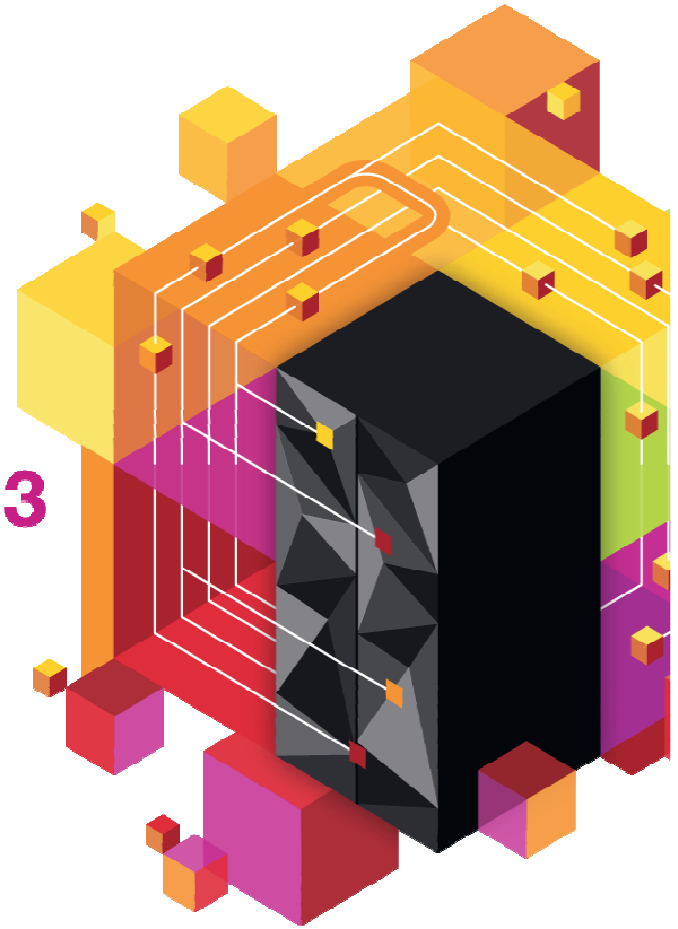
# Université du Mainframe 2013

## 4-5 avril



# Prévoir et diagnostiquer les problèmes :
## du nouveau dans z/OS et zEC12

Floréal Ardeo
Executive IT/Specialist
IBM Systems & Technology Group France

**Université du Mainframe 2013**

**4-5 avril**

# Agenda

- Type of Failures
  - Soft Failures
- Anatomy of an Outage
- IBM z Availability
  - Components involved
- IBM zAware
- Run Time Diagnostics (RTD)
- IBM Heath checker for z/OS
- Predictive Failure analysis (PFA)

# Complex Systems



WebSphere
Information Management
Tivoli
Lotus.
Cognos.
Java
Rational.


IBM Mainframes are a complex environment

- Hardware
- Software
- Compilers
- Database systems
- Transaction processing systems
- Middleware
- User applications

In any system there is the potential for failure

The job of the System Programmer is to deal with failure

# Types of failure on System z and z/OS

Not all failure are alike
Broadly categorize failures into 3 types

## Masked Failure

- Software/Hardware detects failure
- Software/Hardware corrects failure
- No impact to business
- Example: Hardware power supply failure: switch to alternate, IBM alerted, concurrent replacement

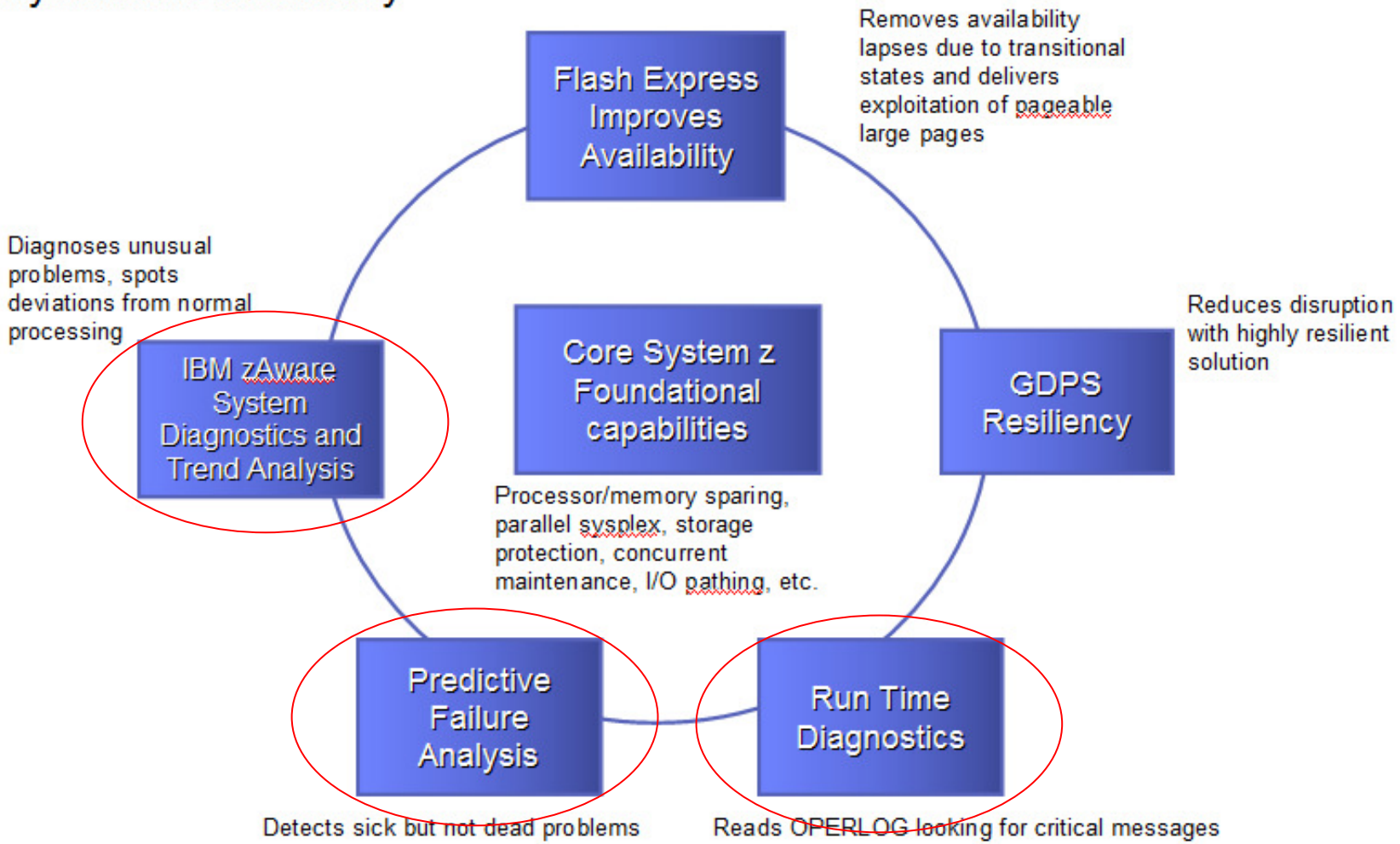
## Hard Failure

- Software/Hardware detects failure
- Automations and operations restart the failing component
- Minimal impact to business
- Example: Application terminates but is restarted by ARM

## Soft Failure

- User detects failure, impact to business.
- Difficult to determine recovery actions
- Example: component is failing, holds resources (locks, enqueues) required by other components, causes sysplex wide stall, leads to sysplex wide IPL.

# System z Availability



# What is a soft failure

Systems don't break, but seem to just stop working:
"sick but not dead" or soft failure

**Symptoms of a Soft Failure**

- 80% of business impact, but only about 20% of the problems
- Long duration
- Infrequent
- Unique
- Can be software or hardware
- Cause creeping failure and "sympathy sickness"
- Hard to determine how to isolate
- Hard to determine how to recover
- Hard for software to detect internally
- Probabilistic, not deterministic

**Manifested as**

- Stalled / hung processes
  - Single system, Sysplex members
  - Sympathy sickess
- Resource contention
- Storage growth
- CF, CDS growth
- I/O issues (paths, response time)
- Repetitive errors
- Queue growth
- Configuration
  - Singe point of failure, thresholds, cache structure size, not enabling new features

# Dealing with soft failure problems

Systems running

Seems impaired
- Slow
- Applications "not working"
- Sysplex problems
- UNIX problems
- Irate users

Action!: Check out the systems for cause

Outstanding contention?
Critical errors detected?
Important tasks looping?
I/O bottlenecks?
Network issues?

Prep for call

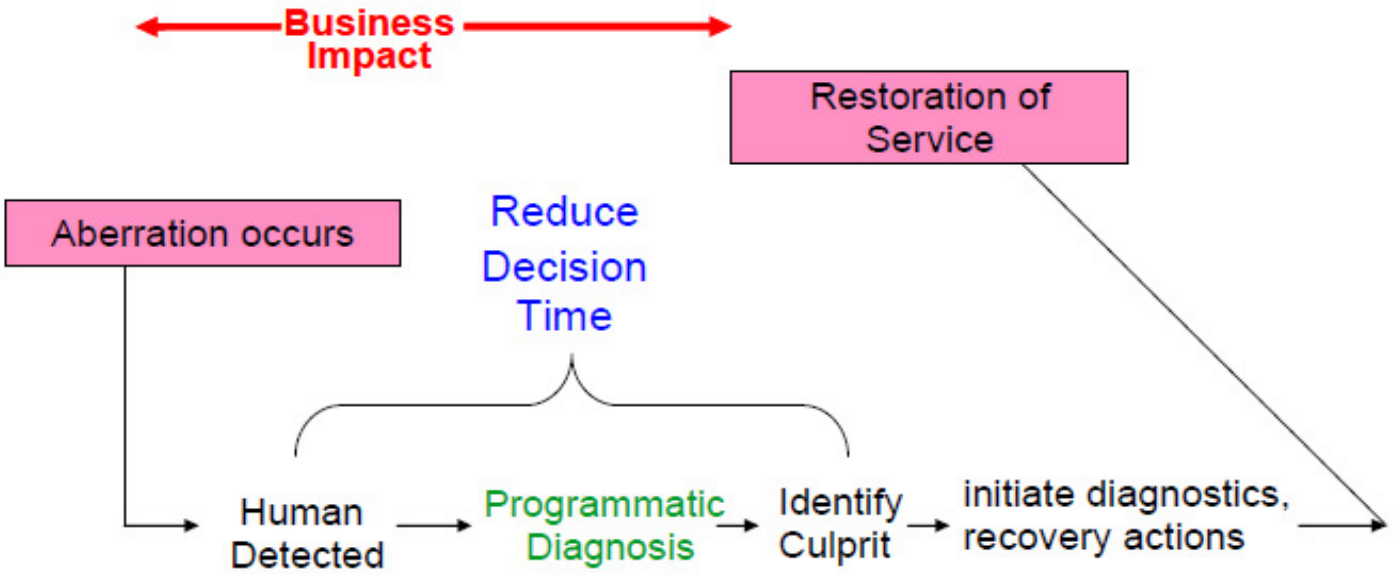
Is it fixed yet?

Investigate the clues using monitors and commands

Get diagnostic data such as dumps and command output

Restart the system
Open a PMR
Fix your resume/CV

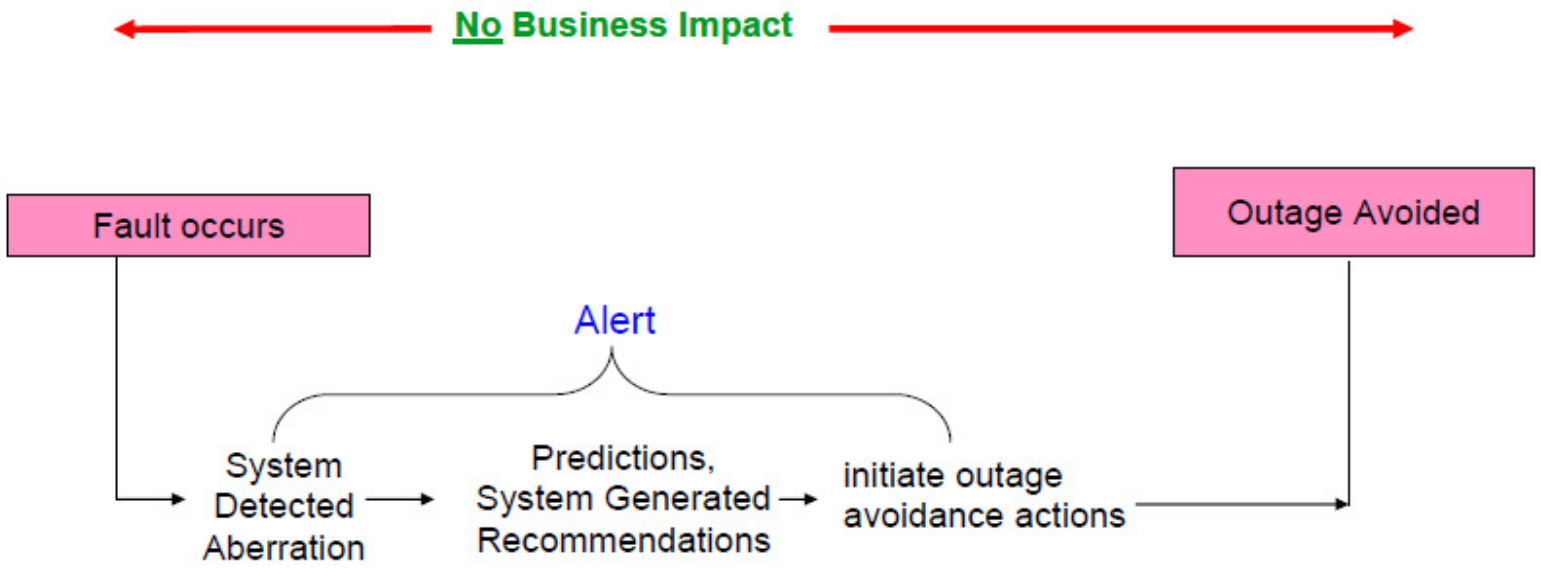
# Anatomy of an Outage – One step further




*RAS Innovation --- Run Time Diagnostics:*

- *Machine-speed understanding*
- *Better tooling to identify the culprit*
- *Enables faster / correct recovery actions*

# Anatomy of an Outage – Objective



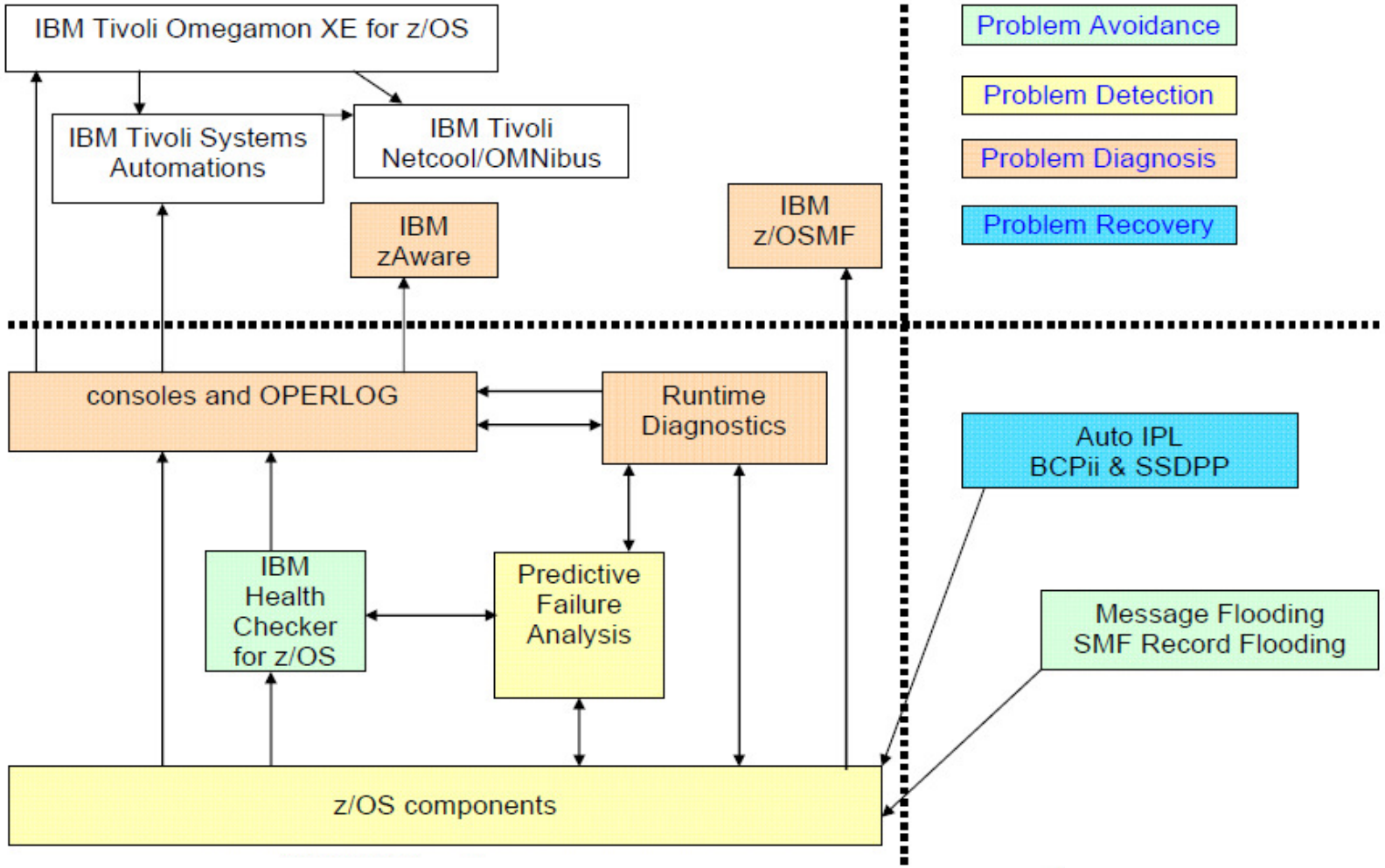

*RAS Innovations: Predictive Technologies*

- *Machine Learning - Convert diagnostic data to knowledge in real time*
- *Convert soft failures to correctable incidents*

