TXSeries (CICS) Recommenations

Architectural, Data Storage, Security & Administration

Iain Boyle, Software Group Services

Iain_boyle@uk.ibm.com

Version 1
19-December-2005

Introduction

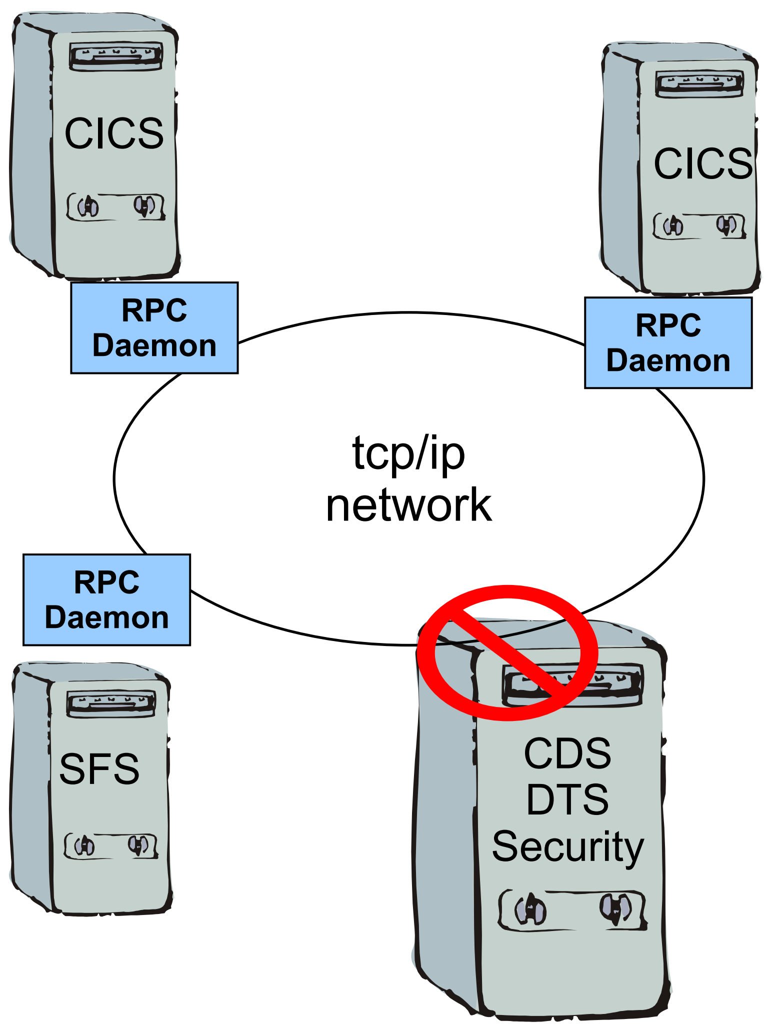
- This presentation is a series of recommendations and guidelines from IBM on how to configure and manage a TXSeries (CICS) environment.

- As with all recommendations:
  - There will be exceptions
  - Good reasons to ignore the recommendation
  - Different alternatives available

- What you should do:
  - Understand the advice
  - Consider your environment
  - Decide if the recommendation is suitable for you

