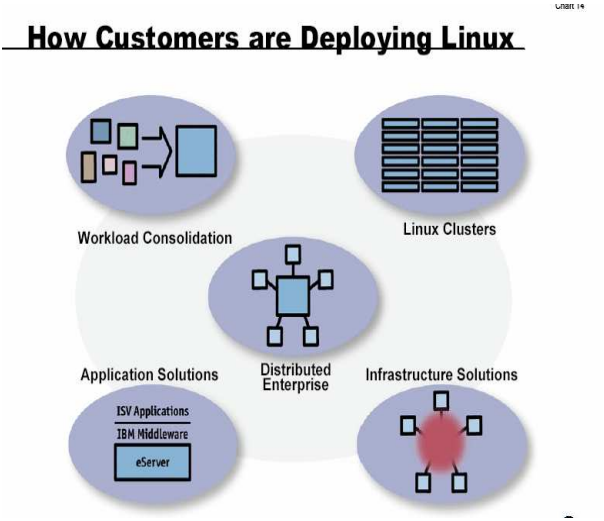


Figura 129 Alcuni utilizzi di Linux

In un'infrastruttura informatica aziendale i server Linux possono svolgere pressoché qualunque funzione, tra cui:

- Web/Application Server
- Database server
- File server e Print Server
- Content/Caching Server
- Security Server

3.1.1 Le applicazioni su z/Linux e il processo di porting

Il numero delle applicazioni portate su z/Linux cresce costantemente e molti dei prodotti middleware di IBM sono disponibili sia per Linux a 31 bit sia per Linux a 64 bit. Anche molti prodotti software sviluppati da importanti ISV, tra cui Oracle e SAP, sono stati portati su z/Linux.

Il Porting consiste principalmente nell'adattare un programma applicativo dall'ambiente operativo in cui fu sviluppato a un altro ambiente operativo in modo tale che esso possa essere utilizzato nel nuovo ambiente.
Portare applicazioni da un ambiente a un altro normalmente richiede considerevole sforzo e competenze specialistiche. Poiché le interfacce di programmazione delle applicazioni di Linux (API) sono molto vicine a quelle delle altre varietà di UNIX e se l'applicazione non richiede alcuna informazione del kernel, informazioni specifiche dell'architettura o estensioni particolari, essa può essere trattata direttamente come compatibile a livello di sorgente. In generale, questo è ciò che accade con la maggior parte del codice open source disponibile in ambiente GNU.
In questo caso la migrazione richiede solamente che l'applicazione sia trasferita sull'ambiente z/Linux e ricompilata. Generalmente, le applicazioni Web basate su Java possono essere portate facilmente a patto che sia presente sull'ambiente target una virtual machine del livello richiesto.

Ci sono varie considerazioni che devono essere affrontate durante un porting di una applicazione per Linux su System z:

- l'applicazione che contiene del codice specifico dell'architettura o specifico dell'assembler deve essere reimplementata per la z/Architecture. Gli opcode devono essere cambiati in opcode per z/Architecture o se il codice utilizza file con intestazione assembler, è necessario utilizzare una versione dell'intestazione per z/Architecture;
- il middleware, le librerie e i database usati dall'applicazione devono essere portati o prodotti specificamente per z/Linux;
- Il sistema z è di tipo big endian. Qualsiasi codice che processa dati orientati ai byte che hanno origine su un sistema little endian

potrebbero richiedere che alcuni dei bytes siano invertiti. Potrebbe essere necessario rigenerare i dati o, se questo non è possibile (per esempio, file condivisi), potrebbe essere necessario ritoccare l'applicazione in modo che processi dati in formato little endian;

- Su una migrazione che coinvolge le applicazioni Web, la maggior parte dei contenuti può essere direttamente trasportata senza alcun cambiamento su qualsiasi Web Server che girano su z/Linux;
- Se si utilizzano applicazioni web basate su Windows che utilizzano DLL implementate ad hoc, esse devono essere riscritte usando linguaggi di scripting compatibili (Perl, ad esempio) e poi portati su Linux per System z;
- Per applicazioni che usano Oracle e DB2 come backend, i database possono essere direttamente trasportati su Linux per System z. Nel caso che si utilizzino altri database (SQL server, ad esempio), è necessario una migrazione verso un RDBMS supportato.

Maggiori dettagli si possono trovare, ad esempio, nell'articolo Linux for S/390 and zSeries porting hints and tips[17].

[17] http://www-128.ibm.com/developerworks/eserver/articles/linux_s390/index.html

3.1.2 Linux sul Mainframe – Modi Operativi e Risorse Gestite

Linux su Mainframe può essere utilizzato nelle seguenti modalità :

- In una partizione Logica (LPAR) come Sistema Operativo Primario. Tutte le risorse dell'Hardware o della Partizione Logica sono viste e possedute da sistema operativo Linux.
- In una macchina virtuale definita e controllata da z/VM. z/VM presenta a Linux le risorse Virtuali come se fossero Reali. Generalmente le risorse hardware sono possedute e gestite da z/VM.

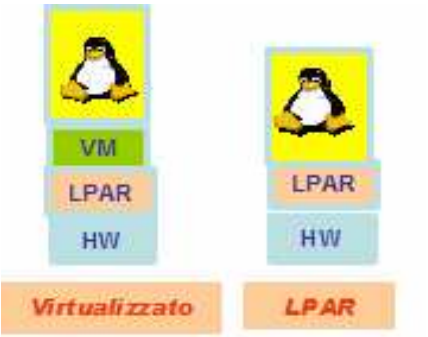


Figura 130 Modalità operative di Linux su mainframe

Linux fornisce metodi di accesso per :

- Dischi DASD tradizionali (formati ECKD)
- Dischi e nastri SCSI
- Schede Ethernet
- TCP/IP Interno (Hipersockets)

Inoltre esistono drivers addizionali per:

- Nastri tradizioni
- Schede Token Ring
- Protocolli di comunicazioni z/VM (CTCA / IUCV)
- Terminali 3270

I dischi (DASD) possono essere acceduti sia con il protocollo di I/O del mainframe (ad esempio FICON) sia con il protocollo SCSI su fibra (FCP). Attualmente il protocollo FCP fornisce, per alcuni workload, prestazioni migliori di quelle ottenibili con il protocollo FICON. Tuttavia con il protocollo mainframe si ottengono generalmente migliori livelli di affidabilità. Nel caso FCP il multipathing, ad esempio, è gestito a livello di sistema operativo, mentre nel caso FICON è integrato nativamente nell'architettura a canali (livello hardware).

In ambiente z/VM, un disco fisico (DASD) può essere suddiviso in più dischi logici (minidischi) assegnabili alle diverse immagini Linux virtuali. Ciascuna macchina virtuale potrà utilizzare il minidisco come se fosse un normale DASD.

Generalmente le risorse dedicate a LINUX vengono gestite su mainframe con uno speciale protocollo di tipo DMA detto Queued Direct I/O (QDIO). Grazie a questo protocollo la path length delle istruzioni di I/O viene sostanzialmente ridotta rispetto all'architettura a canali. Ad esempio, le schede OSA Express (interfacce di rete) possono essere configurate per usare il QDIO.

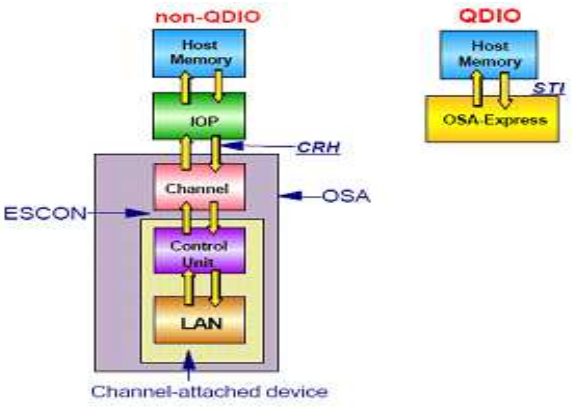


Figura 131 Modalità di accesso ai device

I dettagli relativi ai driver specifici dell'ambiente Linux su mainframe sono reperibili presso il seguente sito di documentazione ufficiale:

http://www.ibm.com/developerworks/linux/linux390/october2005_documentation.html

3.1.3 Uso di processori specializzati (IFL)

Per contenere i costi hardware e software su ambienti mainframe, è possibile utilizzare i processori specializzati IFL (Integrated Facility for Linux) dedicati all'esecuzione esclusiva di carichi di lavoro su z/Linux e che non incrementano la capacità installata per i carichi tradizionali.

Le partizioni logiche (LPAR) sulle quali è attivato Linux o z/VM con soli sistemi ospite di tipo LINUX possono utilizzare processori standard (CPU) o processori Specializzati di tipo IFL sia dedicati che condivisi. Non è consentito usare nella stessa partizione logica CPU ed IFL contemporaneamente.

La Figura 132 mostra una possibile configurazione con processori CP e IFL assegnati a partizioni con sistemi ospiti di tipo Linux.

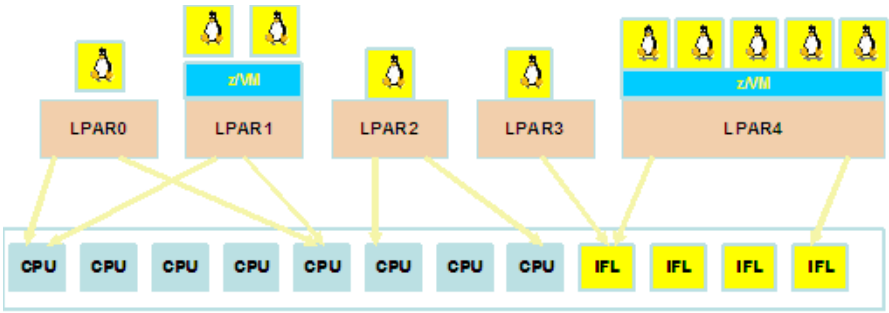


Figura 132 zLinux e processori mainframe

3.1.4 Networking

Linux on System z è caratterizzato da diverse opzioni di networking. E' possibile raggruppare le opzioni disponibili in due macro categorie:

- Tecnologie hardware native (che non richiedono lo zVM)
- Tecnologie basate su z/VM

Le prime sono basate su interfacce fisiche quali le schede OSA Express2 (in fibra o rame) o gli Hypersocket. Questi ultimi sono peculiari degli ambienti mainframe e costituiscono canali in memoria gestiti dall'hypervisor nativo (PRSM) per implementare LAN ad alte prestazioni (bassa latenza e alta banda) tra istanze Linux contenute in un unico sistema.

E' importante notare che in un ambiente mainframe le porte di rete fisiche sono condivisibili nativamente (anche senza z/VM) tra le diverse immagini Linux, ciascuna delle quali può presentarsi in rete con una propria identità pur condividendo lo stesso cavo.

Le opzioni di networking basate su z/VM sono riconducibili alle Guest Lan e alla variante VSWITCH (virtual switch). Le Guest Lan sono LAN virtuali, gestite dall' hypervisor software. Rispetto agli hypersockets, che possono essere definiti fino ad un massimo di istanze per sistema, le guest lan possono essere definite in numero maggiore in funzione delle risorse disponibili. Di contro sono caratterizzate da un maggior overhead per effetto dello strato software di implementazione.

Un VSWITCH è fondamentalmente una GUEST LAN a cui sono associate una o più interfacce di rete fisica (porte OSA Express). Ciò rende possibile integrare in maniera molto semplice segmenti di rete interni al mainframe (con guest costituiti da immagini zLinux virtuali) con un segmento di rete fisico esterno al mainframe.

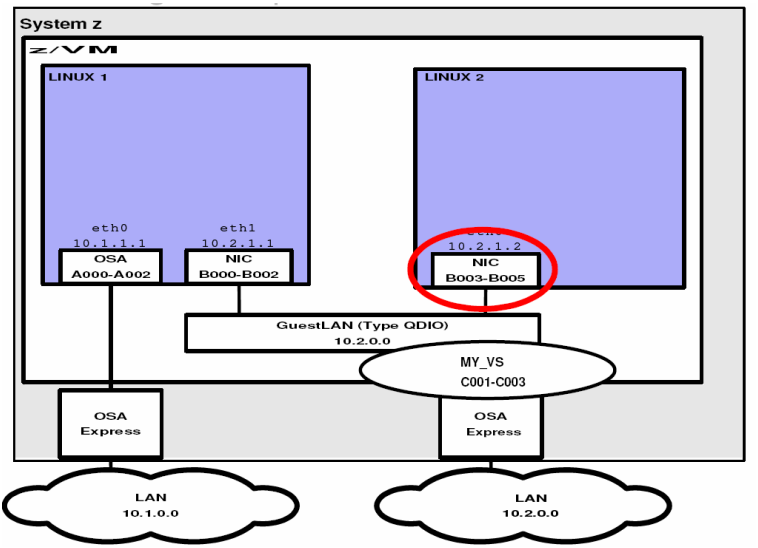


Figura 133 Connettività di Linux in z/VM

A partire dalle versioni più recenti di z/VM i VSWITCH:

- possono operare a livello 2 dello stack OSI, permettendo a ciascun guest Linux di avere un proprio MAC address;
- possono aggregare le porte fisiche al fine di incrementare la banda di comunicazione tra il segmento di rete interno e quello esterno (link aggregation support).

3.1.5 Linux nei Sistemi Virtualizzati con z/VM

La possibilità di ospitare diverse applicazioni, con i relativi livelli distinti dell'architettura, su un unico sistema scalabile contribuisce in maniera significativa a contenere molta della complessità legata al modello distribuito.

La virtualizzazione si è imposta in questi ultimi anni come una delle strategie tecnologiche più promettenti per ottenere tali obiettivi di efficienza e in questo scenario si colloca l'utilizzo di zLinux e z/VM su ambiente mainframe.

In particolare, utilizzare sistemi Linux virtualizzati con z/VM comporta i seguenti vantaggi:

- completa condivisione di risorse tra molte immagini di LINUX sotto il controllo di un unico z/VM e conseguente "over commitment" delle risorse stesse;
- possibilità di Consolidare molti "Serventi" Linux sullo stesso sistema;
- le immagini Linux possono trarre vantaggio dai Servizi offerti da z/VM e quindi utilizzzare a pieno le caratteristiche della z/Architecture;
- comunicazione efficiente attraverso le immagini grazie alle connessioni interne disponibile nella stessa macchina fisica (ad esempio hypersocket, guest lan);
- i dischi gestiti da z/VM possono essere suddivisi in Minidischi e condivisi tra le varie immagini Linux;
- z/VM svolge una serie di funzioni avanzate tipiche della z/Architecture che non sarebbero disponibili sui Linux nativi (ad esempio incremento a caldo del numero di processori utilizzati dall'hypervisor);
- z/VM offre la possibilità di controllare e gestire le istanze da un unico punto.

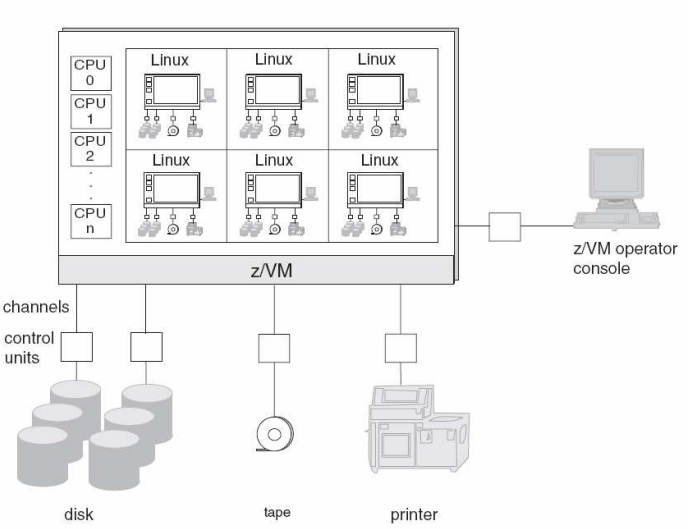


Figura 134 Guest Linux di un sistema z/VM

L'interazione tra z/VM e zLinux è stata progressivamente incrementata al fine di aumentare l'efficienza nella condivisione delle risorse hardware. Tra le tecnologie abilitanti sviluppate per implementare tale sinergia sono da segnalare:

- **Collaborative Memory Management Assist (CMMA)**, tramite il quale z/VM e i guest zLinux scambiano informazioni al fine di ottimizzare l'utilizzo complessivo della memoria del sistema.
- **Virtual Machine Resource Manager (VMRM)**, che permette di controllare l'assegnazione delle risorse in contesa (ad esempio memoria fisica) tra gruppi di guest virtuali in funzioni della loro priorità.

3.1.6 Distribuzioni e installazione

Le distribuzioni disponibili su piattaforma mainframe sono sia di tipo commerciale (Novell SuSE e Red Hat) sia non commerciale (CentOS, Debian, ecc.). I distributori commerciali offrono servizi di supporto del sistema.

Anche IBM offre suoi servizi di supporto[18].

[18] http://www-03.ibm.com/systems/z/os/linux/support_documentation.html

3.1.7 Il processo di installazione

La maggior parte del processo di installazione di zLinux è simile a quello di Linux su altra piattaforma e dipende dalla specifica distribuzione utilizzata. Le differenze principali risiedono nel processo di avvio e nella configurazione dei dispositivi hardware specifici dei mainframe. Linux può essere installato direttamente su partizione LPAR o come ospite di z/VM (opzione più comune).

Per l'installazione è possibile consultare, oltre la documentazione della distribuzione scelta, i seguenti redbook, che forniscono esempi e "ricette" preconfezionate per l'installazione e l'utilizzo di z/Linux su macchine virtuali:

- IBM z/VM and Linux on IBM System z: Virtualization Cookbook for Red Hat Enterprise Linux 4 http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sg247272.html?Open
- z/VM and Linux on IBM System z: The Virtualization Cookbook for SLES 10 SP2 http://www.redbooks.ibm.com/redpieces/abstracts/sg247493.html?Open

In sintesi, nel caso di installazione su LPAR nativa, l'avvio del processo avviene mediante IPL dal lettore DVD dello HMC. Nel caso di installazione su zVM occorre copiare (ad esempio via FTP) i tre file per l'IPL (kernel, parameter file, ramdisk) su un guest zVM del sistema (ad esempio CMS), spedire i file alla macchina virtuale target mediante virtual card writer[19] ed effettuare un avvio (IPL) da virtual card reader dalla macchina virtuale target. Gli step successivi per entrambe le modalità sono uguali e prevedono l'utilizzo di una sorgente di installazione in rete (server ftp, nfs, ecc.) che esponga i pacchetti di installazione (contenuto del DVD).

[19] I virtual card writer e reader sono dispositivi logici associabili a ciascun guest z/VM e sono utilizzati principalmente per la comunicazione tra guest.

3.1.8 La clonazione di Sistemi Linux sotto z/VM

Per clonazione di un Sistema Linux si intende il complesso di operazioni che consentono in poco tempo la creazione di una nuova immagine di Sistema Operativo Linux a partire da una "Copia Principale" esistente.

La clonazione è possibile in quanto il Sistema Clone si differenzierà dalla copia principale ('Golden Image') solo per alcune configurazioni quali l'Hostname, gli indirizzi IP, le password, ecc.

Tramite la clonazione è possibile rendere velocemente disponibili nuove immagini di Sistema in un ambiente mainframe senza dover procedere a complessi passi di installazione.

Le operazioni tipiche da eseguire sono:

1. Installare Linux per la prima volta, ad esempio usando un CD
2. Creare un LINUX Template (Golden Image) dalla quale clonare le altre; la golden image deve rappresentare un sistema minimo avviabile.
3. Copiare la Golden Image per creare la nuova Immagine (ad esempio effettuando copie fisiche dei DASD).
4. Modificare gli opportuni file di configurazione del clone (hostname, indirizzo IP, ecc.).
5. Modificare opportunamente le definzioni di z/VM per poter ospitare il Clone

z/VM facilita le operazioni di creazione di Cloni in quanto mette a disposizione una serie di programmi di utilità per copiare dati all'interno e tra macchine virtuali, creare automaticamente nuove macchine virtuali, avviarle e disattivarle automaticamente.

Su un Sistema z/VM l'operazione di clonazione, compreso il riavvio, se opportunamente programmata può richiedere pochi secondi.

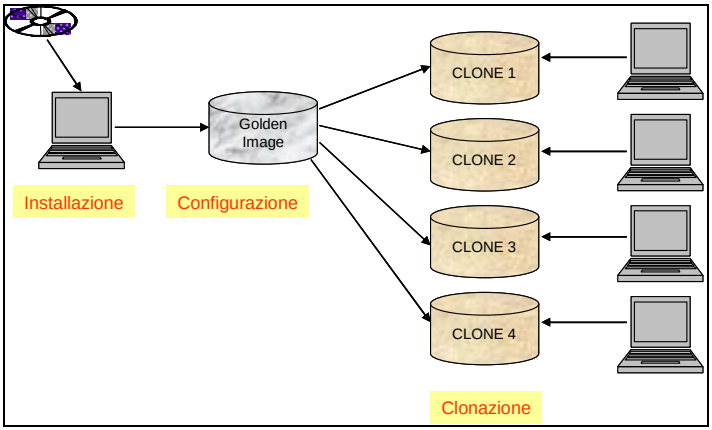


Figura 135 Clonazione di sistemi zLinux

E' anche possibile clonare sistemi zLinux installati su LPAR nativa
http://www.redbooks.ibm.com/abstracts/redp3871.html

4.0 La selezione della Piattaforma Informatica e l'Ottimizzazione della Infrastruttura

In questo capitolo affronteremo il tema della Selezione della Piattaforma Informatica: con questo termine si suole indicare il processo volto a stabilire quale sia la Piattaforma Informatica "migliore" per eseguire una data Applicazione Informatica. Nel fare ciò tenteremo un approccio rigoroso e quantitativo basato sulle caratteristiche non funzionali e vedremo in particolare come alcune peculiarità dei Sistemi Centrali ben si adattino al raggiungimento della "Ottimizzazione della Infrastruttura Informatica" che, alla fine, rappresenta il vero obiettivo della selezione.

In particolare ci riferiremo alla determinazione e al confronto del costo totale di una infrastruttura (TCO) usato come parametro per determinarne il grado di efficienza, riconducendo a tale valore tutte le caratteristiche "non funzionali" relative.

4.0.0 Introduzione e definizioni

Definiamo **Piattaforma Informatica** la coppia ordinata di un ***Sistema Operativo*** e di una ***Architettura Hardware***. Una tale definizione ci permette di identificare con precisione la coppia composta da Sistema Operativo ed Architettura Hardware stante il fatto che a volte lo stesso Sistema Operativo (o quanto meno Sistemi Operativi con lo stesso nome e le stesse Funzionalità) possono funzionare su Architetture Hardware diverse, ovvero che la stessa Architettura Hardware può ospitare sistemi operativi diversi. Il concetto di Piattaforma Informatica risulta particolarmente utile tutte le volte nelle quali si vogliano identificare e

confrontare ambienti diversi in grado di ospitare un'Applicazione Informatica. Con questa definizione un'Applicazione Informatica potrà essere eseguita (hosted) su diverse Piattaforme Informatiche e quindi, a parità di Applicazione avrà un senso il confronto e la selezione tra le Piattaforme Informatiche possibili. La figura seguente rappresenta il concetto di Piattaforma Informatica per come qui definito: come si diceva l'Architettura Hardware denominata z/Architecture potrà ospitare Sistemi Operativi diversi, in questo caso z/OS e Linux. Nella denominazione della Piattaforma Informatica si conviene di mettere al primo posto il nome del Sistema Operativo seguito da quello dell'Architettura Hardware: ci sarà così ad esempio la Piattaforma LINUX/x86 ovvero LINUX/zArchitecture ecc.

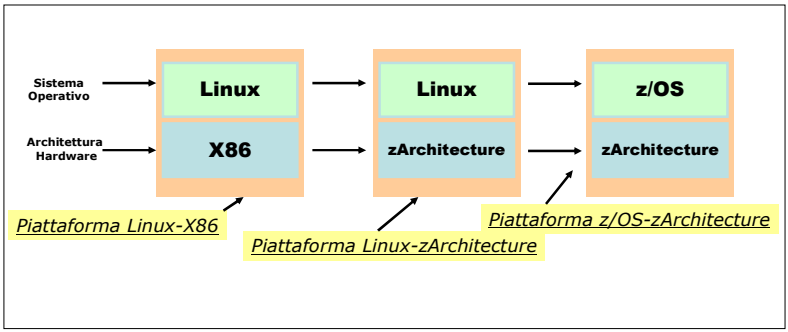


Figura 136 Piattaforma Informatica

Esempi di Sistemi Operativi sono:

- MS Windows
- Linux
- z/OS
- HP/UX
- Solaris
- IBM AIX
- ....

Esempi di Architetture Hardware sono:

- x86
- Apple

- z/Architecture
- PA-RISC
- Sun Sparc
- Power IBM
- ....

Ripetiamo che l'Architettura Hardware da sola o il Sistema Operativo da solo non bastano a definire la Piattaforma Informatica, poichè sulla medesima Architettura Hardware possono funzionare diversi Sistemi Operativi o viceversa lo stesso Sistema Operativo potrebbe essere ospitato su diverse Architetture Hardware.

Osserviamo tuttavia che alcuni Sistemi Operativi possono essere attivati solo su una Architettura Hardware (ad esempio il Sistema Operativo z/OS può funzionare solo su macchine di z/Architecture), parimenti non tutte le combinazioni (coppie ordinate) di un Sistema Operativo ed una Architettura Hardware danno luogo ad una Piattaforma Informatica esistente (ad esempio la Piattaforma Informatica MS Windows/zArchitecture NON ESISTE).

Associamo ora la definizione di Piattaforma Informatica al concetto di Applicazione Informatica.

Diremo **Applicazione Informatica** un insieme di manufatti Software (Programmi, Procedure, Dati) in grado di svolgere un lavoro utile attraverso un Sistema di elaborazione. Una applicazione informatica è l'insieme dei programmi che gestiscono i Conti Correnti di una Banca, le Paghe di una Azienda, le prenotazioni di una Aerolinea, la Segreteria Studenti della Università, ma anche i complessi calcoli per la ricerca petrolifera, per i modelli di simulazione o per un problema matematico di grande complessità.

Una Applicazione Informatica può essere eseguita su una o più piattaforme informatiche differenti.

Una Applicazione Informatica si dirà Tipica o caratteristica di una sola Piattaforma Informatica, quando essa può essere eseguita su una ed

una sola Piattaforma Informatica: in tale caso essa viene comunemente detta "Legata" a quella Piattaforma Informatica [Legacy].
Una Applicazione Informatica si dirà portabile attraverso una lista (insieme) di Piattaforme Informatiche nel caso in cui essa possa essere eseguita su una serie di diverse Piattaforme Informatiche [Open].
Non è detto che una Applicazione Informatica portabile possa essere eseguita su tutte le Piattaforme Informatiche possibili.
Per meglio comprendere quest'ultima affermazione occorre analizzare quali sono gli elementi che realizzano la portabilità della Applicazione Informatica attraverso le diverse Piattaforme Informatiche.
Un elemento certamente essenziale può essere rappresentato dal linguaggio di programmazione utilizzato, o dal tipo di codifica dei dati usato o dalla presenza di un Sottosistema (Middleware) che si interpone tra la Applicazione Informatica ed il Sistema Operativo.

In base a questi elementi avremo quindi:

- Applicazioni Portabili a livello di Codice Sorgente - si tratta delle applicazioni scritte nello stesso linguaggio di Programmazione (ad esempio Java) le quali sono portabili a meno di ricompilazione o reinterpretazione.
- Applicazioni Portabili a livello di Codice Eseguibile (portabilità binaria) nel caso in cui attraverso la presenza di un Sottosistema, di un MiddleWare o di un emulatore lo stesso Codice Compilato potrà essere utilizzato sotto Sistemi Operativi diversi; in questo caso il processo di adattamento tra il Codice Compilato ed il Sistema Operativo sara effettuato dal Middleware o dall'emulatore.