# Soft failure issues and solutions

| Issues | Solutions |
| --- | --- |
| Risk to business<br>•The impact of the symptoms<br>•Risk of recurrence<br>•Impact in getting system stabilized | Prevention<br>•provide policy based tools to prevent predictable failures |
| Complexity of performing the task | Detect/Alert<br>•identify “sick, but not dead” or possible conditions that could lead to larger issues |
| Troubleshooting a live system and recovering from an apparent failure | Diagnosis<br>•better real time diagnosis and diagnostic tools |
| Data collection is very time consuming | Recovery<br>•improve mean time to recovery |
| Significant skill level required to diagnose and analyze problem | Diagnostic data capture<br>•make data capture easier and less time consuming |

# Problem Avoidance, Detection, Diagnosis and Recovery



# IBM z/OS Solutions Address Problem Determination

| Solutions Available: | | Rules based | Analytics / Statistical model | Examines message traffic | Self Learning | Method |
|---|---|---|---|---|---|---|
| **z/OS Health Checker** | ▪Checks configurations<br>▪Programmatic, applies to IBM and ISV tools<br>▪Can escalate notifications | ✓ | | | | Rules based to screen for conditions |
| **z/OS PFA** | ▪Trending analysis of z/OS system resources, and performance<br>▪Can invoke z/OS RTD | | ✓ | | ✓ | Early detection |
| **z/OS RTD** | ▪Real time diagnostics of specific z/OS system issues | ✓ | | ✓ | | After an incident |
| **IBM zAware** | ▪Pattern based message analysis<br>▪Self learning<br>▪Provides aid in diagnosing complex z/OS problems, including cross sysplex, problems that may or may not bring the system down | | ✓ | ✓ | ✓ | Diagnosis Useful before or after an incident |

- IBM zAware Uniquely analyzes messages in context to determine unusual behaviors
- IBM zAware Uniquely understands and tunes its baseline to compare against your current activity
- IBM zAware does not depend on other solutions, manual coding of rules, and is always enabled to watch your system

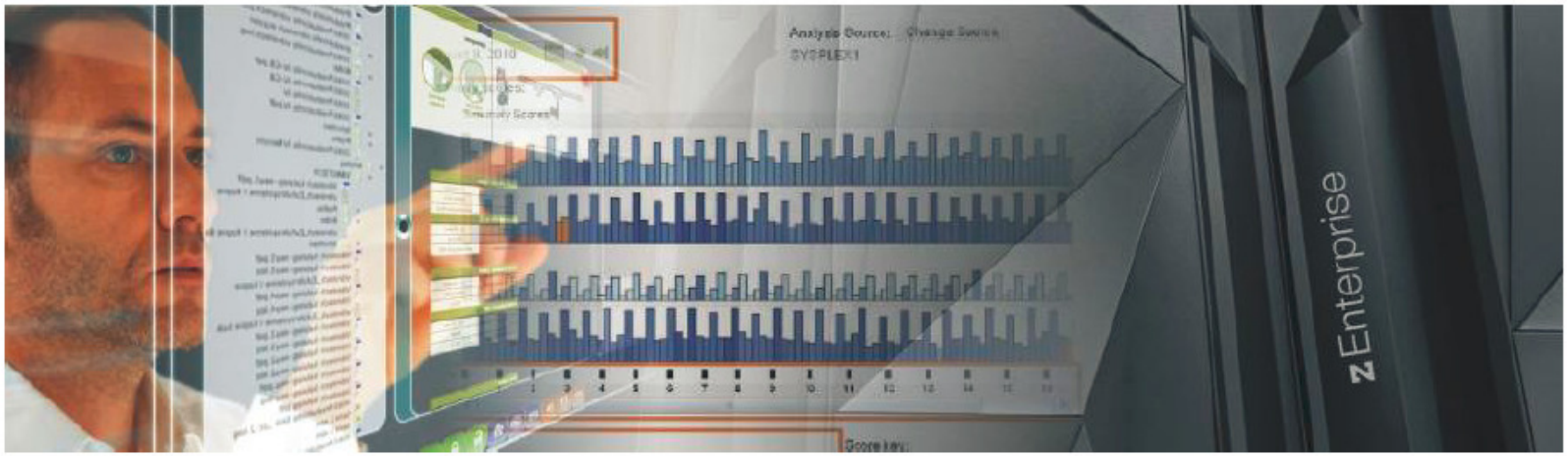
# Resiliency offering on System z

| | Make sure system is likely to work | Find cause of event after event was reported | Report "first" occurrence of event (before externally visible) |
|---|---|---|---|
| **Rules Based Performance** | Capacity planning – RMF®[1] | OMEGAMON® XE | OMEGAMON XE |
| **Rules Based Non Performance** | Health checker for z/OS | RTD | NetView® / TSA |
| **Analytical / Statistically Based Performance** | ITM 6.2.1 | Netcool® Tivoli Performance Analyzer | ITM 6.2.1 |
| **Analytical / Statistically Based Non Performance** | IBM zAware[2] | IBM zAware | PFA – control charts IBM zAware – pattern analysis |

[1] RMF collects the data for customer analysis / customer rules

[2] Changes

# IBM zAware



# IBM zAware –
## *IBM System z Advanced Workload Analysis Reporter*

IBM System z: Mainframe focus

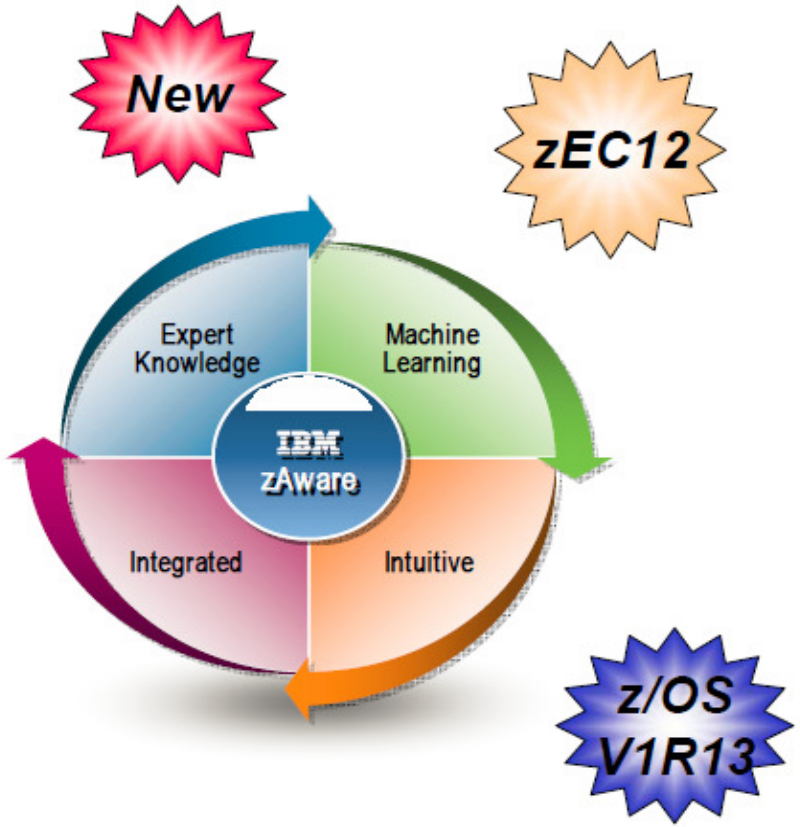
- Server runs on zEnterprise EC12
- Clients are z/OS V1R13

Advanced: Designed by smart people who understand concepts such as Poisson Distribution

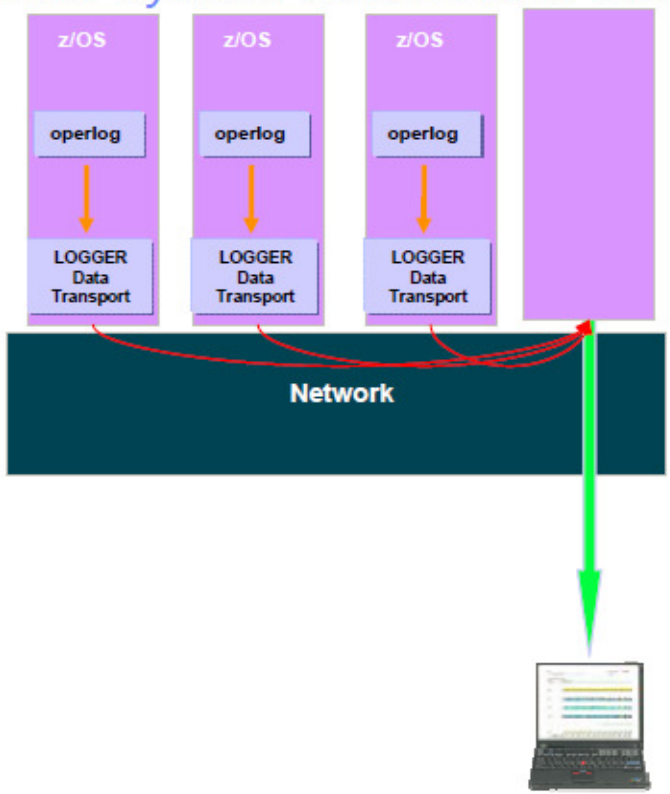
Workload: Data processed is OPERLOG

Analysis: looks at the behavior of messages and message patterns occurring now compared to a model of past behavior

Reporter: message analysis is presented in a graphical view



# IBM zAware –
## *IBM System z Advanced Workload Analysis Reporter*



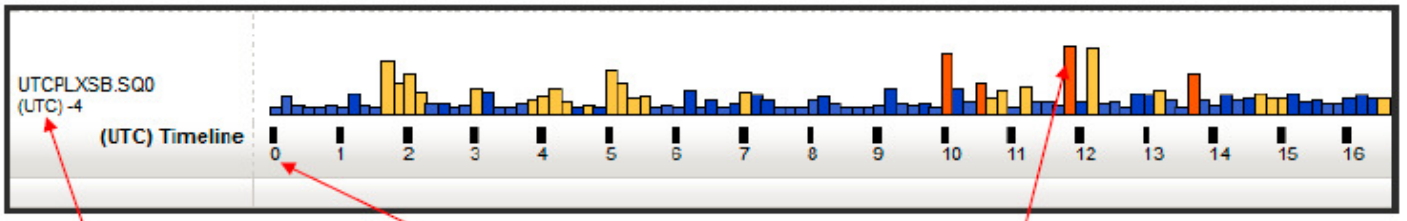

- Monitors z/OS OPERLOG messages including all z/OS console message, ISV and application generated messages
- Can monitor across a sysplex
- Uses a model of past system behavior to compare current message patterns
- Color-coded, browser-based (IE 8, Firefox)
- Visual indicators
  - Number of unique message ids for an interval
  - Anomaly score for an interval
- Detects anomalies monitoring systems miss:
  - Messages may be suppressed or rare
  - Messages may indicate a trend
- XML Output consumable through published API, can drive ISV products

# Analysis

- After a model for the system has been successfully created via training, analysis on real time message traffic can occur.
- This step takes the current messages in real-time and uploads them in a manner similar to the initial upload. Every two minutes, this step performs comparisons with the model and updates the xml and the IBM zAware GUI.
- The comparisons use scoring to detect the following: unique messages (those that were not seen in the data that was used in training), rare messages, anomalous message patterns (such as finding a message out of context of a known pattern), and messages that are occurring more frequently than expected.
- The score is also calculated based on the criticality of messages which may give the message more weight if a message is deemed critical or may reduce the score if a message is deemed inconsequential.
- Messages are retrieved from the system logger prior to being excluded due to message flooding automation.
- A multi-line WTO message is counted as one message.

# IBM zAware Analysis Panel

# IBM zAware Analysis system timeline

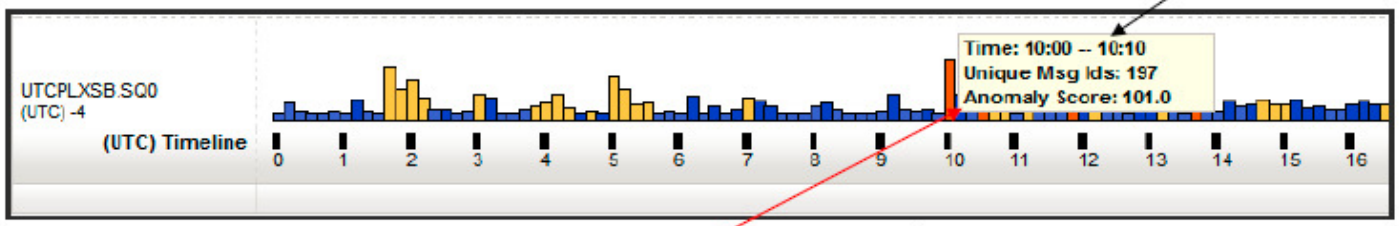



Sysplex & LPAR

24 hour timeline

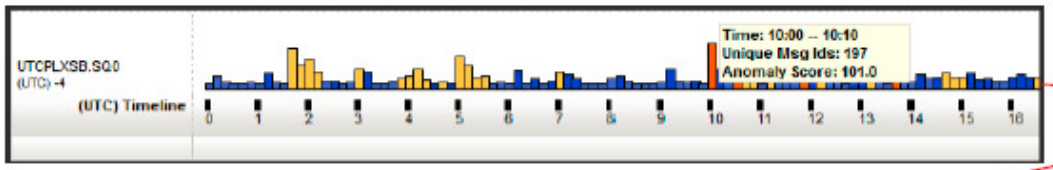
Visual Indicator are bars

Each bar represents a 10 minute interval
Height represents number of unique messages seen during the interval




Color represents anomaly score based on previous history - keyed

# IBM zAware Interval View




Drill down to Interval View by clicking on bar

**Interval View for System SQ0**

The Messages table provides detailed analysis information for each message that occurred during the indicated time interval. To view message details for other intervals use the date and time interval selectors. Click the **Return to Analysis** button to go back to the Analysis view.

