Predictive Failure Analysis – Customer Experience

Sam Knutson, GEICO

Detecting and Diagnosing Problems when z/OS "Thinks" it is Running Okay

Bob Abrams, IBM

Thursday, August 11, 2011
Session Number: 09807

z/OS Predictive Failure Analysis (PFA)

- **Health Checker runs checks supplied by IBM, vendors or written by you to detect existing or potential problems primarily with variation from understand successful practices**
- **GEICO has been running Health Checker for z/OS since the prototype and the first implementation in z/OS in 2005 in z/OS 1.7 base was optional download for z/OS 1.4 to 1.6**
- ***We run Health Checker all the time from IPL. You should too!***
- **PFA will attempt to detect problems which require analysis of history and current trends to detect. PFA can help you avoid soft failures.**
- **PFA was introduced with z/OS 1.10**
- **We run PFA all the time from IPL**

z/OS Predictive Failure Analysis (PFA)

- PFA checks have been delivered in each z/OS release starting with 1.10 currently for me on 1.12 I run these
  - PFA_COMMON_STORAGE_USAGE
  - PFA_FRAMES_AND_SLOTS_USAGE
  - PFA_LOGREC_ARRIVAL_RATE
  - PFA_MESSAGE_ARRIVAL_RATE
  - PFA_SMF_ARRIVAL_RATE
- IBM plans to extend PFA over time PFA so PFA will be useful today but like Health Checker your investment will be returned with interest in the future
- PFA uses the Health Checker for z/OS infrastructure but PFA checks are different and need to be treated differently
- *"Prediction is very difficult, especially about the future" Niels Bohr*

SHARE in Orlando 2011

z/OS Predictive Failure Analysis (PFA)

- **PFA is constructed largely in Java and the majority of the CPU used by PFA is eligible to run on zAAP speciality engines**
- **APAR OA27495 NEW FUNCTION - ZAAP ON ZIIP ENHANCEMENT**
  - **ZZ=YES in IEASYSxx**
  - **zAAP on zIIP does not change in any way what is eligible for specialty processing just allows you to consolidate to have only one pool of specialty engines. If you have z/OS 1.11 or the PTF for APAR OA27495 applied to 1.10 or 1.9 AND you have only zIIP engines but NO zAAP engines AND you set a new operating system parameter ZZ=YES then zAAP eligible workloads will run on the zIIP alongside zIIP eligible workloads.**
- ***Using zAAP on zIIP is like buying colorsafe Cheer for your washing machine and being able to mix the lights and the darks☺***

4

SHARE in Orlando 2011

Dealing with Health Checker Exceptions

- Resolution of a Health Checker exception frequently requires other subject matter experts cooperation
- Understanding of PFA exceptions and tuning has required recurring assistance from IBM Level-2 to adjust the proper “knob” correctly
- Don’t break existing system. Health Checker is reporting a potential problem don’t make an immediate visible one trying to hurry in a change.
- Avoid persistent outstanding exceptions it will lead to 'cry wolf syndrome' and Health Checks being disregarded.
- Running Health Checker and PFA and not looking at the output does not help you☹

SHARE in Orlando 2011

Identifying Exceptions

- Automate notifications from Health Checker!
- We trap HZS* WTO using CA-OPSMVS
  - MLWTO avoid processing same event more than once
  - Consider “normal” error HZS0011E READING THE HZSPDATA DATA SET may want to just process HZS0001I, HZS0002E, HZS0003E, HZS0004I
  - EVENTUAL exceptions are emailed
  - CRITICAL exceptions are sent by email and pager
- Most well known SPOOL browsers and MVS monitors including IOF, SDSF, SYSVIEW, TMON, provide on-line interfaces to view and manage exceptions