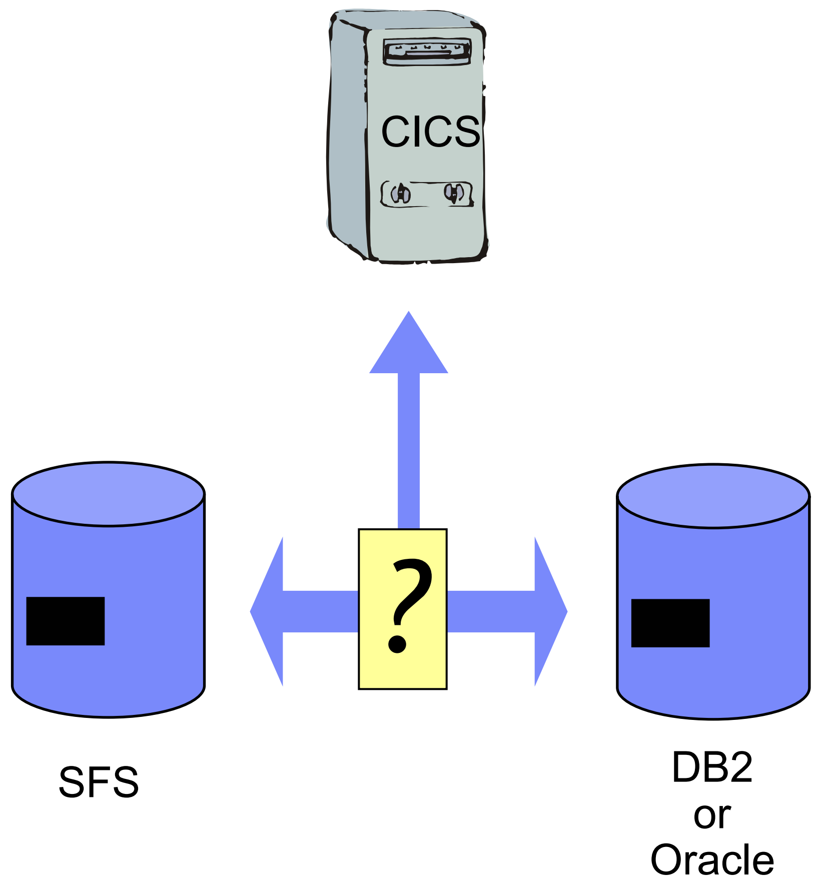
Contents

Use DCE in RPC Mode



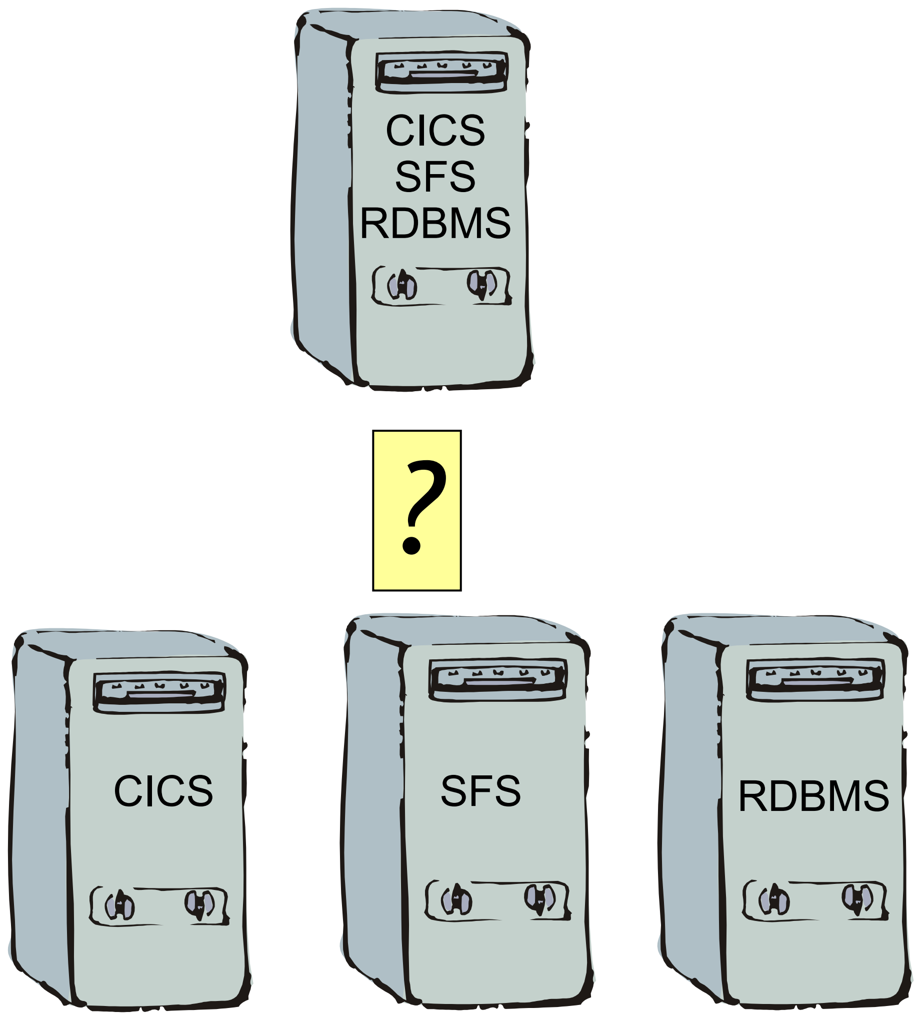
- Removes additional layer of complexity and DCE Server processes
- Only use DCE in following circumstances
  - DCE environment already present and available
  - DCE skills available
  - Secure RPC calls needed between CICS, SFS and PPC Gateway servers

Where to Store VSAM Data?



- For VSAM data, TS and TD queues, CICS can use
  - SFS (from Encina)
  - RDBMS (DB2 or Oracle)
- Use RDBMS if
  - Licenses available
  - Skills exist to Manage, Configure, Tune and Operate
- Use SFS if
  - RDBMS not an option
  - Unrecoverable data widely used

Location of VSAM Data



- Several options for locating CICS, SFS, RDBMS and PPC Gateway servers
- One machine is simplest
- Multiple machines adds complexity
- Decision depends on
  - Machines available
  - Skills available
  - Location of existing data

Contents

DB2 Storage Space

- DB2 storage space can be
  - DMS – Database Managed Storage. DB2 manages to a predefined limit
  - SMS – System Managed Storage. DB2 uses filesystem to manage storage
- DMS is generally 10-15% faster than SMS but requires more management
- Suitable bufferpool caches data and reduces disk access
  - Reduces DMS and SMS differences

Database Security

- Use explicit userid/password on database connection
  - If not defined, authentication uses implicit credentials of service

- Set **TP_MON_NAME** to **CICS** in DB2
  - Identifies CICS as a transaction manager

Using DB2 for VSAM data increases risk of deadlocks

CICS

TD - DeadLockTimeout

- To allow CICS to detect a deadlock

DB2

DB2_RR_TO_RS

- To switch off next key locking
  - Avoid Repeatable Read cursors
  - Only affects non-CICS DB2 applications

locktimeout

- Waiting time for a local

dlcktime

- Time interval for checking for deadlocks

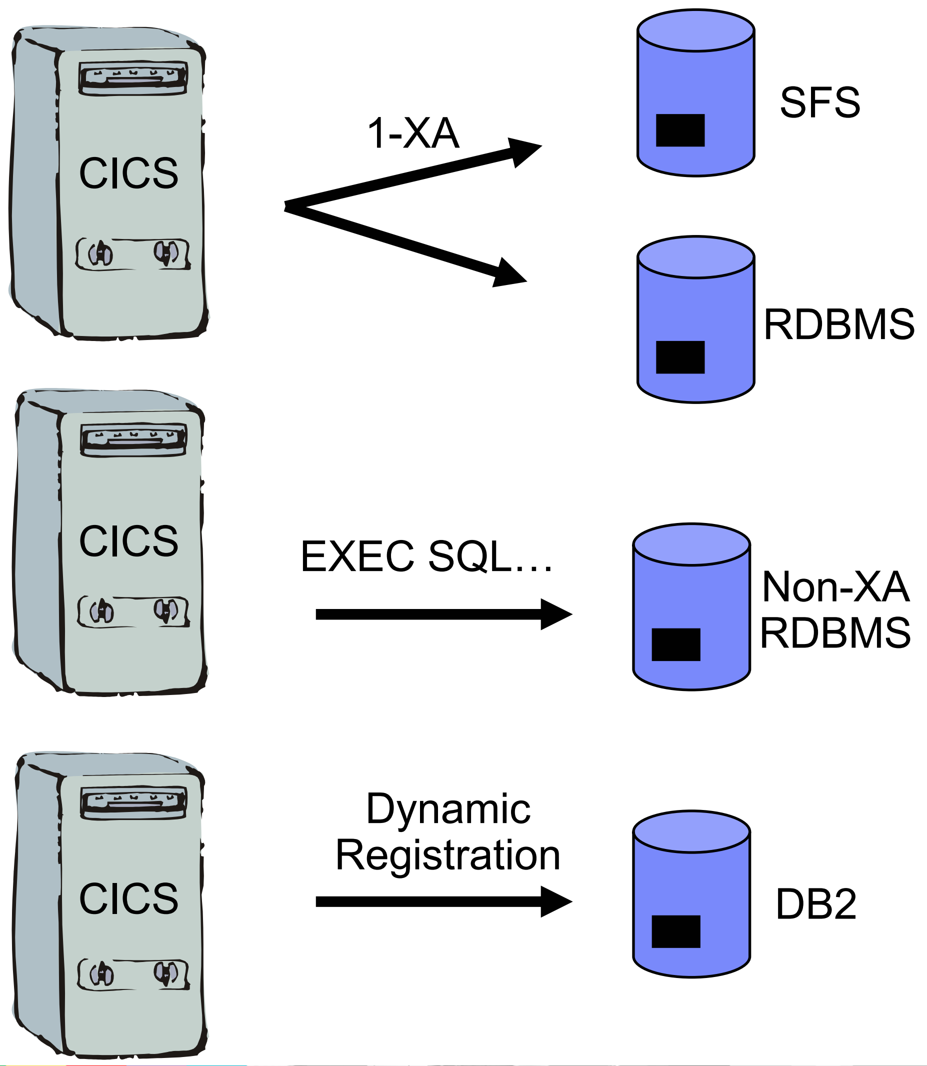
locklist

- Storage for lock list

maxlocks

- Percent of lock list full before escalation starts

XA Connections



- Use **single** phase XA if one non-SFS reosurce updated
  - Application responsible for recovery
  - **Do not use EXEC SQL COMMIT and ROLLBACK if region has an XA connection**
- Must use EXEC SQL statements to Connect, Commit and Rollback
- DB2 supports dynamic registration
  - CICS connects to DB2 only when data is updated