Date: September 11, 2012

Time interval (UTC): 10:00 -- 10:10

Analysis Source: All Monitored Systems

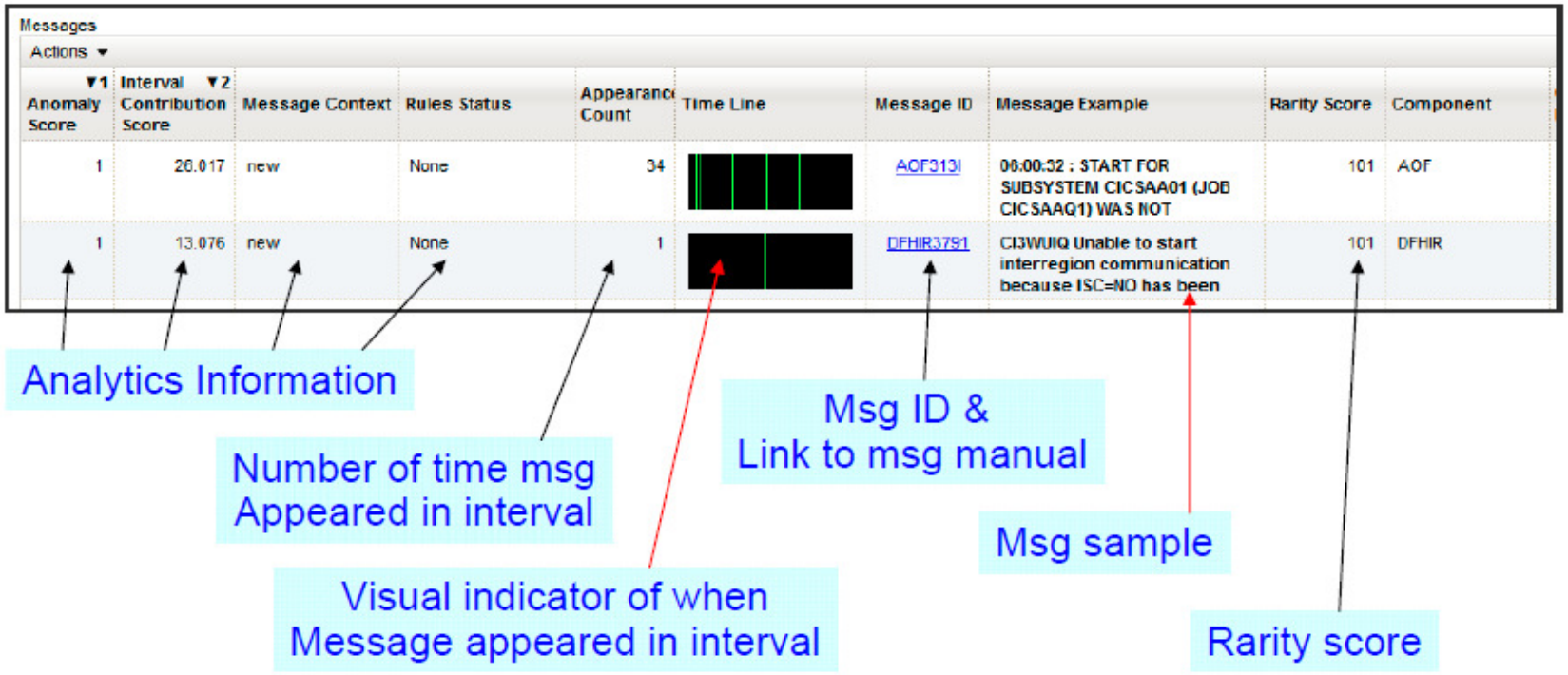
Interval anomaly score: 101.0

Messages

Actions

| ▼1 Anomaly Score | Interval ▼2 Contribution Score | Message Context | Rules Status | Appearance Count | Time Line | Message ID | Message Example | Rarity Score | Component | Cluster ID |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 26.017 | new | None | 34 | | AOF313I | 06:00:32 : START FOR SUBSYSTEM CICSAA01 (JOB CICSAAQ1) WAS NOT | 101 | AOF | -1 |
| 1 | 13.076 | new | None | 1 | | DFHIR3791 | CI3WUIQ Unable to start interregion communication because ISC=NO has been | 101 | DFHIR | -1 |
| 1 | 13.076 | new | None | 1 | | DFHPA1910 | CI3WUIQ SIT OVERRIDE AUTORESTTIME= IS NOT RECOGNIZED. OVERRIDE IS | 101 | DFHPA | -1 |
| 1 | 13.076 | new | None | 1 | | DFHPA1916 | CI3WUIQ SIT OVERRIDE DATA 47185920 IS OUT OF RANGE FOR KEYWORD EDSALIM= . | 101 | DFHPA | -1 |
| 1 | 13.076 | new | None | 1 | | EYUVS0001I | CI3WUIQ CICSPLEX SM WEB USER INTERFACE INITIALIZATION STARTED. | 101 | EYUVS | -1 |

# IBM zAware Interval View Message line



# Analysis

- Anomaly Score indicates the rarity of this specific message ID within the selected interval. Higher scores indicate greater anomaly so messages with high anomaly scores are more likely to indicate a problem.
- Interval Contribution score Indicates the relative contribution of this message to the anomaly score for the 10-minute interval.
- Rarity Score indicates how rarely this message was issued within the selected 10-minute interval.
- Appearance Count specifies the number of times that this message was issued within the selected 10-minute interval.
- Cluster ID provides the identifier of the cluster to which this message belongs. When the message is not part of a recognized cluster, the cluster ID is -1.
  - Clusters are message patterns (messages appearing together as part of some system event) which define the normal context for messages. Identification of Clusters is performed when a model is created.

# Analysis

Message Context indicates whether or not this message is part of an expected pattern of messages (cluster)

- New
  - IBM zAware has not previously detected this message in the client model.
- Unclustered
  - This message is not part of a defined cluster.
- In context
  - This message was issued as expected, within a cluster to which this message belongs.
- Out of context
  - This message is expected to be issued as part of a specific cluster but was issued in a different context.

# Analysis

Rules Status indicates whether or not the IBM zAware server uses an IBM rule to determine the message anomaly score

- Critical
  - critical for diagnosing a potential system problem. For example, message IXC101I, which indicates that a system is being removed from a sysplex.
- Important
  - likely to indicate a problem. For example, message IEA911E, which indicates that an SVC dump has been taken.
- Interesting
  - indicative of a diagnostically useful event, such as a health check exception.
- None
  - No IBM rule is applied for this message.
- Not interesting
  - routine, with little or no diagnostic value. These messages are usually issued frequently and at random.

# Analysis

Interval anomaly score (remember – based on analysis against model)

- 0 through 99.4
  - messages and message clusters that match or exhibit relatively insignificant differences in expected behavior
- 99.5
  - some rarely seen, unexpected, or out-of-context messages
- 99.6 – 100
  - rarely seen messages (these messages appear in the model only once or twice), or many messages that are unexpected or issued out of context
- 101
  - contain unique message IDs that the IBM zAware server has not detected previously in the client model, or messages that IBM rules define as critical messages

# How can IBM zAware Improve Problem Determination?

- **Identify messages indicating a possible z/OS incident is happening**
  - *Which image is behaving abnormally?*
    - Examines unique messages
    - High score generated by
      - unusual messages or message patterns
  - *When did this unusual behavior start?*
  - For a selected 10 minute interval either the current 10 minute interval or past intervals
    - **Which message ids** are unusual?
    - **How often** did the message occur?
    - **When** did the message start to occur?
  - *Were similar messages issued in the past?*
    - Similar characteristics, Same pattern?

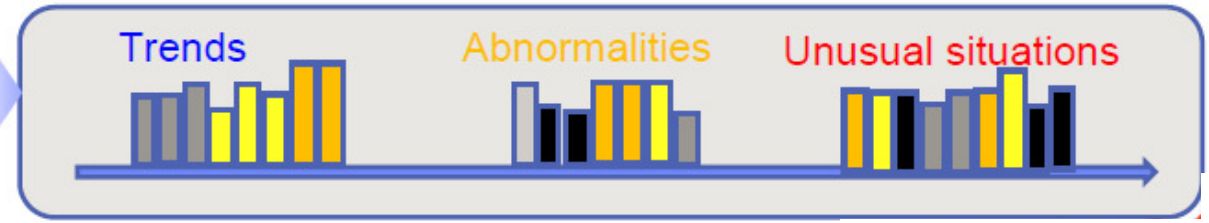
- **After a change has been made**
  - *Are unusual messages being issued following changes ?*
    - New software levels (operating system, middleware , applications)
    - Updated system settings / system configurations

- **When diagnosing the cause of an intermittent problem**
  - *Are new unusual messages being issued in advance of the problem?*
  - Are more messages issued then expected?
  - Are messages issued out of normal pattern or context?

Vertical bar shows the number of unique messages in a 10 minute interval

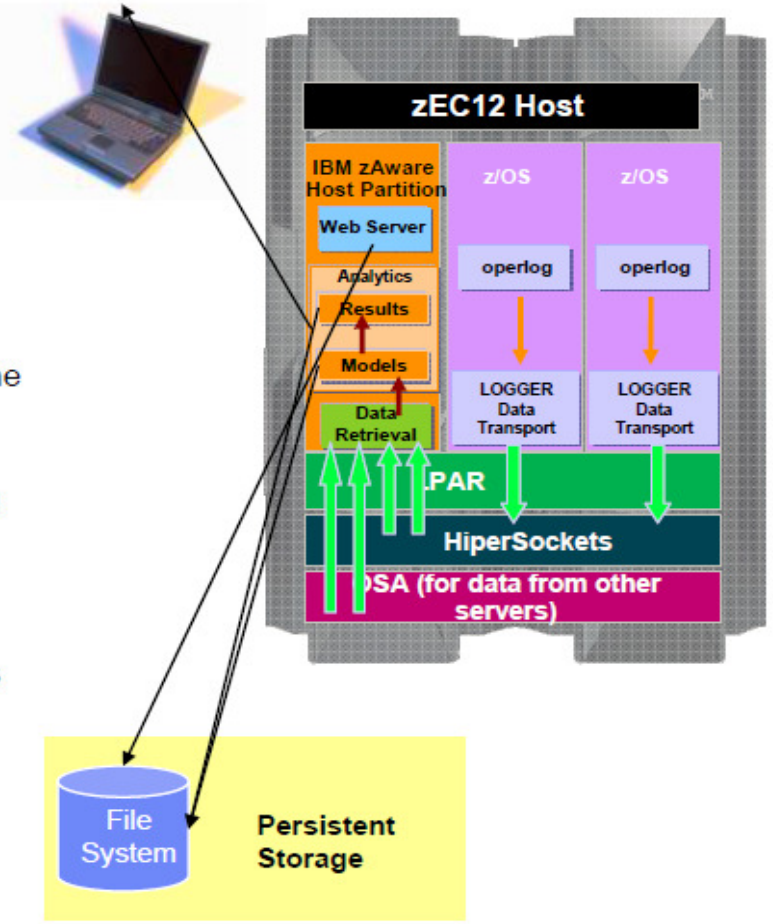
Scoring of messages color coded from no difference (blue) to significantly different (orange)

***Finds Anomalies that would be Hard to Detect***

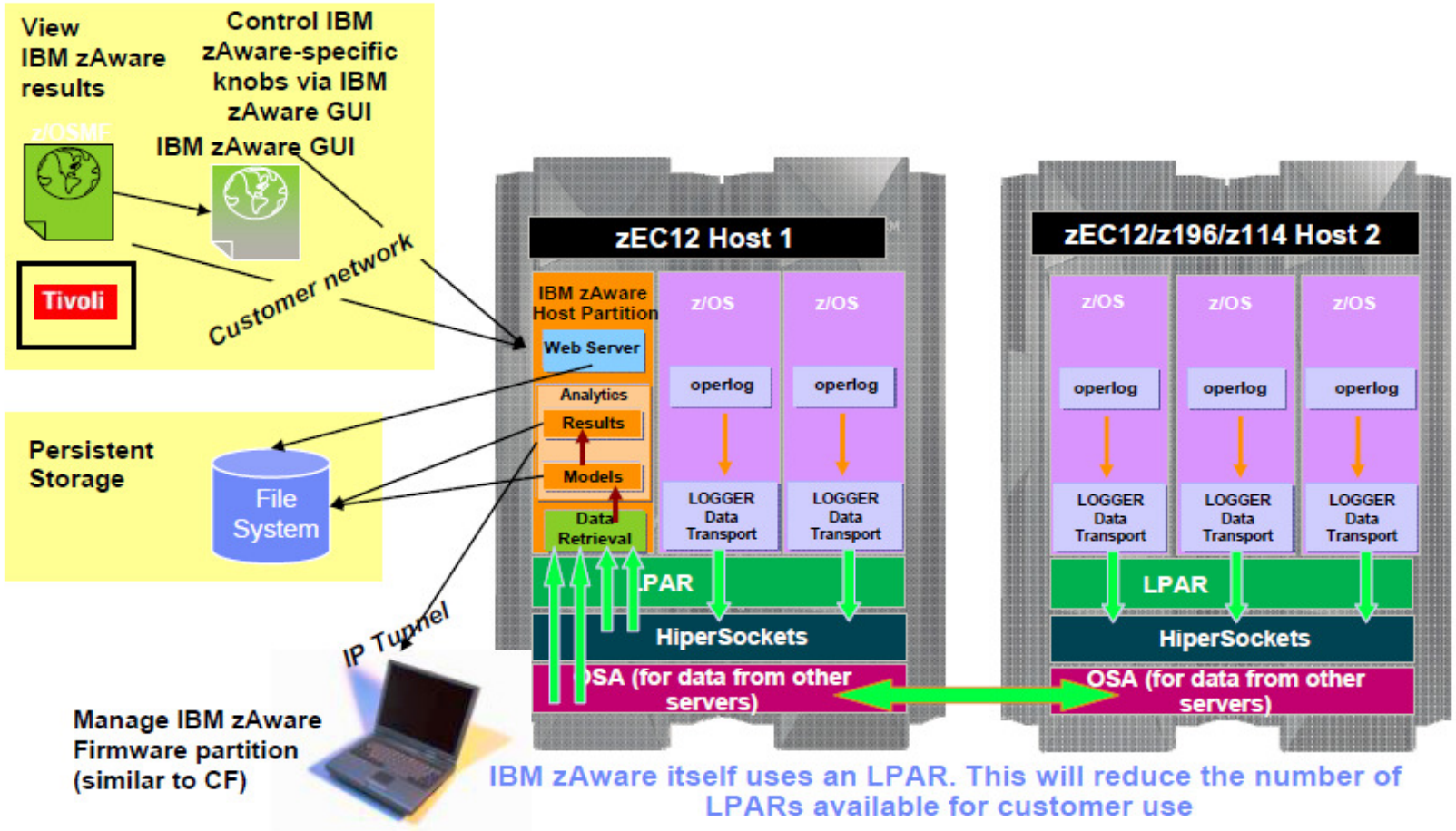


# A closer look inside IBM zAware

- zEC12 to host IBM zAware Server
  - IBM zAware requires it's own LPAR and runs it's own self-contained firmware stack.
    - This will reduce the number of LPARs available for customer use
  - IBM zAware processor resources can be IFL or General Purpose CP (shared or dedicated)
  - Memory and DASD resources are dependent on the number of monitored clients, amount of message traffic, length of time data retained
    - Memory - Min 4 GB + 256 MB per connected client.
    - DASD ~ 250-500 GB (ECKD)
    - IBM zAware uses Logical Volume Manager (LVM) to aggregate multiple physical devices into a single logical device
  - Network: HiperSockets or OSA ports – for both gathering of instrumentation data, and outbound alerting/communications
    - Need dedicated IP address for partition



# Overview




IBM zAware itself uses an LPAR. This will reduce the number of LPARs available for customer use

# IBM zAware Monitored Client Logical Architecture

Enhancements to System Logger to allow transmittal of OPERLOG data to IBM zAware.