SHARE in Orlando 2011

Sending PFA documentation to IBM

- IBM Level-2 will typically need to see the data that is causing PFA to raise an exception
- Package the directory for the exception using the PAX command in batch
- PFA component ID is 5752SCPFA useful to know when opening a PMR

```
//UNPAX    EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSEXEC  DD DSN=SYS1.SBPXEXEC,DISP=SHR
//SYSTSIN  DD *
  oshell cd /u/pfauser/PFA_SMF_ARRIVAL_RATE          ;        +
     pax -wvzf /tmp/PMR93649.499.000.BTST.AUG11.PFASAR.pax .
/*
```

7

SHARE in Orlando 2011

Sending PFA documentation to IBM

- **FTP documentation to IBM this is an example using batch directly from z/OS but you may have different procedures**

```
//FTP   EXEC PGM=FTP,PARM='(EXIT'
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
testcase.software.ibm.com (timeout 720 exit=12
anonymous
you@yoursite.com
cd /toibm/mvs
lcd /tmp
SENDSITE
binary
put PMR93649.499.000.BTST.AUG11.PFASAR.pax
quit
/*
```

PFA : Installation

- RACF userid with OMVS segment we made it unique as doc specified and it’s home directory is where PFA will store historical information
- Initially just added /pfauser to our primary USER file system. Most groups are segregated but odd users like ekmserv and others we have just created in /u. Once PFA had been running for a while saw zFS filling up and needing to be grown i.e. IOEZ00078E zFS aggregate OMVS.BTST.U.ZFS exceeds 95% full (2140/2250)
- Historical collection creates LOTS of small files which add up over time so a separate USS filesystem is a good idea

SHARE in Orlando 2011

PFA : Installation

- **On each system BEFORE starting PFA you need to copy the PROC from SYS1.SAMPLIB(AIRPROC) to your execution PROC. I used SYS1.PROCLIB(PFA)**
- **On each system BEFORE starting PFA you need to use the supplied script to create a directory structure**

```
$ cd /u/pfauser
$ su
# pwd
/u/pfauser
# sh /usr/lpp/bcp/AIRSHREP.sh
All existing data files and directories removed.
 Successfully created the Common Storage Usage Check Directory Structure.
 Successfully created the Logrec Arrival Rate Check Directory Structure.
 Successfully created and populated ini file for the Common Storage Usage Check.
 Successfully created and populated ini file for the Logrec Arrival Rate Check.
```


SHARE in Orlando 2011

PFA : Installation zFS

- Allocated zFS and mounted for PFA. Must have RACF access to PFSCTL or UID 0 to format ZFS
- zSeries File System (zFS) PARMLIB(IOEPRM00) has aggrfull(90,5) and aggrgrow=on
- IDCAMS
  - DEFINE CLUSTER (NAME(OMVS.ASYS.PFA.ZFS) -
    LINEAR CYL(500,500) SHAREOPTIONS(3))
- IOEAGFMT
  - PARM=('-aggregate OMVS.ASYS.PFA.ZFS -compat')
- MOUNT FILESYSTEM('OMVS.ASYS.PFA.ZFS') MOUNTPOINT('/u/pfauser') TYPE(ZFS) MODE(RDWR)
- Don't forget to update SYS1.PARMLIB(BPXPRMxx)

```
MOUNT FILESYSTEM('OMVS.&SYSNAME..PFA.ZFS')
     MOUNTPOINT('/u/pfauser')   /* ZFS for /u/pfauser  */
       TYPE(ZFS)                /* Filesystem type ZFS   */
       MODE(RDWR)               /* Mounted  or read/write */
```

z/OS Predictive Failure Analysis (PFA)

