SHARE August, 2010, Session: 7390

Detecting Soft Failures Using z/OS PFA

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

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

DB2*
DB2 Connect
DB2 Universal Database
e-business logo
GDPS*
Geographically Dispersed Parallel Sysplex
HyperSwap
IBM*
IBM eServer
IBM logo*
Parallel Sysplex*

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Intel is a registered trademark of the Intel Corporation in the United States, other countries or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.

SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Agenda

- Soft failures defined
- How PFA detects and reports soft failures
- The PFA checks
- How to get the most out of PFA
- Installation and Migration
- Summary

Soft Failures: What is a soft failure?

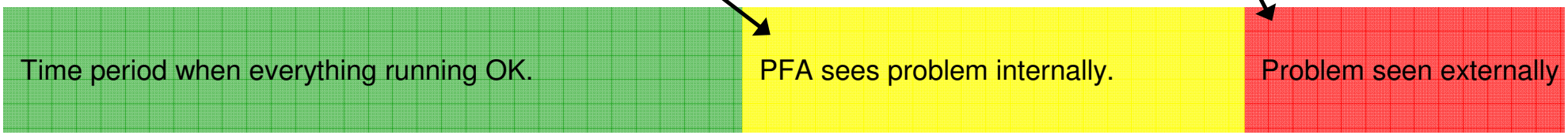
"Your systems don't break. They just stop working and we don't know why."

"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

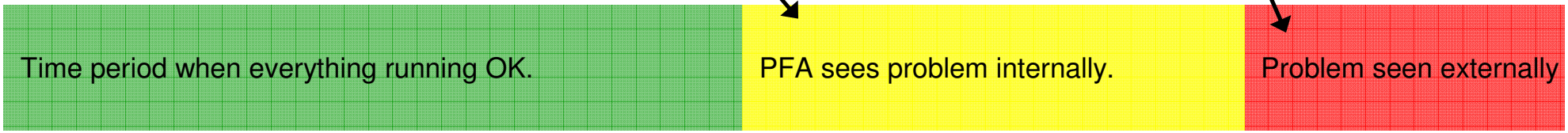
1. A transaction --
   - that has worked for a long time starts to fail, or
   - occasionally (yet, rarely) fails
   - Example – “Reset Password and send link to registered email account”
2. The transaction starts failing more regularly
3. Recovery is successful –
   - Such that the overall, applications 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”



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

Soft Failures – Hypothetical IT Example 2

1. Middleware or application --
   - Has a bug that strands elements which are stored in a data space.
   - Adds additional data spaces as needed to handle volume of requests.
2. After several days or weeks and a high volume of requests,
   - A massive number of frames has been consumed!
3. Request is issued --
   - that causes middleware or application to read through all elements stored in data spaces
4. Application becomes non-responsive
   - because of locks held while searching data spaces



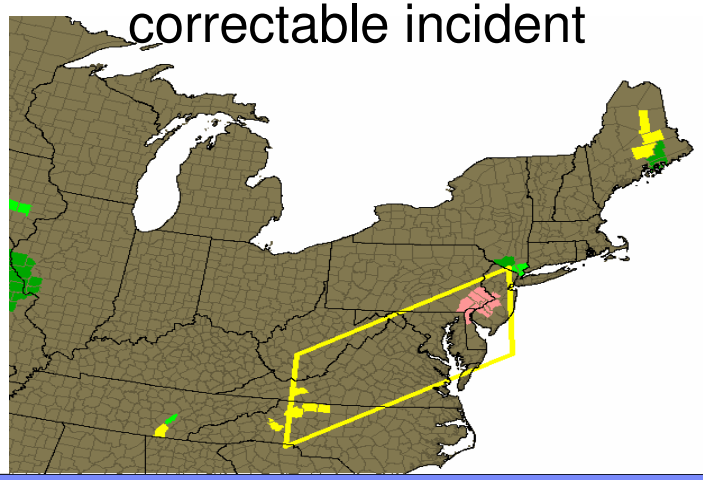
This is a hypothetical problem which is a combination of multiple actual problems