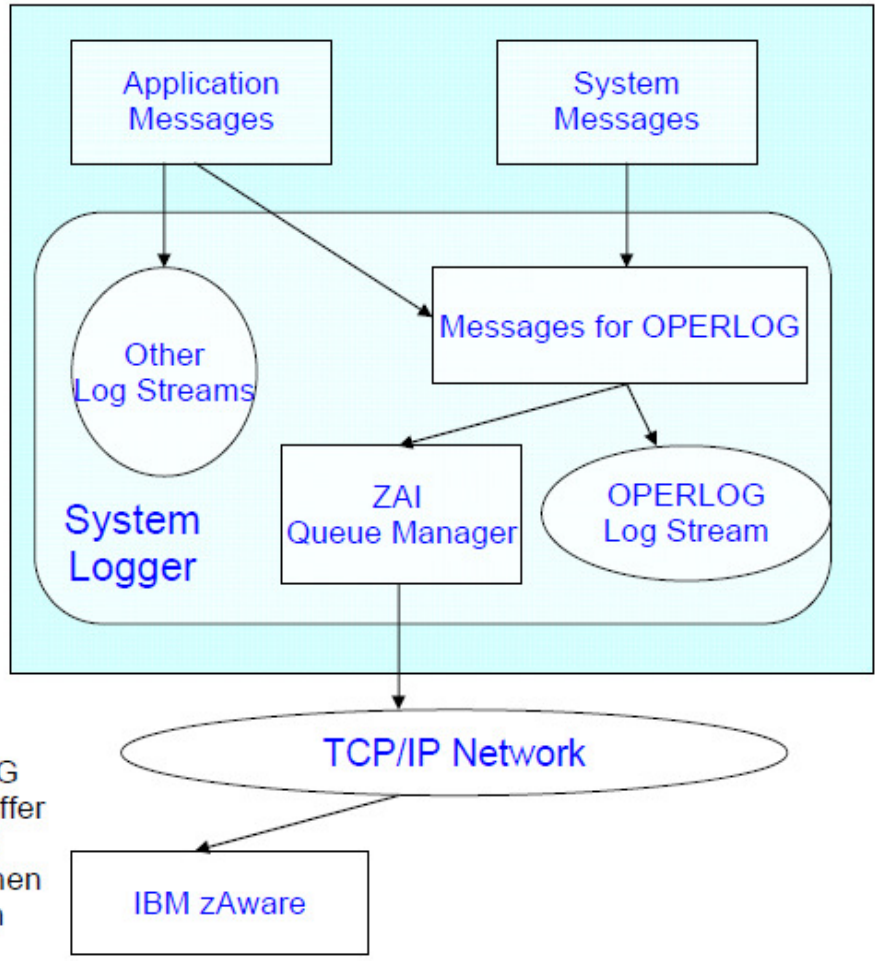
New OPERLOG Log Stream parameters
- •ZAI(YES)
- •ZAIDATA('OPERLOG')

New IXGCNFxx parameters
- •ZAI
  - •SERVER
  - •PORT
  - •LOGBUFMAX
  - •LOGBUFWARN
  - •LOGBUFFULL

Enhancements to DISPLAY LOGGER command
- •DISPLAY LOGGER,STATUS,ZAI
- •DISPLAY LOGGER STATUS,ZAI,VERIFY

Enhancements to SETLOGR command
- •SETLOGR FORCE,ZAICONNECT,LSN=xx
- •SETLOGR FORCE,ZAIQUIESCE,LSN=xx

When a message is to be written to the OPERLOG log stream, the message is copied to a special buffer in System Logger (reserved for use by messages that are destined for IBM zAware). The buffer is then dequeued and sent to the IBM zAware application across a TCP/IP connection.



# Runtime Diagnostics

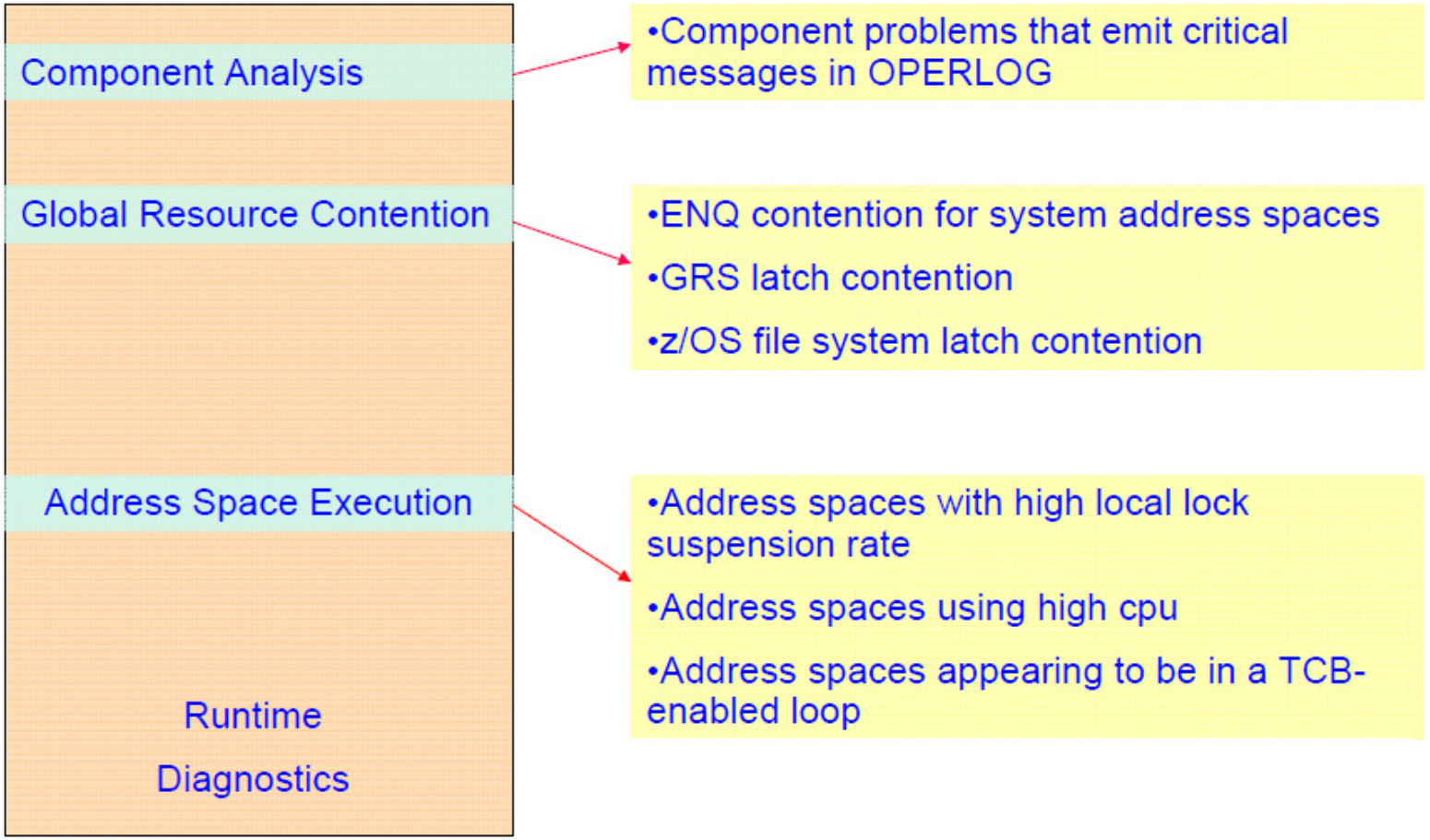
# Runtime Diagnostics

- Analyze system to provide diagnostic data in a timely manner
- Performs analysis similar to an experienced Systems Programmer
  - Faster – goal is 60 seconds or less
  - More comprehensive
  - Looks for specific evidence of “soft failures”
  - Provide suggested next steps
- It is not:
  - Not automations or a monitor
  - Takes no corrective action, but recommends next steps
  - Minimal dependencies on system services

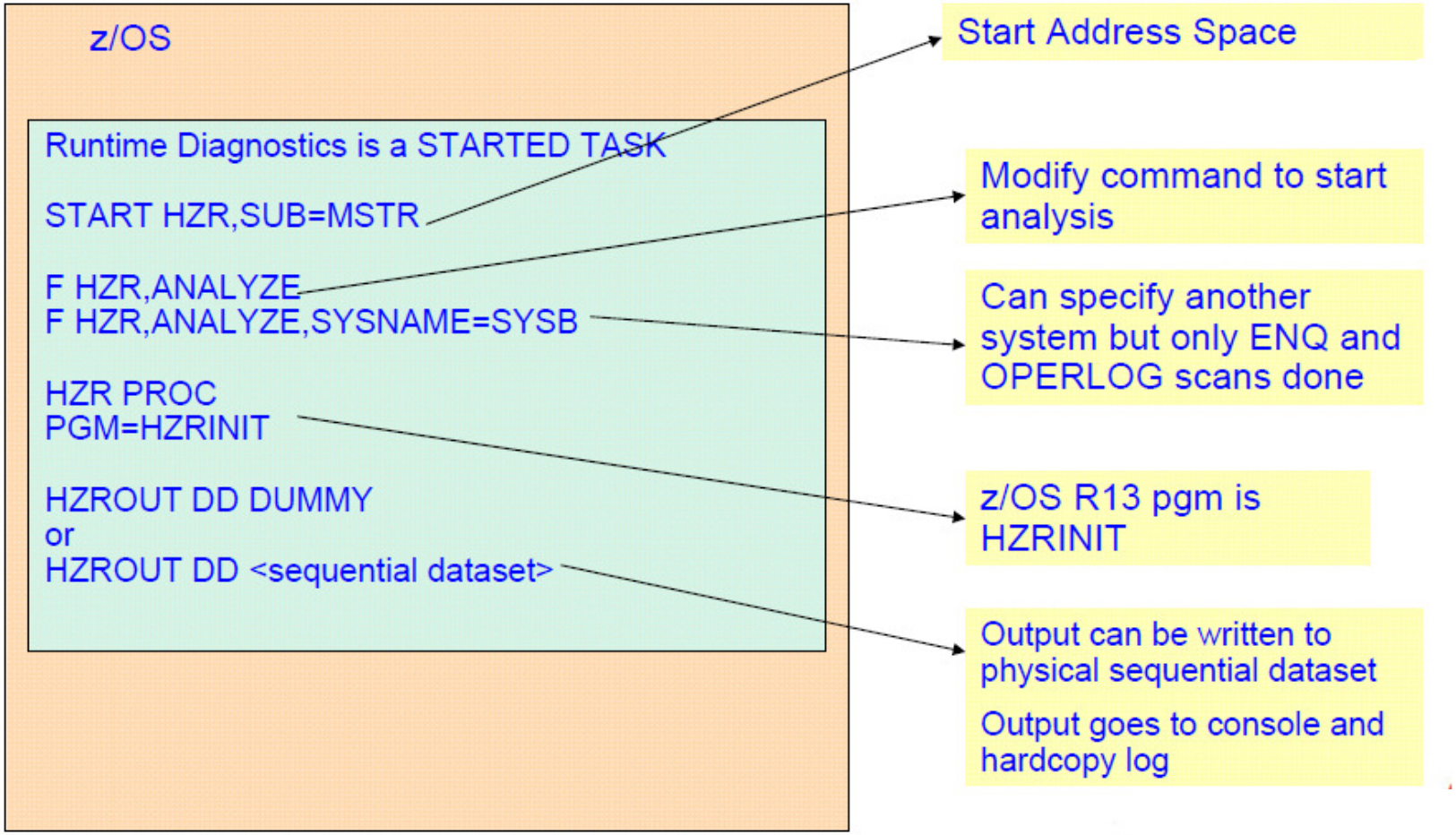
# Benefits

- Reduces the skill level needed by a system programmer for investigating soft failures
  - Provide timely , comprehensive analysis when needed
  - Productivity aid for experienced system programmers
- Quickly discover next actions to take
  - Which jobs to cancel
  - What to investigate further using other tools
- Use when
  - Problem reported on system
  - For problem resolution calls
  - When Predictive Failure Analysis reports a problem

# Categories for diagnostics



z/OS R13


z/OS
Runtime Diagnostics is a STARTED TASK
START HZR,SUB=MSTR
F HZR,ANALYZE
F HZR,ANALYZE,SYSNAME=SYSB
HZR PROC
PGM=HZRINIT
HZROUT DD DUMMY
or
HZROUT DD <sequential dataset>
Start Address Space
Modify command to start analysis
Can specify another system but only ENQ and OPERLOG scans done
z/OS R13 pgm is HZRINIT
Output can be written to physical sequential dataset
Output goes to console and hardcopy log

# Runtime Diagnostics output

- The output of Runtime Diagnostics is a WTO to the operator console (actually, a multiline WTO) that is issued to the console that originally issued the **F HZR,ANALYZE** command.
- If the MCS console that issued the **F HZR** command has an out-of-line display area setup (through a K A,xx) then the output will be displayed in the display area.
- The output of Runtime Diagnostics can also be directed to a sequential data set. DISP=SHR to view output without stopping HZR started task.

# Processing failure messages

- Runtime Diagnostics also reports when part of its processing fails (that is, it is unable to complete processing for one or more events) as QUALIFIED SUCCESS in the SUMMARY: portion of the report

```
f hzr,analyze
HZR0200I RUNTIME DIAGNOSTICS RESULT 751
SUMMARY: QUALIFIED SUCCESS - SOME PROCESSING FAILED
REQ: 001 TARGET SYSTEM: SY1 HOME: SY1 2010/12/21 - 11:25:55
INTERVAL: 60 MINUTES
EVENTS:
FOUND: 01 - PRIORITIES: HIGH:01 MED:00 LOW:00
TYPES: HIGHCPU:01
PROCESSING FAILURES:
OPERLOG....IXGCONN REQ=CONNECT ERROR.......RC=00000008 RS=0000080B
------------------------------------------------------------------------
EVENT 01: HIGH - HIGHCPU - SYSTEM: SY1 2010/12/21 - 11:25:56
ASID CPU RATE:99% ASID:002E JOBNAME:IBMUSERX
STEPNAME:STEP1 PROCSTEP: JOBID:JOB00045 USERID:IBMUSER
JOBSTART:2010/12/21 - 11:22:51
ERROR: ADDRESS SPACE USING EXCESSIVE CPU TIME. IT MIGHT BE LOOPING.
ACTION: USE YOUR SOFTWARE MONITORS TO INVESTIGATE THE ASID.
```

In this case, although Runtime Diagnostics was unable to connect to OPERLOG to examine messages, it continues to find other soft failures (HIGHCPU)

# High CPU Analysis

```
HZR0200I RUNTIME DIAGNOSTICS RESULT 568
SUMMARY: SUCCESS
REQ: 003 TARGET SYSTEM: SY1 HOME: SY1 2010/12/21 - 13:45:49
INTERVAL: 60 MINUTES
EVENTS:
FOUND: 01 - PRIORITIES: HIGH:01 MED:00 LOW:00
TYPES: HIGHCPU:01
------------------------------------------------------------------------
  -
EVENT 01: HIGH - HIGHCPU - SYSTEM: SY1 2010/12/21 - 13:45:50
ASID CPU RATE:99% ASID:002E JOBNAME:IBMUSERX
STEPNAME:STEP1 PROCSTEP: JOBID:JOB00045 USERID:IBMUSER
JOBSTART:2010/12/21 - 11:22:51
ERROR: ADDRESS SPACE USING EXCESSIVE CPU TIME. IT MIGHT BE LOOPING.
ACTION: USE YOUR SOFTWARE MONITORS TO INVESTIGATE THE ASID.
```

- Point-in-time check of any address space that is using more than 95% of the capacity of a single CPU. Takes 2 samples over a 1 second interval.

- The analysis is a one-second sample interval based on the capacity of a single CPU within the LPAR. Be aware that it is possible for the usage to be reported greater than 100% if the address space has multiple TCBs and several of the TCBs are individually using a high percentage of the capacity of a CPU.

# Loop Detection

```
HZR0200I RUNTIME DIAGNOSTICS RESULT 581
SUMMARY: SUCCESS
REQ: 004 TARGET SYSTEM: SY1 HOME: SY1 2010/12/21 - 13:51:32
INTERVAL: 60 MINUTES
EVENTS:
FOUND: 02 - PRIORITIES: HIGH:02 MED:00 LOW:00
TYPES: HIGHCPU:01
TYPES: LOOP:01
------------------------------------------------------------------------
  -
EVENT 01: HIGH - HIGHCPU - SYSTEM: SY1 2010/12/21 - 13:51:33
ASID CPU RATE:99% ASID:002E JOBNAME:IBMUSERX
STEPNAME:STEP1 PROCSTEP: JOBID:JOB00045 USERID:IBMUSER
JOBSTART:2010/12/21 - 11:22:51
ERROR: ADDRESS SPACE USING EXCESSIVE CPU TIME. IT MIGHT BE LOOPING.
ACTION: USE YOUR SOFTWARE MONITORS TO INVESTIGATE THE ASID.
------------------------------------------------------------------------
  -
EVENT 02: HIGH - LOOP - SYSTEM: SY1 2010/12/21 - 13:51:14
ASID:002E JOBNAME:IBMUSERX TCB:004FF1C0
STEPNAME:STEP1 PROCSTEP: JOBID:JOB00045 USERID:IBMUSER
JOBSTART:2010/12/21 - 11:22:51
ERROR: ADDRESS SPACE MIGHT BE IN A LOOP.
ACTION: USE YOUR SOFTWARE MONITORS TO INVESTIGATE THE ASID.
------------------------------------------------------------------------
  -
```

- Investigates all tasks in all address spaces looking for TCB loops
- Takes a snapshot of the system trace
- Looks for consistent, repetitive activity that typically indicates a loop

Runtime Diagnostics looks through all tasks in all address spaces to determine whether a task might be looping. Runtime Diagnostics does this by examining various system information for indicators of consistent repetitive activity that are typical when a task is in a loop.

When both a HIGHCPU event and a LOOP event list the job name, there is a high probability that a task in the job is in a loop. The normal corrective action is to cancel the job name listed.