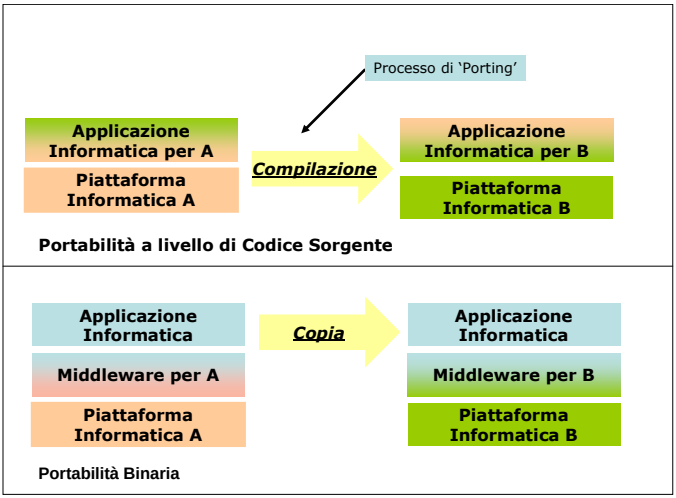


Figura 137 Portabilità dell'Applicazione Informatica

Da quanto detto si intuisce facilmente che la condizione di portabilità di una Applicazione Informatica tra diverse piattaforme informatiche dipende innanzitutto dalla portabilità dell'Applicazione Informatica attraverso i Sistemi Operativi e solo in minima parte dall'Architettura Hardware. Quest'ultima è usualmente "schermata" dal Sistema Operativo ed interviene solo su elementi molto profondi della applicazione come ad esempio il formato di rappresentazione dei dati.

Abbiamo visto come la presenza di uno strato intermedio (MiddleWare) possa facilitare la portabilità dell'Applicazione Informatica attraverso diversi Sistemi Operativi e quindi diverse Piattaforme Informatiche, vedremo ora che un ruolo ugualmente importante può essere rivestito dai cosidetti Standard, in particolare dall'insieme di regole che, scritte da Organizzazioni internazionali o divenute tali "de facto", governano alcuni processi della elaborazione come ad esempio le comunicazioni, l'organizzazione dei dati, la memorizzazione degli stessi, le tecniche di visualizzazione.

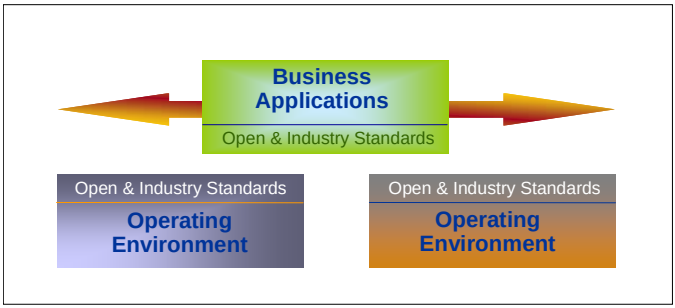


Figura 138 Il ruolo degli Standard

Anche gli Standard costituiscono un elemento utile alla portabilità delle Applicazioni attraverso i Sistemi Operativi e le Piattaforme: gli standard de iure o de facto, come ad esempio TCP/IP, XML, Java, J2EE, si aggiungono ai Middleware (Apache, WebSphere, TomCat, ecc.) nel facilitare la possibilità di esecuzione (deploy) dell'Applicazione Informatica su diverse Piattaforme con lo stesso risultato funzionale e senza modifiche al codice.

L'ambiente costituito dal Sistema Operativo + Architettura Hardware qui definito come "Piattaforma Informatica", unito all'insieme degli strumenti di ausilio alla gestione operativa e Sistemistica prende il nome di Ambiente Operativo (Operating Environment).

Grazie alla portabilità ha quindi senso definire un processo di selezione della Piattaforma Informatica, detto per l'appunto "Platform Selection", in quanto la stessa Applicazione Informatica, con modifiche piccole o nulle al codice, potrà essere eseguita sotto diversi Sistemi Operativi, o più frequentemente sotto lo stesso Sistema Operativo e diverse Architetture Hardware con diverse caratteristiche prestazionali o di costo. La platform selection implicitamente definisce anche il relativo Operating Environment e quindi più in generale le caratteristiche di gestibilità complessiva della Applicazione Informatica.

4.0.1 La Selezione della Piattaforma Informatica [Platform Selection]

Definiamo **Selezione della Piattaforma Informatica** (Platform Selection) il processo atto ad individuare la Piattaforma Informatica più idonea ad eseguire una data Applicazione Informatica in relazione ad una serie di caratteristiche non funzionali assegnate (criteri o drivers).
Criteri non funzionali opportuni per un processo di Platform Selection potranno essere ad esempio, le prestazioni, le caratteristiche di gestibilità operative, i costi di esercizio, ecc.
E' bene precisare che stiamo parlando in ogni caso di variazione dei requisiti non funzionali della Applicazione Informatica (Prestazioni, Costi, Gestibilità, ecc.) poiché solo questo tipo di requisito può essere oggetto di discussione, i requisiti funzionali, al contrario, essendo legati alle funzioni che l'applicazione deve svolgere, non possono essere in alcun modo oggetto di discussione o selezione: essi sono parte integrante della Applicazione stessa.
La Platform Selection ha senso ed è possible solo in funzione del fatto che la portabilità dell'Applicazione possa essere fattibile. Non sempre l'applicazione risulta "portabile" in quanto a volte essa presenta legami stretti con la Piattaforma sulla quale è stata costruita e viene eseguita, ovvero usa funzioni non standard tipiche di una certa Piattaforma e non presenti sulle altre. Questo tipo di legami rende l'Applicazione eseguibile solo su una o più Piattaforme e non su tutte. In questo caso si suole definire l'Applicazione come "legacy".
Il processo di Platform Selection può essere quindi definito ed eseguito solo per applicazioni portabili o non "legacy".
Il processo di Platform Selection viene generalmente eseguito "a posteriori" su Applicazioni Informatiche già esistenti ed in uso con lo scopo di ottimizzarne l'Operating Environment.
Ciò non toglie che tale processo possa essere eseguito "a priori" ovvero in fase di progettazione e rilascio di una nuova Applicazione. In questo caso esso assumerà il ruolo di analisi preventiva allo scopo di determinare "a priori" quale possa essere l'Operating Environment più opportuno per eseguire l'applicazione una volta completata. Sempre più spesso questo secondo modo di operare "a priori" si sta diffondendo presso gli

sviluppatori ed i progettisti, fornendo tra l'altro l'opportunità di selezionare gli strumenti di sviluppo ed i MiddleWare usati rispetto ai criteri di portabilità e standard piuttosto che la conoscenza o popolarità degli strumenti stessi.
Ci aspettiamo quindi nei prossimi anni la nascita di Applicazioni Informatiche sempre più portabili (open) e sempre meno "legacy" e quindi un incremento dei processi di Platform Selection sia "a priori" che "a posteriori".
Osserviamo infine che spesso, preso atto della non portabilità di una data Applicazione e comunque nell'intento di migliorarne l'Operating Environment in base a criteri definiti (ad esempio ridurne i costi), vengono valutati ed avviati processi a volte molto complessi di trasformazione volti a ridefinire o riscrivere l'Applicazione usando strumenti che ne garantiscano la portabilità. Tali processi, di cui più avanti ne analizzeremo specificamente uno, prendono in generale il nome di "Legacy Transformation".
Non è detto che un processo migratorio su vasta scala volto a riportare una Applicazione su standard di portabilià risulti comunque conveniente ed opportuno: per questo è necessario disporre di metodologie e criteri valutativi di tipo oggettivo; di tali criteri e delle relative metodologie ci occuperemo dettagliatamente nei prossimi capitoli.
Anche sul tema della "Legacy Transformation" ci aspettiamo grandi interventi negli anni a venire, per tale ragione e allo scopo di valutare tali processi in maniera quantitativa e asettica ci servono strumenti valutativi obiettivi del processo.

Completiamo la trattazione della Platform Selection precisando meglio le definizioni di Caratteristiche Funzionali e Non Funzionali di una Applicazione Informatica.
Definiamo caratteristiche o **Criteri Funzionali** di una Applicazione Informatica tutto ciò che è relativo al funzionamento della applicazione, ovvero il fatto che l'applicazione faccia le cose per le quali è stata pensata e realizzata, ovvero che funzioni correttamente. Ad esempio, se una Applicazione Informatica deve stampare ogni giorno le fatture prodotte da una azienda per essere spedite ai clienti, un criterio funzionale per l'Applicazione è che sia in grado di "stampare", la capacità di stampare è quindi un requisito funzionale senza il quale l'applicazione di Stampa delle fatture NON FUNZIONA !

Come già accennato prima, ripetiamo che i criteri funzionali non possono essere negoziati in quanto essi sono un prerequisito per l'Applicazione, quindi tutte le Piattaforme Informatiche che non posseggono i requisiti funzionali per l'Applicazione devono essere escluse dal processo di Selezione: non ha senso infatti scegliere o prendere in esame Piattaforme sulle quali l'applicazione non può funzionare; ad esempio se la nostra applicazione richiede funzioni di grafica avanzata per elaborare immagini, non ha senso selezionare per essa Sistemi Operativi che non sono in grado di supportare la grafica.
Definiamo invece caratteristiche o **Criteri Non Funzionali** di una Applicazione Informatica tutti quegli elementi che non attengano esplicitamente al funzionamento della stessa, ma possono solo variarne alcuni aspetti rendendolo più veloce, meno costoso o più gestibile, in poche parole migliorarne o peggiorarne il funzionamento.
Sono tipicamente Criteri Non Funzionali:

- Rapporto Costo/Prestazioni
- Costi di Esercizio/Gestione
- Sicurezza
- Caratteristiche di Gestibilità
- Capacità di Ricoverare disastri
- Continuità di Servizio
- Reattività alle variazioni di carico

Ribadiamo che le caratteristiche Non Funzionali sono la base per il processo di Platform Selection.
Nei processi valutativi della Piattaforma Informatica, assumono spesso valore anche criteri, diversi da quelli sopra elencati, ad esempio una Azienda/Ente potrà decidere di eseguire una Applicazione Informatica su una data Piattaforma Informatica indipendentemente dalla valutazione di criteri non funzionali, basandosi invece sulla presenza in azienda di molte altre applicazioni sulla stessa Piattaforma e ciò allo scopo di ridurre il numero di Piattaforme diverse presenti in Azienda.

La Platform Selection quindi può e probabilmente deve essere mediata da valutazioni Strategiche ed Organizzative complessive. Infine tra gli elementi

Non Funzionali più importanti e quindi determinanti ai fini della Platform Selection possiamo annoverare:

1. I costi di esercizio dell'Applicazione Informatica in relazione alla Piattaforma Informatica analizzata.
2. L'ambiente operativo (Operating Environment) tipico della Piattaforma Informatica analizzata.
3. La presenza in Azienda/Ente di personale con esperienza specifica (Skills) su quella data Piattaforma.
4. Le prestazioni specifiche della Applicazione su quella piattaforma.
5. I Costi di Acquisizione di Hardware e Software

Come si è detto in precedenza, il processo di Platform Selection può riguardare un'Applicazione Informatica già esistente e portabile o una nuova applicazione. Avevamo già accennato al fatto che nel caso di applicazione esistente ed ove questa presentasse legami (legacy) con la Piattaforma Informatica, segnatamente col Sistema Operativo, si può valutare un processo di "legacy transformation" per rendere l'applicazione più portabile e più aderente a standard.
Nel caso in cui un'applicazione esistente e portabile venga trasportata su una diversa Piattaforma Informatica solo in base a criteri non funzionali (tipicamente una riduzione dei costi operativi) si suole parlare di processo di "re-hosting".
Il **re-hosting** è quindi un processo di porting di una Applicazione Informatica da una Piattaforma Informatica a un'altra allo scopo di migliorarne alcuni aspetti non funzionali.
Nella realtà i processi di re-hosting tendono ad essere eseguiti minimizzando l'impatto sul codice: ciò può essere reso possibile grazie ad opportuni strumenti o alla presenza di middleware. La valutazione di un processo di re-hosting è materia sufficientemente complessa: spesso questo processo viene proposto semplicemente sulla base di una evidente riduzione dei Costi Operativi di gestione della Piattaforma Informatica, senza dare molto peso agli aspetti di gestione dell'ambiente finale o ai costi di acquisizione e migrazione.
Per questo motivo e al fine di definire in maniera rigorosa i criteri per una Platform Selection, si rendono necessari, e quindi li proporremo nel seguito,

alcuni elementi quantitativi in grado di fornire valutazioni oggettive al processo.
Questi elementi quantitativi sono:

1. Una Metodologia di Analisi che fornisca i Parametri in base ai quali valutare i Criteri Non Funzionali, ovvero definire quali siano tra essi quelli maggiormente rilevanti.
2. Un modello rigoroso di Analisi dei dati forniti dalla metodologia, in grado di dare un valore a ciascun elemento non funzionale.
3. Un processo accurato, anche se qualitativo, di confronto delle caratteristiche non funzionali delle varie Piattaforme Informatiche; ad esempio dobbiamo considerare di maggior valore le prestazioni o l'affidabilità.
4. Un criterio preciso ed il più possibile oggettivo per stimare le potenze degli elaboratori nelle diverse piattaforme informatiche.
5. Una serie di altri criteri ottimali per valutare tutto ciò che non è numericamente quantificabile (Best Practices). Ad esempio quante persone sono necessarie per gestire una data Piattaforma rispetto ad un'altra.

4.0.2 Ottimizzazione della Infrastruttura Informatica [ITRO]

Definiamo ***Infrastruttura Informatica*** l'insieme degli apparati Hardware (Serventi, Unità a Disco, Unità a nastro, Stampanti, ecc.), del Software, delle linee, le strutture di Rete, tutte le altre interconnessioni necessarie e il Personale, utili per generare e gestire tutte le Piattaforme Informatiche necessarie ad eseguire una Applicazione.

Più in generale ***l'Infrastruttura Informatica è l'ambiente di esecuzione di una o più Applicazioni Informatiche.***

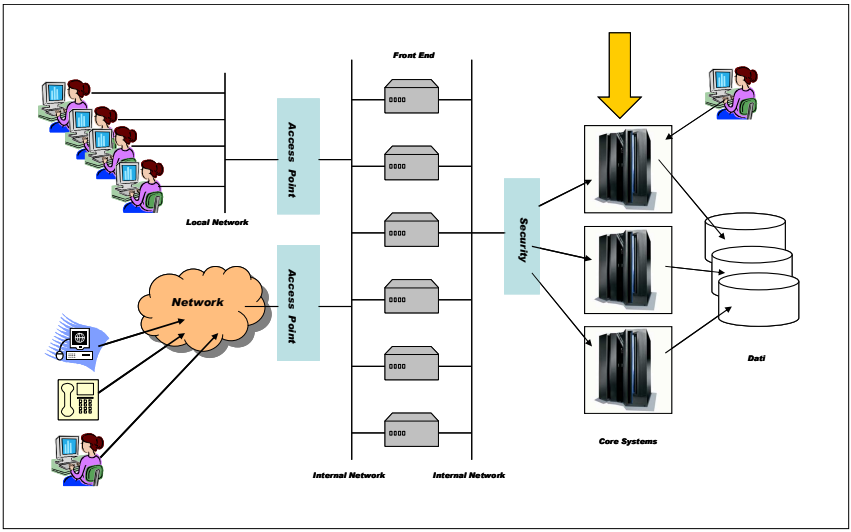


Figura 139 Infrastruttura Informatica

Fanno quindi parte della Infrastruttura Informatica:

- I Serventi (Elaboratori Elettronici) con il loro Software di Base (Sistemi Operativi), Software di Ambiente e Middleware.
- Gli apparati di Memorizzazione su Nastro o su Disco Magnetico o altri supporti (CD, DVD, ecc.).
- Gli apparati di comunicazione locali e remoti (Reti, Hub, Switch, Modems, ecc.).
- I terminali Video, i Personal Computers collegati in rete con i relativi Software di Emulazione Terminale.
- Il Personale di Gestione dei Serventi, dello Storage e della Rete.
- Altri Apparati hardware come ad esempio stampanti.
- Tutti i programmi e le procedure che costituiscono l'Applicazione Informatica.
- I Dati gestiti dall'Applicazione.

Per come qui definita, nell'Infrastruttura Informatica potranno essere presenti una o più Piattaforme Informatiche diverse ed inoltre una Infrastruttura Informatica potrà eseguire una o più Applicazioni Informatiche.
A causa del rapido sviluppo delle Applicazioni Informatiche e della Rete, negli ultimi decenni l'Infrastruttura Informatica si è notevolmente complicata, tanto che oggi essa è diventata un elemento di grande complessità e criticità.
In particolare, negli ultimi anni sono emerse alcune problematiche specifiche ed alcune criticità che qui vogliamo sottolineare; tali problematiche richiedono oggi degli interventi mirati, atti a superare i problemi e a rendere l'Infrastruttura meno critica e più semplice da gestire.
L'insieme degli interventi atti a rendere l'Infrastruttura Informatica più semplice, più stabile, più efficiente e più gestibile prende il nome di **Ottimizzazione della Infrastruttura Informatica**, abbreviato con la sigla ITRO (Information Technology Resources Optimization).

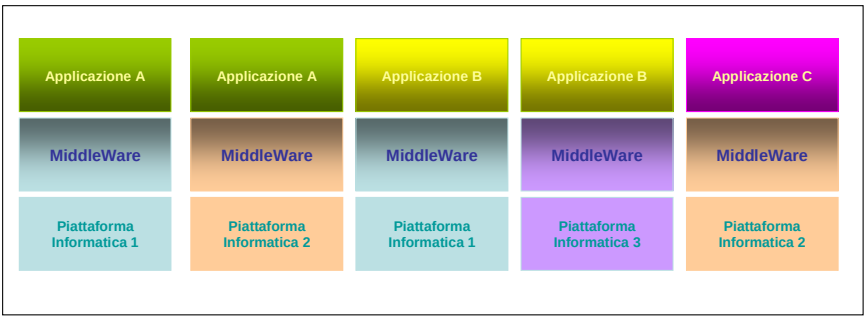


Figura 140 Vista Logica dell'Infrastruttura

Prima di affrontare nel dettaglio il processo di ITRO sarà forse opportuno analizzare quali sono le cause riconosciute della complessità dell'Infrastruttura e quali i problemi più importanti e di maggiore impatto.
A partire dalla metà degli anni '80, il Modello Applicativo denominato "Client Server" ha creato una proliferazione abbastanza incontrollata di Serventi (Elaboratori), che venivano installati in grande numero ed in locazioni differenti.

Lo stesso modello applicativo ha spesso introdotto il paradigma: "Una Funzione = Un Servente" che poi col passare degli anni è diventato "Una Funzione = Più Serventi", da ciò deriva che il numero di serventi installati è diventato molto grande, finendo col costituire esso stesso un problema infrastrutturale.
Sempre a causa del modello Applicativo ed in relazione ad una scarsa pianificazione della Infrastruttura, questa ha finito col comprendere Piattaforme Informatiche diverse e molteplici che risiedono in locazioni geografiche distanti tra di loro: questo di per sé non può considerarsi un "errore di progettazione", ma il risultato inevitabile di carenze tecnologiche e debolezze dei Sistemi Operativi.

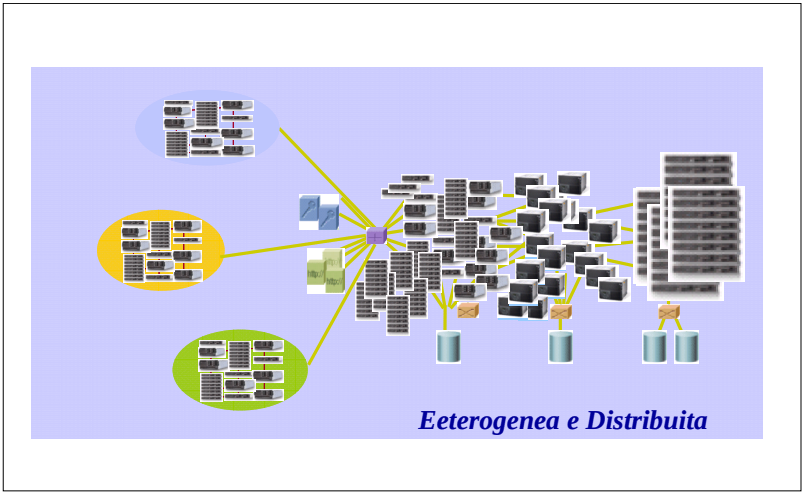


Figura 141 L'Infrastruttura è un elemento di complessità

Anche se oggi queste carenze sono state completamente superate e la tecnologia dei Sistemi Operativi è notevolmente progredita, il risultato di un paio di decenni di proliferazione incontrollata è ancora sotto i nostri occhi e continua a produrre gli stessi disagi.

Pur avendo preso coscienza del problema e pur disponendo oggi delle tecnologie adeguate, in assenza di un intervento specifico, si rischia di peggiorare ulteriormente la situazione.
Possiamo quindi affermare che ancora oggi, anche a causa del permanere di un grande numero di Serventi Eterogenei e distribuiti, l'infrastruttura rimane un elemento di grande complessità del quale ci dobbiamo occupare.
Un processo di ITRO, come già detto, in altri casi può avvenire in modo preventivo, cioè all'atto della definizione di una nuova infrastruttura, ma potrebbe anche essere effettuato "a posteriori" su una infrastruttura già esistente e produttiva: possiamo affermare anzi che quest'ultimo caso è quello più frequente nella pratica.
La Proliferazione dei Serventi (**Server Proliferation**) non è l'unico problema riscontrabile nell'Infrastrura Informatica.
Lo stesso modello applicativo che ha creato la proliferazione - ogni servente svolge sempre una sola funzione - ha generato anche uno scarso utilizzo dei Sistemi Serventi (**Under Utilization**): i Sistemi col tempo sono diventati sempre più potenti e di conseguenza dovendo e potendo svolgere una sola funzione sono diventati sempre meno utilzzati. Anche in questo caso non si tratta di un "errore" di progetto, ma di una semplice conseguenza del disegno architetturale delle applicazioni e dei limiti dei Sistemi Operativi, i quali spesso non consentono lo svolgimento di funzioni diverse sulla stessa Istanza di Sistema Operativo.

Un tema collaterale indotto da quanto finora affermato è quello dei Costi Gestionali (Cost of Management), il tema dei Costi è complesso e sarà trattato in dettaglio nel seguito, per il momento ci limitiamo al concetto intuitivo secondo cui il costo di una Infrastruttura cresce generalmente col crescere del numero dei Sistemi e componenti installati e in esercizio.

Un discorso a parte andrebbe fatto per i problemi legati alla **Qualità del Servizio** (Quality of Service), la **Resilienza** (Resiliency) e la **Ricoverabilità in caso di Disastro** (Disaster Recovery).
Sembra ancora una volta intuitivo il fatto che la possibilità di interruzione accidentale di servizio è funzione del numero di componenti discrete della Infrastruttura e quindi la riduzione del numero di esse possa ridurne sensibilmente il numero: poichè la **Qualità del Servizio** è inversamente proporzionale al numero di fermi accidentali, ne viene che una

Infrastruttura che presenta un minor numero di componenti discreti presenterà anche una migliore Qualità del Servizio.

La **Resilienza** è la capacità dell'Infrastruttura di "reagire" dinamicamente alle variazioni del carico applicativo, vale a dire l'elasticità che essa presenta rispetto ad una sollecitazione di intensità variabile: si può facilmente intuire che un'infrastruttura che comprende un numero minore di Sistemi possa reagire meglio a sollecitazioni esterne a causa della possibilità dei serventi di riutilizzare le capacità elaborative lasciate libere dalle altre applicazioni.

Infine negli anni più recenti stanno prevalendo anche **esigenze di tipo ambientale** legate al contenimento dei consumi energetici, delle necessità di raffreddamento e alla carenza di spazio: avere un grande numero di Sistemi in generale poco utilizzati implica la necessità di grandi potenze, grandi dissipazioni di calore e l'occupazione di grandi spazi, questo può divenire un elemento di grande criticità soprattutto allorquando si abbia carenza nella disponibilità di Energia Elettrrica o di Spazio.

Uno dei temi legati all'Infrastruttura è quindi anche quello della **riduzione dell'impatto ambientale**: molti apparati, soprattutto quelli più "datati", presentano scarsa efficienza energetica e necessitano di molta energia per essere raffreddati: indipendentemente dall'aspetto legato strettamente ai costi dell'energia ciò può avere anche una evidente ricaduta sull'inquinamento e sulla produzione di Anidride Carbonica.

Inoltre alcuni paesi del mondo occidentale cominciano ad avere, soprattutto nelle aree metropolitane densamente popolate, problemi di scarsità di approvvigionamento di energia elettrica in alcune ore del giorno: ciò nei fatti impone la necessità di limitare i consumi e quindi di migliorare l'efficienza elettrica a parità di potenza di calcolo.
Tutto l'insieme degli interventi volti a migliorare l'impatto ambientale della Infrastruttura Informatica prende il nome di "Green Computing" e i Centri di Elaborazione Dati che contengono Infrastrutture a basso impatto ambientale vengono detti "Gren Data Center".

Ritorniamo adesso sul tema della ITRO, che come abbiamo già detto, è l'insieme dei processi che tendono a rendere l'Infrastruttura più gestibile, più efficiente e meno costosa.
Il processo di ITRO in generale è lungo e può essere molto complesso; esso può avvenire intervenendo su fattori ed elementi differenti, ciascun elemento che interviene nel processo viene comunemente detto "driver", può compiersi in fasi distinte nel tempo dette "steps", e può essere facilitato dalla tecnologia mediante accorgimenti detti "enablers".
I driver più comunemente legati al processo di ITRO sono:

- L'Efficienza: si può intervenire sull'Infrastruttura cercando di migliorarne l'efficienza, indipendentemente dagli altri fattori.
- La Gestibilità: si può intervenire sull'Infrastruttura solo per renderla più gestibile.
- I Costi: si può operare sull'Infrastruttura al fine di diminuirne i costi operativi e di acquisizione; questo in effetti è il driver che più frequentemente spinge i processi di Ottimizzazione e del quale ci occuperemo con maggiore dettaglio nel seguito.
- La Resilienza o la Continuità di Servizio: si può operare sulla Infrastruttura solo per migliorarne le capacità di essere ricoverata in caso di disastro o la sua flessibilità operativa.
- I Consumi Energetici: prende sempre più consistenza la valutazione degli impatti ambientali come driver per l'ottimizzazione.

Come già accennato, il tema dei costi è il driver che innesca il maggior numero di processi reali di ITRO, esso quindi merita una particolare attenzione. Una volta determinato il driver o i drivers che innescano il processo, occorrerà determinare quali siano le Azioni da intraprendere o da valutare ai fini del processo stesso. Ad esempio azioni possibili potranno essere:

1. Diminuire il Numero di Serventi.
2. Consolidare gli apparati a disco.
3. Diminuire il numero di Istanze di Sistema Operativo presenti nella infrastruttura.
4. Virtualizzare i Sistemi.

5. Virtualizzare le Reti trasmissive.
6. Usare Sistemi Operativi meno costosi a parità funzionale.

Le azioni sopra indicate, che peraltro non sono le uniche possibili, possono essere facilitate dalla tecnologia: la tecnologia in particolare mette a disposizione degli elementi che sono "abilitanti" ai fini del processo, in quanto lo facilitano e a volte lo rendono possibile con poco sforzo.
Tra gli elementi abilitanti, detti enablers, vorremmo sottolinearne tre, che sono particolarmente funzionali alla nostra trattazione. Essi sono:

- La Virtualizzazione, in quanto consente di ridurre il numero di Serventi Fisici e spesso anche il numero di quelli logici con poco sforzo operativo (**Server Consolidation**).
- L'uso del Sistema Operativo **Linux**, in quanto generalmente poco costoso e facilmente manutenibile.
- L'uso di **Grandi Sistemi Centrali** al posto di molti serventi distribuiti; tale uso, abbinato alla virtualizzazione, può consentire grossi recuperi di efficienza complessiva.

Solo su questi tre elementi concentreremo la nostra trattazione da qui in avanti.

4.0.3 Il Processo di Consolidamento dei serventi [Server Consolidation]

Si definisce "Server Consolidation" il processo volto a diminuire (consolidare) il numero di Serventi Fisici o Logici componenti l'Infrastruttura Informatica.

Tale processo potrebbe, di per sè, essere complesso o di difficile realizzazione a causa delle legacies applicative che abbiamo analizzato in precedenza.

Per tale ragione si suole indicare con "Server Consolidation" il processo che tende a diminuire il numero di Serventi Fisici o Logici di una Infrastruttura Informatica con interventi di modifica piccoli, al limite nulli, sulle Applicazioni Informatiche che su di essa insistono.
Tale operazione può essere effettuata sia sui Serventi Fisici che sui Serventi Logici; essa contribuisce sostanzialmente al processo di Riduzione dei Costi Operativi e quindi è parte del processo di ITRO, anzi di questo molto spesso rappresenta la parte più rilevante.