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

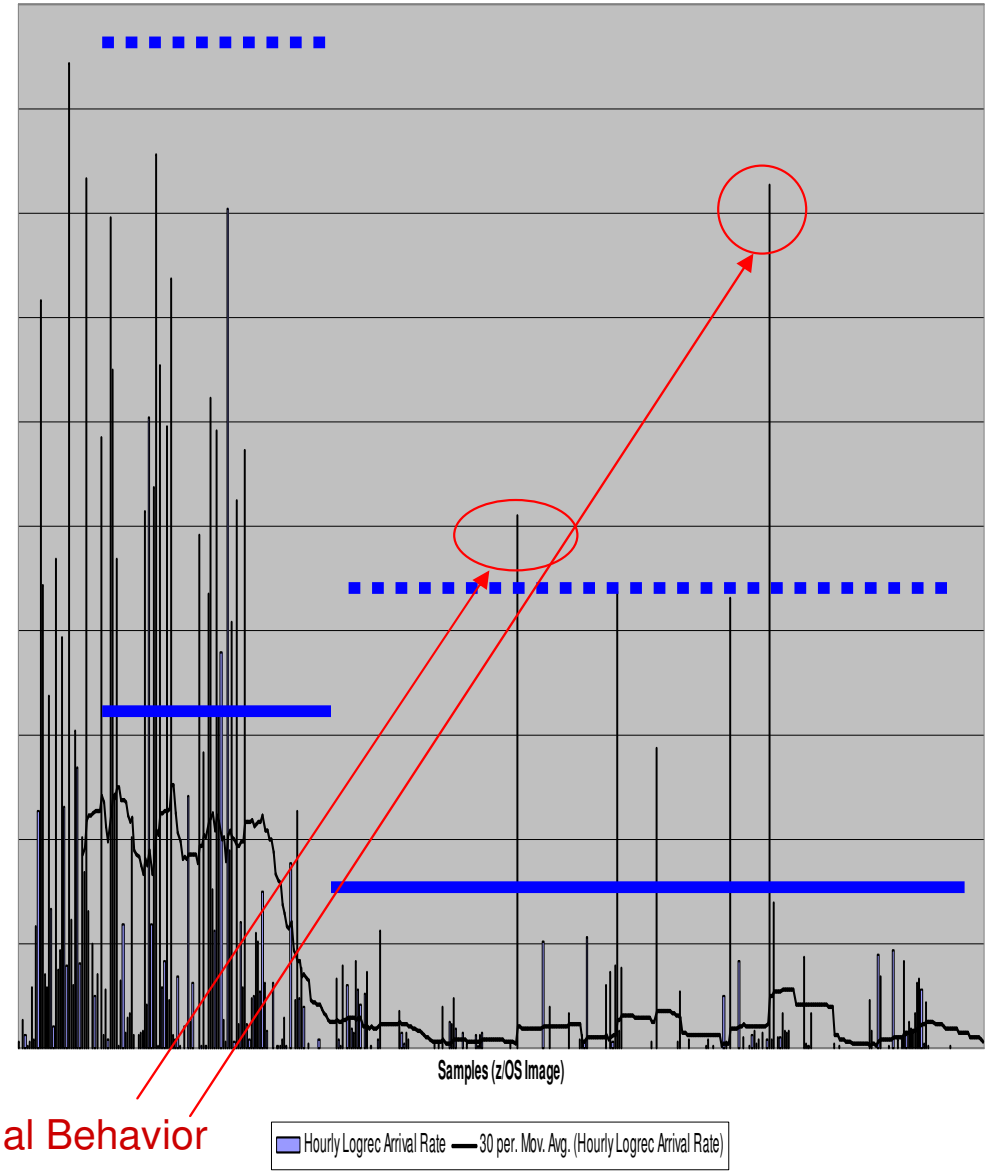
- Predictive failure analysis uses
  - *Historical data*
  - *Machine learning and mathematical modeling*