# Local Lock suspension

```
HZR0200I RUNTIME DIAGNOSTICS RESULT 581
SUMMARY: SUCCESS
REQ: 004 TARGET SYSTEM: SY1 HOME: SY1 2010/12/21 - 13:51:32
INTERVAL: 60 MINUTES
EVENTS:
FOUND: 01 - PRIORITIES: HIGH:01 MED:00 LOW:00
TYPES: LOCK:01
------------------------------------------------------------
EVENT 01: HIGH - LOCK - SYSTEM: SY1 2010/12/21 - 13:51:33
HIGH LOCAL LOCK SUSPENSION RATE - ASID:000A JOBNAME:WLM
STEPNAME:WLM PROCSTEP:IEFPROC JOBID:++++++++ USERID:++++++++
JOBSTART:2010/12/21 - 11:15:08
ERROR: ADDRESS SPACE HAS HIGH LOCAL LOCK SUSPENSION RATE.
ACTION: USE YOUR SOFTWARE MONITORS TO INVESTIGATE THE ASID.
```

Suspend:

Local Lock

Cross Memory Lock

CMS lock

- Runtime Diagnostics provides a point-in-time check of local lock suspension for any address space.
- For the local lock suspension, Runtime Diagnostics calculates the amount of time an address space is suspended waiting for the local lock. If an address is suspended more than 50% of the time waiting for a local lock,

# ENQ contention checking

```
HZR0200I RUNTIME DIAGNOSTICS RESULT 568
---------------------------------------------------------------------
EVENT 01: HIGH - ENQ - SYSTEM: SY1 2010/12/21 - 11:43:33
ENQ WAITER - ASID:0038 - JOBNAME:IBMUSER2 - SYSTEM:SY1
ENQ BLOCKER - ASID:002F - JOBNAME:IBMUSER1 - SYSTEM:SY1
QNAME: TESTENQ
RNAME: TESTOFAVERYVERYVERYVERYLOOOOOOOOOOOOOOOOOOOOOONGRNAME1234567...
ERROR: ADDRESS SPACES MIGHT BE IN ENQ CONTENTION.
ACTION: USE YOUR SOFTWARE MONITORS TO INVESTIGATE BLOCKING JOBS AND
ACTION: ASIDS.
---------------------------------------------------------------------
```

- ENQ contention equivalent to issuing the D GRS,AN,WAITER command
- Lists both waiter and blocker
- It compares the list of job names that are waiters with the list of system address spaces that are started at IPL to determine if any system address spaces are waiters. Looks for system address spaces that is in ENQ ‘waiter’ for > 5 seconds

# z/OS UNIX latch contention – z/OS R13

```
EVENT 01: HIGH - OMVS - SYSTEM: SY1 2010/07/07 - 13:07:32
ASID:000E - JOBNAME:OMVS
MOUNT LATCH WAITERS: 2
FILE SYSTEM LATCH WAITERS: 0
XSYS AND OTHER THREADS WAITING FOR z/OS UNIX: 3
ERROR: z/OS UNIX MIGHT HAVE FILE SYSTEM LATCH CONTENTION.
ACTION: ISSUE D OMVS,W,A TO INVESTIGATE z/OS UNIX FILE SYSTEM
ACTION: LATCH CONTENTION, ACTIVITY AND WAITING THREADS.
```

- If z/OS UNIX file system latch contention or waiting threads exist for > 5 minutes in z/OS UNIX, a Runtime Diagnostics OMVS event is created.

- Normal action is to issue D OMVS,W,A to get the ASID and job names of the waiters

# GRS latch contention – z/OS R13

```
-------------------------------------------------------------------------
EVENT 01: HIGH - LATCH - SYSTEM: SY1 2010/12/21 - 14:32:01
LATCH SET NAME: SYSTEST.LATCH_TESTSET
LATCH NUMBER:3 CASID:0039 CJOBNAME:TSTLATCH
TOP WAITER - ASID:0039 - JOBNAME:TSTLATCH - TCB/WEB:004E2A70
TOP BLOCKER- ASID:0039 - JOBNAME:TSTLATCH - TCB/WEB:004FF028
ERROR: ADDRESS SPACES MIGHT BE IN LATCH CONTENTION.
ACTION: D GRS,AN,LATCH,DEP,CASID=0039,LAT=(SYSTEST.L*,3),DET
ACTION: TO ANALYZE THE LATCH DEPENDENCIES. USE YOUR SOFTWARE
ACTION: MONITORS TO INVESTIGATE BLOCKING JOBS AND ASIDS.
-------------------------------------------------------------------------
EVENT 02: HIGH - LATCH - SYSTEM: SY1 2010/12/21 - 14:32:01
LATCH SET NAME: SYSTEST.LATCH_TESTSET
LATCH NUMBER:3 CASID:003B CJOBNAME:TSTLATC2
TOP WAITER - ASID:003B - JOBNAME:TSTLATC2 - TCB/WEB:004E2A70
TOP BLOCKER- ASID:003B - JOBNAME:TSTLATC2 - TCB/WEB:004FF028
ERROR: ADDRESS SPACES MIGHT BE IN LATCH CONTENTION.
ACTION: D GRS,AN,LATCH,DEP,CASID=003B,LAT=(SYSTEST.L*,3),DET
ACTION: TO ANALYZE THE LATCH DEPENDENCIES. USE YOUR SOFTWARE
ACTION: MONITORS TO INVESTIGATE BLOCKING JOBS AND ASIDS.
```

- Obtains Latch contention information from GRS
- Omits z/OS UNIX file system latch contention
- Returns the longest waiter for each latch set

# Critical Messages

```
HZR0200I RUNTIME DIAGNOSTICS RESULT 361
SUMMARY: SUCCESS
 REQ: 120 TARGET SYSTEM: #@$A     HOME: #@$A     2012/09/25 - 15:26:52
 INTERVAL:  60 MINUTES
 EVENTS:
  FOUND: 01 - PRIORITIES: HIGH:01  MED:00  LOW:00
  TYPES: CF:01
-------------------------------------------------------------------------
EVENT 01: HIGH - CF           - SYSTEM: #@$A     2012/09/25 - 15:10:59
IXL158I PATH B8 IS NOW NOT-OPERATIONAL TO CUID: FFD4
                  COUPLING FACILITY 002817.IBM.02.0000000B3BD5
                                    PARTITION: 2E  CPCID: 00
  ERROR: INDICATED CHANNEL NOT OPERATIONAL.
 ACTION: RUN EREP TO DUMP DATA FROM SYS1.LOGREC AND PROVIDE IT TO IBM
 ACTION: SUPPORT.
-------------------------------------------------------------------------
```

- Checks previous 1 hour of OPERLOG
- For some messages additional analysis done
- Groups related messages into single event
- Weeds out shortage and relieved messages
- In some cases only shows last message if message repeated

Additional analysis done for:

IXC101I, IXC105I, IXC418I

IXL013I

IXC431I

IXC246E

IXC585E

# Runtime Diagnostics Summary

- z/OS 1.12
- Component-specific, critical messages in OPERLOG
  - Looks one hour back, if available
  - Additional analysis for some msgs
  - Message summary found in output
  - Can analyze messages in other systems in sysplex
- Enqueue Contention Checking
  - Looks for system address space waiting > 5 seconds
  - Lists both waiter and blocker
  - Can detect contention in other system in sysplex
- Local Lock Suspension
  - Any address space whose local lock suspension time is > 50%

- z/OS 1.12 (continued)
- CPU Analysis
  - Takes 2 samples over 1 sec. interval
  - Any task using > 95% is considered a potential problem
- Loop Detection
  - Investigates all tasks in all address spaces looking for TCB loops
- z/OS 1.13
- z/OS UNIX Latch Contention
  - Looks for z/OS UNIX latch contention or waiting threads that exit for > 5 minutes.
- GRS Latch Contention
  - Obtains latch contention info from GRS
  - Omits z/OS UNIX file system latch contention
  - Returns longest waiter for each latch set

# IBM Health Checker for z/OS

# Soft Failure Avoidance with IBM Health Checker

Health checker's role is to keep subtle configuration errors from resulting in Soft Failures

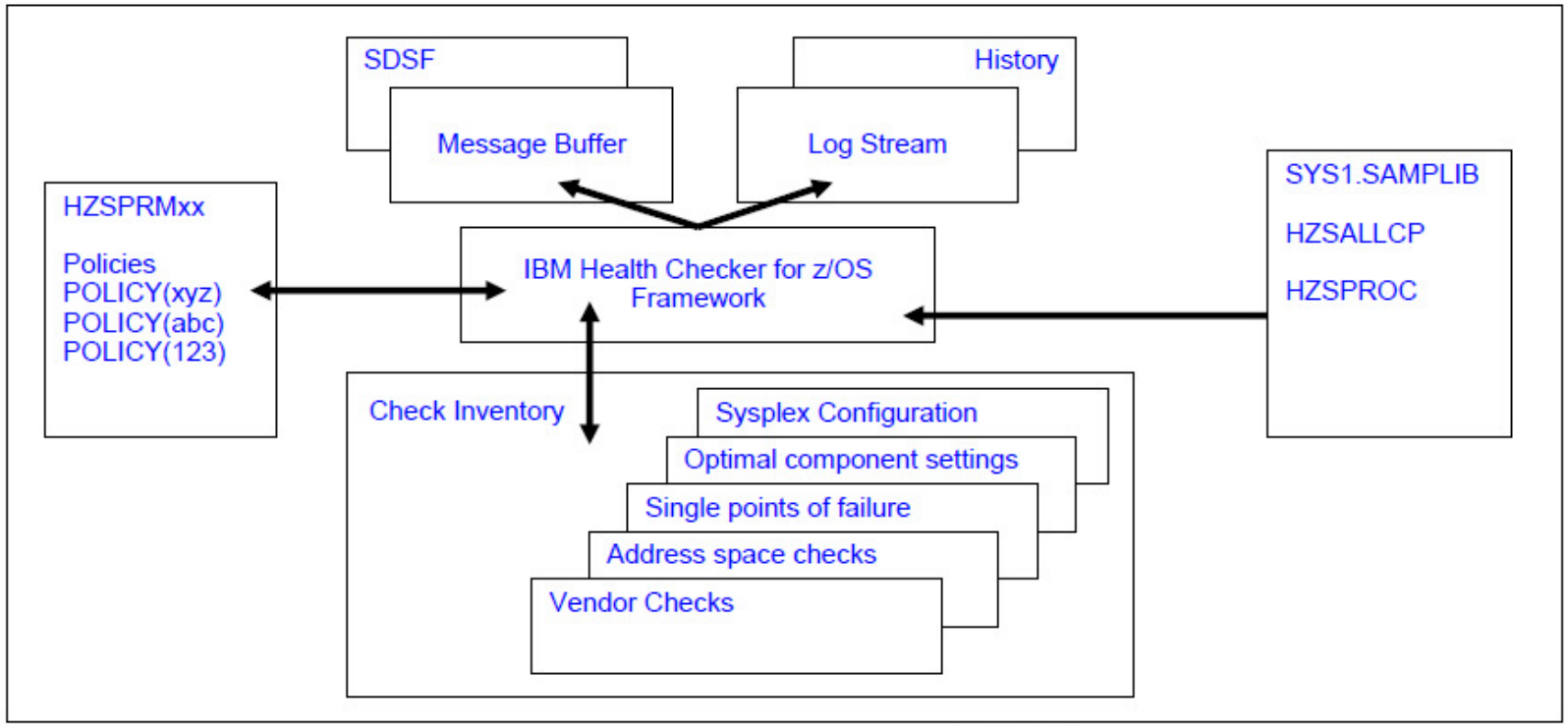
- Performance
- System effects
- Check configuration for best practices
- Single points of failure for log structures, data sets, CDS
- Storage utilization, running out of resources
- How many ASIDs do I have left? LXs? When will I run out?
- Whether DAE is inactive
- VSAM RLS latch contention, CF Cache size, CDS SPOF, etc.
- System Logger structure usage
- I/O timing, protection
- … many others

- Used by Preventive Failure Analysis to emit alerts
- Warnings of detected soft failures
- 187 z/OS Health Checks in z/OS R13
- Just a reminder to enable IBM Health Checker for z/OS

# z/OS Health Checker – History & Objectives

- History
  - Multi-system outage analysis
  - 15-20% system outages attributed to Setup/Configuration
  - A tool developed by ITSO to address component configuration and setup errors commonly made by installations
  - Originally available as web download tool,
  - Is available with its own FMID since z/OS 1.7
  - Migration checks included since z/OS 1.9
- Objectives
  - Identify potential problems before they impact availability or cause outages
  - Checks the current active z/OS and sysplex settings
  - Checks definitions for a system and compares the values to those suggested by IBM or defined by you
  - Run continuously to find deviations from best practices
  - Not a diagnostic or monitoring tool
- Produces output in the form of detailed messages (SDSF support)
  - Indicates both potential problems and suggested actions
  - Does not mean that Health Checker has found problems that you need to report to IBM!
  - Messages simply informs you of potential problems so that you can take action on your installation, before they become critical, thereby avoiding an IPL

# IBM Health Checker for z/OS




The *framework* is an interface that manages services like check registration, messaging, scheduling, command processing, logging, and reporting.

*Checks* are programs or routines that evaluate component, element, or product-specific settings and definitions, looking for potential problems on a running system. Checks are independent of the framework.

- Not new:
- Base function with z/OS 1.7 onwards
- Web download + PTFs for z/OS 1.4 – 1.6

- But:
- Can help avoid soft failures
- Over 180 checks

# z/OS Health Checker - Overview

- Each check has 3 parts
  - The dynamic exit routine that identifies the check to the Health checker
  - The check itself
  - A message table to define messages that are issued by the check
- Each check includes a set of pre-defined values, such as:
  - Interval, or how often the check will run
  - Severity of the check, which influences how check output is issued - (high, medium, and low)
  - Routing and descriptor codes for the check
- You can override some check values:
  - Statements in the HZSPRMxx parmlib member
  - The MODIFY command - F HZSPROC,.....
  - SDSF interactive command panel

# Important Considerations

1. Don't just change the configuration … investigate the exception and then take appropriate action
2. There are 187 Health Checks in z/OS R13
    1. Don't think that you must activate all health checks at once to get benefit
    2. Goal should be to remove all exceptions
        1. by fixing the condition
        2. by tuning the check so that it looks for what you need it to look for
        3. (as a last resort) by deactivating the check
    3. Once you can run cleanly, you will be in the ideal position to know that an exception indicates something has changed
    4. Consider categorizing health checks by
        1. Checks I expect no exceptions from
        2. Checks not turned on because exceptions not cleaned up yet
        3. Plan to move checks to group 1 as you clean up exceptions
3. GDPS recommendations for changing z/OS checks trump z/OS in a GDPS environment
    1. Some z/OS Health Check recommendations conflict with GDPS function, so follow GDPS guidelines

## Resources

- IBM Health Checker's User Guide, SA22-7994
- Exploiting the IBM Health Checker for z/OS Infrastructure
  - IBM Redpaper REDP-4590-01

# Predictive Failure Analysis

# Predictive Failure Analysis (PFA)

- The goal of predictive analysis and early-detection analysis is to notify the system programmer when the system can see the problem is occurring internally rather than later when it is visible externally.
- PFA detects problems before they are visible externally by using resources and metrics at different layers of the software stack that can indicate that resource exhaustion, damaged address spaces, or damaged systems could be occurring.
- PFA is not intended to find immediate problems (on a machine-time scale) that will bring the system down. Rather, it can detect potential problems on a human-time scale.

# How PFA Detects Soft Failures

- Causes of “sick, but not dead”
  - ***Damaged systems***
    - Recurring or recursive errors caused by software defects anywhere in the software stack
  - Serialization
    - Priority inversion
    - Classic deadlocks
    - Owner gone
  - ***Resource exhaustion***
    - Physical resources
    - Software resources
  - Indeterminate or unexpected states

- Predictive failure analysis uses
  - *Historical data*
  - *Machine learning and mathematical modeling*

  to detect abnormal behavior and the potential causes of this abnormal behavior
- Objective
  - Convert “sick, but not dead” to a correctable incident

# PFA focus

- PFA focuses on the **damaged address space or system** and **resource exhaustion** categories that may lead to soft failures.

  **Damaged address space or system:** The indication of a damaged system is typically when there are recurring or recursive errors anywhere in the software stack.