Contents

Avoid Fast Local Transport (FLT)

- Available when Encina Client and Server on same machine
- Use **pipe** or **shared-memory** as transport
- Only available on Unix systems

- **IBM No longer recommends it's use**

- To disable: set following
  - ENCINA_FLT_CLIENT_MAX_DS = 0
  - ENCINA_FLT_SERVER_MAX_DS = 0
- in
  - /var/cics-regions/<region>/environment
  - /etc/environment

Never Cold Start SFS

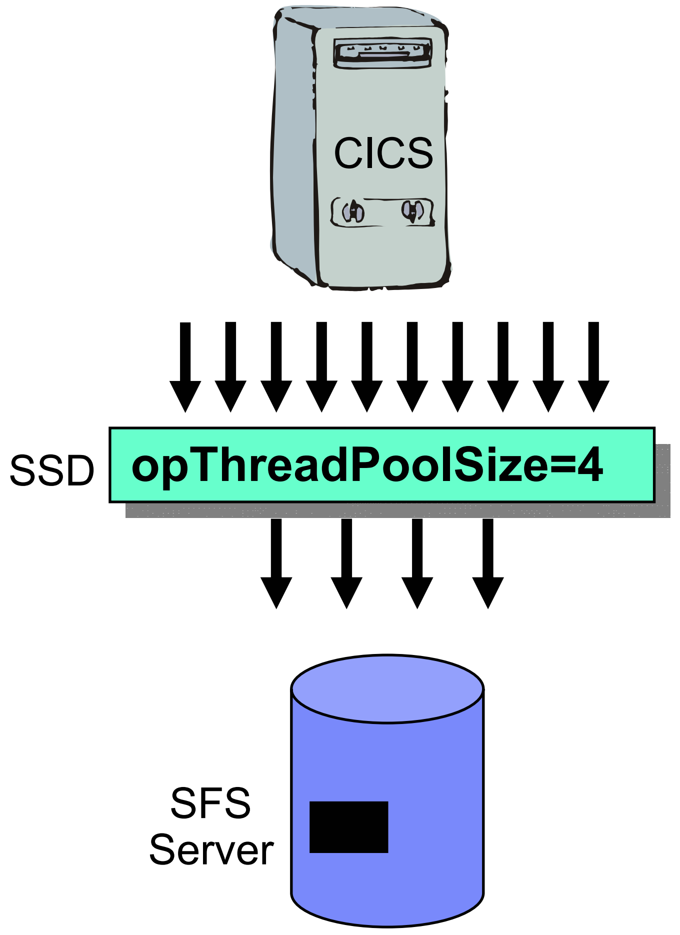
- An SFS **cold** start is **not** same as CICS cold start

- A CICS cold start will …
  - Not recover any transactions
  - Empty TS and TD queues
  - Reload from permanent database

- An SFS cold start will ….

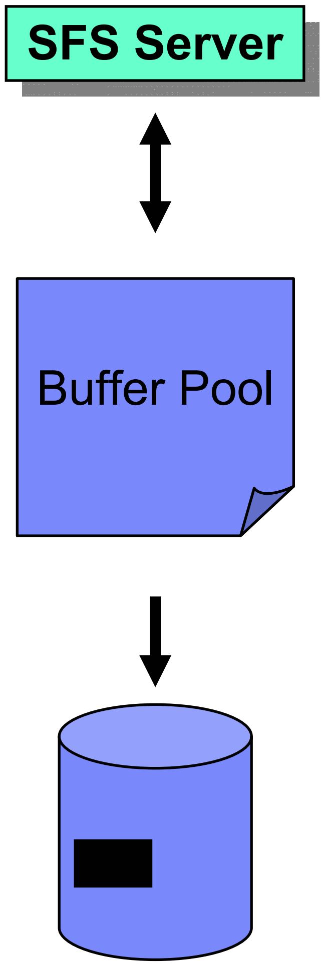
… discard all data and files on a volume !!!

Manage SFS Threads



- Maximum concurrent requests allowed by SFS server
- Defined in SSD stanza
- Default is 12
- Set to MaxServers + 1

Buffer Pool Size



- BufferPoolSize
  Data cache used by SFS server
- Defined in SSD stanza
- Default is 1000 Kb
- Too small – excessive disk I/O
- Too big – excessive paging and memory

Browse Cache

- CICS_BROWSE_CACHE environment variable

- Cache size used for browsing files on an SFS server

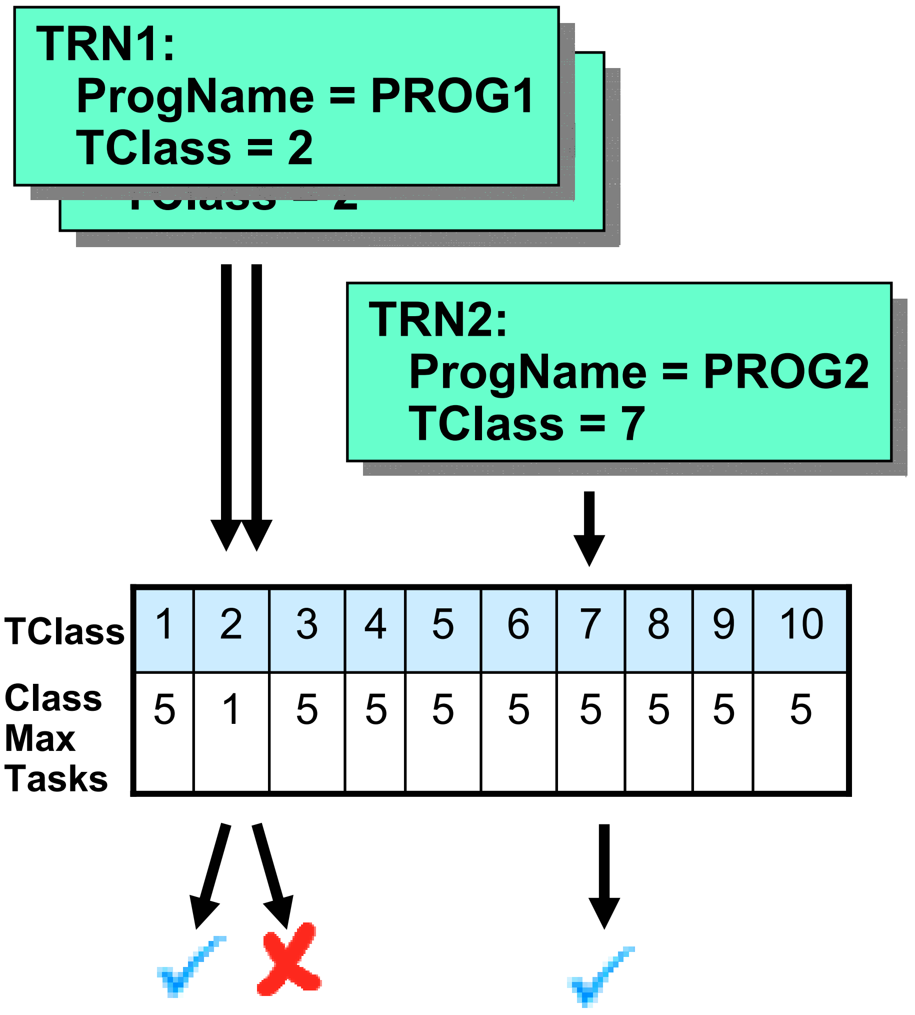
Contents

Disable EDF

```
cicstran –l <lang> -e <file>
```

- Avoid –e option to reduce overhead of EDF check on every CICS API call

Use TClass



TClass	1	2	3	4	5	6	7	8	9	10
Class Max Tasks	5	1	5	5	5	5	5	5	5	5

- 11 classes defined to CICS
  - 1 to 10 plus “NONE” (the default)
- Limit of concurrent transactions per class for classes 1 to 10
- TClass defined in TD stanza