- PFA checks have had to be customized via HZSPRMxx updates to avoid spurious notifications in many cases relief later provided by APARs

```
/* Predictive Failure Analysis Checks    */
ADDREPLACE POLICY STMT(PFA2) UPDATE
CHECK(IBMPFA,PFA_MESSAGE_ARRIVAL_RATE)
PARM='DEBUG(0) STDDEV(16) COLLECTINT(15) MODELINT(360)
COLLECTINACTIVE(1) TRACKEDMIN(0) EXCEPTIONMIN(5)'
REASON('less sensitive to avoid spurious trips')
DATE(20110221)

ADDREPLACE POLICY STMT(PFA4) UPDATE
CHECK(IBMPFA,PFA_MESSAGE_ARRIVAL_RATE)
PARM='DEBUG(0) STDDEV(3) COLLECTINT(15) MODELINT(720)
COLLECTINACTIVE(1) TRACKEDMIN(0) EXCEPTIONMIN(10)'
REASON('less sensitive to avoid spurious trips')
DATE(20110728)
```

SHARE in Orlando 2011

PFA APARs of interest

- OA31644 LOGREC ARRIVAL RATE SPURIOUS AIRH110E MESSAGE
- OA34655 PFA FRAMES AND SLOTS USAGE SPURIOUS. PREDICTIVE FAILURE ANALYSIS
- OA34586 HIGH CPU IN PFA AFTER PTF UA50081 FOR APAR OA29963 APPLIED
- OA35820 PREDICTIVE FAILURE ANALYSIS. Abend0C4 RC4 AIRHMEMA.x'FF' DECIMAL 255 PERSISTENT JOBS CAUSES A LOOP IN PFA THAT OVERLAYS STORAGE
- OA36462 ABEND0C4 IN AIRHMEXC
- OA36837 PREDICTIVE FAILURE ANALYSIS (PFA) TREATS TSO USERID AS PERSISTENT JOBS IN PFA_FRAMES_AND_SLOTS_USAGE CHECK

AIRH109E A problem with common storage usage

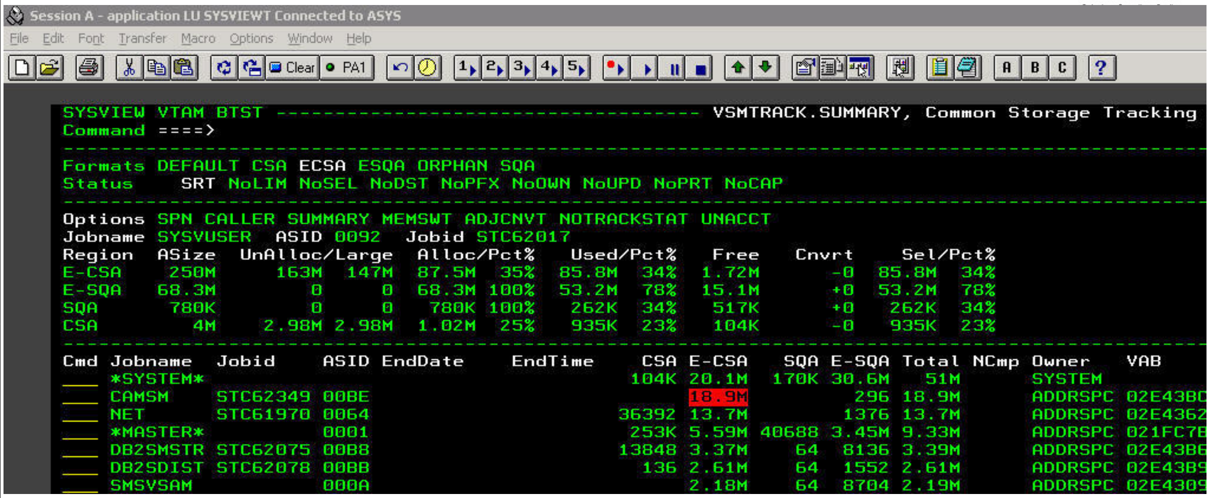
-----Original Message-----
From: Enterprise Event Management
Sent: Friday, July 24, 2009 3:51 PM
To: Knutson, Sam
Subject: z/OS Health Checker BTST 24 Jul 2009 HZS0004I CHECK|IBMPFA,PFA_COMMON_STO

