SHARE March, 2011, Session: 8668

Detecting Soft Failures Using z/OS PFA

Karla Arndt
z/OS Predictive Failure Analysis
Rochester, Minnesota
kka@us.ibm.com

SHARE
Technology • Connections • Results

Trademarks

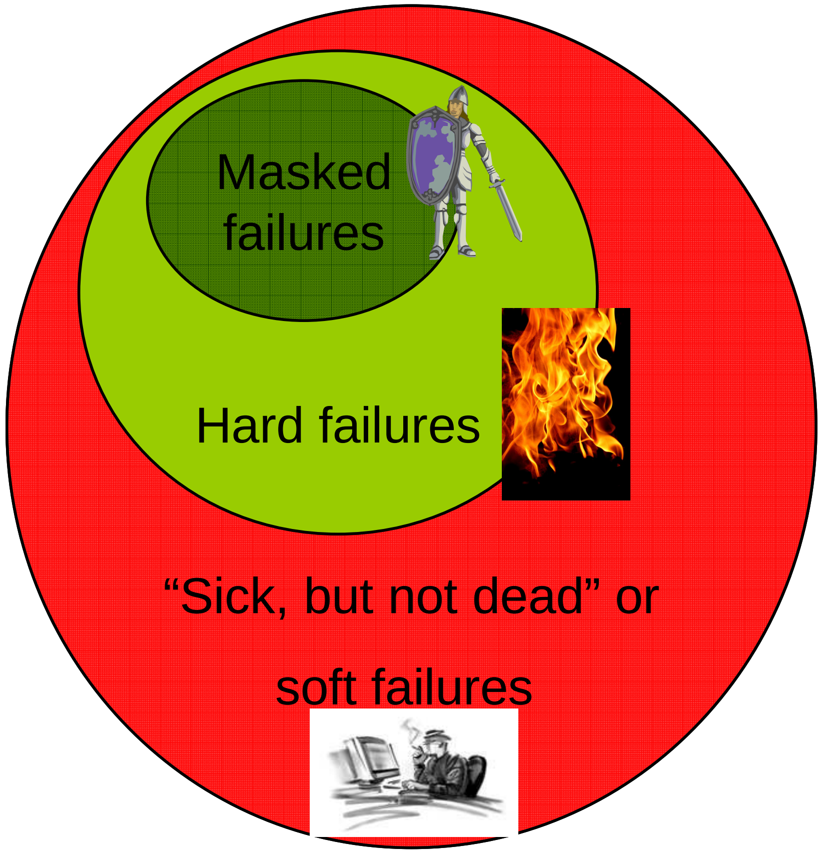
Agenda

- Soft failures defined
- How PFA detects and reports soft failures
- The PFA checks
- **What's new in z/OS 1.13!**
- How to get the most out of PFA
- Installation and Migration
- Serviceability and Troubleshooting
- Summary

PFA Detects Soft Failures: So, what is a soft failure?

"Your systems don't break. They just stop working and we don't know why."

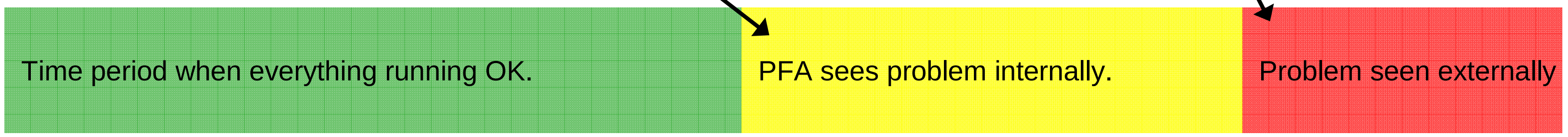
"Sick, but not dead" or Soft failures



- 80% of business impact, but only about 20% of the problems
- Long duration
- Infrequent
- Unique
- Any area of software or hardware
- Cause creeping failures
- Hard to determine how to isolate
- Hard to determine how to recover
- Hard for software to detect internally
- Probabilistic, not deterministic

Soft Failures: Hypothetical IT Example

1. A transcation --
   - that has worked for a long time starts to fail, or
   - occasionally (yet, rarely) fails
   - Example – “Reset Password and send link to registered email account”
2. The transaction starts failing more regularly
3. Recovery is successful –
   - Such that the overall, applications continue to work
   - Generates burst of WTO’s, SMF records and LOGREC entries
4. BUT, THEN! Multiple, failing transactions occur together on a heavily loaded system
   - Recovery occurs
   - Slows down transaction processor
   - Random timeouts of other transactions occur
   - System becomes “sick, but not dead”


Time period when everything running OK.
PFA sees problem internally.
Problem seen externally

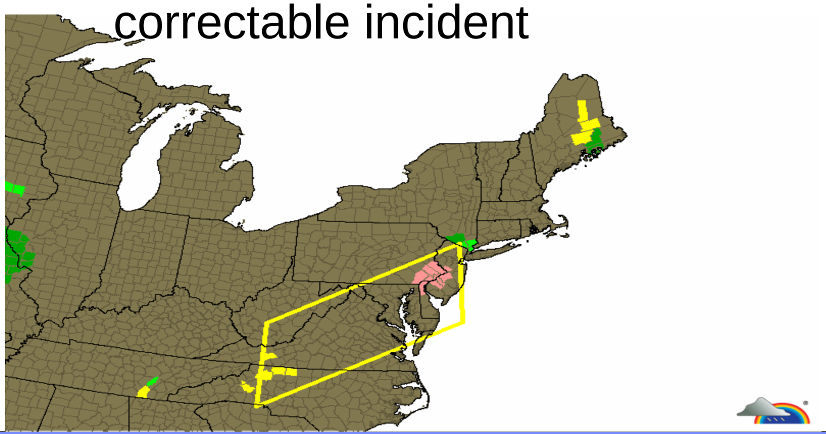
This is a hypothetical problem which is a combination of multiple actual problems