Recommend

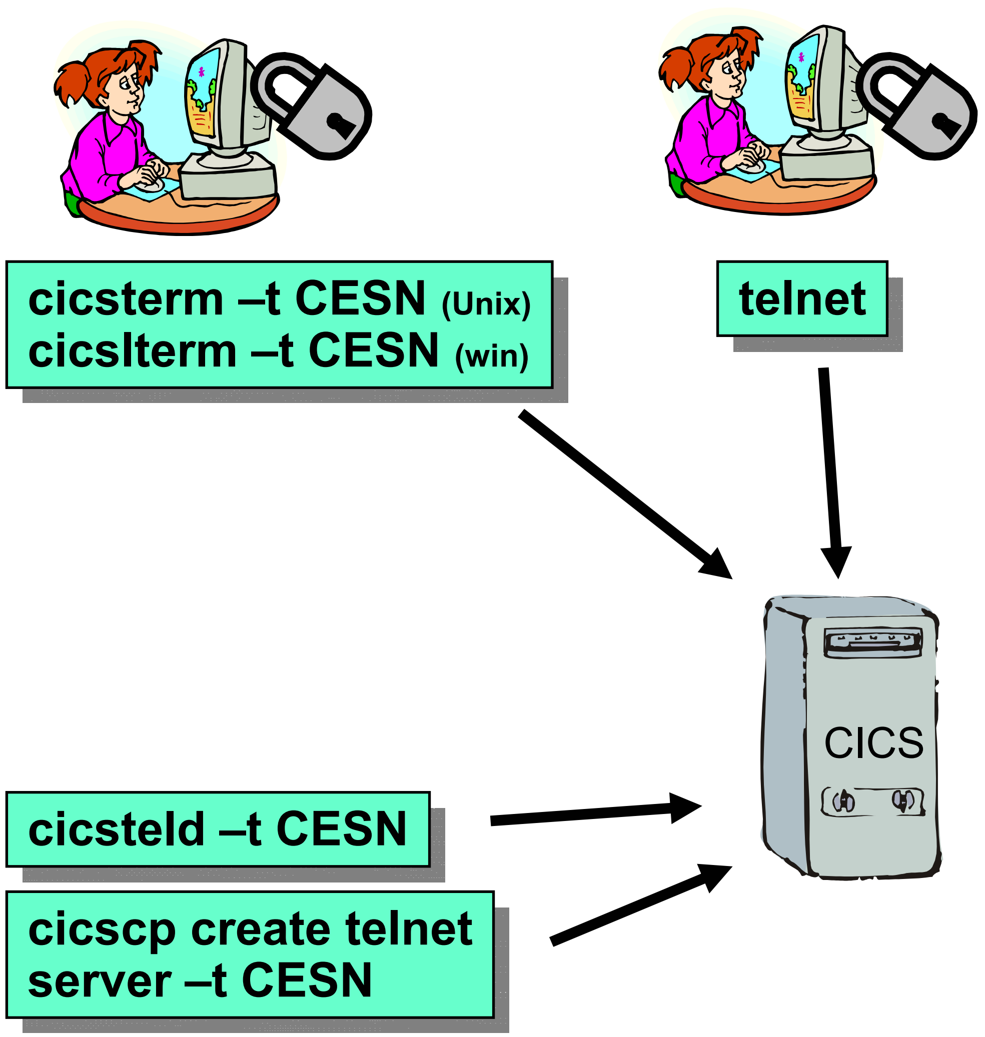
- User transactions have a TCLass
- CICS transactions use NONE
- MaxServers = sum(ClassMaxTasks)

Example

- 1 instance of TRN2. Limit = 5
  - Allowed to run
- 2 instances of TRN1. Limit =1
  - 1 allowed to run, 1 queued

Contents

Force Terminal Users to Signon


```
cicsterm –t CESN (Unix)
cicslterm –t CESN (win)
```

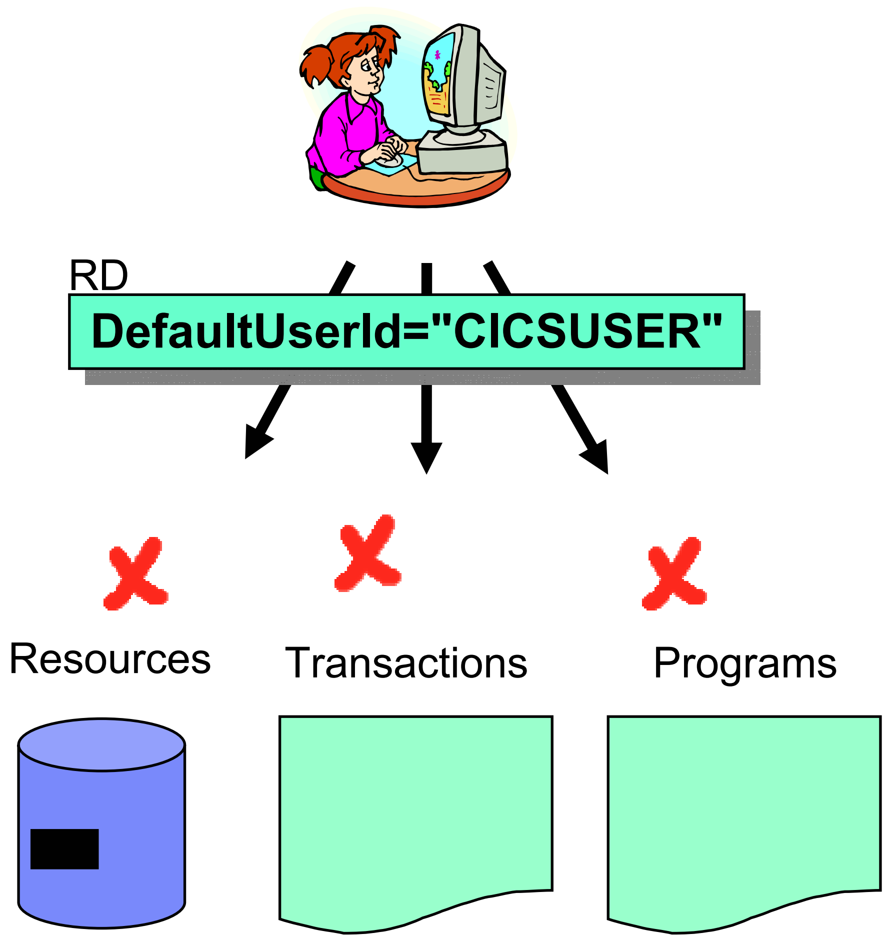
```
telnet
```

```
cicsteld –t CESN
```

```
cicscp create telnet server –t CESN
```