Il processo di Server Consolidation è abilitato fortemente dalle capacità di virtualizzazione dei Serventi ed ha senso e si presenta più efficace in quanto spesso i Serventi di partenza (cioè quelli da consolidare) sono molti e singolarmente poco utilizzati.

Diversamente, nel caso in cui i Serventi di partenza fossero pochi e molto utilizzati, esso risulterebbe un processo poco rilevante al fine della diminuzione della potenza complessivamente necessaria.

Queste affermazioni sono dimostrate da quanto avevamo affermato nel capitolo 2 a proposito dei vantaggi della virtualizzazione in termini di potenza complessiva richiesta per un insieme di Serventi che eseguono la stessa Applicazione Informatica.

La Virtualizzazione rappresenta quindi una **Tecnologia Abilitante** (Enabling Technology) ai fini della Server Consolidation e quindi della ITRO, in quanto consente di diminuire il numero di Serventi Fisici e Logici e di utilizzarli al meglio.

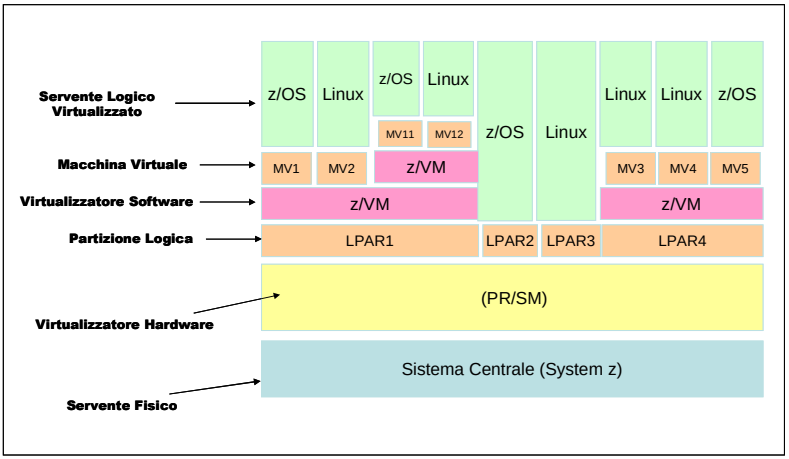


Figura 142 Tipologie di Serventi

Prima di procedere oltre, ricordiamo alcune definizioni di base in termini di Virtualizzazione:

- **Servente Fisico** o Sistema Fisico è un calcolatore elettronico fisicamente indipendente ed isolato, ovvero dotato di alimentazione elettrica indipendente, costituito da componenti elettricamente isolate ed indipendenti da altri Sistemi.
- **Servente Logico** o Immagine di Sistema è una istanza di Sistema Operativo indipendente da altre, cioè attivabile o disattivabile per intero indipendentemente dalle altre ed in maniera non correlata alla sua collocazione su un Servente Fisico.
- Un **Servente** è **Virtualizzato** se esso è attivato su un Servente Fisico insieme ad altri Serventi Logici: in tal caso il Servente fisico disporrà di una apposita componente Hardware o Software o di entrambe, detta "virtualizzatore", ed in grado di condividere il Servente Fisico tra più Serventi Logici (Istanze di Sistema Operativo).
- Diremo "**Macchina Virtuale**" o **Partizione Logica** la porzione o l'insieme di risorse di un Servente Fisico dinamicamente assegnate da un dispositivo virtualizzatore ad un Servente Logico Virtualizzato.

Dalle definizioni segue che un Servente Virtualizzato è solo e sempre un Servente Logico e inoltre su un Servente Fisico possono essere attivati più Serventi Logici mediante uno o più virtualizzatori.

Dopo aver precisato le definizioni relative alla virtualizzazione, cerchiamo di definire meglio il processo di Server Consolidation; in esso possiamo indicare alcuni passi conseguenti, parleremo ad esempio di:

- **Accentramento** (o Centralizzazione) nel caso in cui svariati serventi fisici posti in diverse locazioni geografiche vengano appunto "accentrati" in una sola di esse. L'accentramento è una pratica sempre più frequente e si contrappone a quella della "distribuzione geografica" particolarmente in uso negli anni '80. Essa è resa possibile dalle accresciute capacità di banda delle reti trasmissive e dai diminuiti costi delle linee di comunicazione, nonchè dalle nuove tecnologie trasmissive che eliminano i problemi legati alla difficoltà di comunicazione tra i Client distribuiti ed i Server centralizzati. L'accentramento presenta una convenienza "a priori", indipendentemente dai maggiori costi indotti dalle linee di comunicazione, in quanto concentra in una o due locazioni geografiche tutti i Serventi e si ottiene il vantaggio di non dovere spostare persone (o mantenere persone in diverse località) per gestire i Sistemi; inoltre si possono realizzare importanti sinergie in termini di potenza e di Continuità di Servizio. L'accentramento è una pratica ormai diffusa e consolidata.

- **Consolidamento Fisico** (Physical Consolidation) è il processo per il quale diverse immagini di Sistema Operativo, prima ospitate su Serventi Fisici separati, vengono consolidate su un unico Servente Fisico. Tale processo è tanto più efficace quanto più le capacità di virtualizzazione e di condivisione delle risorse del Sistema Fisico ricevente sono sofisticate. Ad esempio, se il Sistema Fisico ricevente è in grado di virtualizzare con granularità molto "fine" sarà possibile consolidare un grande numero di Serventi Logici in pochi Serventi Fisici completamente utilizzati. Viceversa, se il Sistema Fisico ricevente presenterà un basso livello di condivisione delle risorse, saranno ottenuti risultati modesti in termini di

risparmio complessivo di potenza elaborativa. Il Consolidamento Fisico è la pratica maggiormente diffusa nella realtà produttiva ed è spesso il massimo livello al quale si può giungere senza essere costretti ad operare interventi profondi sulle applicazioni informatiche.

- **Consolidamento Logico** (Logical Consolidation) rappresenta il vero e il più efficace processo di consolidamento, in quanto si pone l'obiettivo di ridurre le immagini di Sistema installate (Sistemi Logici). Esso consiste nel porre il maggior numero di funzioni applicative nello stesso Sistema Logico, il quale a sua volta potrà stare su un Sistema Fisico intero o su un Sistema Virtualizzato. Il consolidamento logico è una pratica complessa in quanto ai Sistemi Operativi richiede particolari caratteristiche, come la capacità di ospitare diverse funzioni e carichi eterogenei. Questa possibilità, come abbiamo già detto, non è presente in tutti i Sistemi Operativi. Il Consolidamento Logico rappresenta un importante elemento di ottimizzazione poichè riduce notevolmente il numero di immagini di Sistema da gestire.

La Figura 143 mostra la differenza tra i processi di consolidamento possibili. Specifichiamo infine che dal punto di vista dell'Infrastruttura, la sola fondamentale differenza tra i processi di consolidamento Fisico e Logico è rappresentata dal fatto che nel primo caso ogni funzione di servente continuerà a risiedere sulla stessa immagine di Sistema Operativo dalla quale proveniva, nel secondo caso invece più funzioni di Servente Logico verranno accorpate sotto il controllo dello stesso Sistema Operativo e, ripetiamo ancora, ciò potrà essere possibile solo se il Sistema Operativo ricevente è in grado di ospitare carichi eterogenei contemporaneamente.

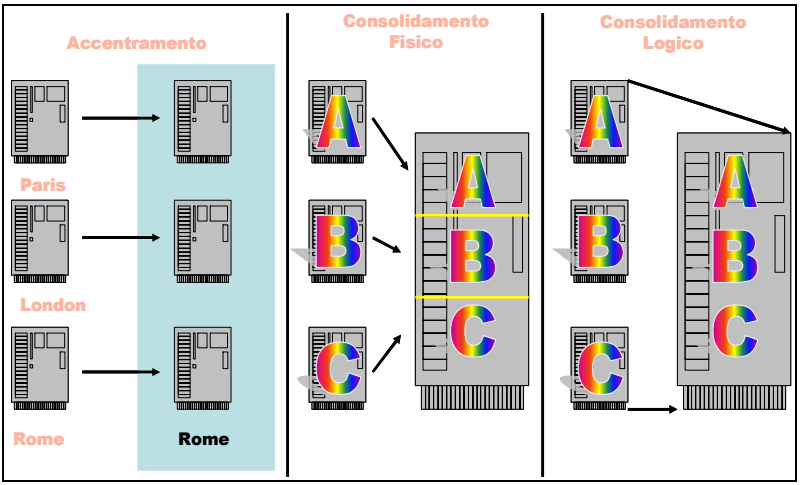


Figura 143 Tipi di consolidamento

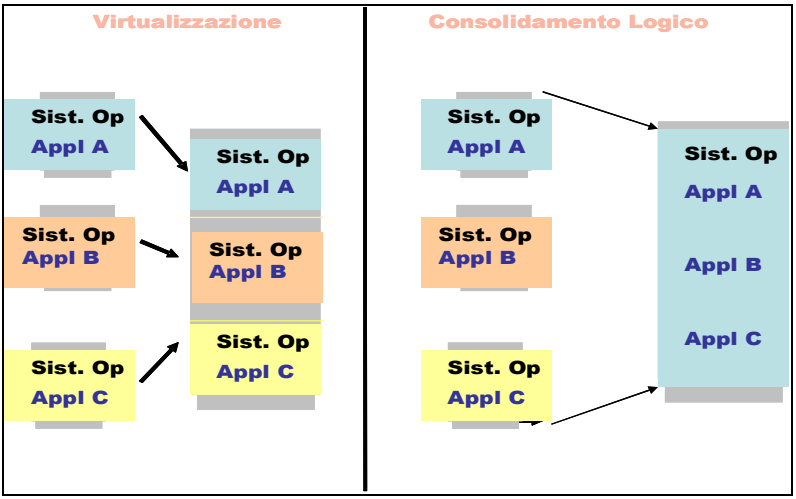


Figura 144 Virtualizzazione e consolidamento logico

Nei processi di Consolidamento Logico il fattore abilitante è rappresentato dalla capacità dei Sistemi Operativi di ospitare sotto la stessa istanza carichi applicativi, cioè Applicazioni Informatiche, diversi ed eterogenei contemporaneamente; ciò non è comune a tutti i Sistemi Operativi, al contrario la Virtualizzazione consente di operare consolidamenti Fisici sullo stesso Servente Fisico di diverse Istanze di Sistema Operativo (Serventi Logici) senza intervenire su alcuna caratteristica dei Sistemi Operativi che restano comunque inalterate; tale processo è di gran lunga più semplice.

Concludendo, il Consolidamento dei Serventi (Server Consolidation) può essere il primo e decisivo passo per un processo di ITRO; esso può avvenire per passi successivi e con livelli di intervento diversi.
Il Consolidamento dei Serventi può essere considerato ormai una necessità stante la complessità raggiunta dalle infrastrutture informatiche: come avevamo già detto a proposito della Platform Selection, esso pone la necessità di disporre di elementi quantitativi in grado di valutare il processo e di stabilire se e quando esso sia conveniente in base ai drivers di ottimizzazione stabiliti.
Poiché il driver più frequentemente usato è la Riduzione dei Costi, ci concentreremo ora sul tema dei Costi e su una metodologia per confrontare i Costi di Infrastrutture Informatiche oggetto di ottimizzazione o di Server Consolidation in particolare.

4.0.4 Il ruolo di Linux nei processi di ITRO

Abbiamo visto come il processo di Ottimizzazione passi spesso attraverso il Consolidamento dei Serventi, analizzeremo brevemente in questo capitolo un altro aspetto essenziale per il processo di ottimizzazione più noto col nome di "**Cambio di Piattaforma**" (o re-hosting).

Si dice che si esegue un processo di re-platforming (o re-hosting) quando una o più funzioni del Servente vengono spostate da una Piattaforma Informatica ad un'altra. Tale processo ha senso solo quando la piattaforma di arrivo è ritenuta maggiormente conveniente, ad esempio dal punto di

vista dei costi, ovvero più affidabile, ovvero più utile al processo stesso di ottimizzazione.

Temi fondamentali del processo di re-platforming sono:

- La **compatibilità applicativa**, ovvero la capacità della piattaforma di arrivo di ospitare la stessa funzione applicativa oggetto di re-platforming (re-hosting) con le stesse caratteristiche fuzionali. Ciò, come già discusso prima, dipende dalla "portabilità" della funzione o della Applicazione Informatica che vogliamo cambiare di Piattaforma.
- La **Pianificazione della Capacità** (Capacity Planning), ovvero la determinazione della potenza del Sistema ricevente che, essendo in generale di un'altra architettura, potrà risultare caratterizzato da diverse metriche. Di tale problema discuteremo in dettaglio più avanti nel paragrafo 3.2.6.

Un caso particolare di **compatibilità applicativa** e di conseguente portabilità si ha per le cosidette Funzioni Infrastrutturali o di Servizio, quali ad esempio quelle che gestiscono la Sicurezza, le autenticazioni, le funzioni di Stampa, di File Serving, di Directory, di Name Serving, ecc. Tutte queste funzioni sono praticamente analoghe su tutte le piattaforme e pertanto sono automaticamente portabili. Ne segue che in relazione ad esse la portabilità è genericamente garantita e quindi il re-platforming (re-hosting) è quasi sempre possibile.
Analoga osservazione può essere fatta tutte le volte in cui la nostra Applicazione Informatica usi dei Sottosistemi o dei Middleware; in questo caso la portabilità e quindi la capacità di essere ospitata sull'altra piattaforma è garantita dal Middleware.

Non si ha invece portabilità e quindi non si può cambiare facilmente la Piattaforma quando si usano strumenti e linguaggi proprietari o tipici della Architettura o Piattaforma, come Assembler, o se si usano Gestori di Basi Dati che funzionano su una sola piattaforma (ad esempio SqlServer).

Nel processo di cambio di Piattaforma assume una particolare importanza il Sistema Operativo Linux; esso risulta infatti particolarmente importante al fine della ottimizzazione di una infrastruttura. E' noto a tutti infatti come Linux possa ospitare a costi di Esercizio e di Acquisizione più contenuti collaterali dell'Infrastruttura Informatica come i già citati DNS, LDAP Server, File Server, ecc.
Inoltre per le caratteristiche di "Open Source" sempre più spesso funzioni Applicative importanti vengono ospitate sul Sistema Operativo Linux risultando nel complesso più gestibili e semplici da manutenere.
Va comunque doverosamente precisato che Linux non è in grado di svolgere con la stessa qualità di Servizio funzioni complesse, ora svolte da Sistemi Operativi più potenti e completi come z/OS, e che quindi al momento non è pensabile che esso possa completamente sostituirli.
Questa osservazione spinge ad affermare che all'interno del processo di ottimizzazione occorrerà tenere conto anche della possibilità di spostare alcune funzioni da piattaforme più costose a Linux, ma ci si dovrà limitare a quelle che non richiedono la qualità di Servizio di Sistemi Operativi più complessi, facendo leva sulle capacità di virtualizzazione offerte dalla architettura e dalla piattaforma.

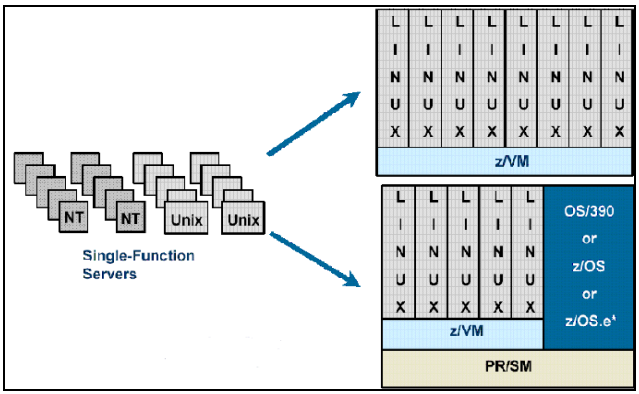


Figura 145 Replatforming

Concludiamo questa trattazione con un'ultima osservazione legata all'uso di ambienti Linux per servizi collaterali: se si riesce a spostare (re-

platforming) su Linux dei carichi applicativi si può ottenere il risultato di spostare su due livelli (TIERS) una applicazione originariamente progettata per funzionare su tre livelli, ottenendo tra l'altro una migliore ricoverabilità della infrastruttura a fronte di disatri o di guasti gravi. La Figura 146 mostra questa caratteristica su un Sistema di z/Architecture.

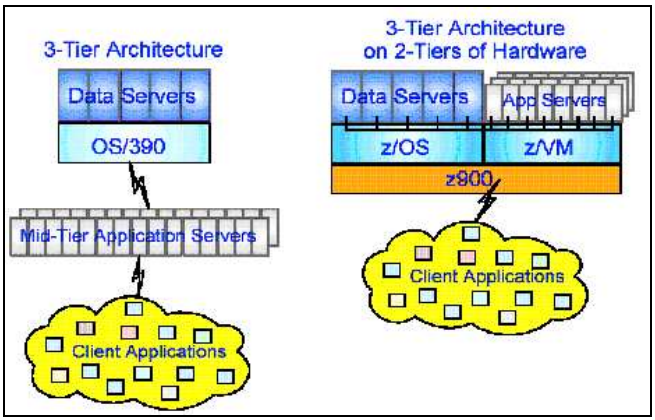


Figura 146 Consolidamento di architetture 3-Tier

4.0.5 Un approccio quantitativo basato sui Costi [TCO]

Abbiamo visto come i Processi di Platform Selection e di I/T Resources Optimizazion si basano su concetti come la Portabilità delle Applicazioni e il re-platforming.
Abbiamo anche detto che qualunque processo di cambiamento di una Infrastruttura Informatica, volto ad esempio alla sua ottimizzazione, è governato e spinto da drivers, uno dei quali, praticamente il più frequentemente usato, è quello dei costi.
Per poter governare processi di cambiamento, come ITRO, servono quindi strumenti quantitativi in grado di fornire metriche di supporto alle decisioni.

In altre parole lo scopo che ci proponiamo è quello di rispondere in maniera oggettiva alla questione:

'Ho individuato una Infrastruttura come possibile alternativa a quella attuale, ho individuato un driver di ottimizzazione (ad esempio il costo), quali sono le condizioni sotto le quali il processo di trasformazione individuato mi fornisce una convenienza in base al driver stabilito (costo inferiore) ed in quali tempi ottengo un ritorno dell'investimento fatto per la trasformazione?'

Per rispondere efficacemente a questa domanda occorre disporre di strumenti quantitativi ed asettici che dovranno consentire di:

- Valutare "a priori" l'efficacia del processo
- Fornire criteri di supporto alle decisioni
- Dare indicazioni sul ritorno dell'investimento
- Verificare i risultati a posteriori

Limitandoci al solo driver dei costi possiamo allora definire un metodo che diremo del Total Cost of Ownershp (TCO). Il metodo fu introdotto alla fine degli anni novanta da alcuni consulenti indipendenti del mercato informatico e si basa sulla definizione di una quantità (espressa in valuta) denominata TCO e definita come segue:

$$TCO = TCOp + TCA$$

Il metodo si pone l'obiettivo di confrontare due "soluzioni" dal punto di vista dei costi fornendo un modello assoluto ed indipendente da valutazioni soggettive.

Il TCO viene calcolato e confrontato per tutte le soluzioni oggetto della analisi che poi verranno confrontate.

Il TCO è dato dalla somma di due componenti diverse, una detta TCOp, rappresenta i costi operativi ricorrenti e pertanto dipende dal tempo ed è variabile con esso:

n

$$TCOp = \sum_{K=1} (TCOp)_K$$

dove $TCOp_k$ rappresenta la componente dei costi ricorrenti per ogni periodo elementare, per il quale sono ritenuti costanti (ad esempio un anno) e la sommatoria è estesa al periodi di n intervalli (ad esempio 3 anni).
Invece TCA rappresenta i costi di acquisizione una tantum, non dipende dal tempo ed è costante per tutto il periodo di valutazione.
Poiché la grandezza TCO contiene una componente (TCOp) che è funzione del tempo, essa sarà funzione del tempo e quindi dovrà sempre essere riferita ad un intervallo di tempo prefissato. Tale intervallo di tempo è detto "Periodo di analisi". Tipicamente in Italia si suole usare periodi di analisi di tre anni, cioè pari ai periodi di ammortamento dei beni informatici.
Il TCO è un concetto intuitivo che può essere applicato a qualunque fatto della vita quotidiana: proviamo ad immaginare il TCO relativo al possesso di una automobile.

Nei costi ricorrenti (TCOp) calcoleremo ad esempio per ogni anno:

1. L'affitto del Box/garage
2. Il Costo dei Carburanti/Lubrificanti
3. Il Costo della manutenzione
4. Il Costo del lavaggio
5. La tassa di Proprietà
6. L'Assicurazione Obbligatoria

Nei costi di acquisizione (TCA), invece calcoleremo:

1. Costo di acquisto
2. Costo delle modifiche /Aggiunte
3. Costo dell'autoradio
4. Costo dei tappetini

Potremmo quindi confrontare due automobili diverse calcolando nei due casi il TCO ad esempio per quattro anni (vita media dell'automobile). Per fare un conto accurato dovremo aggiungere al TCA anche il valore residuo della automobile alla fine dei quattro anni (net present value).
Anche se potrebbe sembrare ovvio farlo, nella pratica nessuno acquista o esclude l'acquisto di una automobile basandosi sul TCO; intervengono in

questo caso "fattori esterni" di tipo emotivo che portano a intraprendere delle scelte indipendentemente dalla convenienza (moda, gradimento, diffusione sul mercato, ecc.). Questo tipo di fattori potrebbe anche influenzare la scelta di una Infrastruttura o la selezione di una Piattaforma Informatica, ma ciò è proprio quello che stiamo cercando accuratamente di evitare, ovvero se fattori non semplicemente quantificabili dovranno essere inseriti nel nostro calcolo, dovremo sforzarci di tramutarli in valori numerici o monetari. Ad esempio il fatto che una Infrastruttura Informatica sia più stabile e resiliente di un'altra potrebbe essere trasformato in diminuizione dei fermi non pianificati e questo dovrebbe essere trasformato in diminuzione della perdita di denaro dovuta ai fermi non pianificati.

In molti casi risulterà particolarmente semplice effettuare una simile valutazione, in altri casi sarà praticamente impossibile.
Ad esempio se l'Infrastruttura Informatica che stiamo analizzando svolge il ruolo di immettere e controllare le prenotazioni di una aerolinea tramite la rete, sarà facile determinare quale sia la perdita di passeggeri (denaro) dovuta al fatto che il servizio sia interrotto per un certo numero di ore: basta sapere quale sia in media il numero di posti/biglietti venduti ogni ora ed assumere che essi vadano tutti alla concorrenza nel caso in cui un utente non riesca a fare la prenotazione a causa di un fermo non pianificato.
Molto più difficile o forse impossibile sarebbe valutare la perdita nel caso in cui l'Infrastruttura gestisca un servizio che non comporti l'esborso di denaro, ad esempio un servizio al cittadino da parte di un ente pubblico. Ci sono infine molti altri casi nei quali un fermo non pianificato non comporta alcun tipo di perdita in denaro.
Ciò premesso, potremmo provare a limitarci, almeno in prima approssimazione, alle componenti di costo certe, lasciando ad una seconda analisi quelle più aleatorie o di difficile quantificazione.

Riferendoci alla Piattaforma Informatica o all'Infrastruttura Informatica sono elementi del TCOp:

- Costi di Personale (FTE)
- Costi di Energia Elettrica, Raffreddamento e Spazio (Environmentals)

- Costi ricorrenti legati al Software (Manutenzione e Canoni)
- Costi ricorrenti legati all'Hardware (Manutenzione)
- Costi ricorrenti legati alla rete
- Altri costi Ricorrenti
- Valore delle interruzioni di Servizio
- .....

Sono invece elementi del TCA:

- Costi di acquisizione Hardware
- Costi di acquisizione Software
- Costi di altri apparati
- Costo della migrazione dei Pacchetti Applicativi
- Consulenze
- Altre spese occasionali (Trasporto, Attrezzaggio, Montaggi)
- Costi di dismissione
- Net Present Value (Ammortamento anticipato)

Alcuni costi sono pesantemente influenzati dalla riduzione dei Sistemi Fisici, tra questi ricordiamo gli environmentals e alcuni costi legati alle licenze Software e alle manutenzioni. Il costo del personale (FTE) dipende in larga parte dal numero delle immagini di Sistema cioè dei Serventi Logici.
Il metodo del TCO può essere usato nel confronto di due o più soluzioni informatiche (Piattaforme o Infrastrutture) secondo lo schema (Figura 147)

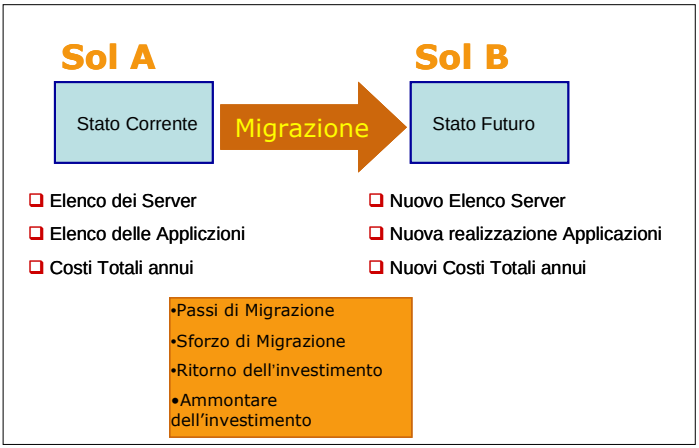


Figura 147 Confronto di due Soluzioni col metodo TCO

In questo caso, se indichiamo con "old" lo stato attuale e con "new" quello ipotizzato dopo la trasformazione, utilizzando le definizioni fino ad ora enunciate avremo:

$$TCO_{OLD} = \sum_{i=1}^{K} (TCOp_{OLD})_i + TCA_{OLD}$$

ed invece per la nuova Infrastruttura:

$$TCO_{NEW} = \sum_{i=1}^{K} (TCOp_{NEW})_i + TCA_{NEW}$$

Sarà bene osservare che il periodo di analisi, ovvero il valore di k, deve essere uguale nei due casi, se vogliamo operare un confronto.

Una operazione di migrazione OLD ---→ NEW risulterà conveniente in base al driver del TCO se accade che:

$$TCO_{OLD} > TCO_{NEW}$$

e quindi:

$$\sum_{i=1}^{K} (TCOp_{OLD})_i + TCA_{OLD} > \sum_{i=1}^{K} (TCOp_{NEW})_i + TCA_{NEW}$$

Se ci riferiamo ad una infrastruttura OLD esistente da un periodo maggiore di k, possiamo assumere che i costi di acquisizione a suo tempo sostenuti siano stati ammortizzati (ciò vale ad esempio per le valutazioni di casi di re-hosting);

sotto queste condizioni possiamo porre:

$$TCA_{OLD} = 0$$

e quindi attraverso semplici passaggi risulterà:

$$TCA_{NEW} < \left[\sum_{i=1}^{K} (TCOp_{OLD})_i - \sum_{i=1}^{K} (TCOp_{NEW})_i \right]$$

La relazione sopraindicata esprime il concetto in base al quale una operazione di cambiamento di Piattaforma per una Infrastruttura già esistente è conveniente dal punto di vista dei costi se il suo costo complessivo di acquisizione (espresso da TCA_{NEW}) risulta minore della differenza tra i costi operativi dell'Infrastruttura attuale e di quella nuova estesa al periodo di valutazione.

Il metodo qui espresso fornisce risultati tanto più accurati quanto più accurata risulterà la determinazione delle quantità TCA e TCOp nei due casi.

I concetti fino a qui esposti possono essere rappresentati in maniera grafica fornendo una rappresentazione visiva immediata del risultato.

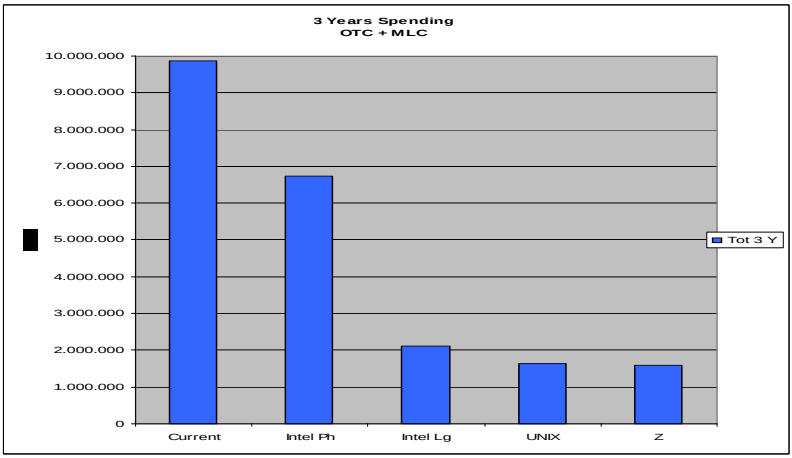


Figura 148 Rappresentazione grafica del TCO

La Figura 148 mostra un esempio di rappresentazione del TCO calcolato su un periodo di tre anni e per cinque diverse soluzioni alternative. Un esame visivo del grafico fornirà una indicazione su quale sia la piattaforma più conveniente dal punto di vista dei Costi.