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

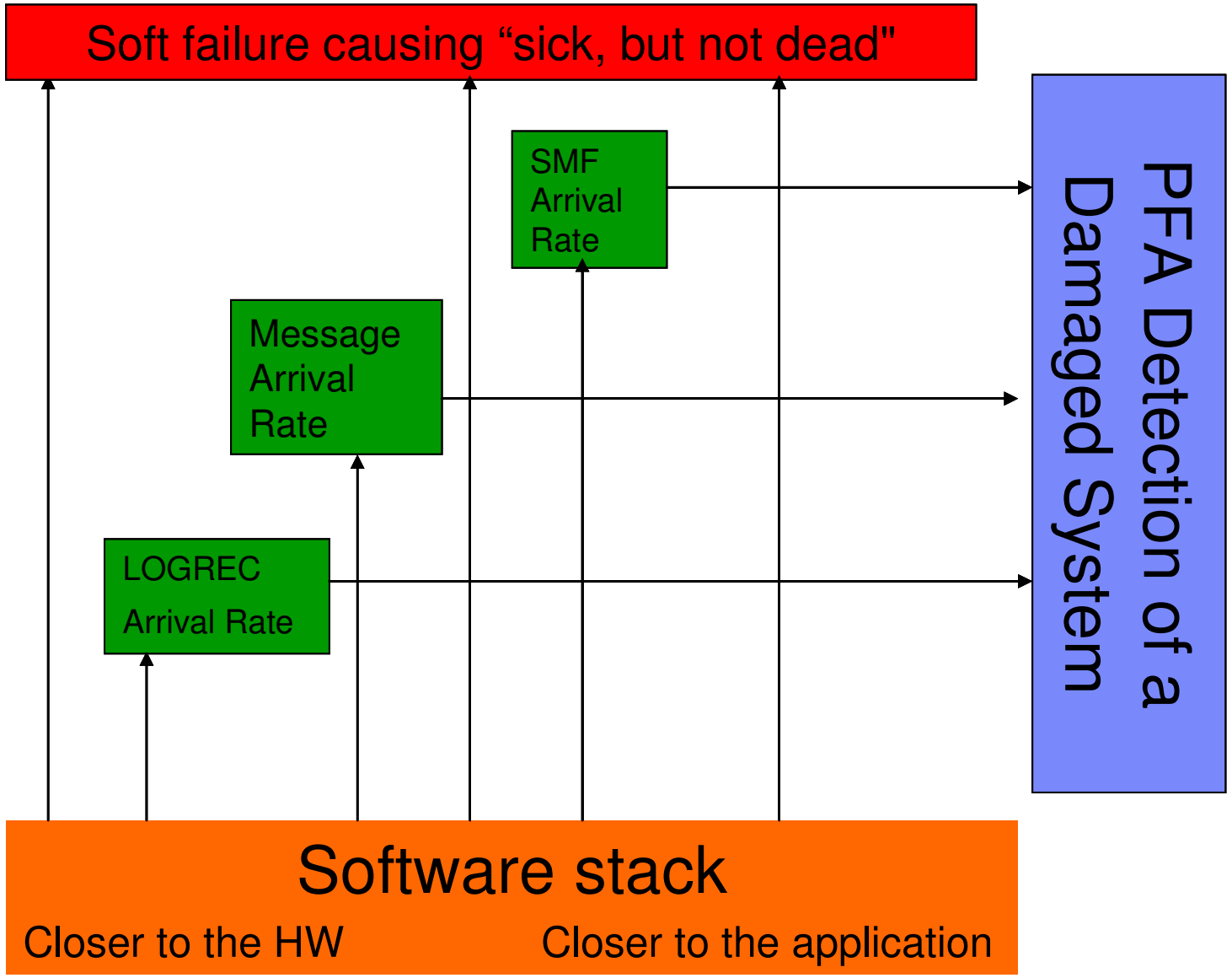


How PFA determines expected or future values

- **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 what to check: Layered approach