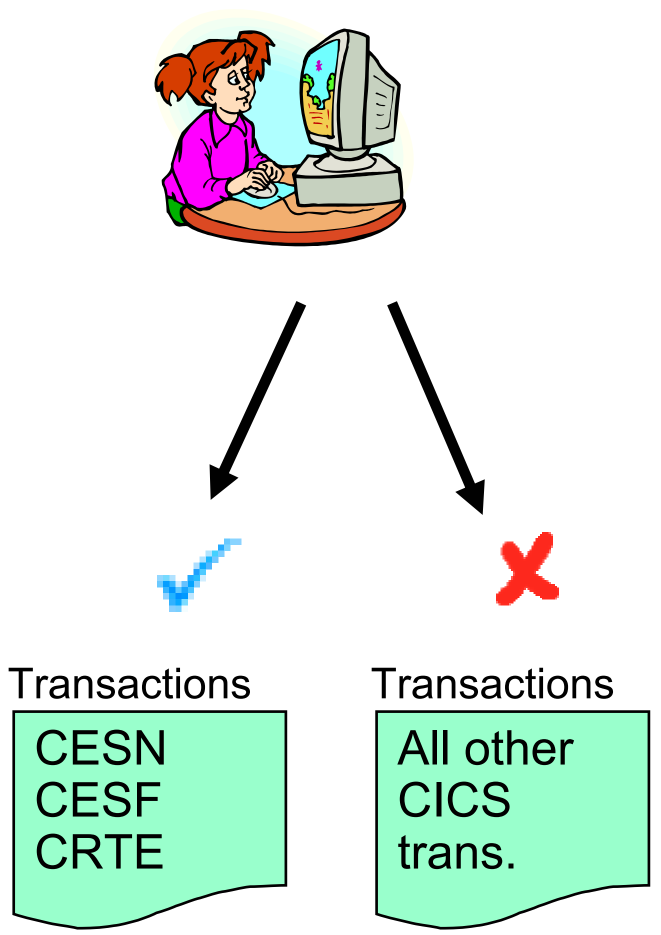
- Always force users to authenticate using:
  - CESN
  - Custom signon transaction
- Option **-t** forces an initial transaction. Applies to
  - cicsteld
  - cicscp create telnet server
  - cicsterm / cicslterm
- Universal Client has similar option in CTG.INI configuration file (INITIALTRANSID)

Use of Default User



- Unauthenticated requests use credentials of Default User
- Restrict access of Default User to
  - Resource (files, TD, TS)
  - Transactions
  - Programs

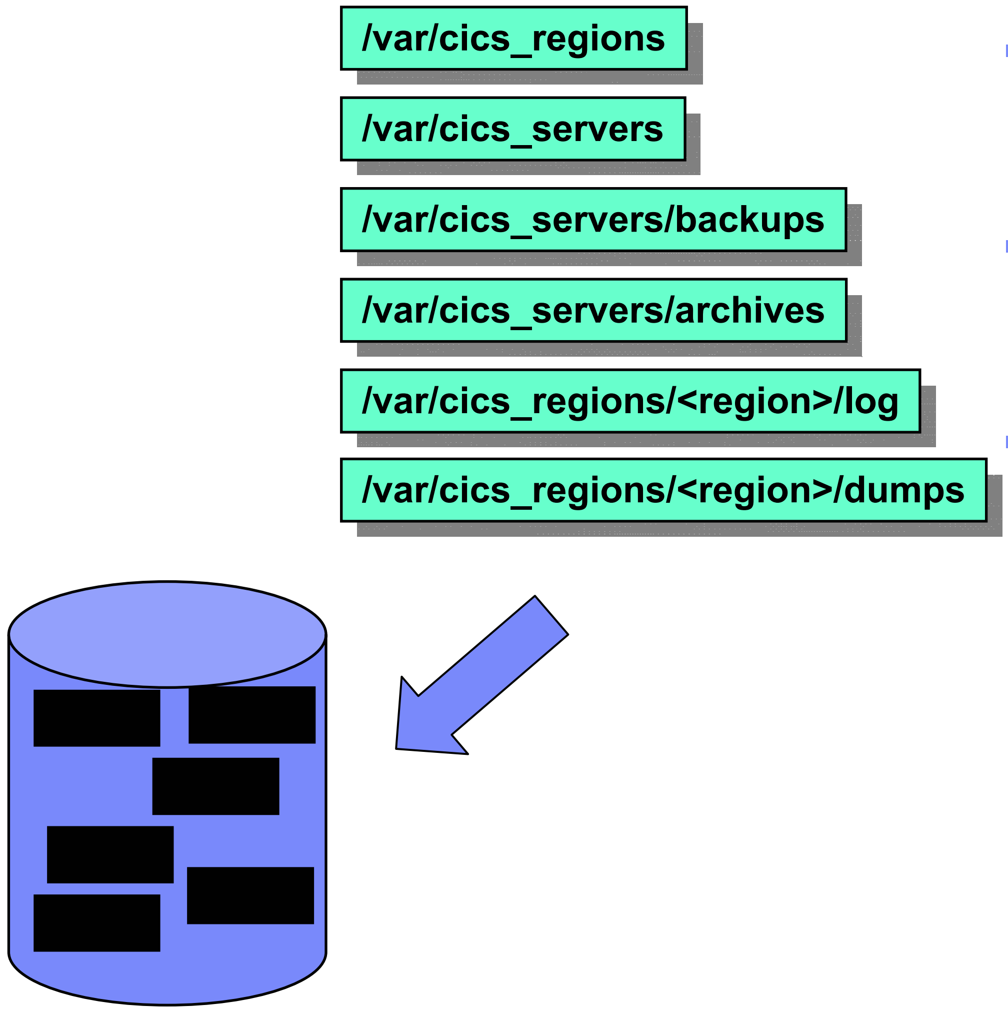
Protect CICS Resources



- By default, all users can run CICS supplied transactions, since TSLKey=public
- Consider protecting
  - CEMT
  - CEDF
  - CECI
- Only allow Systems Programmers to use sensitive transactions

Contents

Use Operating System File Systems


/var/cics_regions
/var/cics_servers
/var/cics_servers/backups
/var/cics_servers/archives
/var/cics_regions/<region>/log
/var/cics_regions/<region>/dumps

- Create separate file systems (logical volume) for selected CICS directories
- Create file system for region log and dumps after region creation
- CICS region log is required for warm start
  - Suggest mirroring