  **Physical or software resource exhaustion** of a shared system resource

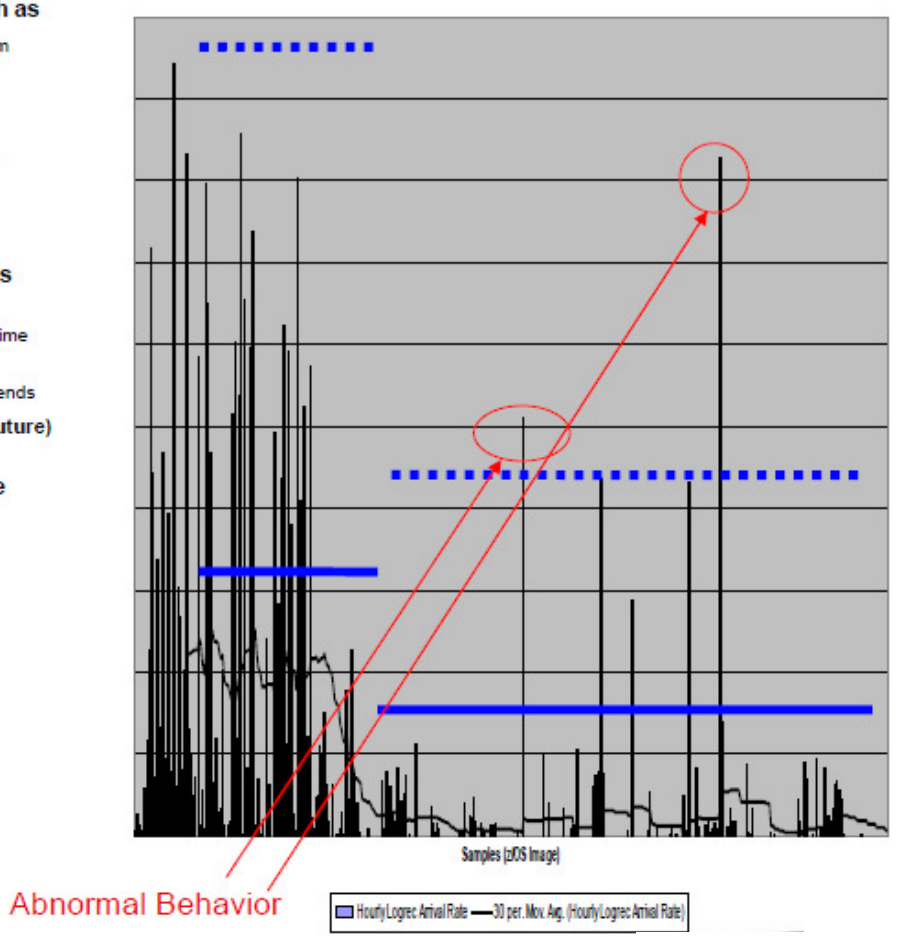
- PFA collects the data from the individual system and models it to determine what is normal for that system as well as to determine what kinds of resource trends are occurring.

# Abnormal behavior detection

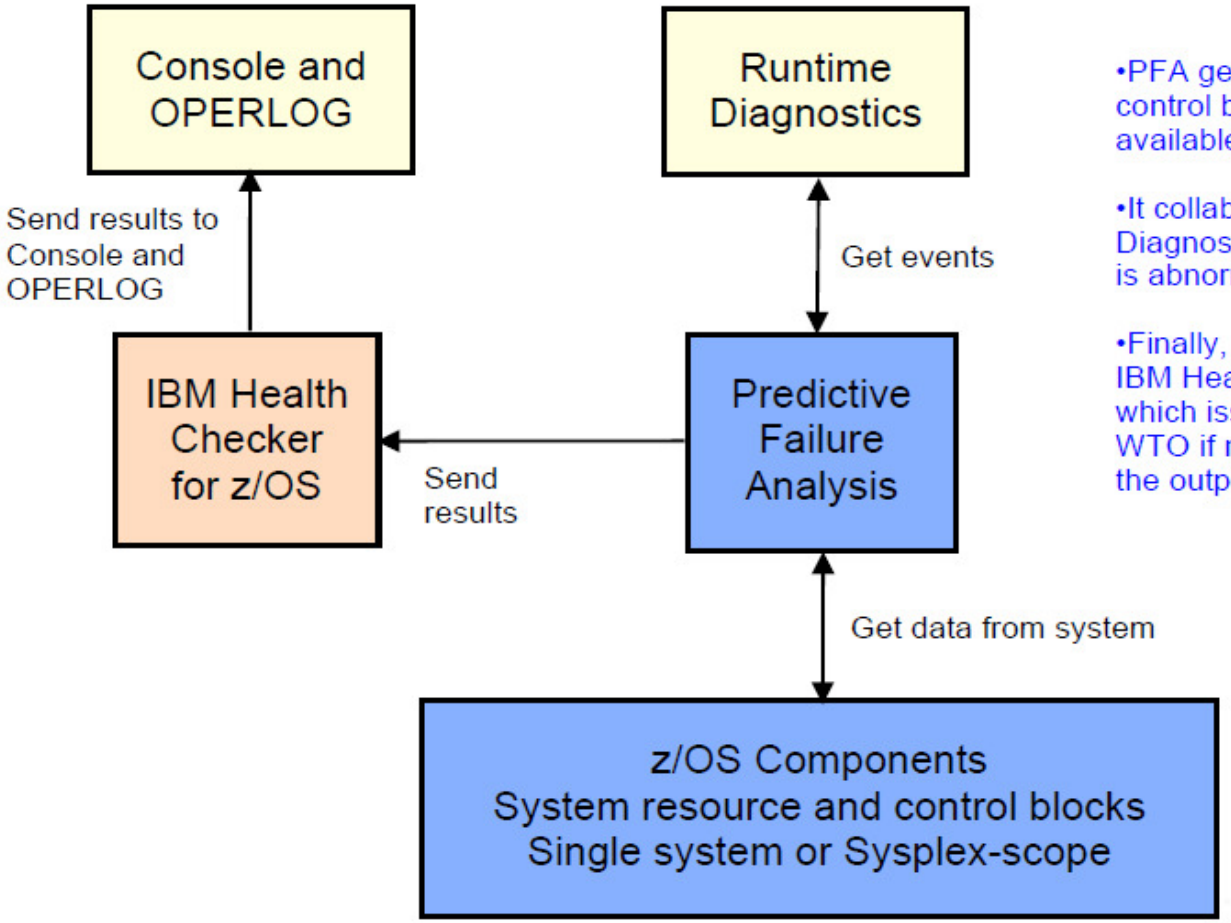
- There are three types of abnormal behavior detection that PFA's algorithms incorporate:
  - **Future prediction**: This processing does trend analysis and models the behavior into the future to predict if the current trend will exhaust a common resource.
  - **Expected value**: This processing does trend analysis and models the behavior to determine what value should be expected at the current time to determine if an abnormal condition is occurring.
  - **Expected rate**: This processing does trend analysis and models the behavior to determine if the current rate when compared to the rates for multiple time periods indicates that an abnormal condition is occurring. The rate is often created by normalizing a count of the metric being analyzed by the CPU being used. By normalizing the rate and by comparing against multiple time period predictions, normal workload changes do not appear as abnormal conditions.
- For both the expected value and the expected rate types of predictions, PFA clusters the historical data so that trends within the data can be identified. It then determines which trend is currently active and uses the prediction for that trend in its comparisons.

# How PFA Determines Expected Values

- Behavior of z/OS system is a function of many factors such as
  - Workload, type of work, hardware and software configuration, system automation, etc.…
- Use historical data to calculate future or expected value to eliminate factors
- Same type of work runs at approximately same time or runs consistently
  - Expected rate = fn(workload, time) -- Can compare different time ranges such as 1 hour ago, 24 hours ago, 7 days ago
  - Expected value = fn(workload) – based on past and current trends
  - Future prediction = fn(workload,time projected into the future)
- Cluster metric by time to calculate expected or future value



# PFA input and output




- PFA gets it data from z/OS control blocks and externally available interfaces.
- It collaborates with Runtime Diagnostics to detect if a metric is abnormally low.
- Finally, it sends the results to IBM Health Checker for z/OS which issues the message as a WTO if necessary and makes the output viewable in SDSF.

PFA Serviceability

# Modify command to display status

**STATUS examples:**

f pfa,display

f,pfa,display,status

```
AIR017I 10.31.32 PFA STATUS
NUMBER OF CHECKS REGISTERED     : 5
NUMBER OF CHECKS ACTIVE         : 5
COUNT OF COLLECT QUEUE ELEMENTS: 0
COUNT OF MODEL QUEUE ELEMENTS   : 0
COUNT OF JVM TERMINATIONS       : 0
```

**SUMMARY examples:**

f pfa,display,checks

f pfa,display,check(pfa*),summary

```
AIR013I 10.09.14 PFA CHECK SUMMARY
                                    LAST SUCCESSFUL      LAST SUCCESSFUL
CHECK NAME                  ACTIVE      COLLECT TIME         MODEL TIME
PFA_COMMON_STORAGE_USAGE     YES     04/05/2008 10.01     04/05/2008 08.16
PFA_LOGREC_ARRIVAL_RATE      YES     04/05/2008 09.15     04/05/2008 06.32
(all checks are displayed)
```

**DETAIL examples:**

f pfa,display,check(pfa_logrec_arrival_rate),detail

f pfa,display,check(pfa_l*),detail

```
AIR018I 02.22.54 PFA CHECK DETAIL
CHECK NAME:   PFA_LOGREC_ARRIVAL_RATE
    ACTIVE                          : YES
    TOTAL COLLECTION COUNT          : 5
    SUCCESSFUL COLLECTION COUNT     : 5
    LAST COLLECTION TIME            : 04/05/2008 10.18.22
    LAST SUCCESSFUL COLLECTION TIME: 04/05/2008 10.18.22
    NEXT COLLECTION TIME            : 04/05/2008 10.33.22
    TOTAL MODEL COUNT               : 1
    SUCCESSFUL MODEL COUNT          : 1
    LAST MODEL TIME                 : 04/05/2008 10.18.24
    LAST SUCCESSFUL MODEL TIME      : 04/05/2008 10.18.24
    NEXT MODEL TIME                 : 04/05/2008 16.18.24
    CHECK SPECIFIC PARAMETERS:
       COLLECTINT                   : 15
       MODELINT                     : 360
       COLLECTINACTIVE              : 1=ON
       DEBUG                        : 0=OFF
       STDDEV                       : 10
       EXCEPTIONMIN                 : 25
  EXCLUDED_JOBS:
       (excluded jobs list here)
```

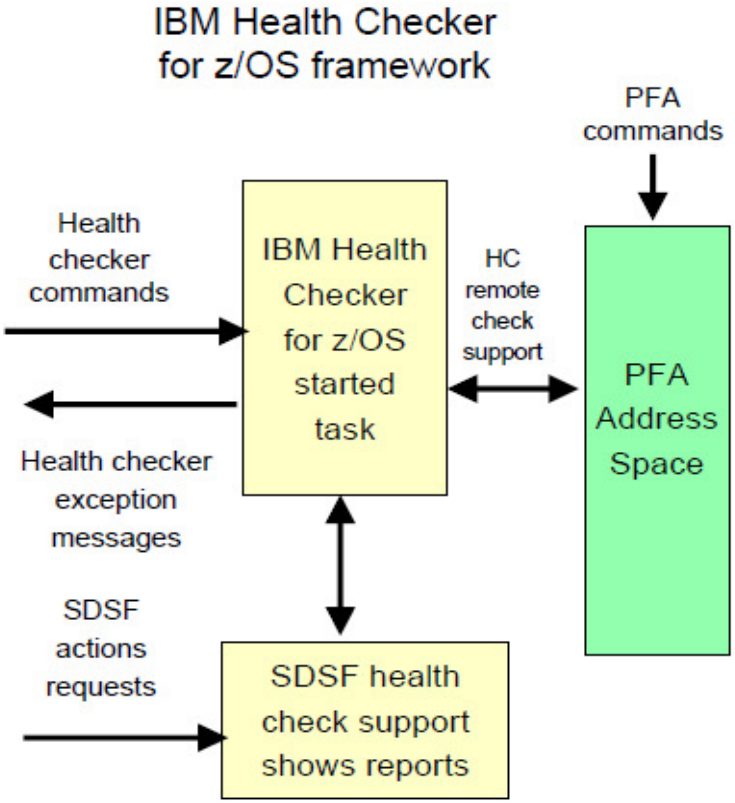
# The PFA checks

| Resource or Metric | Causes detected | Type of Analysis | Release |
|---|---|---|---|
| Common Storage usage | Detects resource exhaustion in common storage by looking for spikes, leaks and creeps | Future prediction | 1.10 SPE |
| LOGREC arrival rate | Detects a damaged address space or system | Expected rate | 1.10 SPE |
| Message arrival rate | Detects a damaged or hung address space based on abnormal rate in the WTOs and WTORs issued, normalized by the amount of CPU being used | Expected rate | 1.11 |
| SMF arrival rate | Detects a damaged or hung address space based on an abnormal rate of SMF arrivals, normalized by the amount of CPU being used | Expected rate | 1.12 |
| JES spool usage | Detects a damaged persistent address space based on an abnormal increase in the number of track groups used | Expected value | 1.13 |
| Enqueue request rate | Detects a damaged or hung address space or system based on an abnormal rate of enqueue requests, normalized by the amount of CPU being used | Expected rate | 1.13 |

# PFA_FRAMES_AND_SLOTS_USAGE

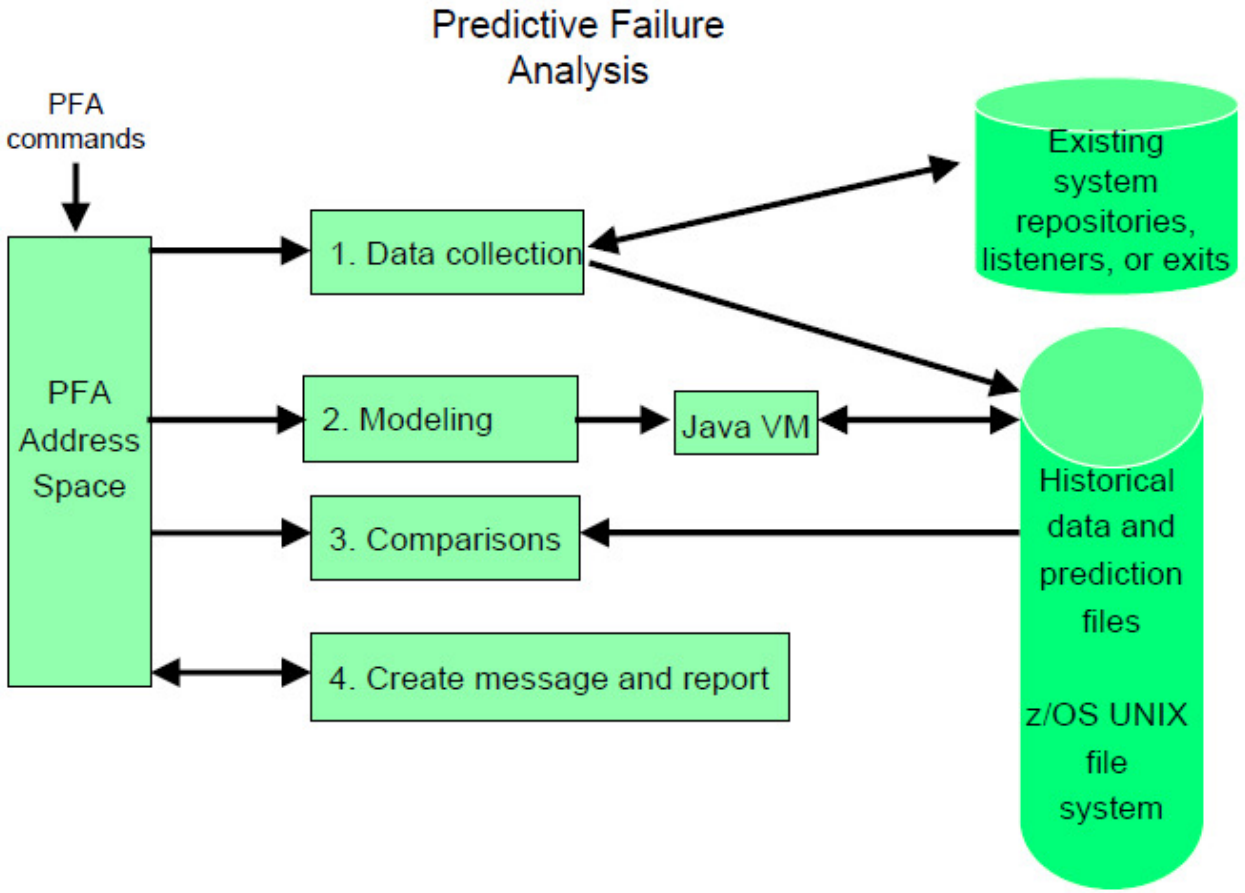
- If you are familiar with PFA from other sources, you may have noticed that the check for frames and slots usage (PFA_FRAMES_AND_SLOTS_USAGE) was omitted from the previous table.
- This check has been permanently removed from PFA with APAR OA40065 due to the fact that it caused unwarranted exceptions that could not be avoided with available mechanisms.
- It is recommended that you apply the PTF for APAR OA40065 as soon as possible. To remove the check from PFA prior to applying the PTF, use the following IBM Health Checker for z/OS command:
- **f hzsproc,delete,check(ibmpfa,pfa_f*)**