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

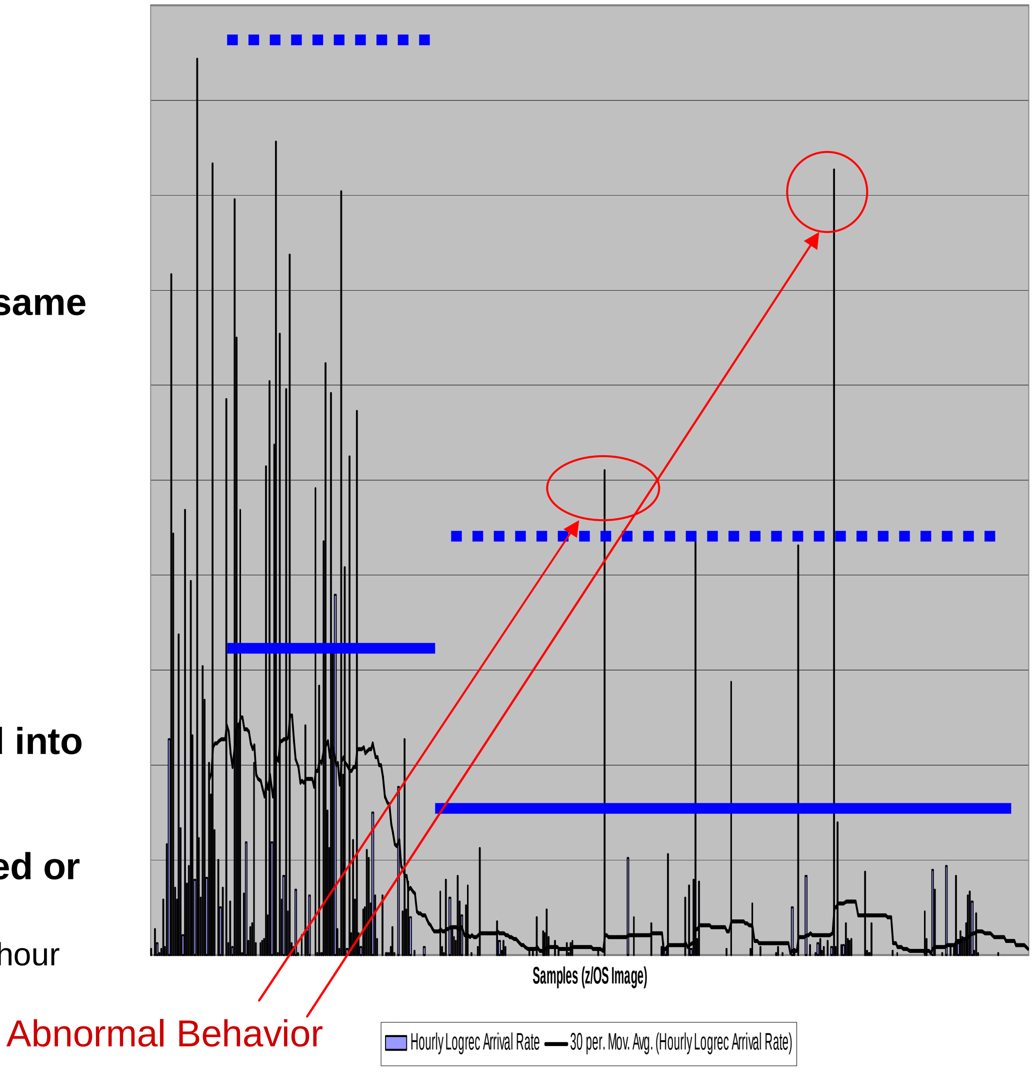
- Predictive failure analysis uses
  - *Historical data*
  - *Machine learning and mathematical modeling*