CICS Application Servers

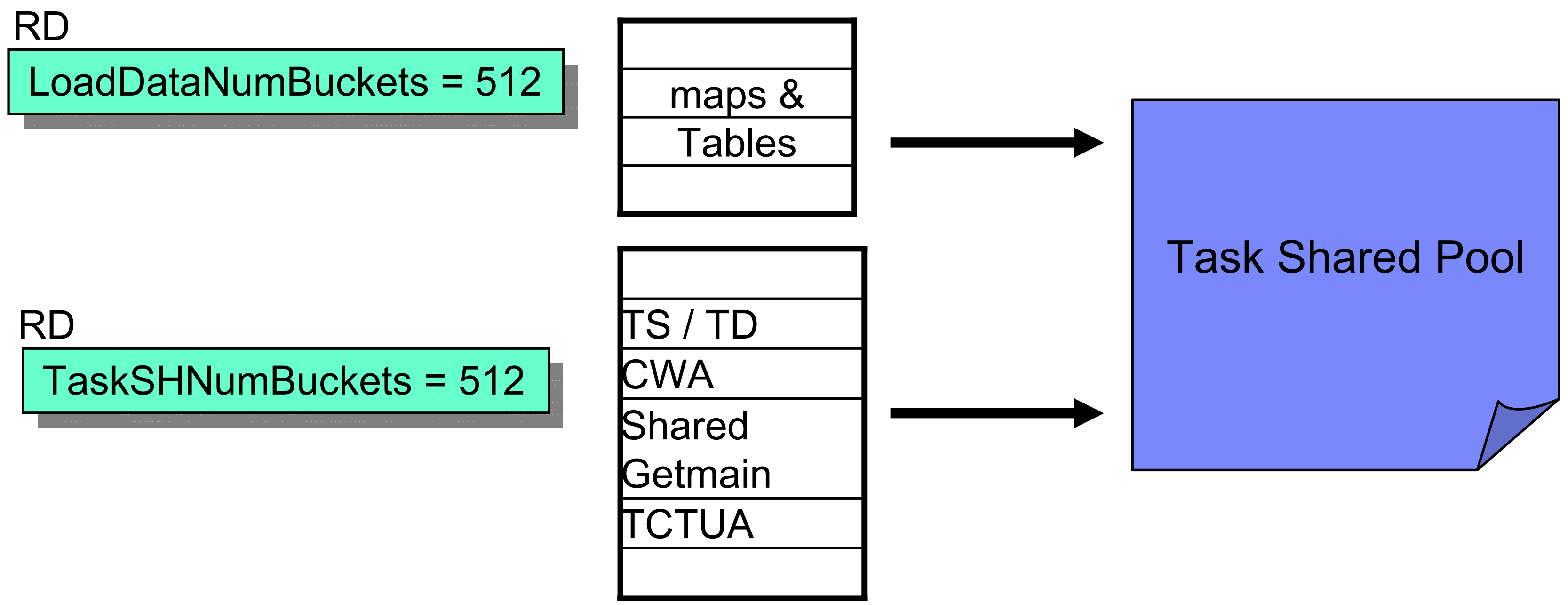
- Responsible for running programs
- Number of **cicsas** processes is controlled by Min and Max Servers in the RD stanza
- Each **cicsas** process also allocates a heap for customer program use
- Well written customer code should therefore use CICS facilities to obtain memory, rather than the standard language facilities (such as the C malloc function for example)
- User written code is compiled into executables which CICS code loads into a **cicsas** process and branches to
- It is important to note that once the user code execution is complete, the **cicsas** process unloads the code (so registering callbacks, for example with the C atexit() function is not a good idea!)

Use of Min and MaxServers

- Defined in RD stanza
- Determines maximum number of tasks in system
- Too small?
  - Requests queue for dispatch
- Too large?
  - Idle prcoesses and wasted resources

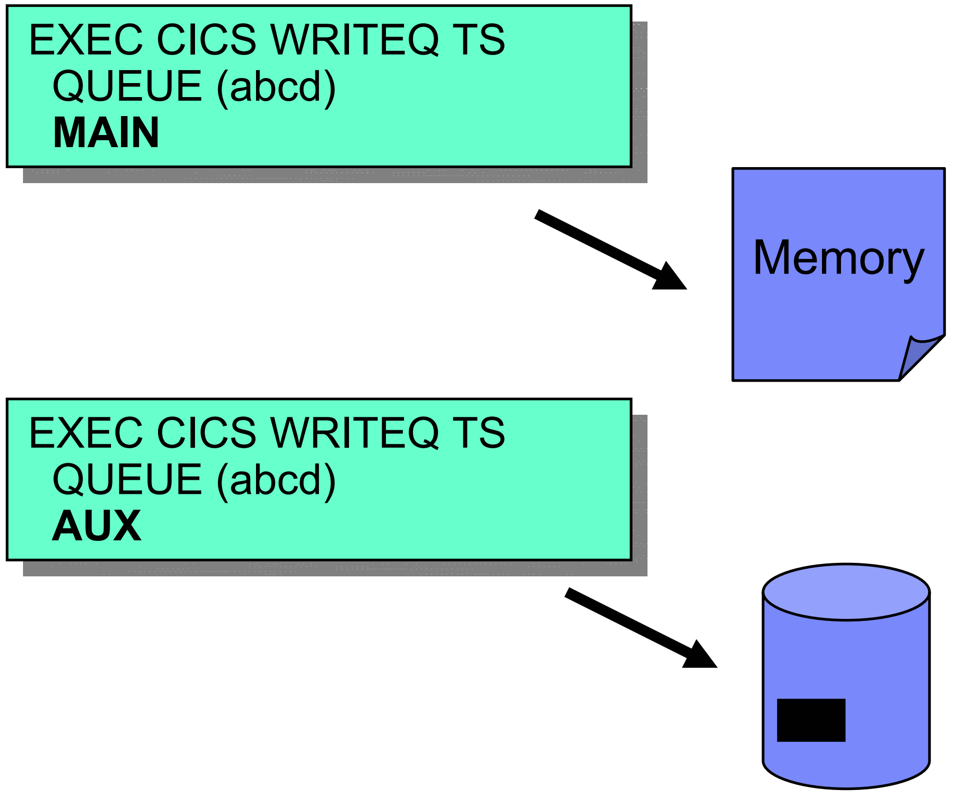
- Identify correct values through testing, tuning and observations
- To modify
  - Change RD stanza
  - Use ***CEMT INQ SYS***