Si osservi che il TCO è composto da due componenti, una fissa nel tempo ed una variabile con esso: dal punto di vista economico/finanziario ciò si traduce dicendo che il TCO è la somma di una componente di Spese di Capitale (TCA) e di una di Costi Operativi (TCOp). Si suole definire la prima col termine inglese Capital Expenditures (abbreviato in CAPEX) e la seconda col termine inglese Operational Expenditures (abbreviato in OPEX). Nella sostanza questa differenza potrebbe essere importante nella valutazione da parte di Enti o Società quotate in borsa e quindi andrebbe tenuta in considerazione.
Per completare la trattazione dell'argomento TCO occorre tener presente un ultimo aspetto, quello cioè del "Ritorno dell'Investimento", sovente abbreviato con la sigla ROI (Return of Investment). Il periodo di ritorno di

un investimento o semplicemente il ritorno dell'investimento è il periodo di tempo entro il quale l'investimento operato per l'acquisizione di un bene o servizio viene ripagato dai risparmi che il bene o servizio produce con il suo uso.
In altre parole, se la nuova infrastruttura presenta dei costi di esercizio TCOp inferiori a quelli attuali, si deve presumere che usando la nuova infrastruttura si possano ottenere dei risparmi. Però il passaggio dalla vecchia alla nuova infrastruttura comporta dei costi rappresentati dal TCA. Il periodo di Ritorno di Investimento rappresenta l'intervallo di tempo a partire dal quale si comincerà a risparmiare una volta recuperati gli investimenti o spese fatti per passare alla nuova infrastruttura.

Per ottenere in modo veloce ed intuitivo (una corretta trattazione potrebbe essere fatta ma non è oggetto di questo volume) ricorriamo ad un metodo grafico, rappresentato dalla Figura 149.
In essa la linea rossa (Cum D) rappresenta la somma dei costi operativi correnti; tale linea è retta in quanto, al netto dei fattori inflattivi, tali costi tendenzialmente sono costanti e quindi per ogni intervallo di tempo la somma cresce della stessa quantità .
Poichè abbiamo ipotizzato che la nuova infrastruttura abbia dei costi operativi inferiori, essi saranno rappresentati da una retta con pendenza minore (slope), nel nostro grafico la linea verde (Cum V).
Le due rette tendono a divergere in quanto i miglioramenti dei costi costituiscono un risparmio crescente, per tempo considerato la distanza tra le due rette rappresenta il risparmio ottenuto. La linea verde infatti si trova sempre al di sotto della rossa.
Poiché però nella realtà il passaggio dall'attuale infrastruttura (linea rossa) alla nuova (linea verde) comporta spese/investimenti dobbiamo sommare tali investimenti (operati nel tempo) ai dati che hanno prodotto la linea verde; otterremo in questa maniera la linea blu (Cum R). Essa rappresenta le spese reali date dalla somma dei Costi Operativi più gli investimenti.

In termini economici la curva blu è la somma degli esborsi reali derivanti dalle spese operative e dagli investimenti.

Per quanto detto sopra risulta evidente che il periodo di ritorno di investimento cercato è dato dalla proiezione sull'asse dei tempi (asse x) del punto di incontro tra la linea rossa e quella blu.

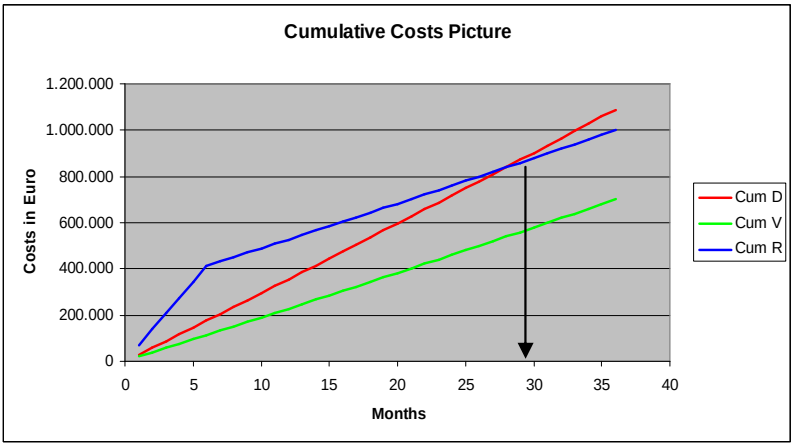


Figura 149 Rappresentazione grafica del ROI

Infatti poichè abbiamo dedotto che la distanza tra due linee nel nostro caso rappresenta il risparmio tra due soluzioni, se consideriamo le linee rossa e blu, tale distanza è negativa (spesa) fino al punto di incontro; da lì in poi diviene positiva (risparmio). Quindi il punto di incontro tra le due linee era il tempo di ROI che stavamo cercando.
Una simile tecnica può essere utilizzata anche per il confronto tra più di due soluzioni, in questo caso come vedremo negli esempi, la linea verde non viene tracciata (essa in effetti non corrisponde ad una situazione reale), e si tracciano due "versioni" della linea blu per le due alternative considerate rispetto alla situazione corrente.
Concludiamo affermando che il metodo qui illustrato fornisce, in maniera quantitativa rispetto al driver del costo, indicazioni sulla scelta della Piattaforma Informatica migliore o dell'infrastruttura migliore.
Il metodo potrà essere utilizzato:

1. Per confrontare due o più piattaforme informatiche dal punto di vista dei costi

2. Per valutare processi di re-hosting determinandone o meno la convenienza
3. Per giustificare l'uso di Linux su Sistemi Centrali
4. Per giustificare processi massivi di Server Consolidation
5. In generale in quasiasi processo di ITRO che abbia la riduzione dei costi come driver.

Alla fine del capitolo forniremo alcuni esempi pratici di applicazione della metodologia.

4.0.6 Cenni al Dimensionamento dei Sistemi su piattaforme Informatiche eterogenee [Cross Platform Sizing]

Il processo che consente di determinare il confronto tra potenze elaborative di due o più Sistemi di diversa Piattaforma Informatica prende il nome di "Cross Platform Sizing" (Dimensionamento multipiattaforma).
Ha senso porsi tale problema tutte le volte in cui si vogliano fare confronti tra le Piattaforme Informatiche, ad esempio il Metodo TCO impone tale processo per stabilire quale sia l'Infrastruttura di Serventi da utilizzare per eseguire un certo carico di lavoro (Applicazione Informatica) quando vogliamo considerare un'infrastruttura alternativa.
Il problema si traduce nello stabilire in sostanza metriche equivalenti o universali in grado di valutare le capacità elaborative indipendentemente dalle Architetture Hardware e dal Sistema Operativo, cioè indipendentente dalla Piattaforma Informatica.
In assenza di standard internazionali per misurare la capacità delle Piattaforme Informatiche il tema risulta a tutt'oggi non risolto nella sua interezza: infatti, mentre alcuni organismi autonomi eseguono misurazioni comparative delle piattaforme basate su UNIX costantemente (tali misurazioni comparative vengono regolarmente pubblicate), ciò accade meno per Linux e non accade per i Sistemi di z/Architecture, per i quali IBM ha sviluppato un Sistema proprietario denominato Large System

Performances Report (LSPR) che mette a confronto i vari modelli di macchina con vari sistemi operativi (ad esempio z/OS, z/VM e zLinux).

Nei fatti non esistono misurazioni pubblicate che mettano in relazione la piattaforma z/Architecture + z/OS con PowerRisc + AIX (solo per restare in ambito di prodotti IBM).

Ulteriore complicazione è data da due altre importanti considerazioni:

- Alcune piattaforme informatiche basate sullo stesso Hardware (ad esempio x86) con Sistemi Operativi diversi (ad esempio MS Windows o Linux) presentano livelli di efficienza radicalmente diversi e quindi valori di potenza differenti per lo svolgimento delle stesse funzioni applicative.
- La stessa piattaforma informatica presenta livelli di efficienza (e quindi potenza relativa) diversi nell'eseguire lavori di diversa natura (ad esempio Batch piuttosto che Transazionali).

Limitandoci in particolare all'oggetto di cui trattiamo, cioè i Sistemi di z/Architecture, esiste ancora una sostanziale differenza di efficienza (in parte dovuta a questioni puramemte architetturali, in parte alla diversa "esperienza" dei Sistemi Operativi) tra le capacità di potenza del Sistema Operativo z/OS e quelle di zLinux.

Proveremo qui ad esprimere una trattazione più generale e a descrivere alcuni casi particolari in cui un approccio più semplice potrà essere usato.
Premettiamo che in generale le prestazioni di un Sistema non dipendono solamente dalla capacità elaborativa del Calcolatore; l'architettura hardware ha un ruolo essenziale ad esempio nel trattamento delle Operazioni di I/O o nell'uso dei dispositivi e della memoria.
Inoltre il Sistema Operativo può presentare caratteristiche di progetto che lo rendano più adatto ad un certo tipo di operazioni e che ne penalizzino altre. Avremo così ad esempio:

- Sistemi specializzati per il calcolo intensivo
- Sistemi specializzati per l'elaborazione dei dati
- Sistemi specializzati per operazioni commerciali (transazionali)

- Sistemi specializzati per il disegno e la grafica
- .....

Premesso quindi che le prestazioni dipendono sia dalla architettura hardware che dal Sistema Operativo (Piattaforma Informatica) ed assodato che diversi tipi di lavoro possono avere prestazioni diverse a parità di Piattaforma Informatica, sarà opportuno valutare le capacità elaborative e quindi le prestazioni precisando anche il tipo di lavoro eseguito; definiremo questa caratteristica Workload Factor e la indicheremo con WLF.

Inoltre è noto che i Sistemi Reali non riescono sempre a lavorare al massimo della potenza che sono in grado di fornire (sulla carta). Per esempio sappiamo dalla pratica che alcune Piattaforme Informatiche sono tipicamente utilizzabili in modo produttivo per percentuali del 60% al massimo, mentre altre si spingono vicino al 95%. Questo non è un errore di progetto, ma una caratteristica globale e peculiare del Sistema, legata in buona parte all'architettura Hardware ed in altra parte al disegno del Sistema Operativo. Questa caratteristica definisce un Punto di Saturazione del Sistema detto **System Saturation Point** (SSP). I Sistemi Centrali presentano un SSP circa uguale ad uno (100%), alcuni Sistemi di architettura Hardware x86 non superano il 60%.

Infine non esiste uno standard internazionale per definire le metriche relative alla potenza di un Sistema: il mercato mette a disposizione dei benchmark, generalmente correlati al tipo di lavoro svolto, oltre a misure di grandezze fisiche come il ciclo base dei Processori o i MIPS. E' opportuno osservare che per questo motivo i confronti tra Piattaforme Informatiche vengono inficiati di una ulteriore imprecisione dovuta alla necessità di dover normalizzare le metriche.

Nell'effettuare il confronto e quindi il dimensionamento dovremo precisare:

1. Architettura Hardware
2. Sistema Operativo
3. Tipo di Lavoro
4. Punto di Saturazione del Sistema
5. Metrica usata

Proviamo ad esprimere in forma quantitativa i concetti fino ad ora espressi nella convinzione che comunque sia sempre necessaria una verifica sperimentale di quanto stiamo affermando.

Risulta intuitivo affermare che dati i **Sistemi** Informatici A e B, rispettivamente di capacita Ca e Cb, rispettivamente utilizzati Ua ed Ub, essi sono **equivalenti** quando:

Ca * Ua * Ka = Cb *Ub * Kb

In questo caso equivalente vuole dire che essi sono capaci di fare lo stesso lavoro nello stesso tempo con gli stessi risultati.

Mentre Ua e Ub sono numeri compresi tra 0 ed 1, Ca e Cb devono essere espressi nella stessa unità di misura, le funzioni Ka e Kb tengono conto delle differenze architetturali e del tipo di lavoro svolto.

Le funzioni K, per quanto detto a proposito di SSP, oltre ad essere variabili in funzione del workload (WKL), saranno variabili in funzione dell'utilizzo ovvero:

Kx = f(WKL , Ux)

E fissato il WKL avremo quindi:

Kx = f(Ux)

Se poniamo Ua=1 (SSP=100%) per il Sistema A e:

WLF = Ka/Kb= f(Ua)/f(Ub)

Potremo scrivere la relazione di partenza nella forma:

Ca = (Cb * Ub) / WLF

Se poniamo Ub uguale a SSP di B, con questa relazione possiamo esprimere la capacità necessaria sul Sistema Centrale A per eseguire il lavoro che porta il Sistema B al System Saturation Point, cioè abbiamo realizzato una equivalenza al limite di utilizzo massimo tra A e B in funzione del solo WLF.
La funzione WLF fornisce risultati espressi in unità di misura dimensionalmente uguali a Cb/Ca.

Per dare un valore pratico a quanto finora trattato dobbiamo quindi essere in grado di fissare una coppia di valori di utilizzo per le piattaforme che vogliamo confrontare e misurare lo stesso Tipo di Lavoro su entrambe. Ad esempio 60% per il Sistema B e 100% per il Sistema A

Per la coppia individuata WLF può definirsi costante e ad esempio di valore $\mathbf{WLF_{ab(100/60)}}$, in questo caso e solo per la coppia di utilizzi considerati ed il tipo di lavoro in esame varrà allora la semplice relazione:

$$Ca = Cb * Ub / \mathbf{WLF_{ab(100/60)}}$$

nella quale compaiono tutte costanti tranne Cb; mediante questa relazione sarà particolarmente semplice, a meno di definire una metrica equivalente, determinare la capacità Ca equivalente a Cb sulle diverse piattaforme Informatiche A e B.

Il problema inizialmente molto complesso si riduce ora a determinare sperimentalmente i valori di WLF al SSP per le coppie di Piattaforme che vogliamo confrontare e a definire una metrica equivalente tra le Piattaforme A e B.

Una prima semplice applicazione del metodo si ha utilizzando la frequenza di Clock del processore per misurare le capacità della Piattaforma. Il problema che vogliamo risolvere è il seguente:

Dato il Sistema A con n processori di frequenza ω_A con un utilizzo U_A, quanti processori saranno necessari sul Sistema B, con frequenza di Clock ω_B per eseguire lo stesso lavoro con un utilizzo U_B.

Avremo quindi:

$$\mathbf{m = (n * \omega_A * U_A) / (\omega_B * U_B * WLF)}$$

dove WLF è la costante misurata nel range di utilizzo previsto per la coppia di Piattaforme Informatiche A e B.

I risultati ottenuti con un simile calcolo sono sufficientemente approssimati in quanto la frequenza di clock del processore non è un elemento esaustivo per definire le prestazioni di un Sistema.

Recentissimamente sono stati introdotti criteri più complessi basati sulla equivalenza tra i MIPS e alcune grandezze utilizzate da altri benchmark come ad esempio RPE ??? introdotti da Ideas International e pubblicati.

Questa introduzione ha dato modo di ottenere relazioni del tipo:

$$\mathbf{MIPS = (rPerf * 100 * U_A) / (U_B * WLF)}$$

dove compare ancora WLF come definito precedentemente ed il valore rPerf che rappresenta il fattore di conversione MIPS/RPE, con una relazione del tipo:

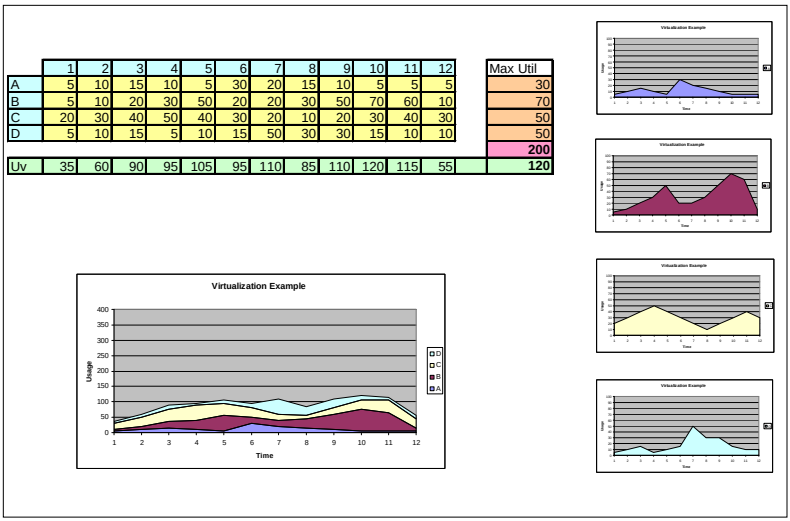
$$\mathbf{rPerf = RPE_A / KRPE}$$

il valore KRPE non è di pubblico dominio.

Questa relazione, una volta conosciuto rPerf, consente di risolvere il problema del calcolo dei MIPS equivalenti ad un Sistema di RPE note.

Poiché l'utilizzo U_X non è costante nel tempo, le relazioni qui introdotte sono tutte funzioni del tempo e quindi presentano un andamento variabile rispetto ad esso. Ai fini del nostro problema tuttavia, la dipendenza esplicita dal tempo non è importante essendo conveniente nei processi di dimensionamento sempre riferirsi all'utilizzo massimo U_{max}. Non è

opportuno infatti come spesso si fa riferirsi a valori medi su grandi periodi, bensì sul massimo calcolato su un intervallo sufficientemente piccolo rispetto ai tempi di risposta attesi dalla Applicazione Informatica. La dipendenza degli utilizzi dal tempo sarà invece molto importante nei processi di dimensionamento di Sistemi Virtualizzati: in questo caso infatti la possibilità di condividere risorse tra vari utilizzatori (Sistemi Operativi) rende fortemente opportuno valutare la dipendenza dal tempo della funzione Utilizzo e rende errato riferirsi ai Massimi.



	1	2	3	4	5	6	7	8	9	10	11	12	Max Util
A	5	10	15	10	5	30	20	15	10	5	5	5	30
B	5	10	20	30	50	20	20	30	50	70	60	10	70
C	20	30	40	50	40	30	20	10	20	30	40	30	50
D	5	10	15	5	10	15	50	30	30	15	10	10	50
													200
Uv	35	60	90	95	105	95	110	85	110	120	115	55	**120**

Figura 150 Dimensionamento di Sistemi virtualizzati

La Figura 150 mostra a titolo di esempio l'utilizzo nell'arco di una ipotetica giornata e per intervalli di tempo fissati arbitrariamente di quattro Risorse Informatiche (ad esempio Server): la tabella in alto ne mostra i valori numerici. Come si può facilmente notare, essendo l'utilizzo variabile nel tempo con andamento differente, se si ipotizza di consolidare le quattro risorse su di una sola (condivisa) si osserveranno due importanti fenomeni:

1. Il Massimo utilizzo della risorsa condivisa (120) sarà sempre minore o uguale della somma dei massimi utilizzi di ogni singola risorsa (200).

2. Il Punto (periodo della giornata) in cui si realizza il Massimo utilizzo della risorsa condivisa potrebbe non coincidere con nessuno dei punti in cui si realizzano i Massimi delle singole risorse.

Queste due importanti osservazioni erano state poste alla base del concetto di virtualizzazione ed usate come giustificazione della virtualizzazione stessa.

Cerceremo ora di fornire un criterio quantitativo per il dimensionamento di processi di Consolidamento dei serventi: quanto stiamo per dire varrà sia nel caso di Serventi che usano la stessa Piattaforma Informatica, sia nel caso di Piattaforme Informatiche eterogenee, fatto salvo quanto indicato all'inizio del capitolo per la normalizzazione delle metriche.

Il problema del dimensionamento di Infrastrutture virtualizzate che consolidano molti Server diversi viene qui esposto al netto del carico aggiuntivo (overhead) dovuto al virtualizzatore, soprattutto quando esso è implementato mediante Software. In questo caso tale carico, generalmente noto, sarà semplicemente aggiunto al risultato finale sotto forma di percentuale o di numero puro.

L'esperienza ci dice che nei processi di dimensionamento di infrastrutture virtualizzate convenga partire sempre da dati di carico dei singoli serventi effettivamente **misurati** e NON dedotti da stime o medie: ciò in quanto gli effetti delle medie su periodi lunghi potrebbero inficiare pesantemente il risultato.

La prima cura da ottenere per tale misura sarà quella di scegliere l'intervallo di campionamento: osserviamo infatti la figura seguente, essa mostra l'andamento in Ghz del "consumo" di quattro serventi fisici (che prevediamo di consolidare) e della loro somma (area blu), avendo misurato i valori medi ogni minuto.

Dal grafico (Figura 151)si evince che il consumo massimo, cioè la dimensione che dovrà avere il Sistema virtualizzato per poter contenere il carico dei quattro serventi divenuti logici, si trova intorno a 8 Gigaherz.

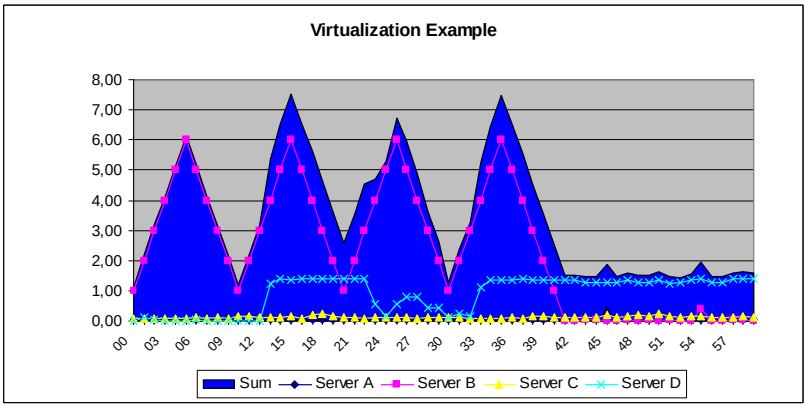


Figura 151 Importanza dell'intervallo di campionatura

La figura 152 invece, a partire dagli stessi dati (relativa ad una ora di lavoro), mostra i massimi che si ottengono calcolando le medie rispettivamente ogni 5, 10, 20 e 30 minuti (Barrette Continue). Come si vede ogni qualvolta l'intervallo di calcolo della media viene ampliato, il valore massimo si abbassa, diventanto alla fine più basso di 3

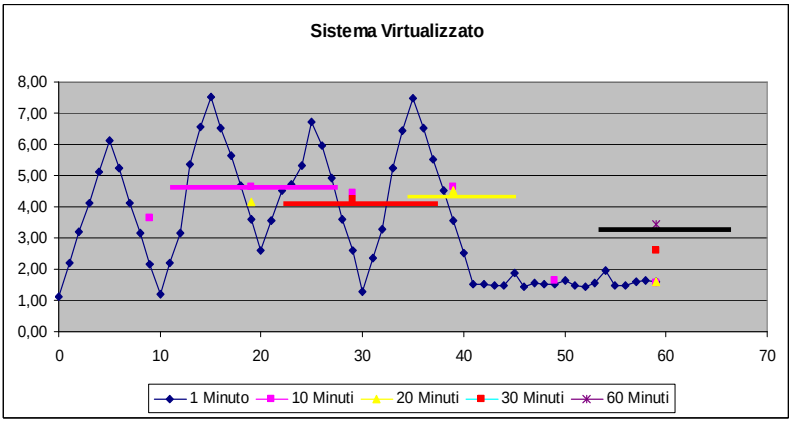


Figura 152 Dipendenze dell'intervallo di campionaturea

Gigahertz,cioè fornendo un'indicazione di un Sistema Virtualizzato in grado di contenere i nostri quattro server sempre più "piccolo".

Ci chiediamo quindi quale possa essere un intervallo di calcolo della media ragionevolmente adeguato ai nostri scopi, anche perché potremmo osservare che riducendolo ulteriormente (cioè passando ad esempio da un minuto ad un secondo) rischieremo l'effetto inverso e cioè quello di percepire un consumo massimo sempre maggiore vanificando l'effetto della virtualizzazione: è facile infatti dimostrare che considerando intervalli sempre più piccoli ne esisterà almeno uno per il quale il consumo della risorsa sarà pari alla potenza complessiva disponibile (100%). Una ragionevole approssimazione può essere quella di considerare intervalli dello stesso ordine di grandezza della durata media di una transazione Applicativa (ad esempio minuti per le operazioni online con intervento umano ovvero ore per le transazioni di tipo Batch), nella cosapevolezza che un sottodimensionamento del Sistema risultante (in media) sortirà l'effetto di svolgere lo stesso lavoro in un tempo maggiore, cioè di peggiorare le prestazioni. Una volta fissato l'intervallo di calcolo delle medie e misurato l'utilizzo delle risorse che vogliamo virtualizzare per un periodo ragionevolmente rappresentativo (ad esempio un giorno, un mese, ecc.) ci ritroveremo in mano per ogni servente una serie di misure del tipo:

$$U_x = [\, U_{x1}, U_{x2}, \ldots\ldots U_{xn}]$$

che rappresenta una serie di misure relative alla risorsa X per n Intervalli di tempo uguali.

Se consideriamo **k** risorse diverse e se sincronizziamo il numero **n** e la durata degli intervalli, le nostre misure costituiranno una matrice rettangolare di **k** righe ed **n** colonne:

$$\begin{vmatrix} U_{11} & U_{12} & U_{13} & & U_{1n} \\ U_{21} & U_{22} & U_{23} & & U_{2n} \\ U_{31} & U_{12} & U_{33} & & U_{3n} \\ \ldots\ldots & & & & \\ U_{k1} & U_{k2} & U_{k3} & & U_{kn} \end{vmatrix}$$

La somma di ciascuna colonna rappresenta per ogni intervallo di tempo l'utilizzo di un Sistema che virtualizzi le k risorse:

$$U_{Vi} = \sum_{j=1}^{k} U_{ji}$$

Considerando quindi la serie numerica data dalla somma di tutte le n colonne della matrice:

$$U_V = \left[\sum_{j=1}^{k} U_{j1} \;, \sum_{j=1}^{k} U_{j2} \;, \ldots \sum_{j=1}^{k} U_{jn} \right]$$

Essa rappresenterà l'utilizzo della risorsa in grado di Virtualizzare le k risorse di partenza per tutti gli n intervalli considerati. L'utilizzo massimo che stiamo cercando, cioè quello che ci consentirà di determinare la capacità del Sistema Virtualizzato, sarà dato da:

$$\tilde{U}_V = \mathrm{Max}\left[\sum_{j=1}^{k} U_{j1} \;, \sum_{j=1}^{k} U_{j2} \;, \ldots \sum_{j=1}^{k} U_{jn} \right]$$

Ricordiamo ancora una volta che il massimo utilizzo così calcolato è sempre minore o uguale alla somma dei massimi utilizzi di ogni risorsa considerata.

Vogliamo ora applicare quanto discusso al calcolo della capacità necessaria a consolidare una Server farm costituita da N Serventi diversi di diversa Piattaforma Informatica. Un primo facile calcolo può essere basato sulla Frequenza di Clock: in questo caso ragruppiamo i serventi in L gruppi.

In ciascun gruppo poniamo i serventi che hanno lo stesso utilizzo massimo che indichiamo con U_{bmax}

la stessa frequenza di clock che indicheremo con Ck_B e che svolgono lo stesso tipo di lavoro.

Contiamo il numero totale di processori (Cores) presenti in ogni gruppo; per ciascuno degli L gruppi si avrà:

$$[Nproc_Z]_h = \frac{[Ck_B * Nproc_B * U_{Bmax}]_h}{[WLF]_h * Ck_Z * U_Z}$$

Il numero di Cores del System z necessari e probabilmente sufficienti al consolidamento in base alla frequenza di clock sarà dato da:

$$Totproc_Z = \sum_{h=1}^{L} [Nproc_Z]_h = \sum_{h=1}^{L} \frac{[Ck_B * Nproc_B * U_{Bmax}]_h}{[WLF]_h * Ck_Z * U_Z}$$

Vediamo ora un semplice esempio.