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

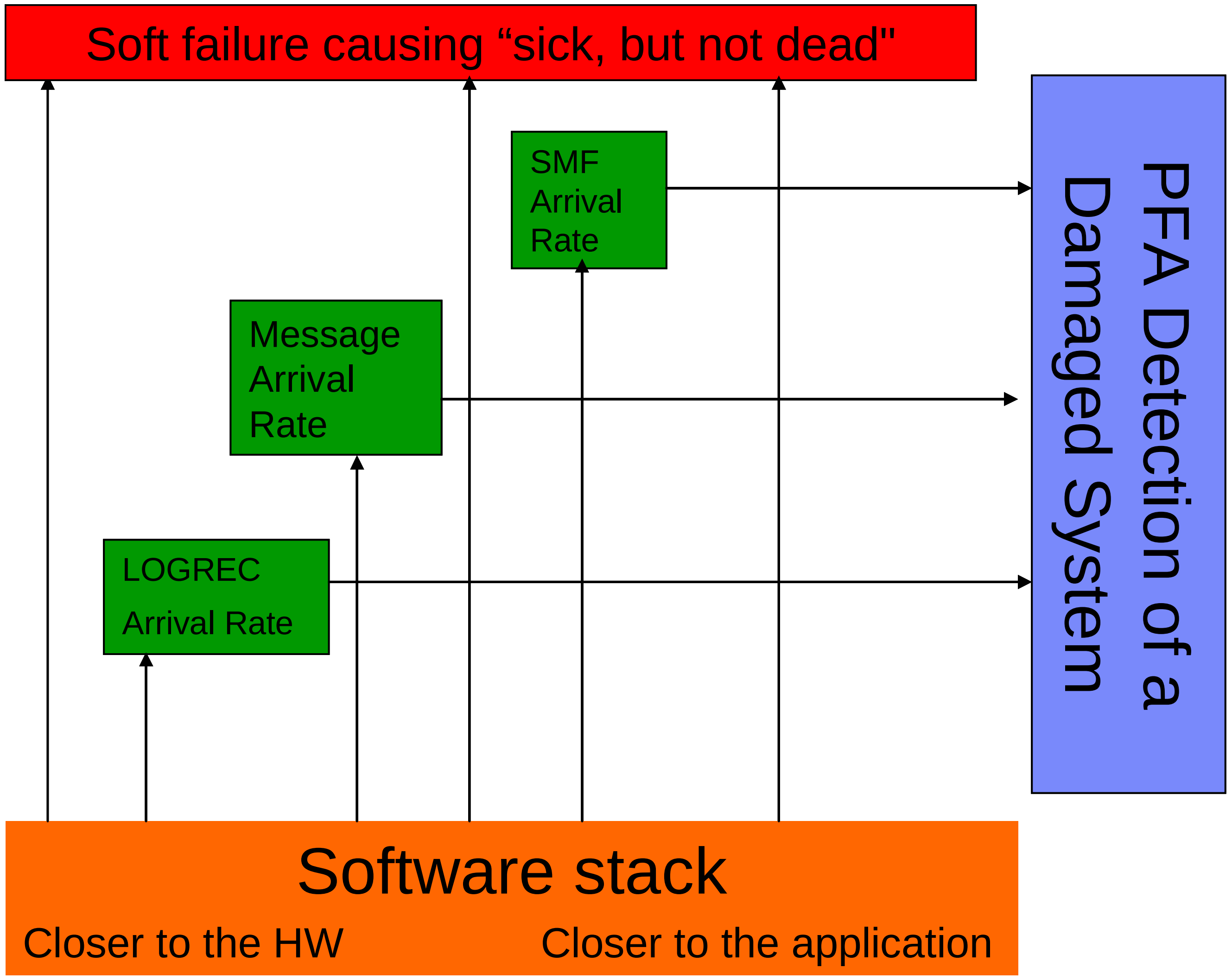


How PFA Determines Normal Values

- **Behavior of z/OS system is a function of**
  - Workload
  - Type of work
  - Hardware and Software configuration
  - System automation
  - …
- **Same type of work runs at approximately same time**
- **Use historical data to calculate future or expected value to eliminate variables**
  - Hardware and Software configuration
  - System automation
  - …
- **Expected value = fn(workload, time)**
- **Future value = fn(workload, time projected into future)**
- **Cluster metric by time to calculate expected or future value**
  - Can compare different time ranges such as 1 hour ago, 24 hours ago, 7 days ago

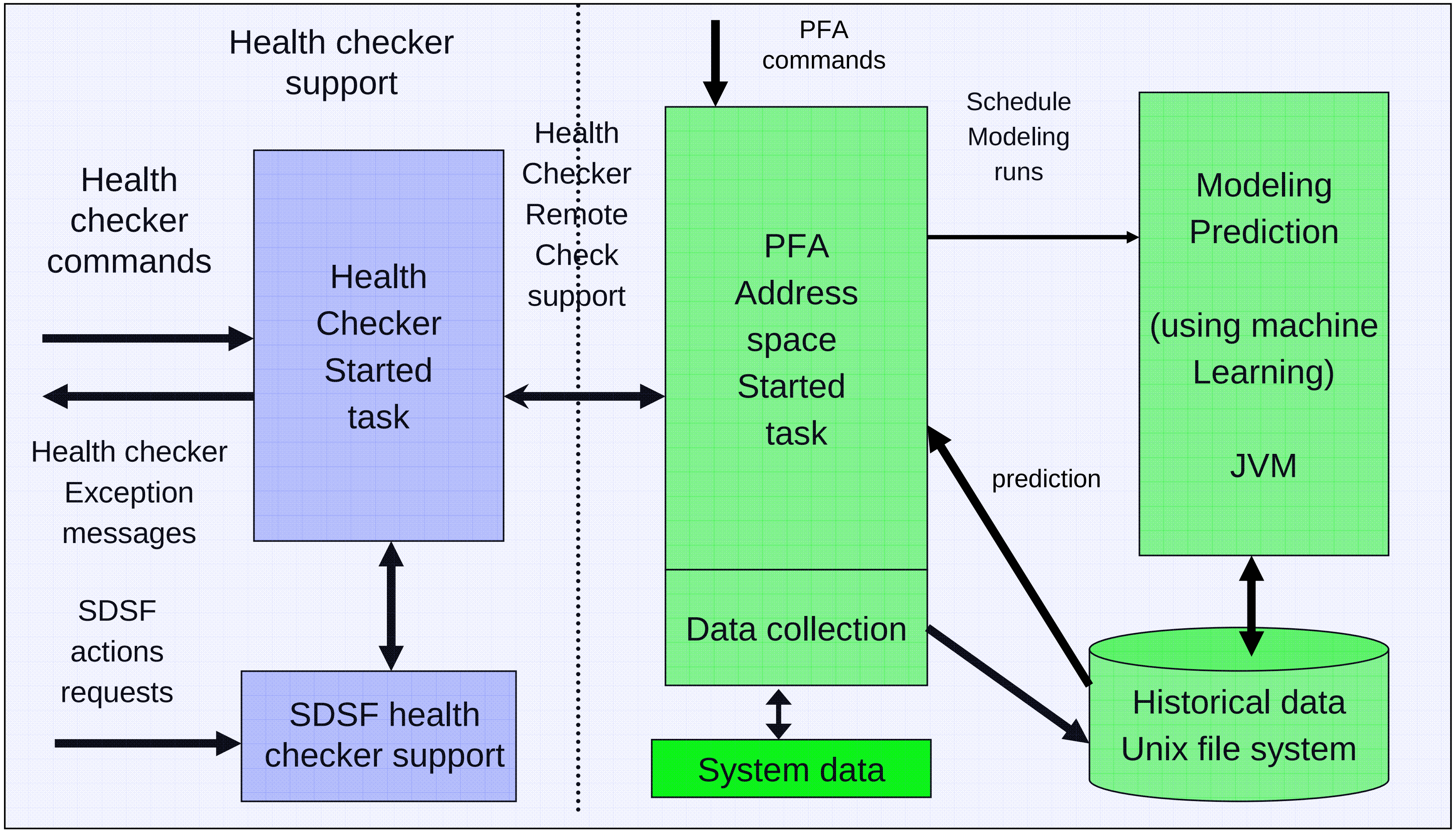


How PFA Chooses the Metrics: Layered approach


Soft failure causing “sick, but not dead"
SMF Arrival Rate
Message Arrival Rate
LOGREC Arrival Rate
PFA Detection of a Damaged System
Software stack
Closer to the HW
Closer to the application