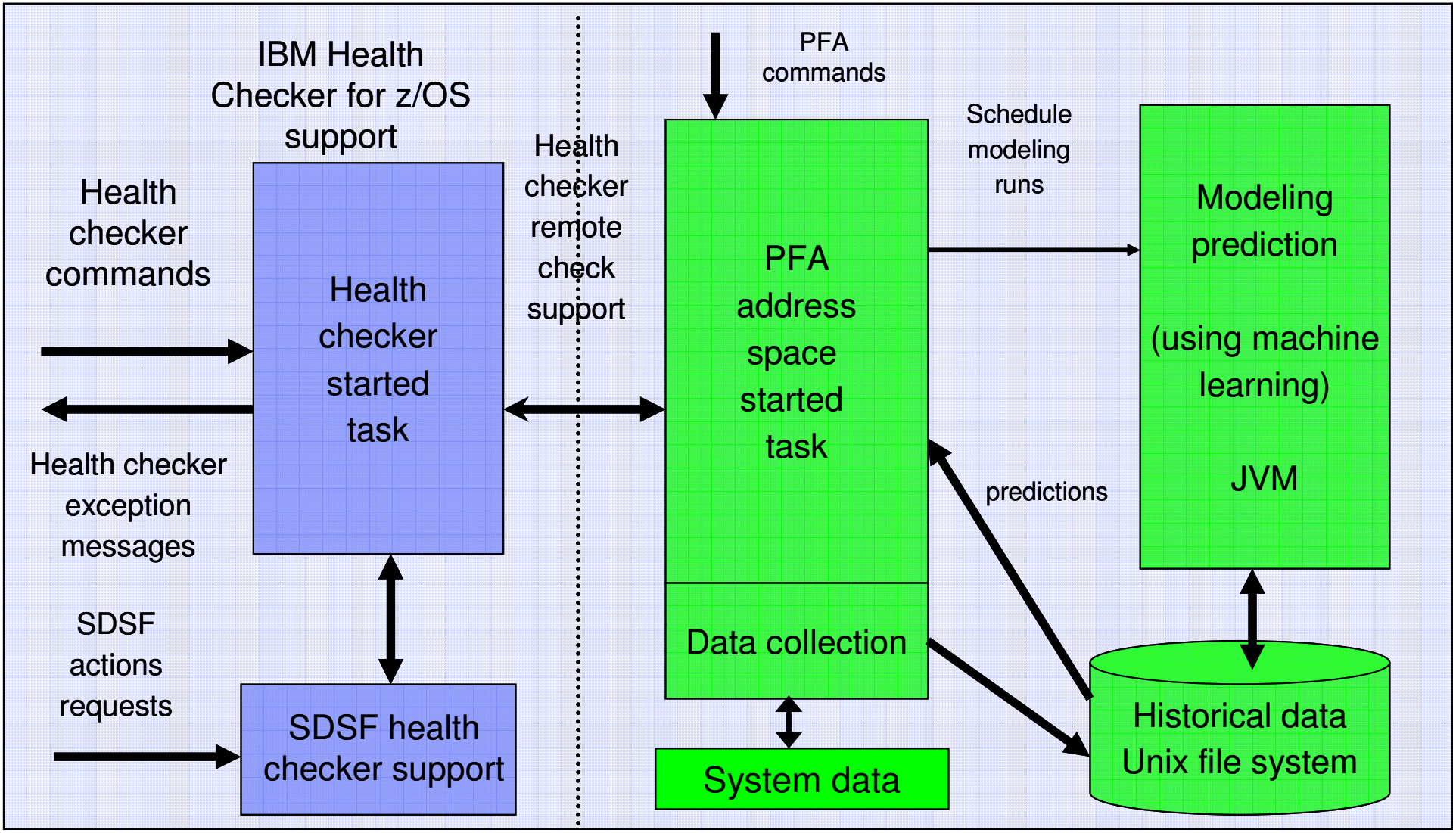


Not to Scale

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, track 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)

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



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*

z/OS 1.10 SPE - PFA_COMMON_STORAGE_USAGE

- Predicts *exhaustion of common storage* by the z/OS image – spikes, leaks, and creeps
- z/OS 1.10 SPE and 1.11 – models two locations
  - CSA + SQA – below the line common storage
  - ESQA + ECSA – above the line common storage
- z/OS 1.12 enhancements
  - Allows 6 categories: CSA, SQA, ECSA, ESQA, CSA+SQA, and ECSA+ESQA
  - Handles expansion of SQA into CSA and ESQA into ECSA
  - Performance improved
- Does not detect
  - *Fragmentation*
  - *Rapid growth* such as on a machine time frame or within a collection interval
  - *Really slow growth* such as less than 750 bytes per second
  - *Usage exceeds a specific threshold* (done by VSM_COMMON_STORAGE_USAGE)
  - *An address space abnormally consuming common storage* without impacting the z/OS image