# PFA and IBM Health Checker for z/OS

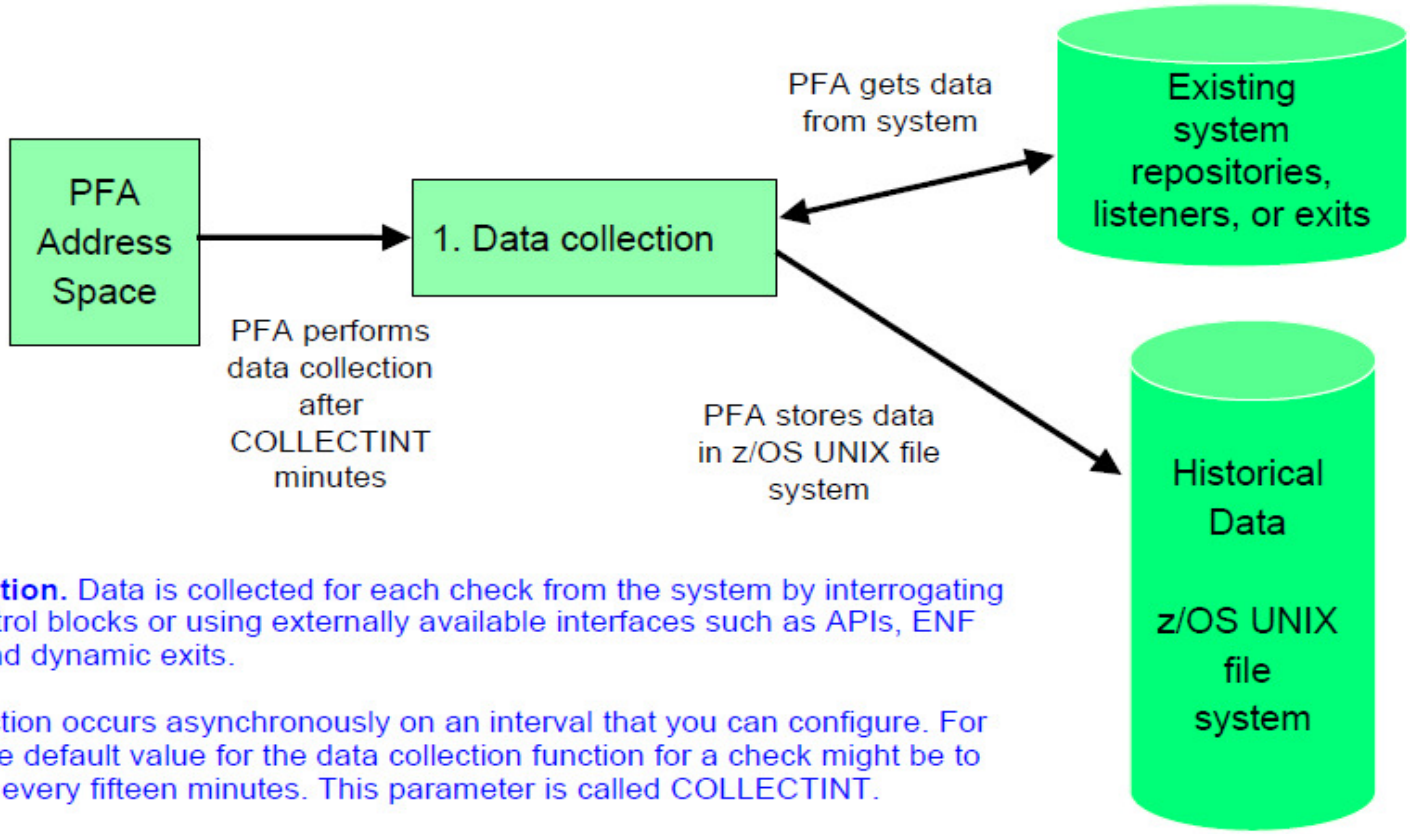



- PFA is built using the remote check feature of the IBM Health Checker for z/OS framework
- The IBM Health Checker for z/OS commands are available for all PFA checks
- The results of the PFA comparisons are available through IBM Health Checker for z/OS interfaces such as the health check support in SDSF
- The results of PFA's comparisons are sent back to IBM Health Checker for z/OS which writes the report. If an exception occurs, a WTO is issued by default
- If IBM Health Checker for z/OS is not active prior to PFA starting, PFA collects and models data, but waits for IBM Health Checker for z/OS to start before performing comparisons or issuing results because those functions are dependent upon IBM Health Checker for z/OS.

# PFA processing



# PFA data collection

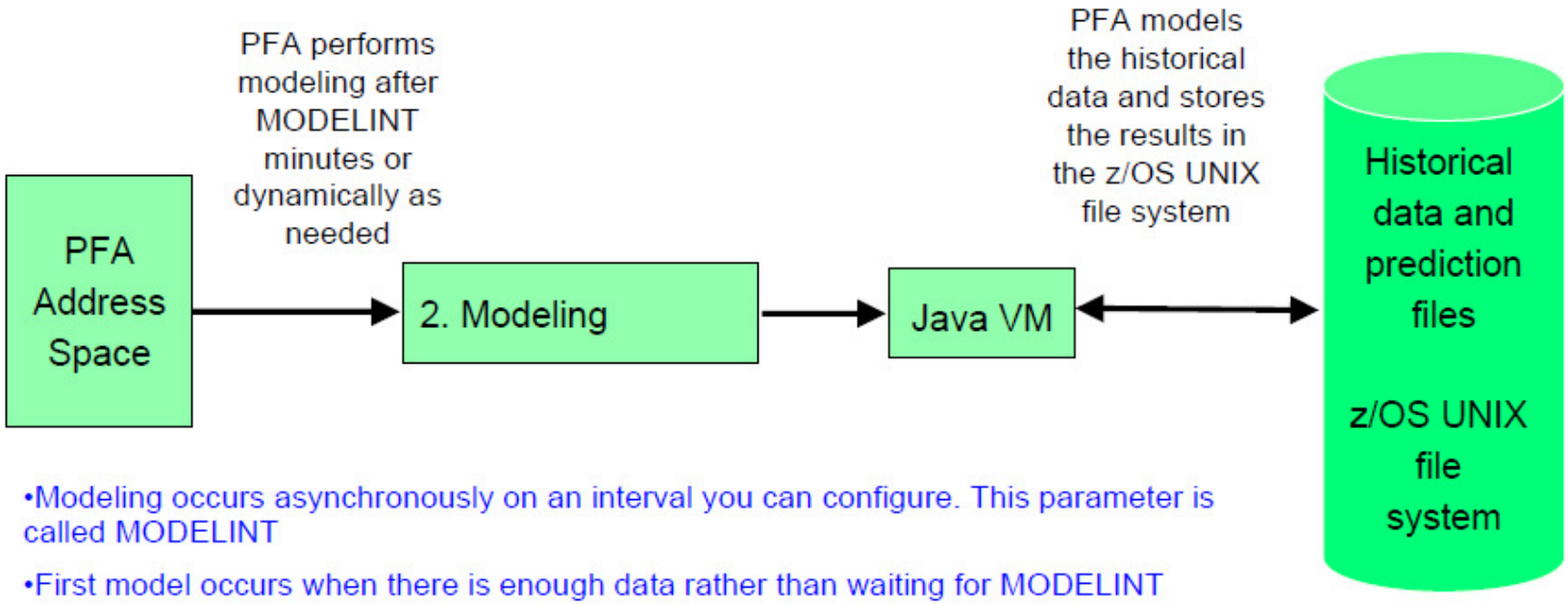



**Data collection.** Data is collected for each check from the system by interrogating system control blocks or using externally available interfaces such as APIs, ENF listeners, and dynamic exits.

- Data collection occurs asynchronously on an interval that you can configure. For example, the default value for the data collection function for a check might be to collect data every fifteen minutes. This parameter is called COLLECTINT.

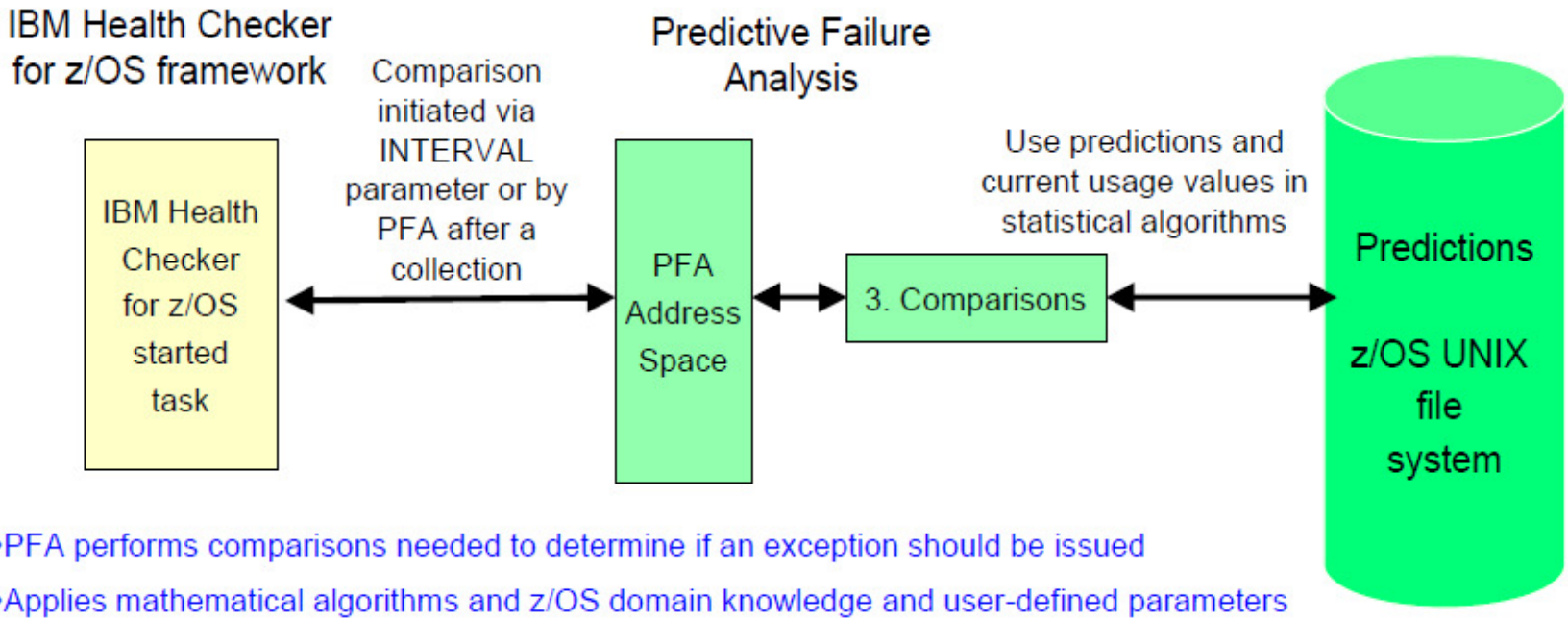
- The data is stored in files in the z/OS UNIX file system and the files are self-maintained by PFA. Data that is deemed too old by PFA for use in making new predictions is deleted.

# PFA Modeling



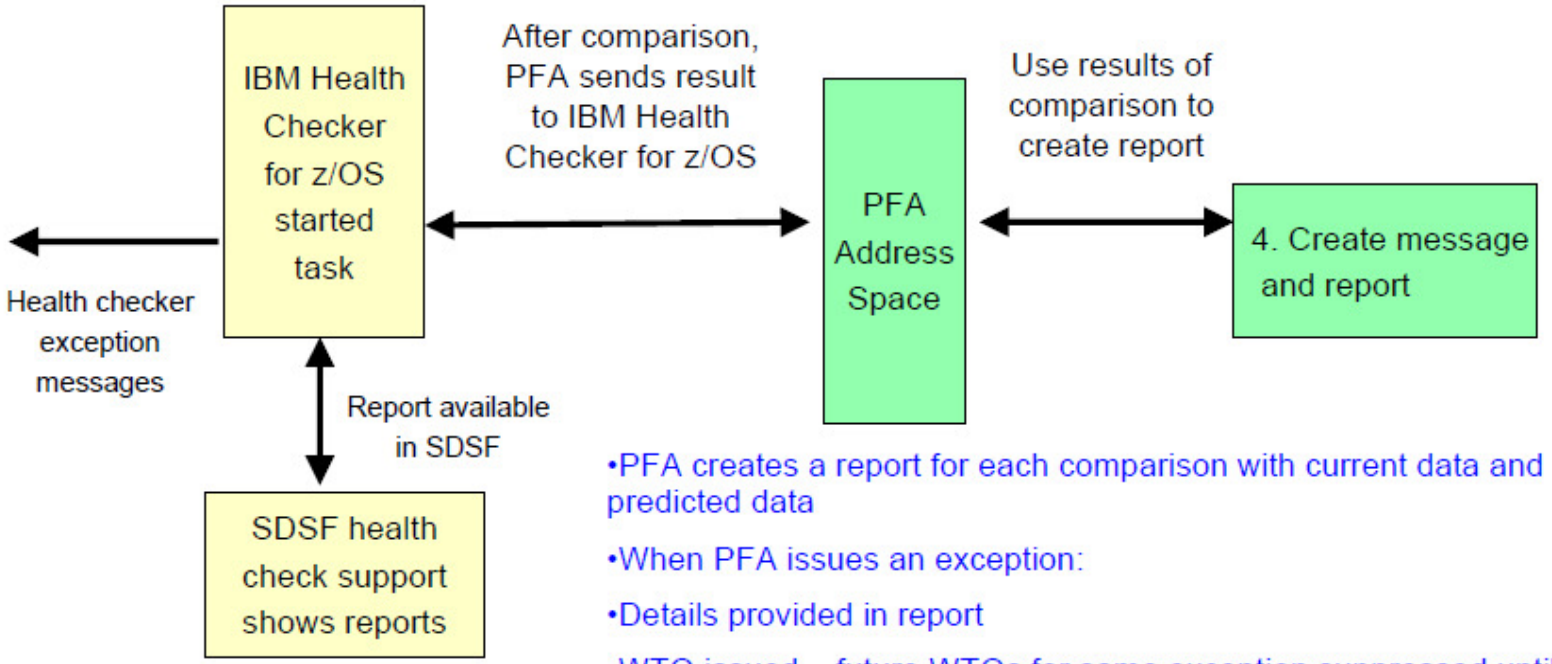

- Modeling occurs asynchronously on an interval you can configure. This parameter is called MODELINT
- First model occurs when there is enough data rather than waiting for MODELINT
- Some checks require 7-13 hours of data collection prior to a model being created
- Four successful collections required before modeling is attempted
- Models are then created based in MODELINT or if PFA determines current model is too old (trends changing)
- Results of modeling are stored in z/OS UNIX file systems – self-maintained by PFA

# PFA comparison processing




- PFA performs comparisons needed to determine if an exception should be issued
- Applies mathematical algorithms and z/OS domain knowledge and user-defined parameters
- Determines if values are normal, abnormal or becoming abnormal
- If abnormal or becoming abnormal may cause a remodel to occur before next MODELINT
- Comparisons are initiated either by INTERVAL parameter of Health Check or by PFA

# PFA reporting




- PFA creates a report for each comparison with current data and predicted data
- When PFA issues an exception:
- Details provided in report
- WTO issued – future WTOs for same exception suppressed until new data available
- Sub-directory created under PFA check's UNIX filesystem directory containing diagnostic data for IBM service
- Modeling may occur more frequently until data stabilizes