Not to Scale

PFA's Relationship to IBM Health Checker for z/OS


Health checker support
Health checker commands
Health Checker Started task
Health checker Exception messages
SDSF actions requests
SDSF health checker support
Health Checker Remote Check support
PFA commands
PFA Address space Started task
Data collection
System data
Schedule Modeling runs
Modeling Prediction (using machine Learning) JVM
prediction
Historical data Unix file system

PFA Functions and Dependencies

- PFA is its own address space – recommend to start at IPL
  1. *Collects data* from the system
  2. *Models the data* from the system to create predictions
  3. *Performs comparisons* on current vs. predictions
  4. *Issues exceptions or "OK" messages and reports* via IBM Health Checker for z/OS
- Dependencies
  - 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 1.4 or later for z/OS 1.10
    - 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.

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 frames and slots usage to detect potential virtual storage leaks in persistent 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)

What happens when PFA detects a problem?

- *Health check exception* written to console
  - New exceptions suppressed until new model is available
- *Prediction report* available in SDSF (s.ck)
  - *"Top address spaces"* = potential villains
  - *Address spaces causing exception*
  - *Current and predicted values* provided
  - Reports also available when no problem occurs
- *Modeling automatically runs* more frequently
- *Best practices and more information* in *z/OS Problem Management*

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

- Available in SDSF (s.ck)
- Heading information
  - Configuration and status
  - Current and predicted information for metric
- Top predicted users
  - Tries to pinpoint potential villains
- Other information
  - CSA – expansion information
  - IBM Health Checker for z/OS message in its entirety

```
                 Common Storage Usage Prediction Report
(heading information intentionally omitted)
                                            Capacity When   Percentage
Storage     Current Usage    Prediction       Predicted     of Current
Location     in Kilobytes  in Kilobytes    in Kilobytes    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         Storage     Current Usage     Predicted Usage
  Name        Location    in Kilobytes      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.
```

z/OS 1.10 (SPE) PFA Checks

- **PFA_COMMON_STORAGE_USAGE** → 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

- **PFA_LOGREC_ARRIVAL_RATE** → 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
  - 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

- Exceptions produced for any key grouping for any time range
- “Jobs having LOGREC arrivals in last collection interval”
  - Lists the jobs contributing to the arrival count.
  - Only displayed if the arrival count > 0.

```
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** → detects a *damaged system by predicting resource exhaustion* by detecting abnormal increased usage of frames and slots by persistent address spaces
  - *Persistent address spaces* are those that start within an hour after IPL
  - Does not detect small usage increases, fragmentation, or really rapid growth (machine time scale)

- **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

Frames and Slots Usage Prediction Report

- “Address spaces with the highest increased usage”
  - Lists the jobs whose frames and slots usage recently increased the most
  - Sorted by expected usage
  - List exists even when no problem
- Exception raised when one or more jobs use substantially more frames and slots than expected
  - Only jobs causing exception listed when exception produced

```
          Frames and Slots Usage Prediction Report
(heading information intentionally omitted)

Address spaces with the highest increased usage:
   Job                   Current Frames     Expected Frames
   Name        ASID     and Slots Usage     and Slots Usage
   ________    ____     _______________     _______________
   ZFS         0029               12223               12329
   XCFAS       0048                1593                1601
   VTAMOSR3    0027                1885                1881
   TRACE       0036                 367                 367
   SMS         0025                 682                 687
```

Message Arrival Rate Prediction Report

- Perform 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. Example “no problem” report and “total system” exception report shown

```
                     Message Arrival Rate Prediction Report
(heading lines intentionally omitted)
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
                      Message                Arrival Rate
  Job                 Arrival
  Name     ASID          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
...
```

z/OS 1.12 – One New Check and Many Enhancements

- **SMF arrival rate check** → 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.
- **Common storage usage check increased granularity**
  - SQA + CSA, ESQA + ECSA, SQA, CSA, ESQA, and ECSA

z/OS 1.12 – Many Enhancements

- Common storage usage and LOGREC arrival rate check **performance improvements**
- Checks enhanced to **improve dynamic modeling** and comparison algorithms
- **Supervised learning** → Excluded jobs list
- **Usability** – One ini file for all checks
- **Serviceability** – Separate log files and files and report data copied at exception to EXC_*timestamp* directory

z/OS 1.13 Enhancements

- **PFA_ENQUEUE_REQUEST_RATE Check**
  - Detects a *damaged address space or damaged system* by comparing the number of enqueue requests per CPU second to the rate expected.

- **PFA_JES_SPOOL_USAGE Check**
  - Detects *abnormalities in persistent jobs* by modeling the change in their JES2 spool usage (i.e., the change in the number of track groups used from one collection to the next)

- **PFA Integration with Runtime Diagnostics**
  - *Detects if certain metrics are too low* by using PFA along with the results of Runtime Diagnostics

z/OS 1.13 PFA_ENQUEUE_REQUEST_RATE Check

- Detects a *damaged address space* or *damaged system* by comparing the number of enqueue requests per CPU second to the rate expected.
- Enqueue request rate = Number of enqueues requested / CPU Utilization
- 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
- Ignores first hour after IPL and last hour prior to shutdown