Vogliamo calcolare quanti processori System z di tipo IFL a frequenza di 1.8 Gigahertz usati al 100% sono necessari e probabilmente sufficienti a consolidare:

- 10 Sistemi x86 (B) con Clock di 1 Ghz, Utilizzo 60%, ciascuno con 4 cores e WLF=2
- 5 Sistemi x86 (C) con Clock di 3,5 Ghz, Utilizzo 60%, ciascuno con 2 cores e WLF=1

System B :
1. Architecture X86
2. Clock Speed = 1 Ghz
3. Utilization Rate = 60%
4. Number of Processors (core) = 4
5. WLF_{ZB} = 2
6. Number of Sys like this = 10

System Z :
1. Architecture System z
2. Clock Speed = 1,8 Ghz
3. Utilization Rate = 100%
4. Number of Processors (core) = x
5. WLF_{ZA} = 2
6. WLF_{ZB} = 1

System C :
1. Architecture X86
2. Clock Speed = 3,5 Ghz
3. Utilization Rate = 60%
4. Number of Processors (core) = 2
5. WLF_{ZC} = 1
6. Number of Sys Like this = 5

IFL = 6,67 + 11,67 ~ 19

Figura 153 Esempio di calcolo della potenza richiesta

In questo esempio occorrono circa 19 processori (Cores) di tipo IFL.

Volendo eseguire infine un calcolo più accurato e disponendo del valore della equivalenza RPE→ MIPS (questo dato non è di pubblico dominio), si potrebbe operare in modo simile raggruppando tutti i Server con lo stesso WLF in gruppi e di questi sommando i valori di Rperf relativi (Rperf è proporzionale a RPE e tiene conto dell'equivalenza sopra detta) ottenendo k gruppi. In questo caso avremo:

$$MIPS_i = \frac{[rPerf_B * 100 * U_B]_i}{[WLF]_i * U_Z}$$

Questa formula vale per ogni intervallo elementare di tempo, come discusso precedentemente, quindi, per ogni intervallo, considerando tutti i gruppi (k), avremo:

$$TotMIPS_Z = \sum_{i=1}^{k} [MIPS]_i = \sum_{i=1}^{k} \frac{[rPerf_B * 100 * U_B]_i}{[WLF]_i * U_Z}$$

A questo punto applichiamo il metodo discusso in precedenza per il campionamento degli intervalli e calcoliamo la matrice degli utilizzi, ponendo in ciascuna riga i valori di ciascuno dei k gruppi per n intervalli di misura:

$$\tilde{U}_V = \text{Max}\left[\sum_{j=1}^{k} U_{j1} \;, \sum_{j=1}^{k} U_{j2} \;, \ldots \sum_{j=1}^{k} U_{jn} \right]$$

IL valore dei Mips totali ricercato sarà dato dal Massimo delle somme così calcolate ovvero:

$$\text{TotMIPS}_Z = \text{Max}\left[M_{j1}, M_{j2}, \ldots\ldots M_{jn}, \right]$$

Nella quale ciascun elemento sarà dato da una relazione del tipo:

$$M_j = \sum_{j=1}^{k} \frac{[rPerf_B * 100 * U_B]_j}{[WLF]_j * U_Z}$$

Il metodo qui illustrato è sufficientemente rigoroso e consente di ottenere risultati molto accurati sotto due importanti condizioni:

1. Disporre di una misurazione significativa degli utilizzi dei Serventi di partenza, secondo le tecniche di campionamento sopra descritte.
2. Disporre di valori accurati per le costanti WLF nell'intervallo di utilizzo misiurato e di RPE ed Rperf.

Sconsigliamo fermamente di operare dimensionamenti a partire da dati di carico stimati e NON Misurati e senza una profonda conoscenza del tipo di lavoro. I valori di WLF possono essere misurati in modo sperimentale e poi utilizzati per casi simili.

4.0.7 Esempi di Applicazione del Metodo

4.0.7.1 Esempio #1: Uso del TCO per giustificare Linux su un Sistema Centrale

Si consideri un'infrastruttura informatica costituita da 50 Serventi di architettura x86 uguali i quali gestiscono un'applicazione di commercio elettronico con prodotti software funzionanti sul Sistema Operativo Linux.

L'infrastruttura ha la seguente composizione:

- 20 Serventi Http Apache
- 10 Serventi Applicativi Java su TomCat
- 4 Serventi di Data Base MySql
- 16 Serventi di Servizio (LDAP, Auth, Firewall)

ed è disposta come nella figura:

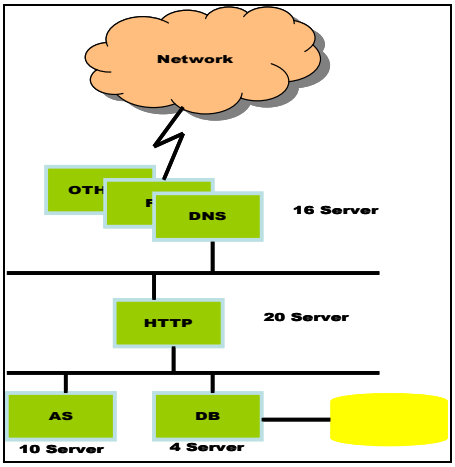


Figura 154 Infrastruttura di partenza

Data la particolare criticità della applicazione (Commercio elettronico), si stima che la infrastruttura fornisca alla azienda che la possiede un ricavo giornaliero pari a €100.000. Inoltre per motivi tecnici l'attuale infrastruttura comporta ogni giorno un fermo di 60 minuti non eliminabile.
La figura sottostante esprime i valori calcolati del TCOp della infrastruttutura attuale a partire dai dati iniziali. Assumiamo di utilizzare un

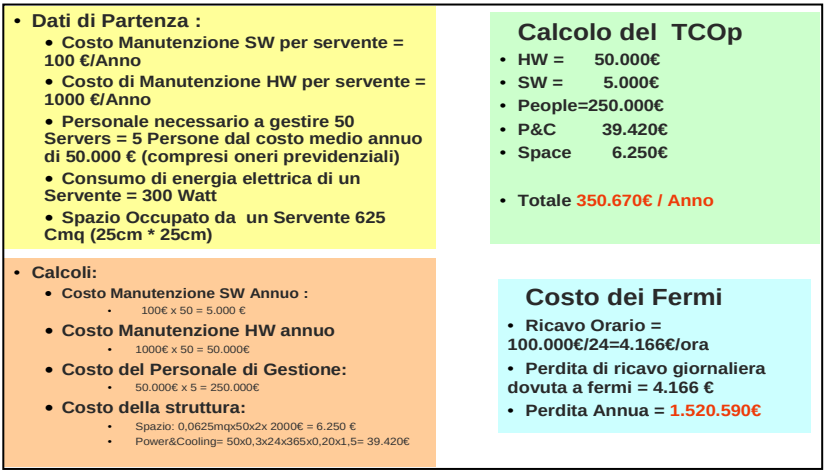


Figura 155 Riepilogo caso di Studio

Sistema di z/Architecture col Sistema Operativo Linux per consolidare ed ottimizzare la nostra infrastruttura secondo lo schema della figura sottostante.

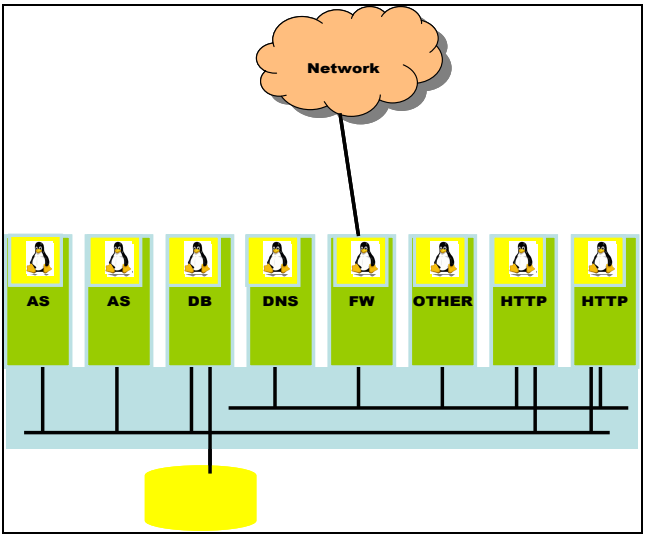


Figura 156 Infrastruttura di arrivo (Target)

Si osservi che grazie alle possibilità offerte dalla nuova architettura, la nuova infrastruttura risulta composta da un numero inferiore di Serventi Logici e da un solo Servente Fisico Virtualizzato. Ciò è dovuto alle migliori caratteristiche di scalabilità del Sistema ed alla maggiore affidabilità dell'hardware.

Stimiamo inoltre che tale infrastruttura, grazie alle possibilità gestionali di z/VM, possa diminuire i fermi giornalieri riducendoli a soli 20 Minuti. L'infrastruttura risultante sarà quindi composta da:

- 2 Serventi Logici di http (Apache)
- 2 Serventi Applicativi Java (TomCat)
- 1 Servente di Data Base MySQL
- 4 Serventi logici di Servizio (Firewall, DNS, ecc.)

Con queste ipotesi le nuove figure di costo sono le seguenti:

Figura 157 Esempio del nuovo calcolo

Per poter determinare il TCA della soluzione consolidata occorrerà prima stabilire i costi della operazione di consolidamento ed ottimizzazione, in particolare:

- Determinare la potenza richiesta ed il Sistema Necessario
- Determinare i costi di migrazione
- Determinare altri costi di investimento

Gli elementi del nostro caso sono contenuti nella figura successiva:

- **Costo dell'Operazione di Consolidation :**
  - **Definiamo un periodo di tempo consistente con la vita media della Applicazione (ad esempio 3anni = 36 Mesi).**
  - **Calcoliamo il TCOp annuale per la nuova (TCOp new) e vecchia (TCOp old) infrastruttura**
  - **Calcoliamo il TCA (total Cost of Acquisition) della nuova Infrastruttura dove :**
    - $TCA = \Sigma(CostoHw) + \Sigma(CostoSw) + \Sigma(CostoMigrazione)$

L'operazione risultera' economicamente conveniente se:

$\Sigma(TCOp\ old) > \Sigma(TCOp\ new) + TCA$

Dove la somma e' estesa al periodo di tempo considerato (36 Mesi) mentre TCA generalmente non dipende dal tempo.

Nel Nostro Caso :

TCOp old x 3 = 1.070.010 €; TCOp new x 3 = 675.072 €

TCA < 394.938 € (non considerando il valore dei Fermi Macchina)

TCA < (3.043.371€ + 394.938€) = 3.438.309€ (considerando la riduzione dei Fermi dovuta alla nuova Infrastruttura).

Figura 158 Riepilogo Conclusioni

Si osservi che è stato utilizzato un periodo di tre anni per il calcolo.

Si osservi infine che l'avere introdotto il costo (valore) della diminuzione del Fermo Macchina comporta una sostanziale variazione nel caso di studio, infatti, senza considerare questa grandezza, il valore del TCA accettabile è di un ordine di grandezza inferiore a quello del caso calcolato con l'introduzione del valore dei fermi macchina.
La determinazione del valore dei fermi macchina non è così immediata e spesso non è possibile determinarla con accuratezza.[20]
Di seguito i grafici con la rappresentazione del TCO a tre anni e del Ritorno dell'Investimento calcolati nel caso in cui non si sono valutati i fermi macchina.

[20] **Attenzione: i dati di costo immessi sono indicativi e non rappresentano valori reali. L'esempio serve a descrivere il metodo e non è indicativo del risultato.**

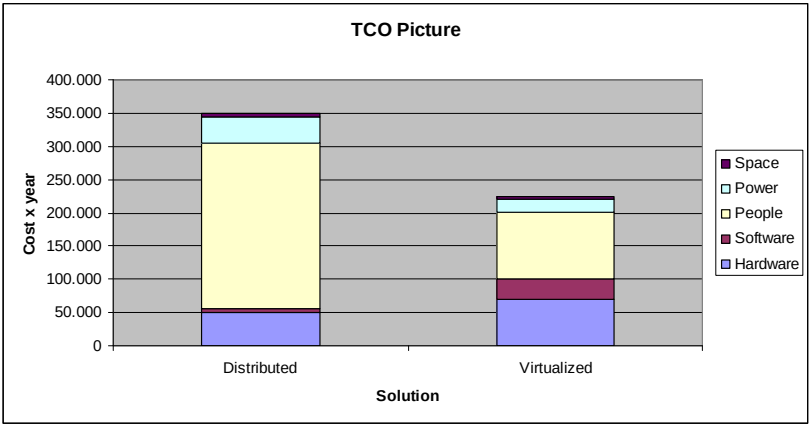


Figura 159 Rappresentazione del TCO per tre anni

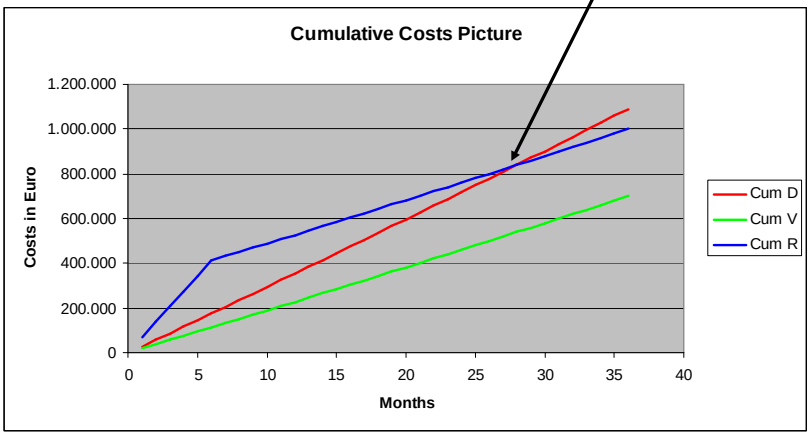


Figura 160 Rappresentazione del Ritorno di Investimento

4.0.7.2 Esempio #2: Uso del Metodo TCO per la Platform Selection

Useremo come secondo esempio un'applicazione del metodo del TCO per valutare la migliore Piattaforma Informatica per eseguire una certa applicazione che, al momento, viene eseguita su un gran numero di Serventi distribuiti sul territorio.

Gli elementi dell'applicazione attuale sono:

- Circa 2000 Serventi Distribuiti di architettura x86
- Sistema Operativo AA
- Data Base Server DD
- Applicazione "Client Server" con scarse funzioni locali e chiamate SQL Standard

Le alternative analizzate sono:

- Consolidamento Fisico a parità di Immagini con l'uso di un Software di virtualizzazione per x86
- Consolidamento Logico con riduzioni di Immagini a parità di Sistema Operativo con un virtualizzatore Software su x86
- Consolidamento logico su Sistemi IBM P5 + IBM AIX con Data Base DDBB
- Consolidamento su z/OS + z/Architecture e IBM DB2

Avendo operato i dovuti calcoli esaminiamo le relative figure di Costi Operativi annui.[21]

[21] **Attenzione:i dati di costo immessi sono indicativi e non rappresentano valori reali. L'esempio serve a descrivere il metodo e non è indicativo del risultato.**

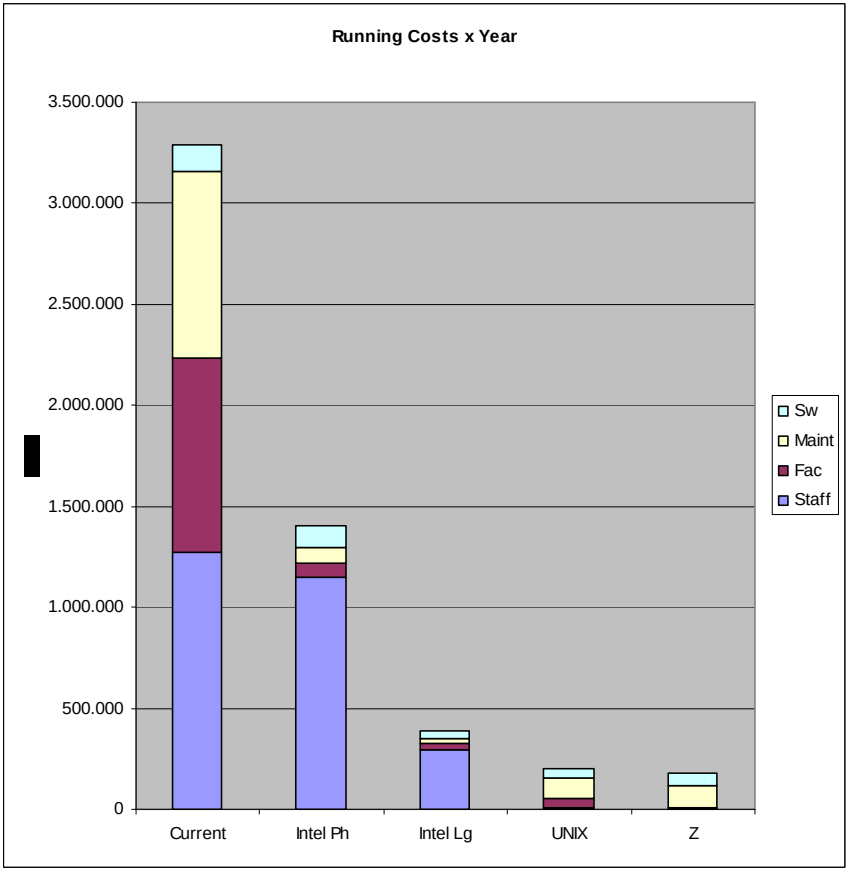


Figura 161 Rappresentazione dei Costi Operativi

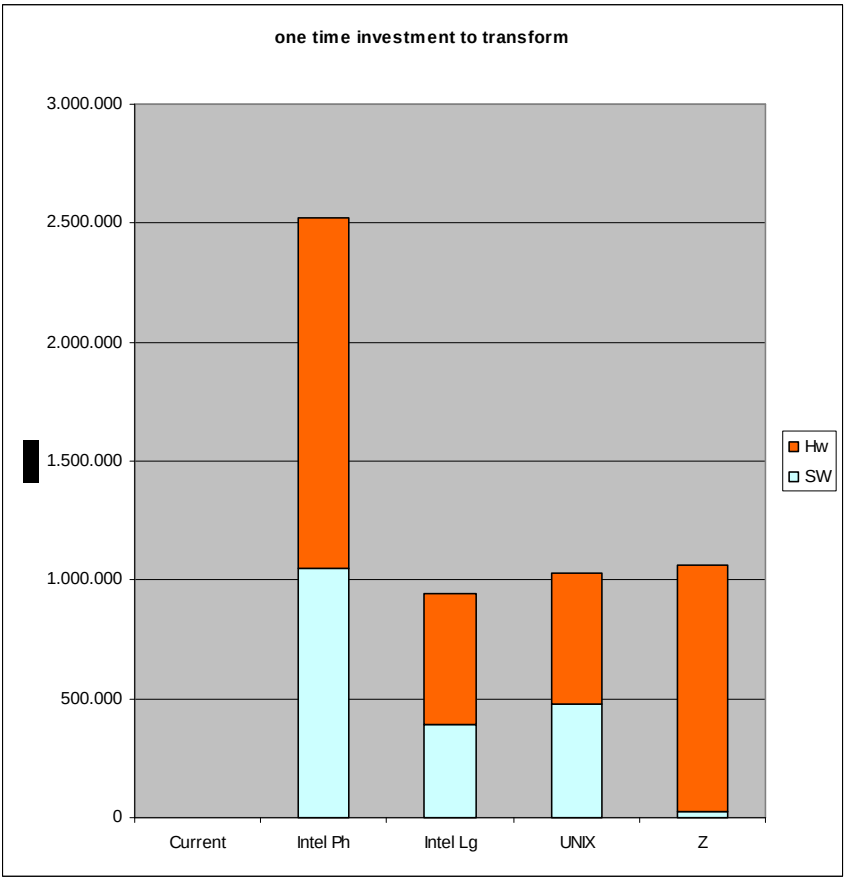


Figura 162 Rappresentazione dei Costi di Acquisizione

Confronto del TCO per un periodo di tre anni:22

22 **Attenzione: i dati di costo immessi sono indicativi e non rappresentano valori reali. L'esempio serve a descrivere il metodo e non è indicativo del risultato.**

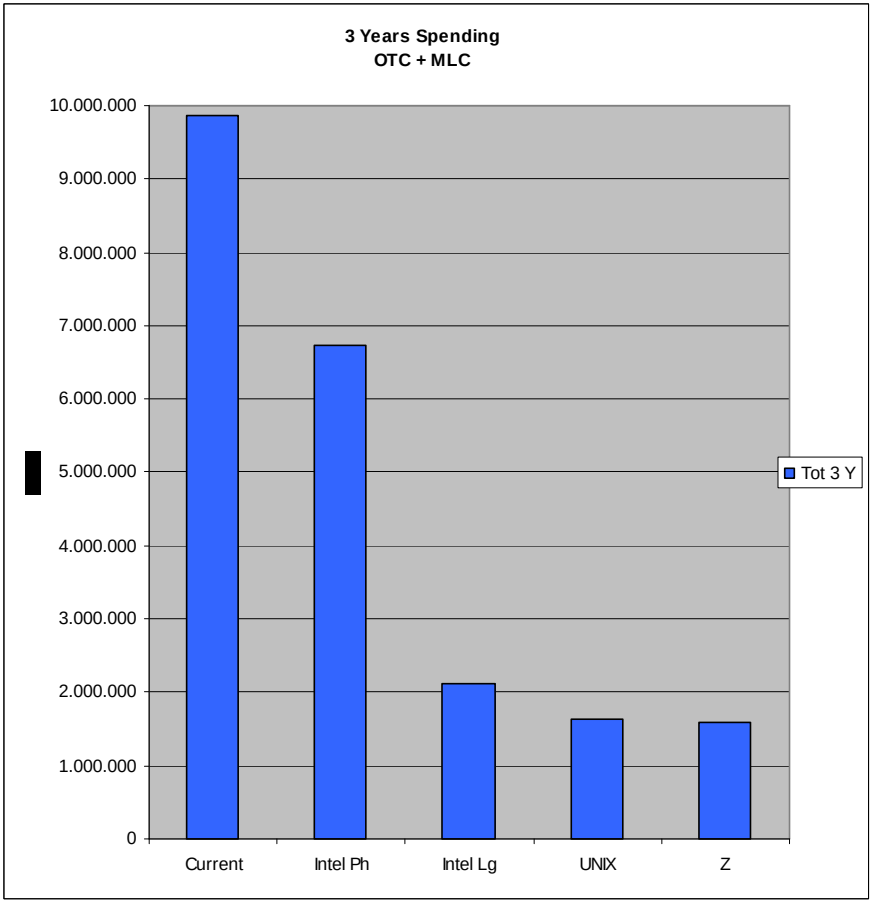


Figura 163 Rappresentazione del TCO per tre anni

4.0.7.3 Esempio #3: Valutazione di una ipotesi di re - hosting

L'ultimo esempio proposto riguarda la valutazione di un ipotesi di re-hosting.[23]

Un utente di una complessa Applicazione industriale ha la sensazione che la sua Infrastruttura Informatica, basata principalmente su un Sistema Centrale, sia eccessivamente "costosa" e pensa quindi di cambiarla operando un re-hosting delle sue applicazioni. Poiché si tratta di una struttura complessa non tutto il Software che costituisce le Applicazioni risulta portabile.

L'utente opera pertanto una prima analisi tecnica del suo Software per verificare la sua portabilità ed ottiene così il valore del Costo necessario al processo di re-hosting. Decide pertanto di confrontare due possibili casi di re-hosting:

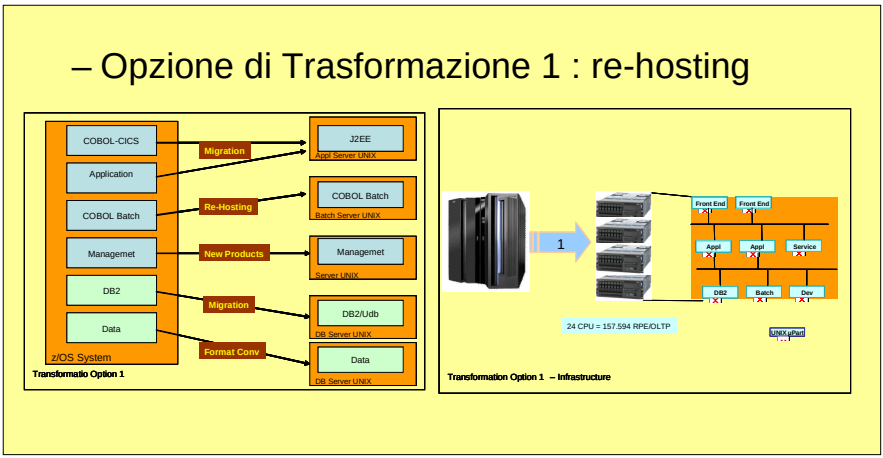
1. Cambiare Piattaforma Informatica da (z/OS + System z) ad (AIX + RISC) operando le attività di porting previste (UNIX)
2. Rimanere all'interno della stessa Piattaforma Informatica modernizzando il Software (anche questa ipotesi richiederà dei lavori di porting) (zNALC)

Le figure seguenti mostrano il cammino di porting nei due casi e le figure di costo derivanti. Osserviamo che entrambe le soluzioni presenteranno un TCO a tre anni maggiore della soluzione corrente: questo è naturale in quanto entrambe le opzioni prevedono forti investimenti applicativi. Parimenti osserviamo che entrambe le soluzioni presenteranno un TCOp più basso e questa era la ragione corretta per la quale si era deciso di iniziare l'operazione di re-hosting.

---

[23] **Attenzione: i dati di costo immessi sono indicativi e non rappresentano valori reali. L'esempio serve a descrivere il metodo e non è indicativo del risultato.**

.



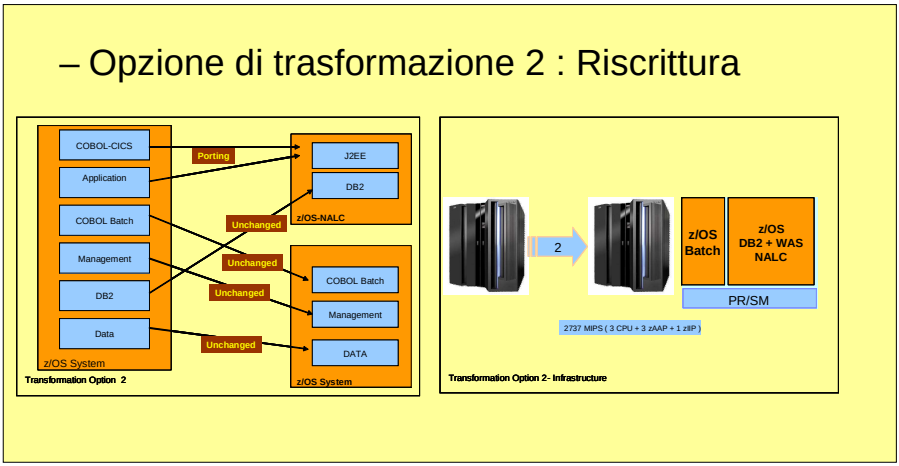


Figura 164 Ipotesi di Trasformazione

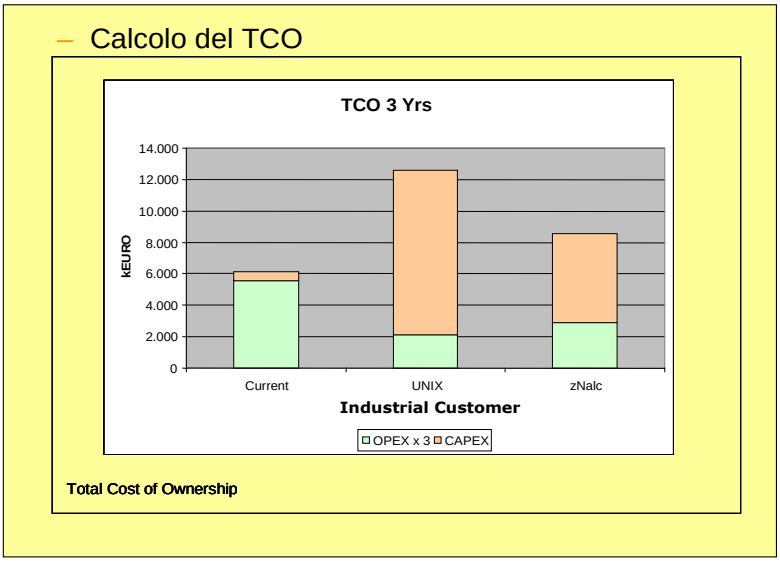


Figura 165 Rappresentazione del TCO a tre anni

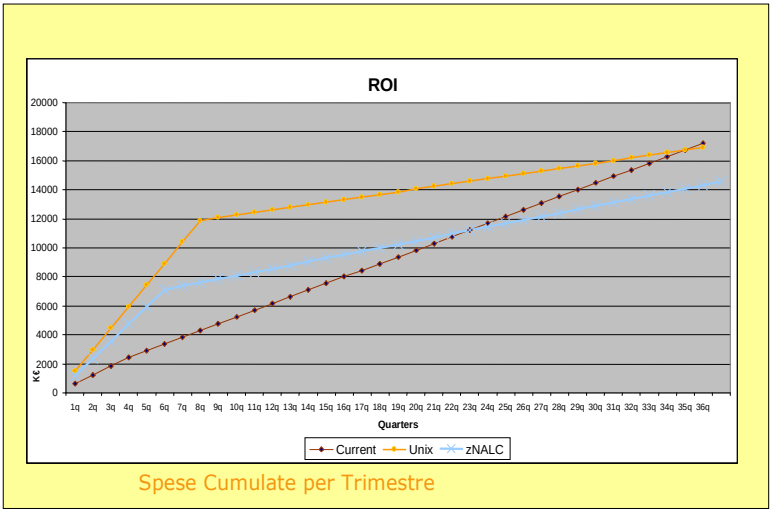


Figura 166 Curva del Ritorno di Investimento

Infine la Curva del Ritorno dell'Investimento mostra che l'operazione non risulta genericamnte conveninte in quanto il ROI viene pesantemente spostato in avanti a causa dei grandi investimenti che occorre fare.