# PFA report format

```
                    Message Arrival Rate Prediction Report

Last successful model time       :  04/05/2012 07:08:01
Next model time                  :  04/05/2012 19:08:04
Model interval                   :  720
Last successful collection time  :  04/05/2012 08:22:15
Next collection time             :  04/05/2012 08:37:16
Collection interval              :  15

Message arrival rate
   at last collection interval      :       83.52
Prediction based on 1 hour of data  :       98.27
Prediction based on 24 hours of data:       85.98
Prediction based on 7 days of data  :      100.22

Top persistent users:
                                                 Predicted Message
                                                   Arrival Rate
  Job                      Message
  Name      ASID           Arrival Rate      1 Hour        24 Hour        7 Day
  ________  ____  _____________  _____________  _____________  _____________
  TRACKED1  001D          58.00          23.88          22.82          15.82
  TRACKED2  0028          11.00           0.34          11.11          12.11
  TRACKED3  0029          11.00          12.43           2.36           8.36
...
```

Heading information: data on modeling and collection

System-level information: includes current values and predictions

Address space Information: Details on tracked address space or address space causing the problem

# How PFA Chooses Address Spaces to Track

- **Some metrics require data for the *entire system* to be tracked**
  - Exhaustion of common storage for entire system
  - LOGREC arrivals for entire system grouped by key
- **Some metrics call for tracking only *persistent address spaces***
  - Those that start within the first hour after IPL.
  - For example, tracks "track groups usage" by persistent address spaces to detect damaged address spaces
- **Some metrics are most accurate when using several categories**
  - *"Chatty" persistent* address spaces tracked individually
    - Start within the first hour after IPL and have the highest rates after a warm-up period
    - Data from first hour after IPL is ignored.
    - After an IPL or PFA restart, if all are running, same address spaces are tracked.
    - Duplicates with the same name are not tracked
    - Restarted address spaces that are tracked are still tracked after restart.
  - *Other persistent* address spaces as a group
  - *Non-persistent* address spaces as a group
  - *Total system rate* ("chatty" + other persistent + non-persistent)

# PFA_COMMON_STORAGE_USAGE (z/OS 1.10 SPE)

- Predicts *exhaustion of common storage* by the z/OS image
- z/OS 1.10 and 1.11 -- Models **two** locations → CSA+SQA and ECSA+ESQA
- z/OS 1.12 -- Models **six** locations, handles expansion, and performance improved → CSA, SQA, ECSA, ESQA, CSA+SQA, and ECSA+ESQA
- *Not a monitor* of individual address spaces
- Does not detect
  - Fragmentation or really rapid growth
  - Usage exceeds a specific threshold (done by VSM_COMMON_STORAGE_USAGE)
  - An address space abnormally consuming common storage without impacting the z/OS image

# Example Report: z/OS 1.12 Common Storage Usage Report

- Top predicted users
  - Tries to pinpoint potential villains
  - Those whose usage has *increased* the most in the last hour
- Other information
  - Expansion information
  - IBM Health Checker for z/OS message in its entirety

Common Storage Usage Prediction Report
(heading information intentionally omitted)

| Storage Location | Current Usage in Kilobytes | Prediction in Kilobytes | Capacity When Predicted in Kilobytes | Percentage of Current to Capacity |
|---|---|---|---|---|
| *CSA | 2796 | 3152 | 2956 | 95% |
| SQA | 455 | 455 | 2460 | 18% |
| CSA+SQA | 3251 | 3771 | 5116 | 64% |
| ECSA | 114922 | 637703 | 512700 | 22% |
| ESQA | 8414 | 9319 | 13184 | 64% |
| ECSA+ESQA | 123336 | 646007 | 525884 | 23% |

Storage requested from SQA expanded into CSA and is being included in CSA usage and predictions. Comparisons for SQA are not being performed.

Address spaces with the highest increased usage:

| Job Name | Storage Location | Current Usage in Kilobytes | Predicted Usage in Kilobytes |
|---|---|---|---|
| JOB3 | *CSA | 1235 | 1523 |
| JOB1 | *CSA | 752 | 935 |
| JOB5 | *CSA | 354 | 420 |
| JOB8 | *CSA | 152 | 267 |
| JOB2 | *CSA | 75 | 80 |
| JOB6 | *CSA | 66 | 78 |
| JOB15 | *CSA | 53 | 55 |
| JOB18 | *CSA | 42 | 63 |
| JOB7 | *CSA | 36 | 35 |
| JOB9 | *CSA | 31 | 34 |

* - Storage locations that caused the exception.

# PFA_LOGREC_ARRIVAL_RATE (z/OS 1.10 SPE)

- Detects *a damaged system* by predicting and comparing LOGREC arrival rates in a collection interval
- Models expected number of LOGRECs in time ranges by key
  - Exceptions produced for any key grouping for any time range
- Not looking for individual LOGRECs, bursts of failures or patterns of failures
- Must be a software LOGREC with a usable SDWA

```
LOGREC Arrival Rate Prediction Report
(heading information intentionally omitted)
                                Key 0      Key 1-7     Key 8-15
                              _________  _________   _________
Arrivals in last
   collection interval:             1          0           2
Predicted rates based on...
     1 hour of data:                1          0           1
   24 hours of data:                0          0           1
     7 days of data:                0          0           1
    30 days of data:                0          0           1
Jobs having LOGREC arrivals in last collection interval:
   Job Name        ASID        Arrivals

   ________        ____        ________
   LOGREC08        0029               2
   LOGREC00        0027               1
```

# z/OS 1.11 PFA Checks

- **Frames and slots usage check** → recently deleted with OA40065
- **Message arrival rate check** → detects *damaged address spaces or a damaged LPAR* by tracking WTO and WTORs normalized by CPU across time ranges
  - Counted prior to possible exclusion by Message Flooding Automation
  - Tracks the four categories: "chatty" persistent; non-chatty, persistent; non-persistent; and total system
  - Does not detect abnormal patterns or single critical messages
  - Performs comparisons after every collection rather than on an INTERVAL schedule in IBM Health Checker for z/OS
  - An appropriate report is printed for each type of exception

# PFA_SMF_ARRIVAL_RATE (z/OS 1.12)

- Detects a *damaged system* based on an SMF arrival rate (normalized by CPU) across time ranges that is too high
- Same four categories as the Message Arrival Rate check
- *Not looking* for abnormal SMF record arrival patterns or single SMF record arrivals
  - *If SMF is not running or stops,*
    - previously collected data is *automatically discarded* so that predictions aren't skewed.
  - *If you change the SMF configuration,*
    - delete the files in the PFA_SMF_ARRIVAL_RATE/data directory or your data will be skewed.
- Report very similar to Message Arrival Rate

# PFA_ENQUEUE_REQUEST_RATE (z/OS 1.12)

- Detects a *damaged address space* or *damaged system* by comparing the number of enqueue requests per CPU millisecond (OA39924) to the rate expected.
- Two categories compared across three time ranges
  - *"Chatty" persistent* address spaces tracked individually and total system rate
  - 1 hour, 24 hour, and 7 day comparisons

```
            Enqueue Request Rate Prediction Report

 (Heading information intentionally omitted.)

 Enqueue request rate
    at last collection interval      :       83.52
 Prediction based on 1 hour of data  :       98.27
 Prediction based on 24 hours of data:       85.98
 Prediction based on 7 days of data  :      100.22

 Top persistent users:

                                          Predicted Enqueue
                                             Request Rate
   Job                    Enqueue
   Name      ASID     Request Rate        1 Hour       24 Hour         7 Day
   _______ ____  _____________  _____________  _____________  _____________
   TRACKED1 001D         58.00          23.88          22.82          35.82
   TRACKED2 0028         11.00          10.34          11.11          12.11
   TRACKED3 0029         11.00          12.43          12.36           8.36
```

# PFA_JES_SPOOL_USAGE (z/OS 1.13)

- Detects a *damaged address space or system* based on persistent jobs usage of the number of track groups
- Models 15 persistent jobs with the highest *increase* in their track group usage from one collection to the next
  - The number of actual track groups used is irrelevant due to the fact we are looking for a damaged address space or system *rather than exhaustion of track groups*.
  - Dynamic modeling occurs when "top jobs" change significantly to model new top jobs
  - JES2 only

# PFA_JES_SPOOL_USAGE (z/OS 1.13)

- The exception is issued based on an *unexpected increase in the number of track groups used* from one collection to the next
  - Often confused – it is looking for a *damaged address space* based on an abnormal *increase* in track groups usage NOT exhaustion!
- The current number of track groups used is provided as additional information and is *totally irrelevant* to the comparisons.

```
                    JES Spool Usage Prediction Report

(Heading information intentionally omitted.)

Address spaces causing exception:

                  Current Change in   Expected Change in          Current
Job                 Number of Track     Number of Track   Number of Track
Name      ASID          Groups Used         Groups Used       Groups Used
________  ____  ____________________  __________________  ________________
JOB1      0019                   252                  10               892
JOB55     000E                   129                   3               400
```

# PFA Integration with Runtime Diagnostics (z/OS 1.13)

- Detects a *damaged or hung address space or system* based on rates being too low
- When PFA detects an abnormally low condition, Runtime Diagnostics is executed
  - If the results of Runtime Diagnostics indicate a problem,
    - the PFA exception is issued
    - the PFA prediction report includes the Runtime Diagnostics output
- **Supported by three checks** → Message Arrival Rate, SMF Arrival Rate, and Enqueue Request Rate
- **Supported by three categories** (if supported by the check) → "Chatty" persistent jobs, other persistent jobs as a group, and total system
- **The Runtime Diagnostics address space (HZR) must be active**

# Exception Report for PFA Integration with Runtime Diagnostics

- **"Too low" exception** message sent as WTO by default
- **Runtime Diagnostics output** included in PFA report
- Prediction report and result message **available in SDSF** (sdsf.ck)
- **PFA current rates and predictions** relevant to category causing exception

```
                Message Arrival Rate Prediction Report
(Heading information intentionally omitted.)

Persistent address spaces with low rates:
                                                Predicted Message
                                                  Arrival Rate
   Job                    Message
   Name      ASID         Arrival Rate      1 Hour       24 Hour        7 Day
   --------  ----  ---------------------  ----------  ------------  ----------
   JOBS4     001F                 1.17       23.88         22.82       15.82
   JOBS5     002D                 2.01        8.34         11.11       12.11

Runtime Diagnostics Output:

   Runtime Diagnostics detected a problem in job: JOBS4
     EVENT 06: HIGH - HIGHCPU - SYSTEM: SY1 2009/06/12 - 13:28:46
     ASID CPU RATE: 96% ASID: 001F JOBNAME: JOBS4
     STEPNAME: PFATEST PROCSTEP: PFATEST JOBID: STC00042 USERID:
   ++++++++
     JOBSTART: 2009/06/12 - 13:28:35
   Error:
     ADDRESS SPACE USING EXCESSIVE CPU TIME. IT MAY BE LOOPING.
   Action:
     USE YOUR SOFTWARE MONITORS TO INVESTIGATE THE ASID.
   ------------------------------------------------------------------------
   -----
     EVENT 07: HIGH - LOOP - SYSTEM: SY1 2009/06/12 - 13:28:46
     ASID: 001F JOBNAME: JOBS4 TCB: 004E6850
     STEPNAME: PFATEST PROCSTEP: PFATEST JOBID: STC00042 USERID:
   ++++++++
     JOBSTART: 2009/06/12 - 13:28:35
   Error:
     ADDRESS SPACE APPEARS TO BE IN A LOOP.
   Action:
     USE YOUR SOFTWARE MONITORS TO INVESTIGATE THE ASID.

   (Additional output intentionally omitted.)
```

# PFA Dependencies and Installation

- IBM Health Checker for z/OS – recommend to start at IPL
- z/OS UNIX file system – where we store the data
- Java (31-bit only) – used primarily during modeling
  - Java 5.0 or later for z/OS 1.11 and z/OS 1.12
  - zAAP eligible (recommended)
- z/OS 1.13 – The Runtime Diagnostics address space must be active for “too low” detection
- Follow the install steps in *z/OS Problem Management*

# How to Get the Most Out of PFA

- Use a zAAP to offload PFA's Java Processing
- Start z/OS Resiliency functions at IPL
  - IBM Health Checker for z/OS
  - PFA
  - Runtime Diagnostics (z/OS 1.13)
- Automate the PFA IBM Health Checker for z/OS exceptions
  - Simplest: Add exception messages to existing message automation product
  - More complex: Use exception messages and other information to tailor alerts
  - See *z/OS Problem Management* for exceptions issued for each check
- Create a policy in an HZSPRMxx member for persistent changes
  - Not all check-specific parameters are required on an UPDATE of PFA checks!
    - UPDATE CHECK=(IBMPFA,PFA_COMMON_STORAGE_USAGE) PARM('THRESHOLD(3)')

# How to Get the Most Out of PFA (continued)

- **Get the latest PTFs!**
  - Configuration value default changes
  - Comparison algorithm tuning changes
  - Changes to design
    - Exclude interactive users from being persistent jobs for some checks
    - Skip comparisons for ESQA
  - zFS space reduction
- Help us to make PFA's results better for everyone!

# How to Get the Most Out of PFA (continued)

- Use PFA's modify command to display parameters
  - Modify command in IBM Health Checker for z/OS does not display cumulative values
- Change the type of WTO of a check if default is not appropriate for your installation
  - Default is SEVERITY(MED) which issues an eventual action WTO
  - `f hzsproc,update,check(ibmpfa,pfa_j*),wtotype=info`
- Quiesce rather than delete PFA checks
  - Optional (only if you want to stop collections and modeling):
    - `f hzsproc,update,check(ibmpfa,pfa_j*,parm('collectinactive(0)')`
  - `f hzsproc,deactivate,check(ibmpfa,pfa_j*)`

# How to Get the Most Out of PFA (continued)

- z/OS 1.12 -- Implementing supervised learning
  - Example: Exclude test programs that issue many LOGRECs and cause exceptions.
  - Example: Exclude address spaces that issue many WTOs, but are inconsistent or spiky in their behavior and cause message arrival rate exceptions.Supported by all checks except Common Storage Usage
  - Create EXCLUDED_JOBS file in the check's /config directory
    - Simple comma-separated value format
      - JobName,Systems,Date,Reason
    - Supports wildcards in both job name and system name
      - KKA*,*,04/05/2011,Exclude all KKA* jobs on all systems
  - Use `f pfa,update,check(check_name)` if PFA running
  - PFA creates an EXCLUDED_JOBS file for some checks during installation

# How to Get the Most Out of PFA (continued)

- **Use check-specific tuning parameters** to adjust *sensitivity of comparisons* if needed
  - To minimize customer configuration
    - Default parameter values constructed from in-house and external data
    - Some defaults changed via PTFs using customers' data

| Parameter | Description |
|---|---|
| **STDDEV** | Increase value to decrease sensitivity.<br>Not available on the Common Storage Usage check. |
| **EXCEPTIONMIN** | Increase value to decrease exceptions issued for relatively low rates.<br>Not available on the Common Storage Usage check |
| **THRESHOLD** | Increase value to decrease sensitivity.<br>Common Storage Usage check only |
| **STDDEVLOW** | Increase value to decrease sensitivity for "too low" checking.<br>Available on checks where "too low" checking is supported. |
| **LIMITLOW** | Defines the maximum rate where "too low" checking is performed<br>Available on checks where "too low" checking is supported. |

# How to Get the Most Out of PFA (continued)

- **Use PFA check-specific parameters** to *affect other behavior*

| Parameter | Description |
|---|---|
| **COLLECTINT** | Number of minutes between collections |
| **MODELINT** | Number of minutes between models<br>• PFA automatically and dynamically models more frequently when needed<br>• z/OS 1.12 default updated to 720 minutes. First model will occur within 6 hours (or 6 hours after warm-up) |
| **COLLECTINACTIVE** | Defines whether PFA should collect and model if check not active/enabled in IBM Health Checker for z/OS |
| **DEBUG** | Use only if IBM service requests it |
| **CHECKLOW** | z/OS 1.13 – Turns on/off "too low" checking with RTD for checks that support it |
| **TRACKEDMIN** | Requires a persistent job to have this minimum rate at the end of the warm-up in order to be tracked (where supported) |
| **Health Checker parameters** | For example, SEVERITY -- All PFA checks default = SEVERITY(MED): Eventual action WTO |

# Summary

- PFA uses *historical data* and *machine learning algorithms* to detect and report soft failures *before they can impact your business*
  - PFA is focused on *damaged systems* and *resource exhaustion*
- Tips
  - Use the PFA reports to help diagnose problems
  - Use a zAAP for PFA's java processing
  - Tune the PFA checks using the configuration parameters and the EXCLUDED_JOBS list if necessary.
  - Stay current on PTFs
  - Start IBM Health Checker for z/OS, Runtime Diagnostics, and PFA at IPL
  - Automate exception messages
  - *z/OS Problem Management* is the main source for documentation
- Use other products to do deep investigation of system or address space problems.

zEND

# Questions ?