```
            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
                      Enqueue                Request Rate
  Job                 Request
  Name      ASID         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
```

z/OS 1.13 PFA_JES_SPOOL_USAGE Check

- 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

z/OS 1.13 PFA_JES_SPOOL_USAGE Check

- The exception is issued based on an *unexpected increase in the number of track groups used* from one collection to the next (rather than the number of track groups used).
- The current number of track groups used is provided as additional information.

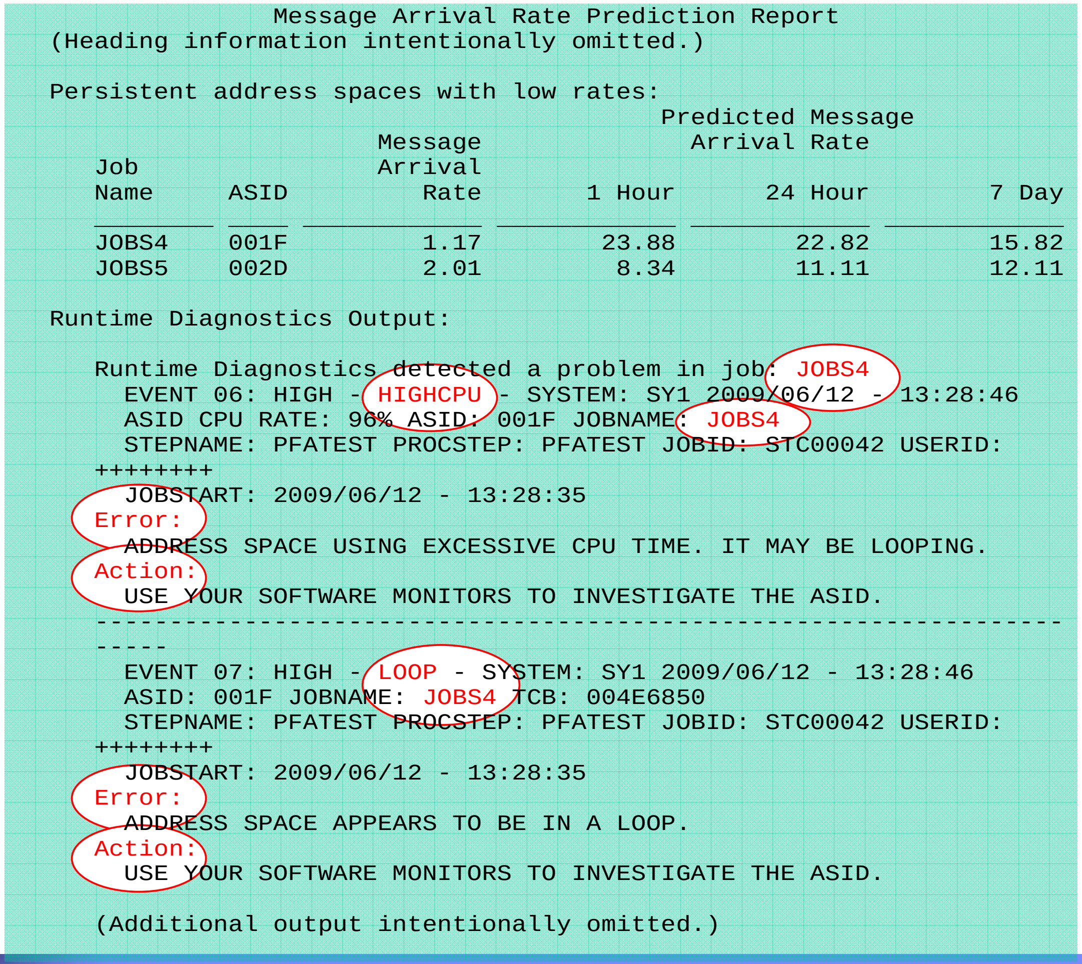
```
                   JES Spool Usage Prediction Report

(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
```

z/OS 1.13 PFA Integration with Runtime Diagnostics

- 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
                       Message              Arrival Rate
    Job                Arrival
    Name      ASID        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.)
```

How to Get the Most Out of PFA

- **Use check-specific tuning parameters** to adjust sensitivity of comparisons if needed
  - Default parameter values were carefully constructed to minimize configuration
  - Refer to *z/OS Problem Management* for definitions of tuning parameters for each check
    - **THRESHOLD** -- Common storage usage check only
    - **STDDEV** -- All checks except Common Storage Usage
    - **EXCEPTIONMIN** -- LOGREC arrival rate (PTF), Message arrival rate, SMF arrival rate, Enqueue request rate
    - **STDDEVLOW** – z/OS 1.13 -- Message arrival rate, SMF arrival rate, Enqueue request rate
    - **LIMITLOW** – z/OS 1.13 -- Message arrival rate, SMF arrival rate, Enqueue request rate

How to Get the Most Out of PFA (continued)

- **Use check-specific parameters to affect other behavior**
  - **COLLECTINT** – All checks -- Number of minutes between collections.
  - **MODELINT** – All checks -- Number of minutes between models.
    - PFA automatically and dynamically models more frequently when needed
    - z/OS 1.12 default updated to 720 minutes. First model will occur within 6 hours (or 6 hours after warm-up)
  - **TRACKEDMIN** – Message arrival rate, SMF arrival rate, Enqueue request rate
  - **COLLECTINACTIVE** – All checks – Defines whether PFA should collect and model if check not active/enabled in IBM Health Checker for z/OS
  - **DEBUG** – All checks -- Use only if having a problem with PFA (service requests it)
  - **CHECKLOW** – z/OS 1.13 – Message arrival rate, SMF arrival rate, Enqueue request rate

How to Get the Most Out of PFA (continued)

- z/OS 1.12 – *Eliminate* jobs causing false positives
  - **Unsupervised learning** is the machine learning that PFA does automatically.
  - **Supervised learning** allows you to exclude jobs that are known to cause false positives. For example,
    - Exclude test programs that issue many LOGRECs and cause exceptions.
    - Exclude address spaces that issue many WTOs, but are inconsistent or spiky in their behavior and cause message arrival rate exceptions.