Quindi pur confrontanto soluzioni che presentano costi di esercizio TCOp più bassi degli attuali in processo di re-hosting, nel caso trattato non risulta conveniente.

In più la soluzione di mantenimento della Piattaforma Informatica corrente modificando le applicazioni in modo da diminuire i costi di esercizio (ad esempio riducendo le licenze Software) consente di ottenere un Ritorno dell'Investimento in tempi più brevi, pur presentando un TCOp leggermente maggiore a causa dei minori investimenti necessari.

4.0.8 Conclusioni

In questo capitolo abbiamo cercato di esprimere ordinatamente tutti i concetti relativi alla Selezione della Piattaforma Informatica e ai processi di ITRO, fornendo un metodo per il confronto dei costi (TCO) e metriche per il dimensionamento dei Sistemi.

Abbiamo anche visto come il Sistema Centrale possa avere un ruolo nei processi di Server Consolidation, che sono sovente il primo passo per i processi di Ottimizzazione basati sulla riduzione dei costi.

Abbiamo visto infine come il Sistema Operativo Linux e la Virtualizzazione siano elementi abilitanti per il Processo di Server Consolidation e più in generale di ITRO, in quanto lo possono facilitare notevolmente.

Concludiamo affermando che ogni processo di ITRO deve individuare prima dei Drivers su cui espimersi; se il Driver individuato è la riduzione dei Costi, il metodo TCO può fornire dei criteri oggettivi di confronto tra la soluzione attuale e una serie di alternative: per operare questo calcolo risulterà essenziale calcolare la capacità richiesta sul Sistema di arrivo, presunto virtualizzato, tale capacità può essere determinata con le tecniche sopra illustrate.

Appendice A: Sicurezza e Crittografia dei Mainframe

[a cura di Massimo Leoni]

Introduzione

Prima di affrontare la sicurezza nel *System z* nei suoi vari aspetti, è necessario dare una panoramica del problema della sicurezza nell'IT, individuarne gli elementi portanti e i cardini di riferimento. Innanzi tutto ci si deve porre la domanda, anche se sembrerà pleonastica: che cos'è la sicurezza? Il termine sicurezza rappresenta generalmente una forma d'assicurazione, con un costo associato, a fronte di un valore che si vuole proteggere e per raggiungere uno stato di confidenza per "sentirsi" sicuri, a fronte di:

- Rischi individuati a fronte di eventi accidentali o deliberati
- Normative cui conformarsi

La tabella 1 seguente riporta una possibile classificazione dei tipi di rischi associati ad un ambiente IT[24].

Umano		**Ambientale**
Volontario	**Accidentale**	
Intercettazioni	*Errori e omissioni*	*Terremoto*
Modifica dell'informazione	*Cancellazione di file*	*Fulmine*
Hacking	*Instradamento errato*	*Allagamento*
Virus (Codice Maligno)	*Incidenti fisici*	*Incendio*
Furto		

Tabella 1

[24] IT – Information Technology

Dalla tipologia di eventi riportati possiamo rilevare un ulteriore raggruppamento in eventi che coinvolgono la sfera fisica ed altri invece che si esauriscono nel dominio logico. Gli eventi che afferiscono al mondo fisico sono studiati delle discipline del "*Disaster Recovery*" e della "*High Availability*" che sviluppano teorie e soluzioni in questi campi. Noi invece ci occuperemo della sicurezza logica, cioè dello studio e della definizione di soluzioni che riguardano eventi volontari o accidentali che possono compromettere l'integrità del sistema di elaborazione e/o dei dati. Procediamo ora a sviluppare un metodo e un modello che permetta di tracciare una strada per la costruzione e la comprensione di una soluzione di sicurezza. Innanzitutto è necessario classificare i così detti "*asset*", cioè le risorse (dati, programmi, comunicazioni, ecc. per quanto riguarda la sicurezza logica) che sono ritenuti necessari all'operatività aziendale e che sono alla base dell'integrità dei processi dell'impresa. Classificare, in questo contesto, significa identificare le risorse ed attribuire loro un valore aziendale. Un valore che deve potere essere "assicurato".

Ad un primo livello di astrazione questa "assicurazione" si traduce nel definire delle politiche che appunto mirino a mitigare il rischio, difendendo gli *asset* da minacce o adeguando comportamenti nel rispetto di normative richieste dal contesto di mercato in cui opera l'azienda (es. Legge sulla Privacy, Sarbanes Oxley Act ecc.). Ora che abbiamo le politiche, si pone la questione di come farle applicare. Due alternative sono possibili, l'utilizzo di:

1. **Disciplina -** Vengono diramate e messe a conoscenza dei dipendenti delle norme comportamentali e organizzative a protezione degli *asset* e volte anche al controllo del loro accesso;
2. **Meccanismi -** Vengono utilizzate delle tecnologie e dei meccanismi che permettono di costringere i comportamenti dei dipendenti in aderenza alle politiche aziendali.

I meccanismi sono la modalità che, dal punto di vista della tecnologia ed in questa trattazione, ci interessano maggiormente. Quindi vediamo ora come collocare e classificare i vari meccanismi per poterli impiegare profittevolmente nella sicurizzazione di ambienti *System z*.

Partiamo con una classificazione o domini della sicurezza logica, che per comodità baseremo su quella proposta nello standard ISO 7498-2 (rappresentato graficamente in Figura 167).

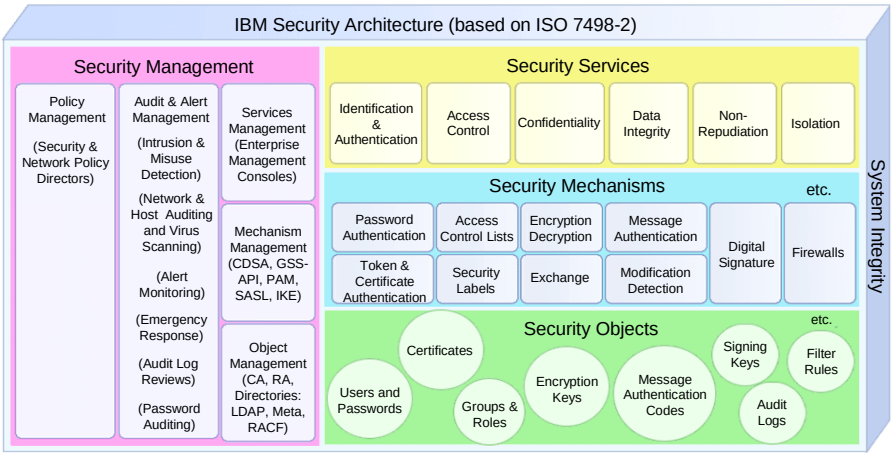


Figura 167 IBM Security Architecture

Questa classificazione permette di catalogare gli ambiti di applicazione dei meccanismi di sicurezza nelle seguenti categorie:

- Autenticazione
- Autorizzazione e controllo degli accessi
- Riservatezza/Confidenzialità
- Auditabilità e Non Ripudio

che andiamo a collocare sul modello riportato in Figura 168.

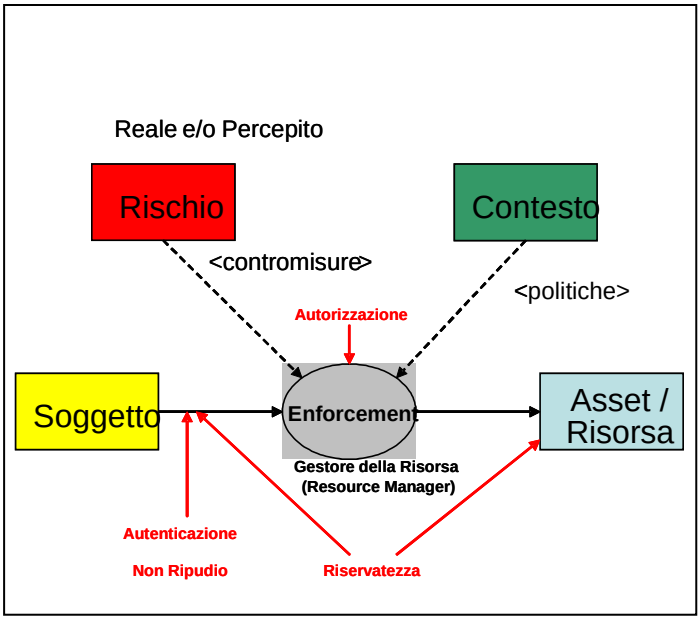


Figura 168 Catalogazione dei meccanismi di sicurezza

Nel modello si evidenzia come un soggetto, che accede ad un asset o ad una risorsa tramite un mediatore (gestore delle risorse o *resource manager*), deve essere in qualche modo autenticato. Inoltre anche il mezzo di comunicazione utilizzato per lo scambio di informazioni tra il soggetto e il gestore delle risorse può essere protetto (riservatezza e non ripudio) così come i dati stessi (*Asset*/Risorsa). La decisione di lasciar accedere il soggetto alla risorsa viene presa dal gestore delle risorse in base a delle politiche derivate dal contesto e da meccanismi (contromisure) che invece sono correlate al livello di rischio legato all'accesso di queste risorse.

Tutto questo, per poter rispondere in modo coerente alle aspettative di chi implementa soluzioni di sicurezza, si deve fondare obbligatoriamente sulla integrità del proprio contenitore (Piattaforma Hardware, Sistema Operativo, ecc.), in modo da garantire che le politiche e la semantica dei meccanismi siano rispettate.

Il paragrafo precedente suona come l'enunciazione di un assioma. Tutta la sicurezza che si va a costruire si fonda sull'integrità. Ma come si valuta o si verifica la bontà di una soluzione di sicurezza o le affermazioni di integrità fornite dai vari produttori di Hardware e Software? La risposta è la certificazione. Per garantire che una soluzione, un meccanismo, un sistema hardware o software rispetti le specifiche di disegno, le pratiche riconosciute di realizzazione e sia progettato per resistere ad attacchi conosciuti (e sconosciuti), esso deve essere certificato. Le metodologie e gli standard di certificazione più rilevanti per quanto riguarda la sicurezza sono:

- ISO/IEC 17799 (ex BS 7799 – *British Standard 7799*)
- ISO/IEC 15408 (conosciuto come *Common Criteria* o *CC*, i livelli di validazione e certificazione sono riportati in Tabella 2)
- FIPS 140 (*Federal Information Processing Standard* – conosciuto soprattutto per la certificazione di Hardware Crittografico)

Common Criteria Assurance levels
EAL1: *Functionally Tested*
EAL2: *Structurally Tested*
EAL3: *Methodically Tested and Checked*
EAL4: *Methodically Designed, Tested and Reviewed*
EAL5: *Semiformally Designed and Tested*
EAL6: *Semiformally Verified Design and Tested*
EAL7: *Formally Verified Design and Tested*

Tabella 2

L'integrità del System z

Iniziamo ora a percorrere e a collocare, all'interno della piattaforma *System z* e z/OS, i concetti che abbiamo appena introdotto. Analizziamo per primo l'integrità del sistema hardware. Come sappiamo il *System z* è un elaboratore che, attraverso un ipervisore a livello firmware, il PR/SM, può essere virtualizzato, cioè le risorse fisiche della macchina possono essere condivise da più immagini di sistema operativo. In questo caso è necessario garantire che ogni immagine di sistema operativo ospitato in una partizione logica della macchina (LPAR) sia equivalente ad un elaboratore fisico isolato. Per questo il *System z* è certificato, rispetto a questa caratteristica, con i *Common Criteria (CC)* a livello EAL5 (in Germania) e EAL4 a livello mondiale.

Un'altra importante caratteristica hardware del System z sono le chiavi di memoria o "*Storage Key*". Ad ogni blocco di 4KB (pagina) di memoria è associata una chiave (*storage key*) che permette di controllare a livello hardware l'accesso in scrittura e/o lettura al contenuto della pagina. Se un programma ha associato la chiave di memoria 0, significa che può accedere a qualsiasi pagina di memoria reale (di solito questa è la chiave del sistema operativo). Altre chiavi sono assegnate dal sistema operativo ai sottosistemi principali (es. VTAM, JES2, IMS ecc.) che utilizzano a loro volta questo meccanismo per proteggere le proprie aree di memoria da accessi non autorizzati (per un errore o intenzionale) da parte di programmi utente.

Passiamo ora all'integrità del software. L'integrità nel sistema operativo z/OS è un aspetto fondamentale del disegno della piattaforma. Questa caratteristica è talmente importante che viene supportata da una forte dichiarazione da parte di IBM attraverso l' *IBM STATEMENT OF MVS (z/OS) SYSTEM INTEGRITY*, che afferma l'impegno di IBM ad apportare correzioni (a qualsiasi costo) al sistema z/OS per malfunzionamenti che inficino l'integrità del sistema stesso: deve essere impossibile per un programma non autorizzato da un meccanismo sotto il controllo del cliente:

1. aggirare o disabilitare la protezione di *store* (scrittura in memoria) o *fetch* (lettura dalla memoria),

2. accedere ad una risorsa di Sistema protetta da password o da RACF[25], o
3. ottenere il controllo in uno stato autorizzato da programma, cioè essere in supervisor state con una chiave di protezione inferiore a otto (8), o essere autorizzato in *Authorized Program Facility* (APF).

Oltre a questa dichiarazione di IBM, il System z congiuntamente con lo z/OS 1.7 è certificato con i *Common Criteria (CC)* a livello EAL4.

Autenticazione, Autorizzazione e controllo Accessi nello z/OS

Come abbiamo visto in Figura 168, i meccanismi di Autenticazione e Autorizzazione (Controllo degli Accessi) sono utilizzati dal *resource manager* per prendere delle decisioni riguardo l'accesso alle risorse che controlla (ad esempio il sottosistema CICS - *resource manager* - e le sue transazioni - *resources*). Il modello generale di Figura 168 può essere ora contestualizzato nel caso dello z/OS nella figura seguente:

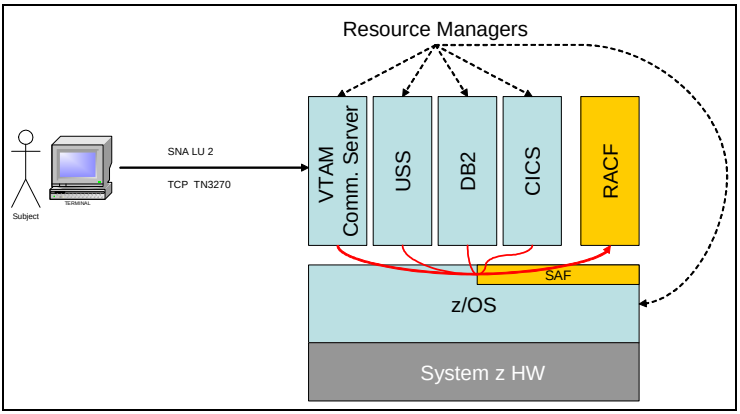


Figura 169 Modello di sicurezza dello z/OS

[25] RACF – Resource Access Control Facility, è il servente di sicurezza del sistema z/OS.

Vediamo ora come le varie componenti e in particolare il RACF, contribuiscono a realizzare la soluzione di sicurezza per lo z/OS.

SAF

La *System Authorization Facility* (SAF) è parte del sistema operativo z/OS e fornisce una serie di interfacce a servizi di sistema per operare l'autenticazione, l'autorizzazione e la tracciatura (*auditing* e *logging*) degli eventi di sicurezza. La SAF fornisce una modalità che si occupa di rispondere a richieste di un gestore di risorse (*Resource Manager).* L'elemento chiave della SAF è il *SAF router* (modulo ICHSFR00). Il *router* è sempre presente in una installazione di z/OS anche se RACF non è installato. Il *SAF router* rappresenta il punto comune di ingresso alle funzioni di sicurezza che possono quindi essere condivise tra prodotti e sistemi. Un gestore delle risorse (*resource manager*) può richiamare il *SAF (z/OS) router* attraverso una macro assembler (RACROUTE) come parte di un flusso che comporti decisioni di controllo di accessi (REQUEST=VERIFY) o di autenticazione di un soggetto (REQUEST=AUTH). Queste funzioni sono chiamate *control points*. Questa nuova infrastruttura di sicurezza fornisce molti benefici per la realizzazione di un sistema sicuro e promuove il suo uso da parte di programmi e componenti del sistema operativo.

RACF

RACF è l'acronimo per *Resource Access Control Facility* ed è una componente fondamentale nell'architettura di sicurezza dello z/OS. Esso incorpora al suo interno diversi elementi del modello della sicurezza, come l'autenticazione e l'identificazione degli utenti e il controllo degli accessi. RACF è quindi un fornitore di servizi alla SAF e fornisce le risposte alle richieste dei vari gestori di risorse. Per completezza di informazione dobbiamo anche dire che esistono sul mercato alternative a RACF come security manager fornite da altri fornitori di software come CA-ACF2 e CA-TopSecret della Computer Associated.

Architettura

Dal punto di vista architetturale, RACF si avvale di una base dati (RACF database) nella quale vengono memorizzati i profili:

- degli utenti (USER)
- dei loro raggruppamenti (GROUPS)
- delle risorse da proteggere
- dei suoi parametri di funzionamento.

Il base ai suoi algoritmi di verifica RACF, basandosi sulle informazioni contenute nella sua base dati, può fornire una risposta alle richieste ad esso pervenute dalla SAF, invocate dai resource manager o da programmi utente. Nella Figura 170 è rappresentata sinteticamente l'architettura di RACF, dove viene mostrato il flusso di una richiesta di accesso ad una risorsa all'interno di un ambiente z/OS. L'utente o il programma innanzitutto devono essere autenticati dal resource manager o dal componente di sistema operativo a cui richiedono il servizio. Risultato dell'autenticazione, che avviene richiamando appunto RACF, è la creazione da parte dello z/OS di un ACcessor Environment Element (ACEE). L'ACEE rappresenta l'identità di un utente o di un programma (Address Space o Task/Thread) ed è l'elemento che viene utilizzato in seguito per il processo di autorizzazione.

Quando un utente o un programma autenticato richiede l'accesso ad una risorsa, il resource manager chiede a RACF se l'utente è autorizzato ad essa. In questa richiesta vengono passate a RACF seguenti informazioni:

- l'ACEE dell'utente (o del programma)
- il nome della risorsa che l'utente vuole accedere e
- il tipo di accesso richiesto (es. READ, WRITE, UPDATE, ...).

RACF verifica se l'utente è autorizzato, eventualmente registra un record di audit avvalendosi della System Management Facility (SMF), e restituisce un "parere" al resource manager.

Questo parere può essere del tipo:

- l'utente è autorizzato all'accesso (RC[26]=0)
- RACF non può prendere una decisione perchè la risorsa non è protetta da un profilo (RC=4)
- l'utente non è autorizzato all'accesso (RC=8).

A questo punto il resource manager, sentito il parere di RACF, decide operativamente se garantire o meno l'accesso alla risorsa (normalmente in modo congruente alla risposta di RACF).

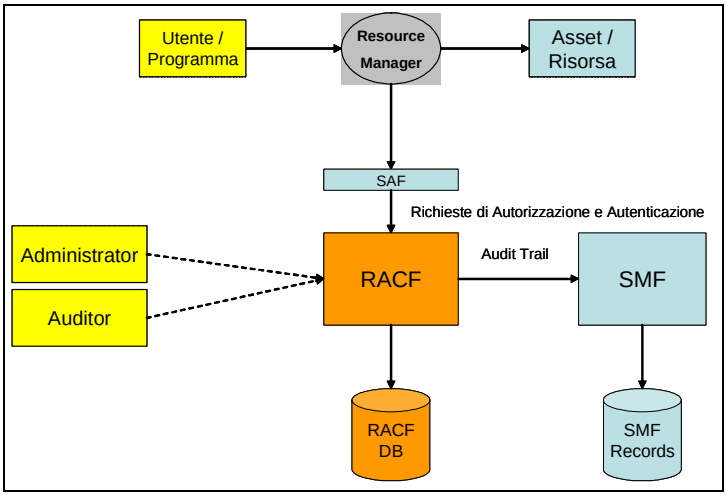


Figura 170 Architettura RACF

[26] RC – Return Code

Identificazione e Autenticazione dei soggetti (utenti, programmi, ecc.)

L'autenticazione nello z/OS, come abbiamo visto, avviene a cura del resource manager, cioè quel programma che, attraverso un protocollo o API[27], interagisce con un utente per fornire un servizio. Se il processo di autenticazione termina positivamente, viene creato un ACEE, che come abbiamo mostrato in precedenza, è l'elemento che viene utilizzato per verificare il controllo dell'accesso alle risorse. Per poter abilitare questo meccanismo, l'amministratore di RACF deve definire i profili degli utenti (USER PROFILEs) ed eventualmente anche gli insiemi di raggruppamento degli utenti o gruppi (GROUP PROFILEs) che verranno memorizzati nel database RACF. All'interno del profilo dell'utente sono memorizzate sia informazioni di tipo informativo (es. USER DATA) sia informazioni relative a specifiche applicazioni (USS[28], CICS, IMS SEGMENTs).

A questo punto per autenticare un utente il resource manager deve ottenere, attraverso il protocollo di comunicazione, delle credenziali che certifichino che l'utente è quello che afferma di essere. In seguito, effettuata la verifica delle credenziali, è necessario, per la creazione dell'ACEE, associare il soggetto riconosciuto tramite le credenziali con l'utente definito all'interno del database RACF (quello che è riconosciuto a livello di sistema). Quest'ultima funzione prende il nome di "*identity mapping*" che all'interno del sistema z/OS e RACF può essere sia di tipo banale, come nel caso della user/password, nella quale la user usata corrisponde a quella registrata in RACF, sia di tipo più complesso, come il *mapping* di un soggetto di un certificato digitale X.509 con un utente RACF o utilizzando i servizi di Enterprise Identity Mapping (EIM).

Come riferimento, riportiamo di seguito alcuni esempi di credenziali:

- Utenza e Password

[27] API – Application Programming Interface
[28] USS – Unix System Services

- Certificati digitali X.509 (SSL e TLS[29])
- RSA SecureID
- Kerberos
- ...

Nella tabella 3 seguente riportiamo invece degli esempi di abbinamento Resource Manager / Protocollo / Credenziali.

Resource Manager	***Protocollo/API***	***Credenziali[30]***
TSO	• TN3270 • SNA LU2	• User/Password o Passticket[31] • Certificato X.509
WebSphere/HTTP	• HTTP • RMI/IIOP	• User/Password o Passticket • Certificato X.509
JES (z/OS Batch)	• JCL	• User/Password o Passticket
CICS	• CTG[32] • SNA LU2	• User/Password o Passticket
DB2	• DRDA • JDBC	• User/Password o Passticket • Kerberos

Tabella 3

[29] SSL – Secure Socket Layer; TLS – Transport Layer Security
[30] Non tutte le credenziali elencate si applicano ai protocolli elencati nella casella adiacente. Sulla stessa riga.
[31] Passticket – E' un meccanismo di "*one time password*" offerto da RACF
[32] CTG – CICS Transaction Gateway

Autorizzazione e controllo degli accessi

Quando un utente (o programma) è stato autenticato, può operare all'interno dello z/OS e interagire con componenti del sistema operativo o con resource manager come il CICS o il DB2. Al fine di verificare se un utente può avere accesso ad una risorsa, è necessario definire un profilo nel database RACF che controlli gli accessi alla risorsa.

I profili di risorse in RACF sono identificati attraverso la classe di appartenenza (RACF CLASS), che ne identifica il tipo, e da un nome (es. Per un dataset SYS1.USERS.ROSSI). Ne consegue che il resource manager (es. DB2, CICS, ...), quando richiama RACF per un parere sull'autorizzazione all'accesso, deve inserire nella richiesta di autorizzazione sia l'ACEE dell'utente sia la classe e il nome della risorsa.

Per poter prendere delle decisioni sull'accesso alla risorsa, RACF può utilizzare due schemi di autorizzazione, singolarmente o in combinazione:

1. *Discretionary Access Control* (DAC) nel quale le autorizzazioni sono esplicitamente assegnate agli utenti o ai gruppi di utenti. Queste autorizzazioni prendono il nome di *Access Control List* (ACL) e fanno parte del profilo della risorsa.
2. *Mandatory Access Control* (MAC) nel quale l'autorizzazione viene valutata in base ad attributi definiti sia a livello di profilo utente che di profilo della risorsa. Gli attributi utilizzabili per questo metodo di autenticazione in RACF sono: il SECLEVEL, le CATEGORY e le SECLABEL (la SECLABEL di fatto è una combinazione di CATEGORY e SECLEVEL).

Vediamo in maggior dettagli i due modelli di autorizzazione.

Il modello DAC richiede che siano definiti gli utenti, i gruppi di utenti e le risorse. L'amministratore della sicurezza modificherà la ACL del profilo di una risorsa in modo tale che garantisca (o neghi) esplicitamente l'accesso alla risorsa (o alle risorse) protetta da quel profilo RACF, indicando anche il tipo di accesso che viene autorizzato (es. READ, UPDATE, ...). Per far questo l'amministratore inserisce nella ACL il nome di un utente o di un

gruppo. E' buona pratica autorizzare l'accesso alle risorse ai gruppi di utenti, in questo modo l'accesso alla risorsa avviene semplicemente connettendo un utente ad un gruppo di utenti, semplificando e separando la fase di disegno del controllo degli accessi da quella di amministrazione ordinaria.

Il modello MAC, rispetto al DAC, non necessita del concetto di gruppo e l'amministratore della sicurezza di fatto garantisce un accesso implicito alle risorse assegnando ad esse e agli utenti degli attributi (es. SECLABEL). Vediamo di chiarere meglio questo concetto con un esempio. Supponiamo di creare una tabella di classificazione utilizzando livelli di sicurezza (SECLEVEL) di classificazione dell'informazione identificati da Top Secret, Confidential e Unclassified. Ortogonalmente, definiamo delle category, che per esempio corrispondono ad unità di business come HR, Personnel, Engineering, ecc. Otterremo una tabella come quella in Tabella 2. Ora creiamo dei "tesserini" (SECLABEL) associando categorie e classificazioni di sicurezza, che nel nostro caso chiamiamo:

- SECLABEL A per la combinazione (HR, Top Secret)
- SECLABEL B per le combinazioni ((HR, Top Secret), (HR, Confidential), (Personnel, Confidential))

In questa situazione se un utente ha un tesserino che "include" il tesserino della risorsa (primo esempio in Figura 171, allora l'accesso può essere garantito, nel caso contrario (secondo esempio in Figura 171) l'accesso viene negato.

Nella realtà, l'implementazione delle modalità di autorizzazione MAC in RACF è più complessa e articolata di quanto mostrato in quest'esempio. Inoltre la modalità MAC è alla base della cosiddetta *Multi Level Security* (MLS), che aggiunge al modello MAC il modello di sicurezza Bell-La Padula[33].