z/OS 1.10 and 1.11 Common Storage Usage Report

- Top predicted users
  - Printed only if DEBUG(1) or exception occurs to reduce overhead
  - Address spaces whose usage has recently increased the most
  - UNAVAILABLE is displayed for *SYSTEM to further eliminate overhead

```
                    Common Storage Usage Prediction Report
(heading information intentionally omitted)
Below line CSA+SQA (in kilobytes):
   Current usage           :          750
   Future prediction       :          613
   Capacity when predicted:         5212
Above line CSA+SQA (in kilobytes):
   Current usage           :       205555
   Future prediction       :       235408
   Capacity when predicted:       526112
Top predicted users:
   Job             Storage      Current Usage       Predicted Usage
   Name            Location     (in kilobytes)      (in kilobytes)
   ____________    _________    ___________________ ___________________
   CSATST4         ABOVE                     35002               40023
   CSATST3         ABOVE                     32364               33530
   CSATST1         ABOVE                     12456               12478
   ZTTLARM0        ABOVE                      3102                3110
   *SYSTEM*        ABOVE               UNAVAILABLE                 190
```

z/OS 1.12 Common Storage Usage Prediction Report

- Six storage locations
- Asterisk indicates storage location(s) of exception
- Percentage of current to capacity
- If expansion occurs,
  - Message printed on report
  - Comparisons on SQA (or ESQA) stop
  - Expanded usage accounted for in location expanded into for usage and predictions

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.

z/OS 1.10 - PFA_LOGREC_ARRIVAL_RATE

- Predicts a *damaged system* by predicting and comparing LOGREC arrival rates in a collection interval
- Models expected number of LOGRECs for entire system in *time ranges by key*
- Provides a list of jobs that caused the software failures
  - If the LOGREC arrivals are isolated to a job or small number of jobs, an address space is potentially damaged
  - Otherwise, the z/OS image is potentially damaged
- Unable to detect
  - A single critical failure
  - Burst of failures that don't generate software LOGRECs
  - Burst of failures that don't provide usable SDWA with LOGREC
  - Patterns of LOGRECs (issues exception by rate, not pattern)

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
```

LOGREC Arrival Rate Check Enhancements

- New check-specific parameter **EXCEPTIONMIN**
  - Used to reduce false positives on systems that have normally low arrival rates
  - z/OS 1.10 – PTF UA54819
  - z/OS 1.11 – PTF UA54820
  - z/OS 1.12 – in base
- z/OS 1.12 Enhancements
  - Improved time range comparisons across all time ranges to reduce false positives when “normal” spikes occur
  - Data will be reused across an IPL
    - Omits data in hour prior to shutdown and 1 hour after IPL
  - Does not require a 24 hour warm-up period (1 hour of data required)
  - Performance improvements

z/OS 1.11 - PFA_FRAMES_AND_SLOTS_USAGE

- Detects a damaged system by predicting *resource exhaustion* of frames and slots by persistent address spaces
  - Each individual persistent address space is checked
  - Abnormal increase typically indicative of a virtual storage leak

- The usage for each persistent job is the sum of the following:
  - Number of 4K frames used (includes data spaces)
  - Number of AUX slots used

- Unable to detect
  - Small usage increases
  - Fragmentation
  - Rapid growth (on a machine time scale)

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
  - At the most, 14 top users can be printed or displayed in the report
  - 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 ommited)

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
```

z/OS 1.11 - PFA_MESSAGE_ARRIVAL_RATE

- Detects a *damaged system* based on a message arrival rate that is too high
- Messages = WTO and WTOR messages (not BEWTO)
  - Counted prior to possible exclusion by Message Flooding Automation
- Message Arrival Rate = Count of Messages / CPU Utilization
- Four categories compared across time ranges
  - *"Chatty" persistent* address spaces tracked individually
  - *Other persistent* address spaces as a group
  - *Non-persistent* address spaces as a group
  - *Total system rate* ("chatty" + other persistent + non-persistent
- Provides a list of jobs that caused the message burst
  - If the high rate is isolated to a job or small number of jobs, an address space is potentially damaged
  - Otherwise, the z/OS image is usually potentially
- Does not detect abnormal patterns or single critical messages

z/OS 1.12 - PFA_SMF_ARRIVAL_RATE

- Detects a *damaged system* based on an SMF arrival rate that is too high
- SMF Arrivals = All SMF records sent within the PFA collection interval
  - Not by SMF collection intervals
  - Not by record type
- SMF Arrival Rate = Count of SMF Arrivals / CPU Utilization
- Four categories compared across time ranges
  - *"Chatty" persistent* address spaces tracked individually
  - *Other persistent* address spaces as a group
  - *Non-persistent* address spaces as a group
  - *Total system rate* ("chatty" + other persistent + non-persistent)

z/OS 1.12 - PFA_SMF_ARRIVAL_RATE

- Provides a list of jobs that caused the burst of SMF records
  - If the high rate is isolated to a job or small number of jobs, an address space is potentially damaged
  - Otherwise, the z/OS image is potentially damaged
- 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.
- Unable to detect
  - Abnormal SMF record arrival patterns
  - Single SMF record arrivals