How to Get the Most Out of PFA (continued)

- z/OS 1.12 -- Implementing supervised learning
  - Supported by all checks except Common Storage Usage
  - Create EXCLUDED_JOBS file in the check's /config directory
    - /u/pfauser/PFA_LOGREC_ARRIVAL_RATE/config/EXCLUDED_JOBS
    - Comma-separated value file
      - Jobname,system,date_time,reason_added
      - Jobname and system name are required
    - Sample in /usr/lpp/bcp/samples/PFA
  - Start PFA or use f pfa,update,check(*check_name*) if PFA running
  - Supports wildcards in both job name and system name
    - * or ? Supported
    - For example,
      - KKA,*,03/15/2010 12:08,Exclude KKA job on all systems.
      - ABC*,LPAR1,03/03/2010,Exclude all ABC* jobs on LPAR1
  - The Message Arrival Rate Check on z/OS 1.12 installs an EXCLUDED_JOBS file by default that excludes all JES* jobs on all systems.

How to Get the Most Out of PFA (continued)

- 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
- Start IBM Health Checker for z/OS at IPL
- Start Runtime Diagnostics at IPL (z/OS 1.13)
- Start PFA at IPL
- 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
  - **OA33669 – Coexistence support** for fall back from z/OS 1.12 to either z/OS 1.11 or z/OS 1.10
  - **OA34586 – PFA_COMMON_STORAGE_USAGE Check Updates**
    - Delay processing of the module mapping until modeling
    - Serviceability improvements
  - **OA34992 – Reliability problem with certain checks**
    - z/OS 1.12 only
    - PFA_LOGREC_ARRIVAL_RATE, PFA_MESSAGE_ARRIVAL_RATE, and PFA_SMF_ARRIVAL_RATE
  - **OA35064 -- Inconsistent summarization** for PFA_COMMON_STORAGE_USAGE and PFA_FRAMES_AND_SLOTS_USAGE
  - **OA31644 -- Add EXCEPTIONMIN parameter** to PFA_LOGREC_ARRIVAL_RATE to reduce false positives on systems that have normally low arrival rates
  - **OA34654 and OA34655 (target availability end of April)** – **Too many exceptions** in the message arrival rate and frames and slots usage checks, respectively

- If on z/OS 1.10 → Shipped as SPE -- Get all PFA and MAPMVS PTFs

Installation and Migration

- Follow install and configuration instructions in *z/OS Problem Management*
  1. Ensure dependencies are configured and working
  2. Create PFA user ID to own the PFA started task and directories
  3. Run install script from the pfauser's home directory
     - Use "*new*" if you are installing for the first time or want to delete data from previous releases (prior to z/OS 1.12, use AIRSHREP)
     - Use "*migrate*" if you want to retain data from previous releases (recommended)
       - z/OS 1.11 → AIRSHKP
     - For z/OS 1.12 and z/OS 1.13, run AIRSHREP.sh directly or use the JCL file provided in SYS1.SAMPLIB(AIRINJCL).
       - If using AIRINJCL, you must update it to specify your pfauser's home directory.
  4. Update the Java configuration in each check's ini file (or in z/OS 1.12, /etc/PFA/ini) if needed
  5. If you install the PFA code in a place other than the default (/usr/lpp/bcp), update the PROC to have the correct path.

Installation and Migration (continued)

- z/OS 1.12 - PFA now supports one or multiple ini files
  - Uses ini file in /etc/PFA unless check-specific ini exists
    - /etc/PFA directory automatically created when z/OS 1.12 installed
      - If an installation does steps that would cause the /etc/PFA directory to no longer exist, the new ini file processing cannot be used.
      - The /etc/PFA directory can be created manually as described in *z/OS Problem Management*.
    - If "*new*" was chosen,
      - /etc/PFA/ini copied from /usr/lpp/bcp/samples/PFA
      - Update Java configuration in /etc/PFA/ini if necessary
    - If "*migrate*" was chosen,
      - the ini file copied to /etc/PFA/ini was from an existing check so that the Java configuration should be accurate already.
      - and multiple ini files desired, copy the file from /etc/PFA/ini to the checks' directories and update them if needed.

Fall back considerations by release

- From z/OS 1.11 to z/OS 1.10
  - Falling back → No action
  - Moving forward to z/OS 1.11 after rollback → rerun an install script if
    - you deleted any directory structures for z/OS 1.11 checks (use AIRSHREP.sh to replace data or use AIRSHKP.sh to keep data)
    - you want to delete z/OS 1.10 data (use AIRSHREP.sh)
    - If using AIRSHREP.sh, update ini files for each check

- From z/OS 1.12 to z/OS 1.11 or z/OS 1.10 *without* Coexistence APAR OA33669 (it’s available!)
  - Falling back
    1. If using /etc/PFA/ini, copy that file prior to fall back for simplicity.
    2. After fall back, run install script for release to replace directory structures (AIRSHREP.sh)
    3. Update ini files for Java paths (use data in copied /etc/PFA/ini if the same)
  - Moving forward to z/OS 1.12 after rollback
    1. Run install script (AIRSHREP.sh -- either *new* or *migrate*)
    2. If using /etc/PFA/ini, restore it if needed. Otherwise, recreate individuals and update them.