[33] http://en.wikipedia.org/wiki/Bell-LaPadula_model

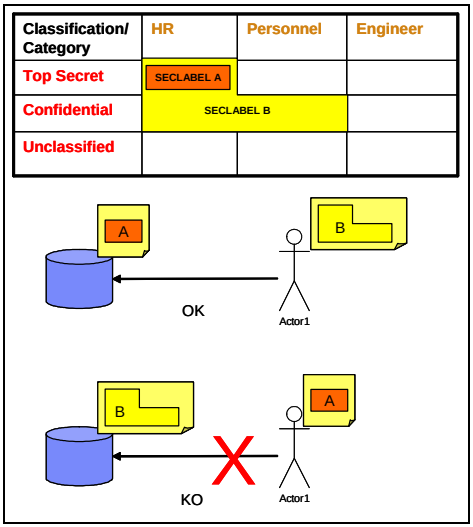


Figura 171 Esempi modello MAC

Amministrazione e Auditing

L'amministrazione ed il controllo di un ambiente protetto da RACF è in genere operata da più ruoli con autorità in ambiti amministrativi disgiunti, in modo da avere una separazione di potere che garantisce un minor rischio rispetto ad attività malevole di tipo volontario. Gli ambiti di amministrazione/gestione definiti in RACF sono:

- *Amministrazione e Gestione del RACF* - E' caratterizzato dall'attributo SPECIAL nel profilo RACF dell'utente. Questa tipologia di utente è in grado di definire profili (utente, gruppo ed risorse) ed impostare i parametri di funzionamento di RACF.
- *Libera operatività e accesso alle risorse del sistema* - E' caratterizzato dall'attributo OPERATIONS nel profilo RACF dell'utente.

- *Impostazione e rapportistica dei parametri di auditing del sistema e delle risorse* - E' caratterizzato dall'attributo AUDITOR nel profilo RACF dell'utente.

Gli amministratori si avvalgono, per amministrare RACF, o delle interfacce native, TSO e ISPF, o di prodotti come IBM/CONSUL.

IBM Security Server

L'IBM Security Server è un "impacchettamento" di prodotti e componenti di sicurezza relativi alla piattaforma z/OS. I componenti che ne fanno parte, oltre a RACF, sono i seguenti:

- *DCE*[34] *Security Server*. Questo server fornisce le funzionalità di un server di sicurezza OSF[35] DCE.
- *Lightweight Directory Access Protocol (LDAP) Server*. Questo server è basato su un modello client/server e fornisce accesso tramite LDAP ad una directory di tipo generico e ai dati relativi alle utenze e gruppi memorizzati nel database di RACF.
- *z/OS Firewall Technologies* (da z/OS 1.8 parte integrante di z/OS Communication Server). E' l'implementazione sia di un firewall di rete IPV4 con il suo meccanismo di controllo accessi di rete basato su filtri, sia di un server IPSEC[36] per la realizzazione di reti private virtuali (VPN) che vedono la partecipazione di istanze z/OS.
- *Network Authentication Service for z/OS*. Fornisce i servizi di sicurezza di Kerberos integrati con RACF.
- *Enterprise Identity Mapping (EIM)*. Sono dei servizi infrastruttuarli a disposizione di applicazioni e sistemi operativi per gestire più facilmente il flusso delle identità tra sistemi e base dati utenti (*User Registry*) diverse all'interno di una soluzione d'impresa.

[34] DCE – Distributed Computing Environment. http://www.opengroup.org/dce/
[35] OSF – Open Software Foundation, now The OpenGroup http://www.opengroup.org/
[36] IPSEC – IP Security http://www.ietf.org/html.charters/OLD/ipsec-charter.html

- *PKI*[37] *Services*. Fornisce i servizi di Certification Authority e di gestione del ciclo di vita dei certificati digitali X.509 (emissione e amministrazione) necessari a soluzioni di firma digitale o di autenticazione forte (es. SSL v3 o TLS Client Authentication).

Riservatezza (Confidentiality): la crittografia

La **crittologia** è la scienza che studia i modi e le tecniche per segretare delle informazioni o come rivelare informazioni segretate senza esserne autorizzati. Essa si compone quindi di due branche:

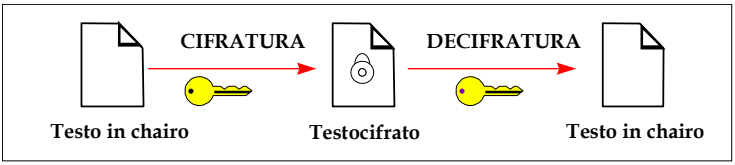
- La **crittografia** (nome derivato dal greco κρυπτός kryptós "nascosto" e il verbo γράφω gráfo "scrivere") che si occupa dello studio della segretazione delle informazioni. Attualmente è diventata un ramo della teoria dell'informazione, spostandosi dall'originario ambito linguistico a quello matematico. Ron Rivest, noto crittografo e co-inventore della crittografia a chiavi pubbliche RSA [38] ha osservato che "la crittografia è relativa alla comunicazione in presenza di avversari".
- La **crittoanalisi** (nome derivato dal greco κρυπτός kryptós "nascosto" e il verbo αναλύειν analýein "sciogliere, risolvere") si occupa invece dello studio di metodi per ottenere l'informazione cifrata, senza aver accesso all'informazione segreta che è stata usata per cifrarla. Nel liguaggio comune questa pratica viene anche chiamata "*codebraking*" o "*cracking the code*" cioè "rompere" il codice.

La crittografia moderna, come abbiamo accennato prima, si fonda sull'utilizzo della matematica per inventare dei metodi che, utilizzando un'informazione segreta detta chiave, permettano di segretare delle informazioni. Naturalmente, il disegno dell'algoritmo crittografico garantisce

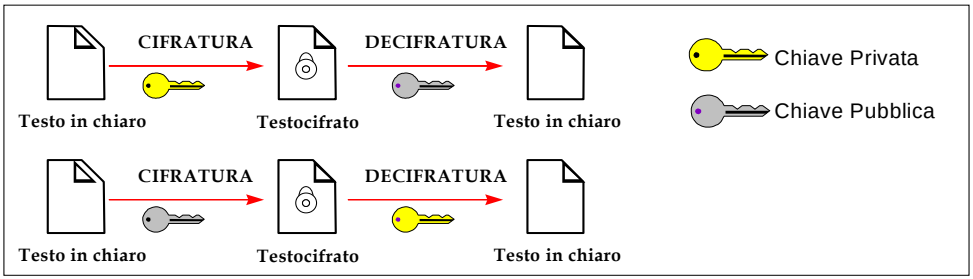
[37] PKI – Public Key Infrastructure o Infrastruttura a Chiavi Pubbliche.
[38] RSA – Rivest, Shamir e Adelman, matematici e fondatori della società RSA, proprietaria del brevetto della crittografia a chiavi pubbliche detta comunemente crittografia RSA.

che più una chiave è "lunga" più è difficile (se non impossibile) effettuare la crittoanalisi in tempi accettabili. Attualmente possiamo classificare i metodi (e algoritmi) crittografici in tre categorie principali:

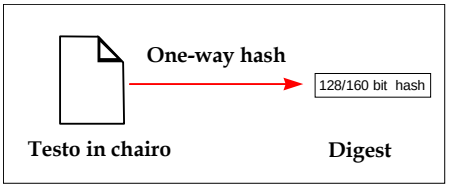
- *Crittografia a chiavi simmetriche*. Questa classe di algoritmi utilizza la stessa chiave sia per cifrare che per decifrare l'informazione.



- *Crittografia a chiavi asimmetriche o a chiavi pubbliche*. Questa classe di algoritmi utilizza due chiavi, univocamente associate l'una all'altra. Una chiave, detta privata, è usata per cifrare ed un'altra chiave, detta pubblica, per decifrare. Vale inoltre la proprietà inversa, cioè quello che è stato cifrato con la chiave pubblica può essere decifrato unicamente con la chiave privata associata.



- *Hash o digest*. Questa classe di algoritmi non necessita di una chiave per operare. Il loro scopo è quello di trasformare un'informazione di lunghezza arbitraria, nel modo più univoco possibile, in una stringa di lunghezza fissa e relativamente limitata (detta *hash* o *digest*). Queste funzioni operano in un solo senso ossia l'informazione originaria non può essere ricostruita partendo dall'*hash*.



Nella tabella 4 sono riportati alcuni algoritmi di crittografia più comunemente usati.

Algoritmi Simmetrici	Algoritmi Asimmetrici	Algoritmi Hash
DES, Triple-DES (3DES), AES, RC4, Blowfish	RSA, DSA, ELGAMAL	RIPEMD, SHA-1, MD5

Tabella 4

Come e quando vengono utilizzati questi algoritmi e perché? Per rispondere a questa domanda dobbiamo analizzare le caratteristiche di questi algoritmi e il contesto in cui vengono impiegati.

Prestazioni. Sotto questo aspetto gli algoritmi simmetrici sono decisamente più efficienti di quelli asimmetrici. Quando si deve cifrare una notevole quantità di dati o effettuare la cifratura frequentemente, in genere si utilizzano algoritmi simmetrici.

Contesto. Con l'avvento di Internet il contesto informatico operativo coinvolge quasi sempre (se non sempre) due sistemi che si scambiano informazioni e servizi attraverso un mezzo di comunicazione accessibile a tutti. In questo caso se, come abbiamo detto in precedenza, utilizziamo la crittografia simmetrica per cifrare la comunicazione, sorge il problema di condividere la stessa chiave tra i due interlocutori. Questo problema naturalmente cresce con il numero degli interlocutori. In nostro aiuto però viene la cifratura a chiavi pubbliche che può essere impiegata nella

cifratura di pochi dati, come le chiavi, e in attività eseguite non frequentemente, come lo scambio di chiavi. In effetti le soluzioni di comunicazione con cifratura, come l'SSL, il TLS e IPSEC, impiegano una combinazione di algoritmi di cifratura simmetrica e asimmetrica per raggiungere l'obiettivo di realizzare una comunicazione sicura efficiente ed efficace su internet.

Le funzioni di hash, in combinazione con cifratura simmetrica o asimmetrica, sono invece utilizzate per garantire l'integrità delle informazioni tramite tecniche di MAC e di firma digitale o per derivare chiavi crittografiche e password da un testo od una frase.

Prima di passare all'architettura crittografica del System z, spendiamo ancora qualche parola su alcuni concetti che ci torneranno utili a breve. Come abbiamo visto nei paragrafi precedenti, l'elemento portante della sicurezza è la chiave di cifratura. Essa è un elemento di controllo e non deve essere rivelata a persone non autorizzate a conoscerla; se ciò avvenisse, l'informazione protetta dalla chiave sarebbe esposta a manipolazioni, alterazioni e così via. Come però ben sappiamo, all'interno di un elaboratore, per poter eseguire un programma (in questo caso l'operazione di cifratura o decifratura) è necessario che i dati (in questo caso anche la chiave) siano in memoria: questo significa che la chiave di cifratura è "in chiaro" e potrebbe essere letta da chi è in grado di leggere nella memoria dell'elaboratore.

Per ovviare a questa situazione è necessario realizzare un ambiente di elaborazione protetto sia logicamente che fisicamente nel quale operare la cifratura o la deciffratura delle informazioni. Solamente in questo ambiente, detto *Hardware (o Host) Security Module* (HSM), la chiave viene messa in chiaro. In questo caso parliamo invece di chiave di cifratura "protetta", che si contrappone alla cifratura con chiave "in chiaro". Gli HSM possono prendere forma di dispositivo esterno all'elaboratore o di scheda interna (es. PCI/PCIX) e sono acceduti tramite delle API come quelle definite nello standard RSA PKCS#11 (conosciuto anche come *Cryptoki*).

L'architettura del sottosistema crittografico

L'architettura del sistema crittografico si basa sulla *IBM Common Cryptographic Architecture* (IBM CCA), che definisce i componenti e le interfacce applicative (API) nonché le modalità di interfacciamento ai componenti crittografici di sistema. La caratteristica principale della CCA, oltre alla definizione dei servizi crittografici e a come invocarli, risiede nella "architettura" delle chiavi di cifratura. Lo scopo è quello di avere un disegno di riferimento per la cifratura con chiavi "protette" avvalendosi di un HSM. La principale classificazione delle chiavi comprende due categorie principali:

1. Le *Master Key* (MK). Sono chiavi che vengono inserite nell'HSM (e non possono poi essere più estratte) che servono a proteggere le chiavi operative. Le Master Key cifrano le chiavi operative generate o inserite all'interno dell'HSM in modo tale che possano essere memorizzate con sicurezza esternamente all'HSM, per esempio su un database. In questo modo si è sicuri che l'utilizzo di quelle chiavi può essere fatto unicamente conoscendo la Master Key di protezione, e quindi, per quanto abbiamo appena detto, solo all'interno dell'HSM. In genere le Master Key sono chiavi a cifratura simmetrica di tipo 3DES o AES.
2. Le *Operational Keys* (OPKey) o chiavi operative. Sono le chiavi utilizzate per effettuare la cifratura, possono essere sia di tipo simmetrico che asimmetrico ed in genere sono specializzate per un particolare tipo di cifratura (legata quindi ai servizi della CCA). Ad esempio una chiave può essere utilizzata solo per generare PIN[39] bancari, un'altra per cifrare delle chiavi, un'altra ancora per effettuare firme digitali. Questo controllo sull'utilizzo avviene a cura dell'HSM tramite l'utilizzo di *Control Vector*, cioè delle maschere che si combinano con la chiave operativa per ottenere l'effettiva chiave di cifratura.

[39] PIN – Personal Identification Number. Il codice segreto utilizzato dalla carte bancarie come ad esempio il Bancomat per autorizzare delle transazioni (es. Prelievo di contante).

L'hardware: CPACF e schede crittografiche CEX2C

Nel System z le "*facility*" crittografiche, ovvero il complesso dei componenti crittografici, è caratterizzato da due elementi hardware:

- CPACF (CP[40] Assist for Cryptographic Functions) - è un componente hardware del processore (CP) che può essere utilizzato attraverso istruzioni (assembler) dell'architettura z. Il CPACF è un'estensione dell'Instruction Set della z/Architecture presente in ogni processore z e la sua esecuzione è sincrona rispetto al flusso di esecuzione delle istruzioni del programma. Esso è molto veloce nell'esecuzione di istruzioni crittografiche ed opera con chiavi "in chiaro". Gli algoritmi sono solo di tipo simmetrico, come il 3DES e l'AES, di hash, come lo SHA-1, e di generazione di numeri casuali (PRNG[41]).
- Il coprocessore IBM 4764 - è un "*embedded system*" alloggiato su una scheda PCIX ed implementa un HSM per le piattaforme hardware IBM. Sul System z il coprocessore IBM 4764 è chiamato *Crypto Express2 Coprocessors* (CEX2C) che al suo interno include due IBM 4764. Esso può essere configurato sull'elaboratore in due modalità:
    - Come acceleratore crittografico (CEX2A) per eseguire velocemente operazioni di cifratura asimmetrica con chiavi "in chiaro"
    - Come co-processore crittografico (CEX2C) per eseguire funzioni crittografiche simmetriche, asimmetriche e di tipo bancario/finanziario con chiavi "protette".

  Il CEX2C è una scheda PCI che, al contrario del CPACF, è acceduta in asincrono e attraverso un sottosistema crittografico ICSF[42]. Non è possibile accedere con delle API native ai servizi CEX2C ma bisogna transitare attraverso i servizi ICSF.

[40] CP – Central Processor. Il processore del System z.
[41] PRNG – Pseudo Random Number Generator
[42] ICSF – Inegrated CryptoGraphic System FAcility

Esiste nell'architettura System z un altro componente hardware (opzionale) che indirizza la gestione sicura delle chiavi. Questo componente è una workstation, la *Trusted Key Entry* (TKE) *workstation,* che appunto è in grado di comunicare, in modo protetto, le chiavi crittografiche ai cripto-processori senza farle transitare in chiaro nella memoria dell'elaboratore.

Nella Figura 172 è riportata l'architettura dell'ambiente crittografico nel System z.

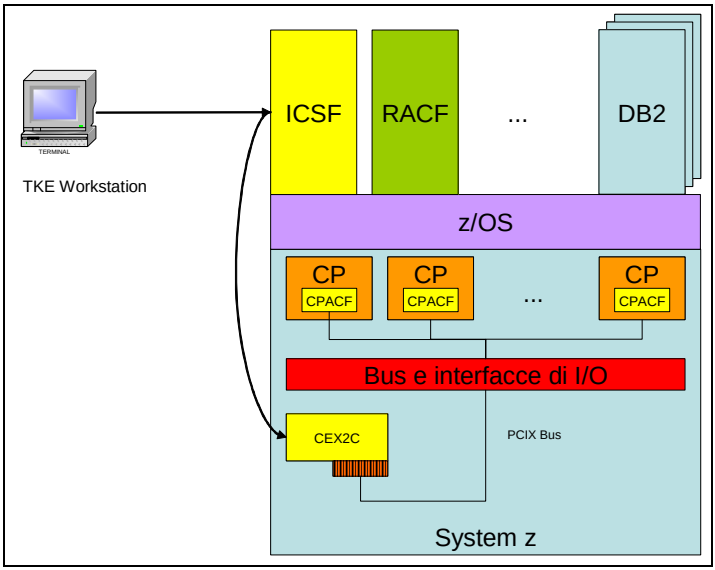


Figura 172 Architettura ambiente crittografico System z

Il software: ICSF e le applicazioni/framework crittografici

Il gestore delle risorse crittografiche, cioè i servizi (algoritmi) e le chiavi, in z/OS è l'Integrated Cryptographic Service Facility (ICSF). Il compito di ICSF è quello di offrire un'interfaccia applicativa (API) in conformità alla IBM CCA per accedere ai servizi crittografici forniti dall'hardware. Dal punto di vista della sicurezza e del controllo degli accessi, ICSF si appoggia a RACF per prendere queste decisioni, in particolare si avvale di due classi di risorse, la CSFSERV e la CSFKEYS, che proteggono rispettivamente l'accesso ai servizi e alle chiavi crittografiche secondo il modello di sicurezza RACF implementato. ICSF è quindi utilizzata sia da diversi "framework" di sicurezza, come la z/OS System SSL o l'implementazione OCSF (CDSA) 43, sia direttamente da vari linguaggi di programmazione, come il C/C++, COBOL, FORTRAN, ecc., come rappresentato in Figura 173.

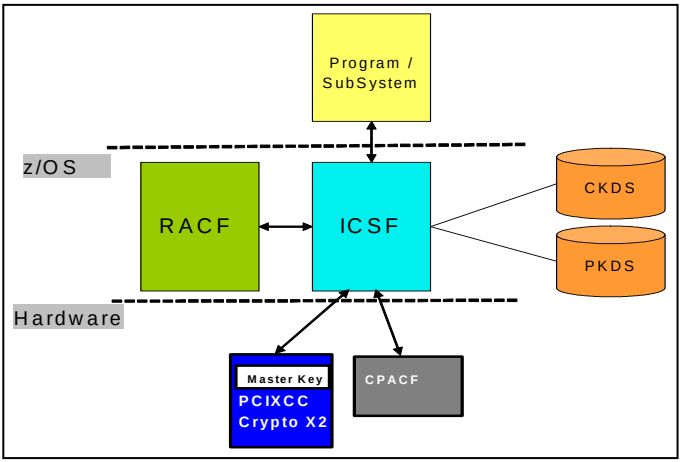


Figura 173 ICSF – Architettura

[43] OCSF - Open Cryptographic Service Facility è una implementazione della Intel Common Data Security Architecture (CDSA),

Un'altra funzione di ICSF, oltre ai servizi crittografici, è quella di offrire un ambiente per la gestione delle chiavi crittografiche. Per far questo – vedi Figura 174 - ICSF si avvale di due basi dati (dataset VSAM[44]):

- Una chiamata CKDS[45], che contiene chiavi operative simmetriche,
- Una chiamata PKDS[46], che contiene chiavi operative asimmetriche.

Le chiavi contenute nei due dataset sono cifrate con le master key memorizzate e rese sicure nel criptoprocessore. Il CKDS è protetto dalla Master Key Simmetrica (SYM-MK) mentre il PKDS è protetto dalla Master Key Asimmetrica (ASYM-MK). E' da notare che, a dispetto del nome, entrambe sono chiavi per cifratura simmetrica Triple DES (3DES). L'inserimento delle Master Key nel criptoprocessore può avvenire attraverso le interfacce amministrative di ICSF o, in modalità più sicura, utilizzando la workstation TKE.

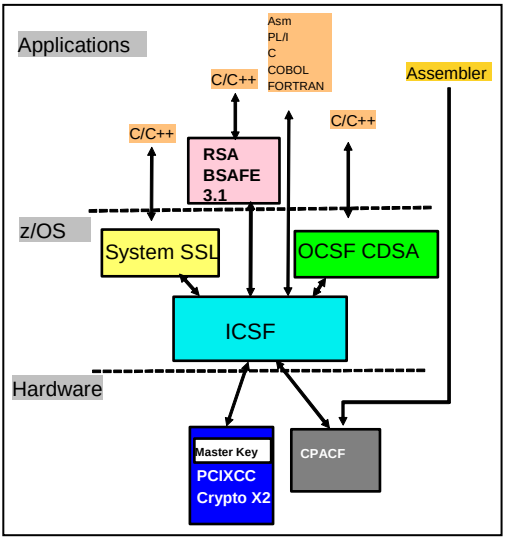


Figura 174 ICSF - Gestione chiavi

[44] VSAM – Virtual Storage Access Method
[45] CKDS – Cryptographic Key Data Set
[46] PKDS – Public Key Algorithm Data Set

Linux

Anche Linux, se ospitato su di un elaboratore System z in modo nativo o come ospite dell' ipervisore z/VM[47], è in grado di sfruttare le funzionalità hardware della piattaforma. In questo ambiente non esiste un sottosistema come l'ICSF dello z/OS ma viene fornito un apposito device driver, chiamato zcrypt, al quale richiedono l'accesso, tramite la libreria libICA, il *framework* o API di sicurezza come l'OpenSSL, PKCS#11 e l'implementazione SSL di IBM, il GSKit. La Figura 175 mostra l'architettura crittografica (Hardware e Software) su zLinux.

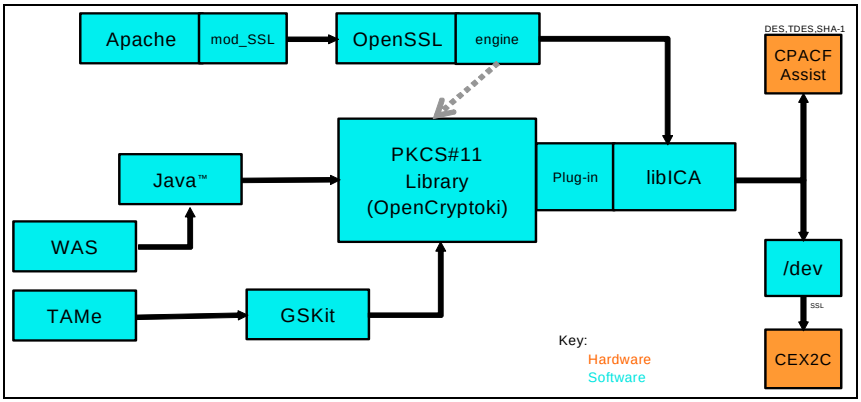


Figura 175 Architettura crittografica su zLinux

[47] z/VM – System z Virtual Machines

Appendice B: I sistemi ibridi - una possibile evoluzione dei sistemi mainframe

[a cura di Fulvio Capogrosso]

Il sistema z10, annunciato e reso disponibile nel 2008, ha introdotto notevoli elementi di evoluzione ed innovazione nella tecnologia del Mainframe.

A livello architetturale, le novità più significative riguardano le numerose istruzioni per l'ottimizzazione di codice già compilato, le istruzioni per il calcolo floating point decimale e le nuove istruzioni di sistema che, attraverso interazioni dirette fra hardware e programma di controllo, permettono di ottimizzare le prestazioni sopratutto in ambienti altamente virtualizzati.

A livello di disegno sono di notevole rilievo il nuovo chip quad-core che contiene le unità elaborative ed opera ad una frequenza di 4.4 GHz ed il nuovo SMP hub chip che contiene 24 MB di cache di secondo livello. Inoltre la struttura modulare multi-book del sistema è resa più efficiente dal collegamento a stella invece che ad anello come nei sistemi precedenti; ulteriori significativi miglioramenti sono stati portati alla capacità della infrastruttura di I/O (288 GB al secondo complessivi) e alla dimensione massima della memoria di sistema (1,5 Terabytes) per rendere il sistema molto ben bilanciato per quanto riguarda la quantità di risorse esterne. Sono inoltre da segnalare la nuova architettura di *provisioning* per permettere di aumentare "on demand" ed in modo facile e rapido la potenza elaborativa del sistema e la funzionalità di Hyperdispatch per ottimizzare le prestazioni in presenza di alti livelli di virtualizzazione.

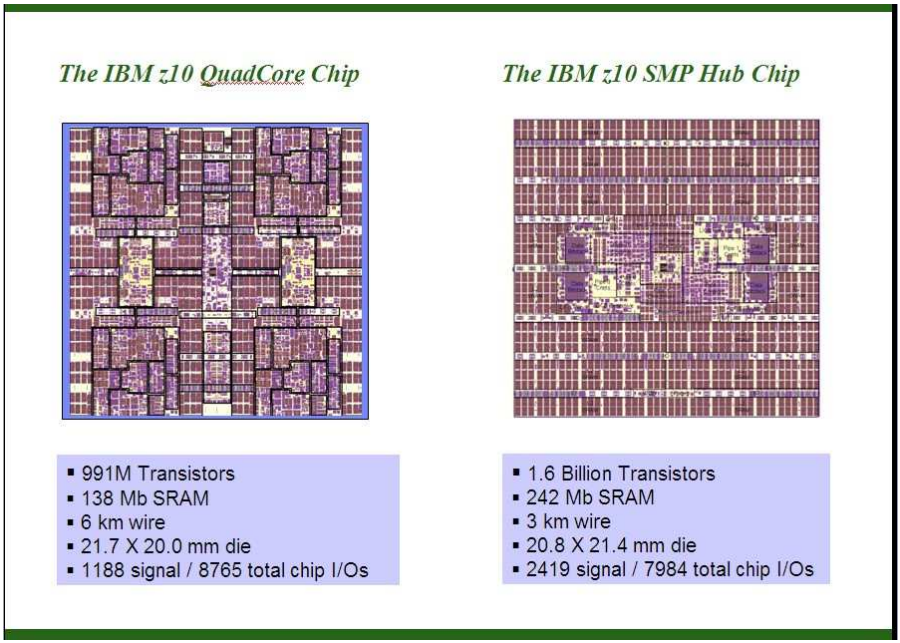


Figura 176 Chip del Processore z10

Tutto questo conferma la tradizionale capacità del mainframe di sfruttare al meglio le possibilità sempre maggiori offerte dalla tecnologia di base, di fornire le migliori opportunità di crescita applicativamente compatibile oggi esistenti sul mercato e ribadisce l'insostituibile ruolo che il mainframe ha oggi in ogni moderna infrastruttura IT. La sempre più rapida evoluzione della tecnologia dei sistemi e delle richieste del mercato suggeriscono però di volgere comunque lo sguardo verso il futuro più o meno prossimo per chiedersi quale potrebbero essere i prossimi cammini di evoluzione della tecnologia del mainframe.

Per far questo è necessario prima di tutto analizzare quali sono i principali trend tecnologici che prevedibilmente caratterizzeranno gli anni futuri e quali sono le sfide che il mercato dell'informatica proporrà ai fornitori di tecnologia.