Prediction Reports – Message Arrival Rate and SMF Arrival Rate

- Very similar layout for these checks
- 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 for Message Arrival Rate/

```
                  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
...
```

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
  - **THRESHOLD** (Common storage usage check only)
    - A higher value requires a larger current usage and a larger future prediction of common storage before an exception is issued.
  - **STDDEV** (All checks except Common Storage Usage)
    - Most problems causing soft failures show very erratic behavior such that the problem may be *many* standard deviations higher than normal.
      - A lower **STDDEV** value allows an exception to be issued if the actual rate is closer to the expected rate and the predictions across the time ranges are consistent.
      - A higher **STDDEV** allows an exception to be issued if the actual rate is significantly greater than the expected rate even if the predictions across the time ranges are inconsistent.
  - **EXCEPTIONMIN** (LOGREC arrival rate (PTF), Message arrival rate, SMF arrival rate)
    - Use to require a higher number of events before an exception is issued
    - A higher **EXCEPTIONMIN** value requires the current value and for some comparisons, the prediction value, to be higher than this value before an exception will be issued.

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.
    - Most installations should use default.
  - **MODELINT** – All checks -- Number of minutes between models.
    - Most installations should use default.
    - 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 and SMF arrival rate
    - Useful for systems where “chatty” jobs selected have very low rates
    - A higher value requires a persistent job to have a rate higher than this value during the warm-up period before it will be considered “chatty” enough to be tracked individually.
  - **COLLECTINACTIVE** – All checks
    - If on (which is the default) and check not active/enabled in IBM Health Checker for z/OS, PFA continues to collect and model data so that there is no loss of data during this time.
  - **DEBUG** – All checks -- Use only if having a problem with PFA (service requests it)
    - More information written to PFA LOG files
    - This parameter is *not* the debug parameter available via IBM Health Checker for z/OS because that parameter did not apply to all PFA functions

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 spikey in their behavior and cause message arrival rate exceptions.