Use Hash Buckets



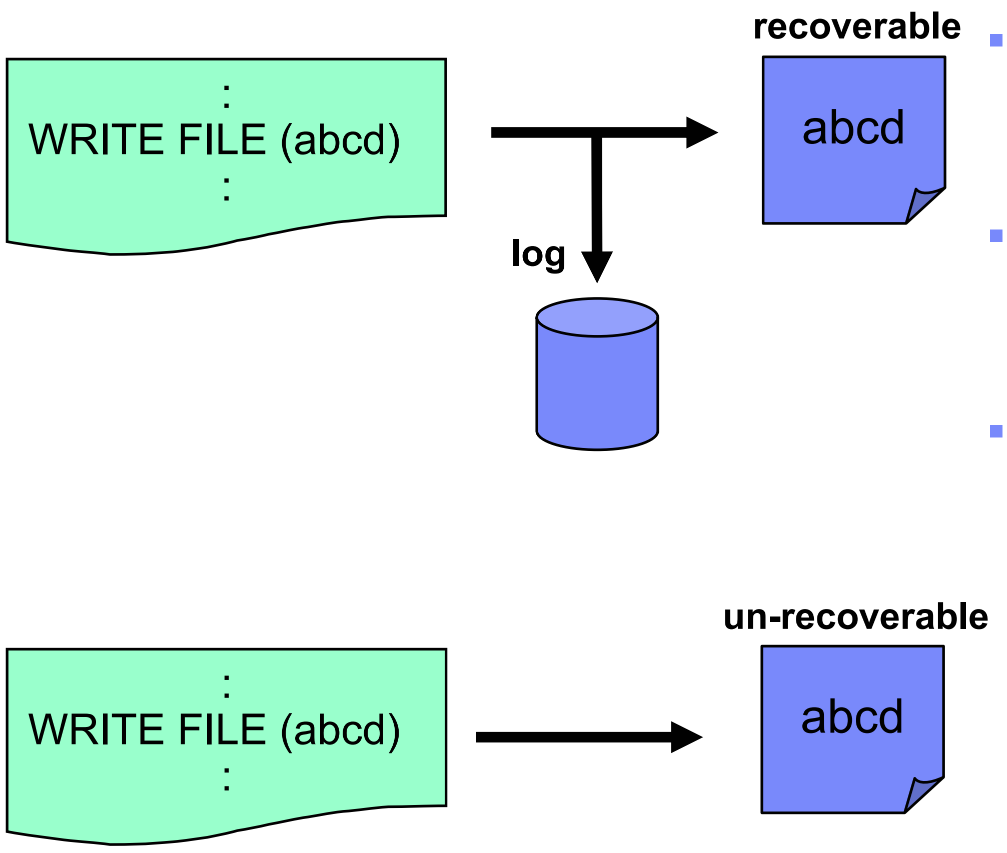
- LoadDataNumBuckets: size of hash table to find **maps & tables** in task shared pool
- TasksSHNumBuckets: size of hash table to find **everything else** in task shared pool
- Default is 512
- Keep same or make TasksSHNumBuckets 10x bigger
- Use Statistics to tune

Use Main over Aux for TS Queues



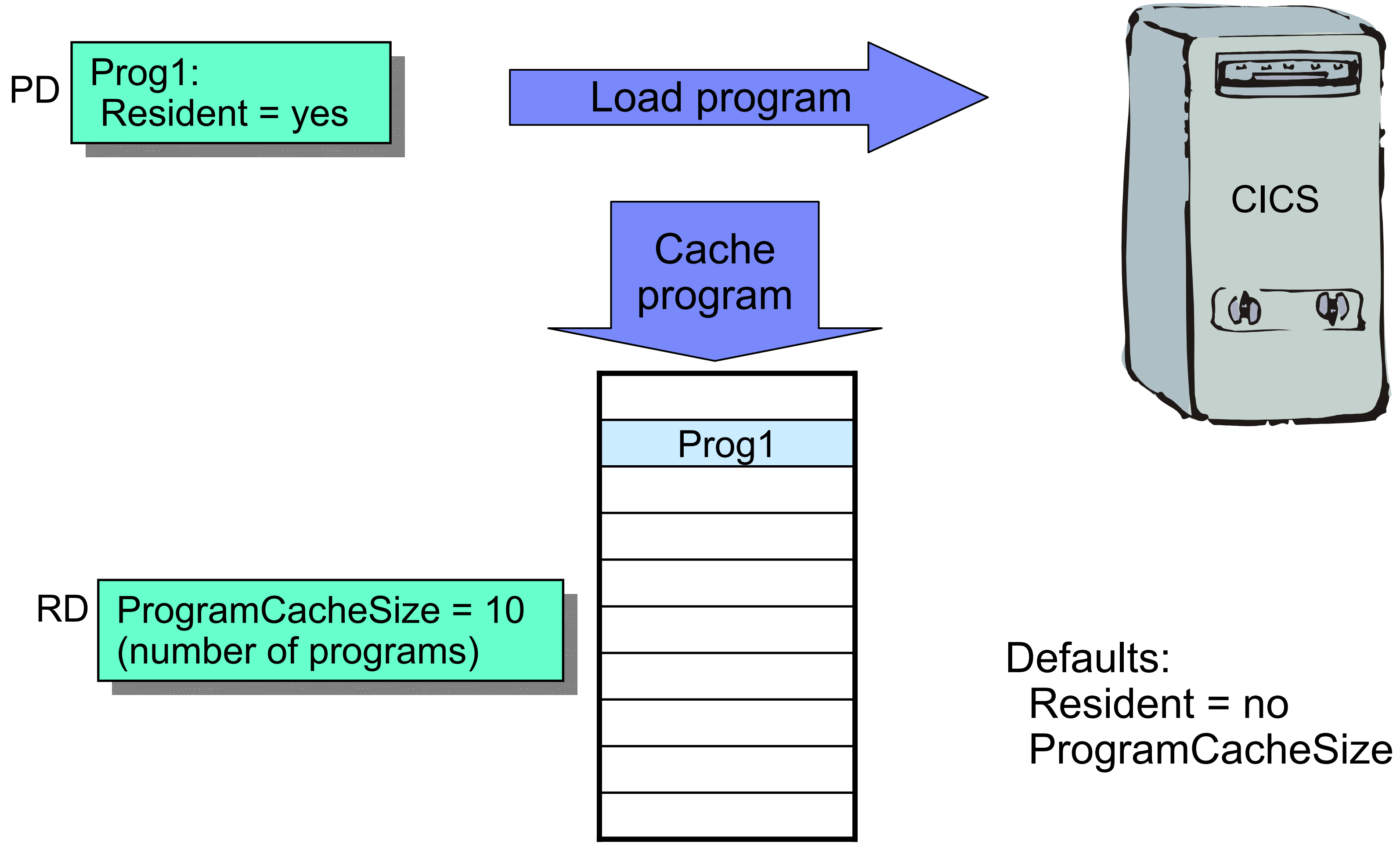
- MAIN
  - Stored in region storage (memory)
  - Non-recoverable only
  - Short retention
  - Lost on CICS restart
  - Faster
  - Use if possible
- AUX
  - Stored on file system (or rdbms)
  - Recoverable & non-recoverable
  - Long retention
  - Retained on CICS restart
  - Slower

Use Unrecoverable Resources



- Recoverable resources require additional logging (for recovery).
- Un-recoverable resources do not require this logging.
- Make read only resources unrecoverable