Fall back considerations by release (continued)

- From z/OS 1.12 to z/OS 1.11 or z/OS 1.10 *with* Coexistence APAR OA33669
  - If using /etc/PFA/ini,
    1. Copy /etc/PFA/ini prior to fall back
    2. After fall back with PTF, restore /etc/PFA/ini and update it (and any individual ini files) if necessary.
    3. Moving forward →
       - No action if directories created with original z/OS 1.12 install were left intact and you want to keep previous release data.
       - Otherwise, rerun AIRSHREP install script (either *new* or *migrate)*
  - If not using /etc/PFA/ini,
    1. After fall back, no action necessary unless your java paths are different (update ini files)
    2. Moving forward →
       - No action if directories created with original z/OS 1.12 install were left intact and you still want separate ini files.
       - Rerun install script if any directories need to be recreated or you want to consolidate ini files (use *new* or *migrate* depending on your preference)
         - If directories need to be recreated and you want to use separate ini files, copy the ini files prior to rerunning install script and then restore them afterwards.

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)
```

PFA Serviceability (continued)

- Reports issued with a PFA check exception which can be seen in SDSF provide additional data for analysis.

- PFA documentation outlines best practices for each check.

- Each check has a *check_name*/data directory in the pfauser’s home directory
  - Files with additional details (described in PFA documentation)
  - Diagnostic files (i.e., logs)

- A “debug” parameter that is check-specific.
  - Additional diagnostic information is generated in the log files when debug is on.
  - This parameter is *not* the debug parameter available via Health Checker because the Health Checker parameter did not apply to all 3 major functions (i.e., collection, modeling, and perform comparison).

PFA Serviceability (continued)

- Individual checks can be quiesced → Check exists, but performs some or no functions
  1. Set COLLECTINACTIVE(0) for the check to stop data collection and modeling
     - Recommended to stop all functions. If COLLECTINACTIVE(1), the check will continue to collect and model
     - f hzsproc,update,check(ibmpfa,pfa_logrec_arrival_rate),parm('collectinactive(0)')
  2. Deactivate the check in IBM Health Checker for z/OS to stop the check from performing comparisons
     - f hzsproc,deactivate,check(ibmpfa,pfa_logrec_arrival_rate)

- Individual checks can be deleted → Does not exist until PFA restart! (not recommended)
  - f hzsproc,delete,check(ibmpfa,pfa_common_storage_usage)

PFA Serviceability (continued)

- Log additional data by turning on PFA’s debug parameter (if requested by service)

- z/OS 1.12 – Serviceability Enhancements
  - Separate log files for each function
    - *systemName*COLLECT.LOG, *systemName*MODEL.LOG, etc.
  - Files and report data copied at exception to EXC_*timestamp* directory for check
    - /u/pfauser/PFA_COMMON_STORAGE_USAGE/EXC_*timestamp*

Basic Troubleshooting

- **AIR010I** PFA CHECK *check_name* WAS UNABLE TO OPEN LOG FILE, ERRNO= 00000081 ERRNOJR=0594003D
  - **Problem:** Didn't run install script at this release level to create new directories
  - **Solution:** Run the install script as described in *z/OS Problem Management*

- **AIR022I** REQUEST TO INVOKE MODELING FAILED FOR CHECK NAME= *check_name.* UNIX SIGNAL RECEIVED= 00000000 EXIT VALUE= 00000006 (or EXIT VALUE= 00000002)
  - **Problem:** It is very likely that PFA cannot find the Java code.
  - **Solution:** Update the ini file(s) to specify paths to PFA's Java code and the Java code for your system
    - **SMP/E install path for PFA's java code:** JAVAPATH= /usr/lpp/bcp
    - **System's executable java code for JNI calls:**
      - PATH= /usr/lpp/java/J6.0/lib/s390/classic:/usr/lpp/java/J6.0/lib/s390
      - LIBPATH= /usr/lpp/java/J6.0/lib/s390:/usr/lpp/java/J6.0/lib/s390/classic:/lib:/usr/lib:

Basic Troubleshooting (continued)

- **SUCCESSFUL MODEL COUNT = 0, but there are no AIR022I messages**
  - **Problem:** Your installation likely had SMP/E install some of PFA's executable code in the z/OS UNIX file system in a path other than the default (path=(/usr/lpp/bcp))
    - Look in the *.LOG file for the check (z/OS 1.12 -- use MODEL.LOG), the following line should exist:
      - AIRHLJVM failed BPX1SPN errno=00000081 errnojr=053B006C
  - **Solution:** Update the PFA procedure your installation uses to specify the path in which you installed the executable code.