I trend tecnologici prossimi futuri

Secondo le opinioni dei maggiori analisti di settore e le più recenti indagini di mercato, nel prossimo futuro l'informatica dovrà affrontare importanti sfide per soddisfare importanti esigenze di mercato quali, ad esempio:

- semplificare o meglio razionalizzare l'infrastruttura IT per migliorarne le prestazioni e per controllare la crescita dei costi di gestione che oggi avviene a ritmi inaccettabili
- aumentare la "business resilience" cioè la capacità dell'infrastruttura informatica di fornire servizi anche in presenza di fenomeni potenzialmente catastrofici
- aumentare l'efficacia degli apparati di sicurezza a protezione dell'infrastruttura per permetterne l'operatività secondo le normative sempre più stringenti
- migliorare la "capacità di risposta" dell'infrastruttura informatica ovvero la capacità di adattarsi velocemente, e possibilmente senza bisogno di interventi umani, a nuove condizioni di operatività
- aumentare la flessibilità dell'infrastruttura ovvero la capacità di supportare nuovi processi di business senza rappresentare un vincolo al cambiamento
- fornire e supportare nuovi modelli applicativi che permettano di "modernizzare" le applicazioni esistenti usando le nuove tecnolgie di settore (per esempio la SOA) ma conservando quanto di valido ancora esiste nelle applicazioni correnti
- supportare i nuovi modelli applicativi emergenti

Quest'ultimo punto è forse il più importante e merita un ulteriore approfondimento. Lasciando da parte gli ambiti applicativi più specializzati tipici del mondo del calcolo intensivo e dei supercomputer, possiamo invece concentrarci sui modelli applicativi emergenti in un campo tradizionale dell'informatica quale il modello OLTP/DB. Questo modello applicativo, che è alla base dell'informatica moderna (si pensi alle decine di miliardi di transazioni al giorno gestite oggi da CICS ed IMS nel mondo) è da sempre caratterizzato da altissimi volumi di traffico, tempi di risposta cosiddetti sub-second ed enormi requisiti di sicurezza, integrità e

disponibilità. Tutto questo richiede transaction monitor di altissimo livello quali appunto il CICS e l'IMS, tecniche di programmazione avanzate per non appesantire l'esecuzione delle singole transazioni e per supportare gli altissimi volumi, grande integrazione con i database managers (ad esempio il DB2) per un accesso ai dati rapido, efficiente e sicuro, ed una avanzata capacità di integrazione con le applicazioni batch per completare l'intero flusso applicativo nei tempi richiesti. Su questa base di tradizioni oramai consolidate e spinte dalle più recenti innovazioni tecnologiche, sopratutto nell'area della velocità di esecuzione delle istruzioni e nella capacità di memorizzazione ed accesso ai dati, emergono esigenze applicative sempre nuove e sempre più complesse quali:

- integrare le attuali transazioni con componenti applicativi real time ad alto contenuto di calcolo elementare e/o analitico (si pensi per esempio alle opportunità di online risk management per le applicazioni finanziarie o alle ricerche online su grandi masse di dati per applicazioni nel campo della medicina)
- utilizzare le vaste possibilità offerte dalla cosiddetta *digital convergence* per permettere la gestione contemporanea di dati, immagini, voce, animazioni, ecc.
- accedere in modo rapido ed efficace a enormi quantità di dati strutturati e non strutturati (si pensi alle più moderne applicazioni di *streaming computing* e ai progetti in corso nel campo della sicurezza)
- sviluppare applicazioni di tipo "sociale" (community driven computing) con utilizzo di tecniche sofisticate di real time collaboration, gaming, ecc.

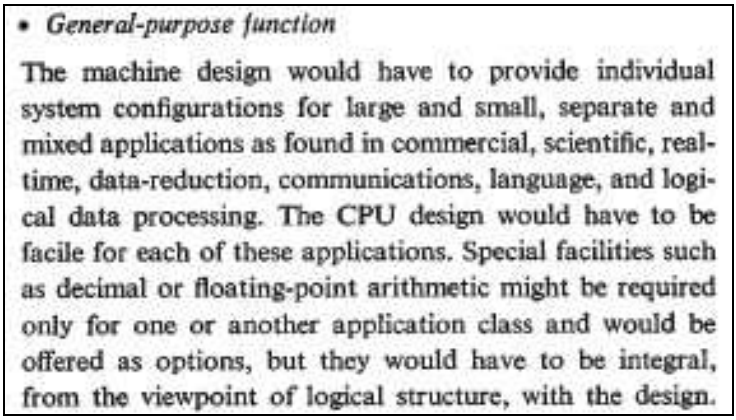
A fronte di questa vasta gamma di esigenze applicative emergenti è lecito chiedersi quale sia l'architettura di sistema e l'infrastruttura informatica più adatta a soddisfarle. In particolare è importante chiedersi se sarà ancora possibile accogliere in futuro tutte queste esigenze applicative in un'unico disegno architetturale di tipo *general purpose* quale il mainframe e mantenere così tutti i vantaggi associati a questa tecnologia oppure se non sarebbe più opportuno adottare architetture *special purpose* specializzate per eseguire particolari tipi di calcolo rischiando però di introdurre insostenibili livelli di complessità nell'infrastruttura, oppure se non sia

possibile ipotizzare soluzioni intermedie, quali, per esempio, i cosiddetti sistemi ibridi. Per affrontare questo tema è opportuno ricordare alcuni elementi di differenziazione fra sistemi *general purpose* e sistemi *special purpose*.

I sistemi general purpose

Si dicono *general purpose* quei sistemi progettati per eseguire (in modo efficiente) applicazioni di qualunque tipo tipo. La definizione è molto generica e serve in pratica solo per differenziare tali sistemi dai sistemi *special purpose* (vedi nel seguito), che sono progettati per eseguire solo applicazioni di un tipo particolare. Nel tempo il concetto di *general purpose* è stato anche esteso per comprendere la capacità di eseguire contemporaneamente un numero elevato di applicazioni. Il mainframe è l'esempio tipico di sistema *general purpose* in quanto in grado di gestire contemporaneamente e con alti livelli di efficienza un numero elevato di applicazioni di tipo batch, interattivo e transazionale, di tipo commerciale e scientifico, e con un vasto apparato di funzionalità di controllo e di gestione fornite dal software di sistema. Il sistema IBM S/360, l'archetipo del mainframe, è definito fin dall'origine (1964) come sistema "general purpose" ed il nome stesso ne indica un possibile utilizzo su tutto lo spettro applicativo immaginabile, "a 360 gradi" appunto.

Ecco la definizione presa da "The Architecture of the IBM System /360, G.M.Amdhal, G.A.Blaauw, F.P.Brooks, IBM J.Res.Dvlp, vol 44, n.1/2, January 2000" dei sistemi *general purpose:*



• *General-purpose function*

The machine design would have to provide individual system configurations for large and small, separate and mixed applications as found in commercial, scientific, real-time, data-reduction, communications, language, and logical data processing. The CPU design would have to be facile for each of these applications. Special facilities such as decimal or floating-point arithmetic might be required only for one or another application class and would be offered as options, but they would have to be integral, from the viewpoint of logical structure, with the design.

Figura 177 Definizione Sistema General Purppose

Questa caratterizzazione del sistema S/360 come sistema *general purpose* è riflessa in molte delle decisioni architetturali prese allora quali: il formato e la lunghezza delle istruzioni, la presenza di istruzioni logiche di grande generalità per permettere la manipolazione di qualunque configurazione di bit e anche dei singoli bit, il meccanismo degli interrupt, i meccanismi di protezione, la gestione dell'apparato di I/O. Nel tempo la dotazione tecnologica dei mainframe è andata gradualmente aumentando fornendo funzioni sempre più avanzate ed essenziali alla natura *general purpose* del sistema quali:

- il supporto di diversi linguaggi di programmazione e relativi compilatori
- le funzioni di dispatching e di resource management per accomodare sia le esigenze della multiprogrammazione, sia le esigenze di utilizzo ottimale delle risorse
- i principi e regole di sistem integrity
- il multiprocessing per svincolare la capacità elaborativa complessiva del sistema dalla potenza massima erogabile con un singolo processore e quindi dai ritmi di evoluzione della tecnologia di base
- la virtualizzazione per offrire la possibilità di gestire contemporaneamente ambienti elaborativi diversi; questa funzionalità, sviluppata negli anni '70 per rendere più efficiente il lavoro di un gran numero di programmatori di sistema, conserva

ancora oggi tutto il suo valore sopratutto nell'area del consolidamento e della ottimizzazione della gestione delle infrastrutture distribuite
- l'enorme apparato di funzionalità hardware e software a supporto della RAS (Reliability, Availability, Serviceability) per garantire i più alti livelli di disponibilità del servizio anche in presenza di malfunzioni hardware, software o di ambiente.

I sistemi special purpose

A differenza degli elaboratori *general purpose* che, come abbiamo appena visto, sono in grado di gestire in modo efficiente e contemporaneamente un gran numero di ambiti applicativi diversi, i sistemi cosiddetti *"special purpose"* sono disegnati per eseguire in modo molto efficiente un ambito applicativo molto ristretto, spesso di un solo tipo. Un sistema *special purpose* è di solito caratterizzato da una piattaforma hardware a larga diffusione oppure, nei casi più particolari, da hardware disegnato appositamente per il particolare ambito applicativo (ASIC, Application Specific Integrated Circuit, chip). Il programma di controllo (sistema operativo) è anch'esso molto semplice dal punto di vista delle funzionalità richieste ed è di solito scelto anch'esso fra i prodotti a larga diffusione, ad esempio Linux . Il programma di controllo fornisce anche le funzionalità necessarie a collegare questi sistemi al resto della infrastruttura informatica utilizzando i tradizionali protocolli standard.

Trattandosi di sistemi per l'esecuzione di un unico tipo di applicazione con programmi di controllo relativamente semplici, non sono richiesti sofisticati meccanismi di schedulazione, di controllo della multiprogrammazione e dell'utilizzo delle risorse. Analogamente, stante l' ambito applicativo ristretto, non sono richiesti sofisticati meccanismi di protezione della memoria, di isolamento degli utenti l'uno dall'altro, di sicurezza e di RAS. Le particolari caratteristiche dei sistemi *special purpose*, detti anche talvolta *accelerators* nelle versioni più semplici, permettono grandi livelli di efficienza in fase di esecuzione dei programmi con costi di acquisto relativamente bassi, elementi questi che rappresentano altrettanti punti di forza per queste tecnologie.

Fra i sistemi *special purpose* più noti ricordiamo: i sistemi di caching, i routers e gli switch di comunicazione, i processori per la crittografia e la compressione dei dati, i dispositivi per la gestione di applicazioni grafiche e multimedia, gli acceleratori per esecuzione di codice Java, i firewall, i sistemi per il load balancing del traffico di rete e tutti quei sistemi detti genericamente internet appliance. Rientrano anche nella categoria degli elaboratori *special purpose:*

- i grandi supercomputer, quali il sistema IBM BlueGene, il supercomputer più potente del mondo, specializzato per l'esecuzione di calcoli particolarmente complessi quali la simulazione delle reazioni nucleari, il modo di replicarsi dei virus ed i processi interni legati allo sviluppo e formazione delle proteine
- il sistema IBM Cell Broadband Engine, progettato inizialmente per l'ambiente *gaming*, ma dotato di tali capacità di calcolo da aver trovato immediatamente utilizzo in numerosi altri campi applicativi
- la famiglia di prodotti DataPower specializzati per l'esecuzione di applicazioni XML.

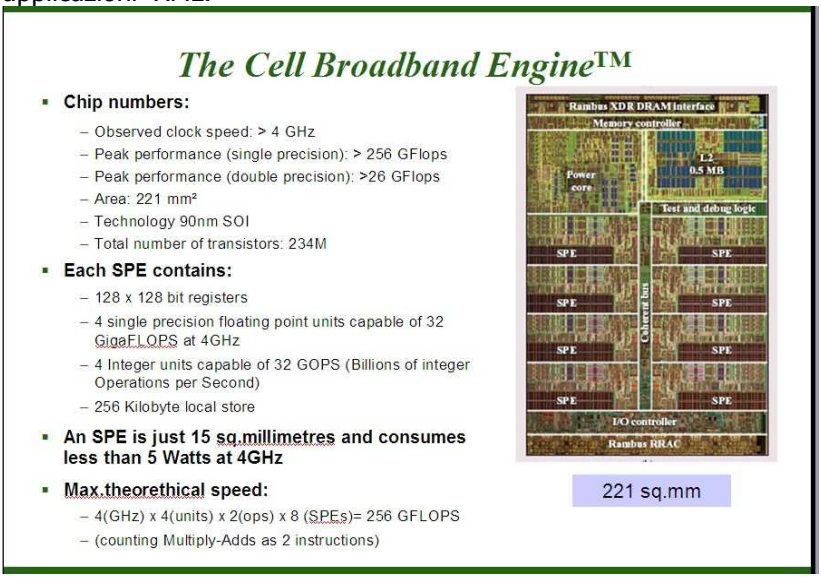


Figura 178 CELL BE Processor

I sistemi ibridi

Per tornare al tema dei sistemi ibridi, la discussione su tali sistemi nasce dalla seguenti considerazioni:

- la possibilità di aumentare le prestazioni di un processore (single thread) semplicemente aumentandone il ciclo base ha già da tempo evidenziato grossi limiti tecnologici; le soluzioni multi-core e multi-thread, se pur forniscono potenze elaborative sempre crescenti, richiedono però complessi modelli di sviluppo applicativo di tipo parallelo che tardano ad emergere
- sia i sistemi g*eneral purpose* che i sistemi *special purpose* presentano punti di valore e indirizzano ciascuno esigenze sicuramente presenti nella maggior parte delle infrastrutture IT moderne, sopratutto di dimensioni medio grandi
- ragionando in termini di evoluzione risulta impraticabile pensare di modificare il disegno e l'architettura dei sistemi *general purpose* per renderli competitivi sia come prezzo che come prestazioni con gli elaboratori di tipo specializzati
- analogamente e sempre ragionando in termini evolutivi non è pensabile dotare i sistemi specializzati di tutte le funzionalità tipiche dei sistemi *general purpose* e pensare di poter mantenere allo stesso tempo l'efficienza di prestazioni e gli interessanti aspetti economici
- un assetto infrastrutturale che semplicemente veda affiancati sistemi *general purpose* e numerosi *sistemi specializzati*, con livello di integrazione minimo o nullo, risulta poco praticabile sulla base dell'esperienza già oggi maturata con le infrastrutture di tipo distribuito che vedono un incremento vertiginoso dei costi totali di gestione al crescere della molteplicità dei sistemi da gestire individualmente.

Una possibile risposta a questa complessa serie di considerazioni è oggi rappresentata dai *sistemi ibridi* ovvero da sistemi costituiti da una combinazione integrata di elaboratori eterogeneii, di tipo mainframe o comunque *general purpose* con elaboratori di tipo *special purpose* per

eseguire applicazioni di tipo tradizionale e di tipo emergente e mantenendo allo stesso tempo l'immagine di sistema unico.

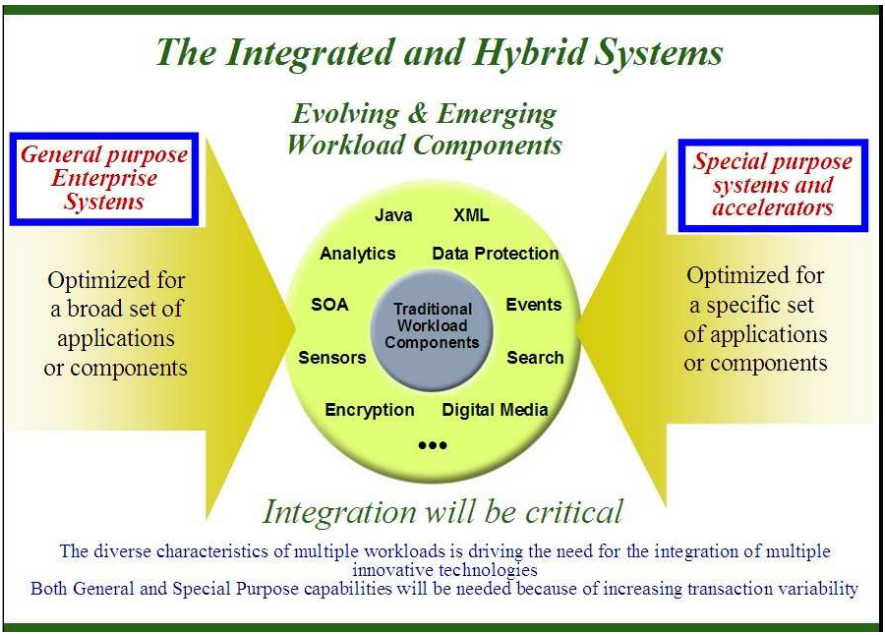


Figura 179 Sistema Ibrido Integrato

Il livello di integrazione fra i componenti di un sistema ibrido può essere di vario tipo, passando da soluzioni di tipo "loosely coupled" (interconnessione attraverso rete di comunicazione utilizzando protocolli più o meno standard) o integrazioni più strette (tightly coupled) ove l'interconnessione è a livello hardware e la velocità di comunicazione ovviamente più elevata. In questo senso si parla anche di acceleratori o co-processori "attached" o "integrated". La realizzazione di sistemi funzionalmente avanzati mediante l'integrazione di sistemi eterogenei non è un discorso nuovo e soluzioni di tipo ibrido sono state spesso disponibili, a cominciare dalle prime configurazioni di Attached Support Processor (ASP) degli anni '70 in cui un elaboratore S/360 veniva usato per controllare l'esecuzione dei lavori di un

altro sistema S/360, di solito più potente attraverso una connessione di tipo channel to channel operante con protocollo proprietario. L'obiettivo era di eseguire a livello sistema operazioni complesse di schedulazione che altrimenti avrebbero dovuto essere eseguite dal personale operativo. Più di recente e per restare all'interno del mondo mainframe, ricordiamo le diverse soluzioni di integrazione di sistemi *special purpose* e di accelerators già presenti oggi negli elaboratori System z quali:

- i processori specializzati presenti all'interno del chip per le funzioni crittografiche e la funzione di compressione dei dati
- i processori specializzati derivati dai motori standard per fornire supporto a particolari funzioni di sistema (motori ICF) o per ottimizzare i costi complessivi in presenza di particolari ambiti applicativi quali Linux (motori IFL), programmi Java (motori zAAP) e tipologie particolari di accesso ai dati (motori zIIP).

Infine ricordiamo gli ultimi importanti sviluppi di integrazione di sistemi eterogenei, cioè integrazione del sistema z con il processore Cell Broadband Engine e integrazione del sistema z con i sistemi DataPower.

Il processore Cell Broadband Engine (Cell BE), annunciato da IBM nel 2005 come risultato di un progetto congiunto Sony, Toshiba e IBM iniziato nel 2001, è già di per se un sistema ibrido in quanto combina in un unico chip di dimensione contenute un motore general-purpose di architettura Power e di modesta capacità elaborativa con numerosi (attualmente 8) processori specializzati che impiegano modelli di esecusione SIMD (Single Instruction Multiple Data) secondo la classificazione di Flynn (Flynn, M., Some Computer Organizations and Their Effectiveness, IEEE Trans. Comput., Vol. C-21, pp. 948, 1972) per l'esecuzione di carichi applicativi ad alto ontenuto computazionale quali i calcoli analitici, le applicazioni multimediali ed il calcolo vettoriale. Il programma di controllo è basato su Linux.

Le prime realizzazioni di integrazione fra i sistemi z ed il processore Cell BE (disponibile da IBM come sistema Blade Center QSXX) sono del tipo loosely-coupled con collegamento via rete e protocollo TCP/IP per supportare una nuova generazione di applicazioni internet 3D.

(www.ibm.com/industries/financialservices/doc/content/news/pressrelease/2478857103.html).
E' prevedibile un passo successivo verso un livello maggiore di integrazione a livello hardware (tightly coupled) che permetta di realizzare un sistema ibrido ove l'enorme capacità elaborativa del Cell BE è facilmente accessibile da un sistema con gli elevati livelli di qualità del servizio tipici dell'ambiente manframe.

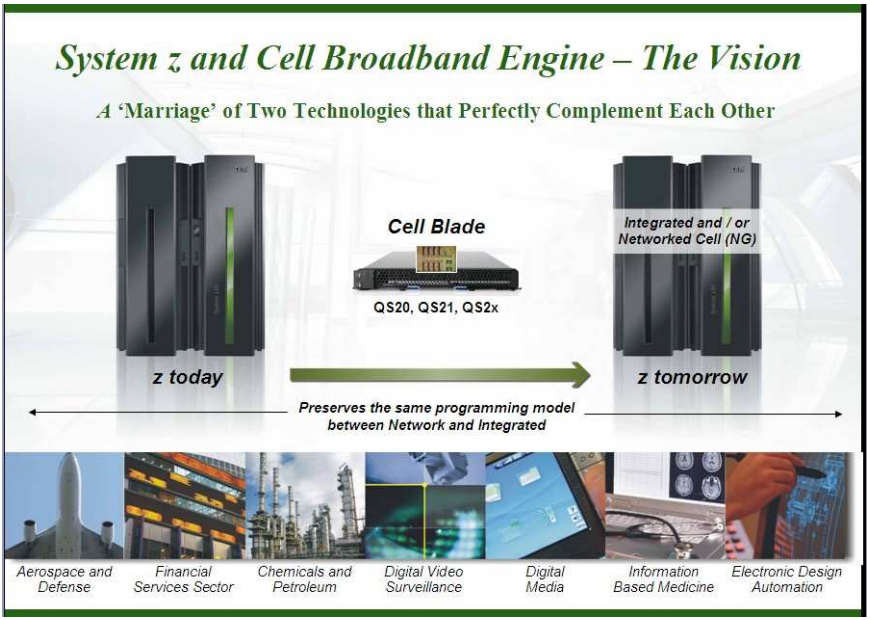


Figura 180 Il primo esempio di Sistema Ibrido

Un discorso molto simile può esser fatto per l'integrazione del sistema z con le cosiddette IBM WebSphere DataPower SOA appliances. DataPower, originariamente un costruttore di dispositivi di rete di base a Cambridge, Massachusetts, USA, è dal 2005 una divisione prodotti della IBM che produce appliances XML per l'elaborazione di messaggi XML e per la più generica trasformazione di messaggi per le comunicazioni server-to-server.

Tali appliances usano hardware ASIC ed un programma di controllo basato su Linux. Queste soluzioni permettono già oggi di aumentare le prestazioni, la sicurezza e la gestione di applicazioni che fanno uso di standard XML con configurazioni ibride loosely-coupled.
www.ibm.com/industries/financialservices/doc/content/resource/3868697103.html

Anche per questo tipo di soluzioni è prevedibile una soluzione ibrida più integrata che permetta di associare le importanti funzionalità DataPower con la qualità di servizio dei sistemi z. In termini più generali è possibile prevedere per il futuro la disponibilità di sistemi ibridi che integrino in modo più o meno stretto funzionalità di tipo *general purpose* con funzionalità di tipo *special purpose*. Le caratteristiche principali di queste soluzioni saranno, sempre prevedibilmente, le seguenti:

- uno strato di hardware costituito da un numero variabile di sistemi di tipo sia *general purpose* che *special purpose* affiancati dai vari tipi di co-processors ed accelerators che l'industria informatica renderà disponibili
- una serie di dispositivi di interconnessione per consentire comunicazioni veloci e sicure fra i vari componenti hardware e per consentire l'accesso ai dispositivi esterni
- uno strato di virtualizzazione (Hypervisor) che mascheri all'esterno la molteplicità anche architetturale dei componenti hardware e fornisca importanti funzionalità quali il provisioning on demand
- una serie di istanze di ambienti elaborativi caratterizzate ciascuna dal proprio sistema operativo e dalle proprie applicazioni e con la possibilità di condividere i dati
- uno strato di software di controllo per la gestione dell'intero sistema che mantenga nei confronti dell'utilizzatore esterno l'immagine del sistema unico. Stante la criticità sopratutto per le prestazioni di questo elemento di controllo è prevedibile che questa funzionalità venga realizzata in firmware.

I punti di forza di una evoluzione della piattaforma z verso soluzioni di tipo ibrido sono:

- la possibilità di eseguire sulla stessa piattaforma workload di tipo tradizionale e workload di tipo emergente, rimanendo sempre in un ambiente operativo con le caratteristiche della piattaforma z
- la possibilità di utilizzare nuove tecnologie e mantenere una visione unica di sistema, con le caratteristiche tipiche del mondo mainframe, per quanto riguarda le tecniche di gestione e l'accesso ai dati
- la possibilità di far crescere le dimensioni dell'infrastruttura informatica al crescere delle esigenze applicative senza per questo incidere in modo determinante sui costi di gestione.

Tabella delle figure

Riferimenti Bibliografici

Documentazione di riferimento della z/Architecture

- z/Architecture Principles of Operation http://www-01.ibm.com/support/docview.wss?uid=isg26480faec85f44e2385256d5200627dee

Introduzione al mainframe e ai suoi sistemi operativi

- Introduction to the New Mainframe: z/OS Basics *http://www.redbooks.ibm.com/abstracts/SG246366.html*
- Introduction to the New Mainframe: z/VM Basics *http://www.redbooks.ibm.com/redpieces/abstracts/sg247316.html*
- Introduction to the New Mainframe: z/VSE Basics *http://www.redbooks.ibm.com/abstracts/sg247436.html*
- Linux for IBM System z9 and IBM zSeries *http://www.redbooks.ibm.com/abstracts/sg246694.html*
- Introduction to the New Mainframe: Networking *http://www.redbooks.ibm.com/abstracts/sg246772.html*
- Introduction to the New Mainframe: Large-Scale Commercial Computing *http://www.redbooks.ibm.com/abstracts/sg247175.html*
- Introduction to the New Mainframe: Security *http://www.redbooks.ibm.com/redpieces/abstracts/sg246776.html*
- IBM System z Strengths and Values *http://www.redbooks.ibm.com/abstracts/sg247333.html*

E.F. Codd, "A Relational Model of Data for Large Shared Data Banks", Comm. of the ACM, Volume 13, Number 6, June 1970, reperibile alla pagina *http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf*

C.J. Date, "Edgar F. Codd August 23rd, 1923 - April 18th, 2003 A Tribute", 2003, reperibile alla pagina *http://www.sigmod.org/codd-tribute.html*

K. Beyer and al., "DB2 goes hybrid: Integrating native XML and XQuery with relational data and SQL, IBM Systems Journal, Volume 45, Number 2, 2006, reperibile alla pagina
http://www.research.ibm.com/journal/sj/452/beyer.html

Hardware

- IBM System z10 *http://www-03.ibm.com/systems/z/news/announcement/20080226_annc.html*
- Il processore z990 http://researchweb.watson.ibm.com/journal/rd/483/slegel.pdf
- IBM System z9 Business Class Technical Introduction *http://www.redbooks.ibm.com/abstracts/sg247241.html*
- IBM System z9 Enterprise Class Technical Guide *http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sg247124.html*
- IBM System z Connectivity Handbook *http://www.redbooks.ibm.com/abstracts/sg245444.html*

ABC della Programmazione di sistema dello z/OS

- **Volume 1:** Introduction to z/OS and storage concepts, TSO/E, ISPF, JCL, SDSF, and z/OS delivery and installation *http://www.redbooks.ibm.com/abstracts/sg246981.html*
- **Volume 2:** z/OS implementation and daily maintenance, defining subsystems, JES2 and JES3, LPA, LNKLST, authorized libraries, SMP/E, Language Environment *http://www.redbooks.ibm.com/abstracts/sg245652.html*
- **Volume 3:** Introduction to DFSMS, data set basics storage management hardware and software, catalogs, and DFSMStvs *http://www.redbooks.ibm.com/abstracts/sg246983.html*
- **Volume 4:** Communication Server, TCP/IP, and VTAM *http://www.redbooks.ibm.com/abstracts/sg245654.html*
- **Volume 5:** Base and Parallel Sysplex, System Logger, Resource Recovery Services (RRS), global resource serialization (GRS), z/OS system operations, automatic restart management (ARM),

Geographically Dispersed Parallel Sysplex (GDPS)
http://www.redbooks.ibm.com/abstracts/sg246985.html

- **Volume 6:** Introduction to security, RACF, Digital certificates and PKI, Kerberos, cryptography and z990 integrated cryptography, zSeries firewall technologies, LDAP, and Enterprise identity mapping (EIM)
  http://www.redbooks.ibm.com/abstracts/sg246986.html?Open
- **Volume 7:** Printing in a z/OS environment, Infoprint Server and Infoprint Central
  http://www.redbooks.ibm.com/abstracts/sg246987.html
- **Volume 8:** An introduction to z/OS problem diagnosis
  http://www.redbooks.ibm.com/redpieces/abstracts/sg246988.html
- **Volume 9:** z/OS UNIX System Services
  http://www.redbooks.ibm.com/abstracts/sg246989.html
- **Volume 10:** Introduction to z/Architecture, zSeries processor design, zSeries connectivity, LPAR concepts, HCD, and DS8000
  http://www.redbooks.ibm.com/abstracts/sg246990.html
- **Volume 11:** Capacity planning, performance management, WLM, RMF, and SMF
  http://www.redbooks.ibm.com/abstracts/sg246327.html

Il consolidamento dei serventi

- Linux on IBM zSeries and S/390: Server Consolidation with Linux for zSeries *http://www.redbooks.ibm.com/abstracts/redp0222.html*