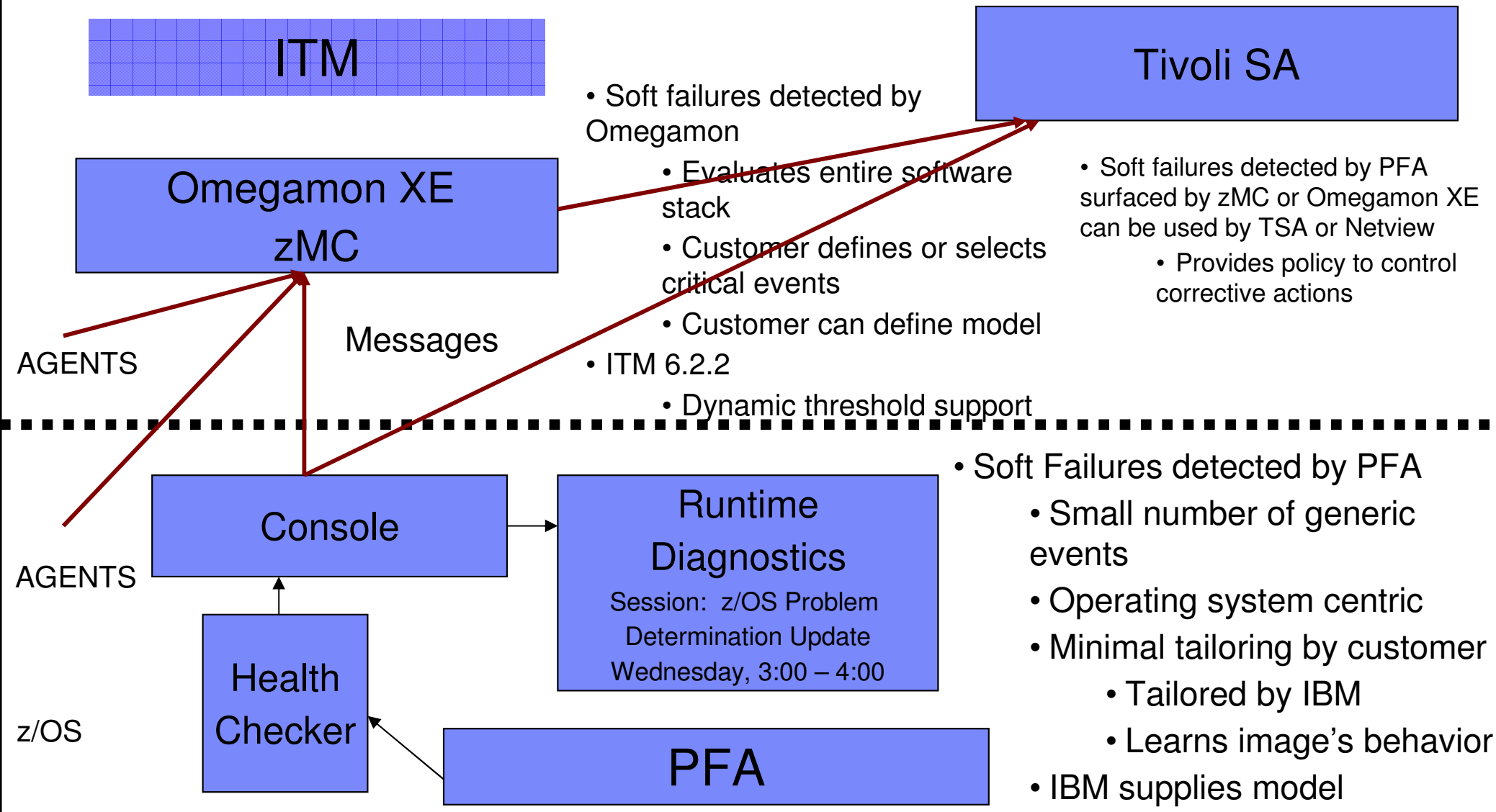
How to Get the Most Out of PFA (continued)

- 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 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)



PFA 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)
```

Dependencies

- Software Dependencies
  - IBM Health Checker for z/OS
  - z/OS UNIX file system
  - Java (31-bit only)
    - 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

- z/OS 1.10 – SPE – get the latest PTFs. Also need the latest MAPMVS PTFs.
  - OA29259/UA52928
  - OA30939/UA51988
  - OA30174/UA51953
  - OA28540/UA47850
  - OA30106/UA50305

- z/OS 1.11 and following releases – shipped with z/OS as part of BCP

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
     - Use "*migrate*" if you want to retain data from previous releases (recommended)
     - For z/OS 1.12, 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 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 (manual steps until co-existence APAR 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 future co-existence APAR
  - 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.

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

- **SUCCESSFUL MODEL COUNT = 0, but there are no AIR022I messages**
  - **Problem:** Your installation likely installed PFA into a different location than the default.
    - 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 have the correct path.

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

PFA Serviceability

- Modify PFA command to display status
- 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 which contains
  - Files with additional details (described in PFA documentation)
  - Diagnostic files (i.e., logs)

PFA Serviceability (continued)

- Individual checks can be deleted or deactivated via IBM Health Checker for z/OS modify command
  - For example, f hzsproc,delete,check(ibmpfa,PFA_COMMON_STORAGE_USAGE)

- Individual checks can be quiesced
  1. Set COLLECTINACTIVE(0) for the check to stop data collection and modeling
     - f hzsproc,update,check(ibmpfa,pfa_logrec_arrival_rate),parm('collectinactive(0)')
  2. Deactivate the check in IBM Health Checker for z/OS
     - f hzsproc,deactivate,check(ibmpfa,pfa_logrec_arrival_rate)

- 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*

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.

Appendix

- 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 presentation (contains information through z/OS 1.11)
  - *Predictive Failure Analysis Overview*
  - http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp?topic=/com.ibm.iea.zos/zos/1.11/Availability/V1R11_PFA/player.html
- SHARE presentation by Jim Caffrey
  - *IBM Experience Building Remote Health Checker Checks*
  - http://ew.share.org/proceedingmod/abstract.cfm?abstract_id=19167
- 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