```
//PFA      EXEC PGM=AIRAMBGN,REGION=0K,TIME=NOLIMIT,
//         PARM='path=(/usr/lpp/bcp)'
```

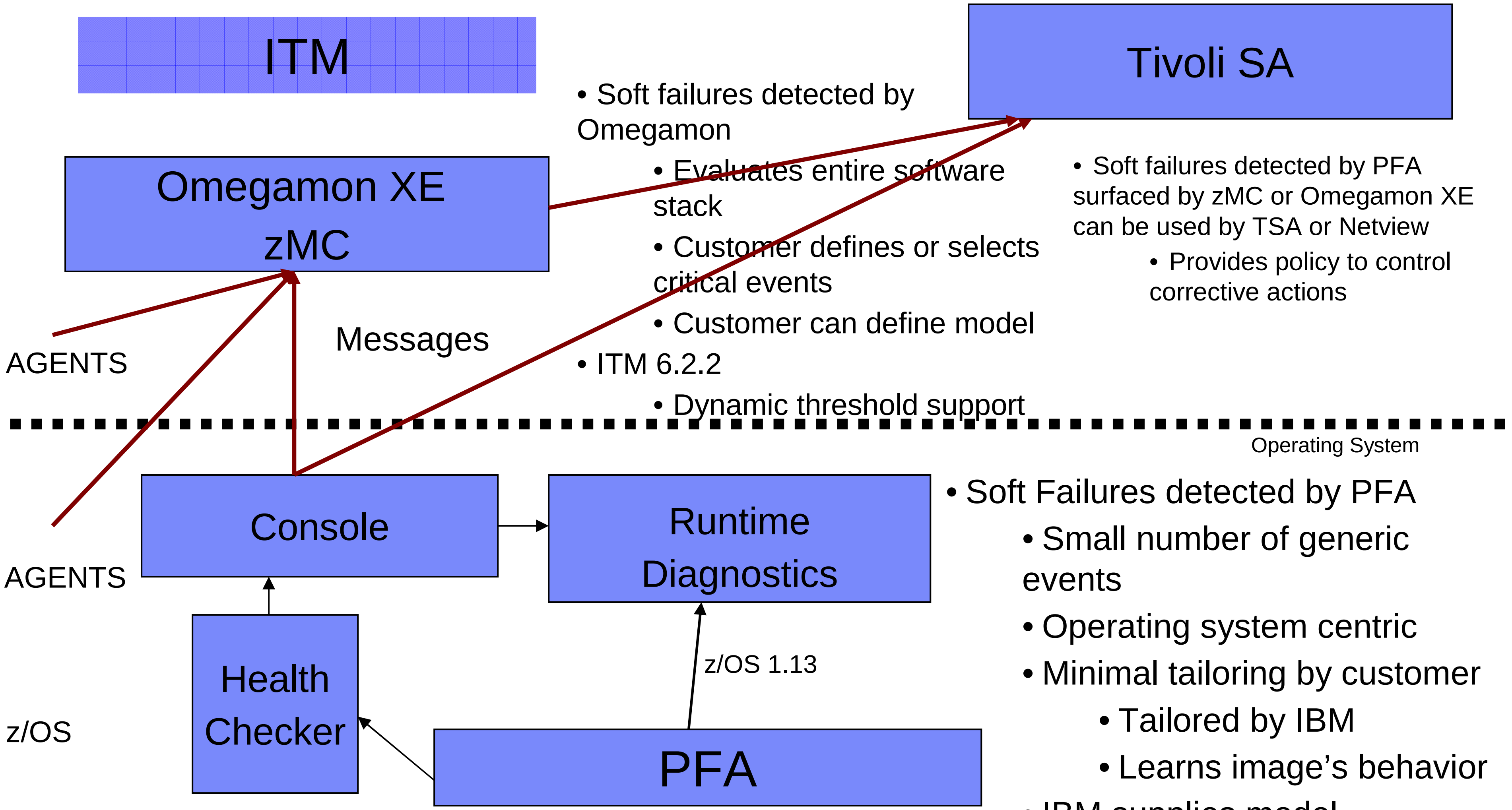
Basic Troubleshooting (continued)

- **The check is issuing too many exceptions!**
  1. Update configuration parameters as described on slides 27 and 28
  2. Add the job to the EXCLUDED_JOBS file as described on slides 29 and 30.
  3. Quiesce or delete the check as described on slide 39.
  4. If tuning parameters or excluding jobs don't help, contact IBM service

- **The CPU usage is too high!**
  - Java processing is zAAP-eligible
  - OA34586 – Delays processing of the module mapping until modeling for the PFA_COMMON_STORAGE_USAGE check
  - z/OS 1.12 has several performance improvements

- **I've run out of z/OS UNIX file system space! How much DASD do I need to allocate for each LPAR?**
  - z/OS 1.10 = 150 cylinders primary; 50 cylinders secondary on a 3390 device
  - z/OS 1.11 and z/OS 1.12 = 200 cylinders primary; 50 cylinders secondary on a 3390 device
  - z/OS 1.13 = 300 cylinders primary; 50 cylinders secondary on a 3390 device

Soft Failure Detection Across Products



- Soft failures detected by Omegamon
  - Evaluates entire software stack
  - Customer defines or selects critical events
  - Customer can define model
- ITM 6.2.2
  - Dynamic threshold support

- Soft failures detected by PFA surfaced by zMC or Omegamon XE can be used by TSA or Netview
  - Provides policy to control corrective actions

- Soft Failures detected by PFA
  - Small number of generic events
  - Operating system centric
  - Minimal tailoring by customer
    - Tailored by IBM
    - Learns image's behavior
  - IBM supplies model

Summary

- PFA uses *historical data* and *machine learning algorithms* to detect and report soft failures *before they can impact your business*

- Focused on *damaged systems* and *resource exhaustion*

- Get more out of PFA by
  - Automating exception messages
  - Using the PFA reports to help diagnose problems
  - Tuning the PFA checks using the configuration parameters and the EXCLUDED_JOBS list if necessary.
  - Using other products to do deep investigation of system or address space problems.

- Consult *z/OS Problem Management* (G325-2564) for more information.

Additional Resources

- Documentation: *z/OS Problem Management* G325-2564
  - z/OS 1.11 -- http://publibz.boulder.ibm.com/epubs/pdf/e0z1k131.pdf
  - z/OS 1.12 -- http://publibz.boulder.ibm.com/epubs/pdf/e0z1k140.pdf
- IEA presentations
  - http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp?topic=/com.ibm.iea.zos/zos/1.11/Availability/V1R11_PFA/player.html
  - http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp?topic=/com.ibm.iea.zos/zos/1.12/Availability/V1R12_Availability_PFA_Enhancements/player.html
- *z/OS Hot Topics* Newsletters
  - #20 (GA22-7501-16) -- *Fix the Future with Predictive Failure Analysis* by Jim Caffrey, Karla Arndt, and Aspen Payton
  - #23 (GA22-7501-19) – *Predict to prevent: Let PFA change your destiny* by Jim Caffrey, Karla Arndt, and Aspen Payton
  - http://www.ibm.com/systems/z/os/zos/bkserv/hot_topics.html
- *IBM Systems Magazine - Mainframe Edition*
  - *A Soft Touch* by Karla Arndt, Jim Caffrey, and Aspen Payton
  - http://www.ibmsystemsmagmainframedigital.com/nxtbooks/ibmsystemsmag/mainframe_20101112/index.php#/48