Cache Programs



Defaults:
Resident = no
ProgramCacheSize = 0

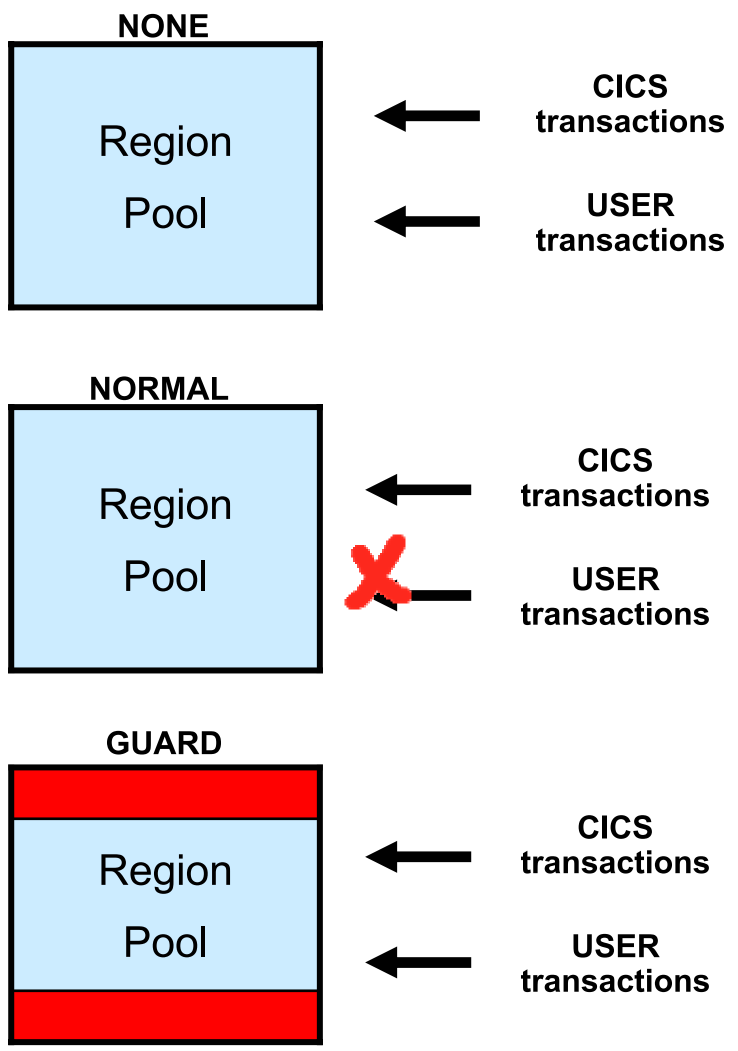
Use Full Path Name

PD

```
Prog1:
 PathName = <program path and file name>
```

- Reduce program load times by
  - Using absolute pathname to file followed by filename
  - Using program extension (if one exists)
- Avoid using **CICS_PROGRAM_PATH** environment variable to search multiple directories

Use of SafetyLevel



RD

```
SafetyLevel = none | normal | guard
```

- Ignored on Solaris (same as none)
- Normal incurs performance cost
  - Only use if storage corruption occurs
- Guard is Windows only
  - Less performance cost than Normal

- Default is None
  - Avoid changing

Use Timeouts

TD

```
DeadLockTimeout = 0
Timeout = 0
```

- DeadLockTimeout
  - Time (seconds) transaction allowed to wait when deadlock detected
- Timeout
  - Time (seconds) to wait for terminal input
- Default is 0 for both
- Only effective if deadlock is not in program

Increase Introspect Interval

RD

```
IntrospectLevel = minimal
IntrospectInterval = 10
```

- Introspect is CICS self checking
- **Level**
  - Fixed at **minimal**
- **Interval**
  - Default is 10 minutes
  - Frequent intervals decreases performance
  - Once region is stable increase to several hours

Check for Memory Growth

RD

```
ServerMemCheckInterval = 3600
ServermemCheckLimit = 4
```

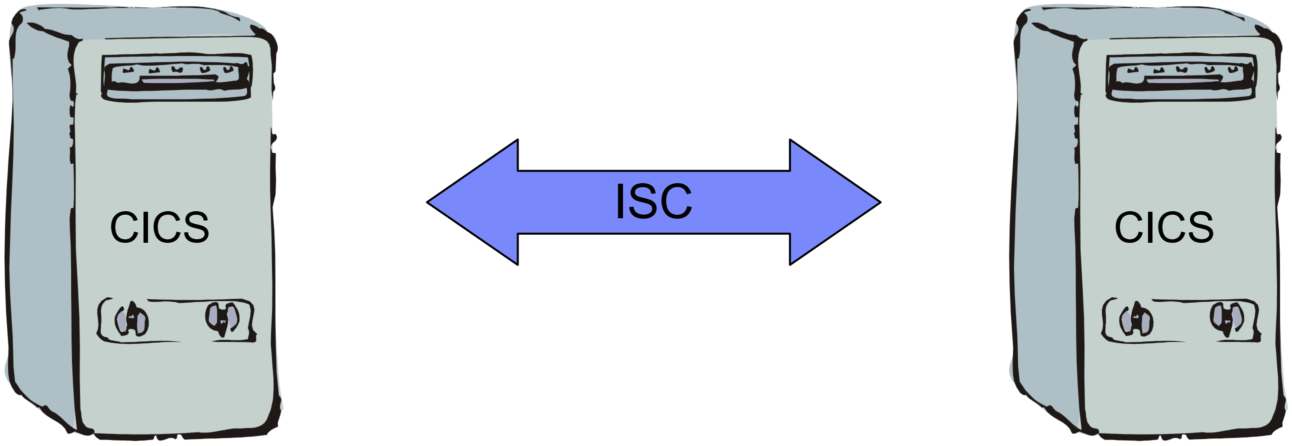
- Manage memory growth checking
  - **ServerMemCheckInterval**
    - Time in seconds between memory growth checks
    - Default is 3600 (0 is disabled)
  - **ServermemCheckLimit**
    - Number of consecutive checks before CICS reports growth
    - Default is 4 (0 means disabled)
  - Messages written to console

Check for Leaks

```
CICS_LEAKDEBUG="LOGDIR=/var/cics_regions/leak MEM=heap
LANG=c,cpp,java FILEDES=allowcore,minlimit=1000,maxlimit=1100
TIMESTAMP=ON TRAN=SAMP"
```

- Used to check for memory growth and file descriptor leaks
- Set in the regions “environment” file
- File created for each cicsas process called **cicsas.<pid>**
- Following options are available:
  - LOGDIR=<Location of the directory to store report files>
  - MEM=<heap | taskprivate | taskshared>
  - LANG=<c | cpp | cobol | ibmcob | ibmpli | java | cbmfnt | ALL>
  - FILEDES=minlimit=<value>,maxlimit=<value>[,allowcore]
  - TIMESTAMP=<ON | OFF>
  - LEVEL=<1 for entry/exit, 2 for full debug>
  - TRAN=<List of transactions>

Minimise Intersystem Communication



- Intersystem Communication (ISC) is
  - Function Shipping, Transaction Routing, Distributed Program Link, Distributed Transaction processing, Asynchronous Starts
- All ISC is expensive in resources and performance
  - Try to minimise where possible, use local resources