z/OS Health Checker BTST 24 Jul 2009 HZS0004I CHECK|IBMPFA,PFA_COMMON_STORAGE_USAGE|: AIRH109E A problem with common storage usage |CSA and SQA| above the line was predicted to occur by 07/24/2009 21:51:17. The current usage is higher than expected based on an evaluation of the total capacity plus the threshold, the current usage, the current prediction, and the future prediction modeled at 07/24/2009 15:51:17.

```
----------------------------------------
* EMAIL NOTIFICATION SENT FROM OPS/MVS *
*                                      *
* SYSTEM :BTST                         *
*   RULE :MSGACTN.HZSINFO              *
*  GROUP :SKNUTSON                     *
*   DATE :24 Jul 2009  TIME: 15:51:17  *
----------------------------------------
```


SHARE in Orlando 2011

AIRH109E A problem with common storage usage



- Good exception system is still healthy only 34% used on ECSA but a task was out of profile. Testing on test Sysplex had found new bug.
- Recycle of started task temporarily resolved the problem and CA-CCI RO10195 and CA-DATACOM PTFs RO10065 were created

SHARE in Orlando 2011

PFA_LOGREC_ARRIVAL_RATE

CHECK(IBMPFA,PFA_LOGREC_ARRIVAL_RATE)
START TIME: 08/18/2009 12:03:47.499169
CHECK DATE: 20080330 CHECK SEVERITY: MEDIUM
CHECK PARM: DEBUG(0) STDDEV(2) COLLECTINT(60) MODELINT(360) COLLECTINACTIVE(1)

* Medium Severity Exception *

The LOGREC entry arrival rate is higher than expected and could cause a system problem by
08/18/2009 17:49:35.
When the LOGREC entry rate is higher than expected, it can indicate recurring failures on the system which can eventually lead to system hang. The prediction used in the comparison was modeled at 08/18/2009 11:49:35.

SHARE in Orlando 2011

PFA_LOGREC_ARRIVAL_RATE

```
LOGREC Arrival Rate Prediction Report
Last successful model time      :  08/18/2009 11:49:35
Next model time                 :  08/18/2009 17:49:35
Model interval                  :  360
Last successful collection time:  08/18/2009 11:52:37
Next collection time            :  08/18/2009 12:52:37
Collection interval             :  60
                                 Key 0      Key 1-7     Key 8-15
                              __________  __________  __________
Arrivals in last
   collection interval:           77         364          8
Predicted rates based on...
     1 hour of data:                9          0           5
Jobs having LOGREC arrivals in last collection interval:
   Job Name         ASID         Arrivals
   ________         ____         ________
   DSMD0D           0205            438
   NONE-FRR         00D0              1
   U38T20           0344              1
```


SHARE in Orlando 2011

PFA_LOGREC_ARRIVAL_RATE

- ABEND rate can vary widely and quickly
- Out of the box check tends to trip spuriously
- Remember this is saying “look at me” not I am broken
- Adjusted STDEV temporarily
- APAR OA31644 LOGREC ARRIVAL RATE SPURIOUS AIRH110E MESSAGE was the real resolution to spurious trips from this check and has been effective

PFA Customer Experience Summary

- **I run Health Checker and PFA and you should too**
- **There are many opportunities for new and improved PFA checks. I would like to see PFA doing global monitoring of private virtual storage use for persistent address spaces**
- **Please work with IBM to tune your checks to work accurately and improve PFA for everyone**
- **z/OS 1.12 contains some useful enhancements to PFA which will not be rolled back to previous releases**
- **The goal of automatically configured proactive detection failures or looming problems is laudable**
- **This is a hard problem and I am not sure when PFA exceptions will be sufficiently accurate to treat them as operational alerts**
- ***HAL: I've just picked up a fault in the AE35 unit. It's going to go 100% failure in 72 hours. 2001: A Space Odyssey (1968)***

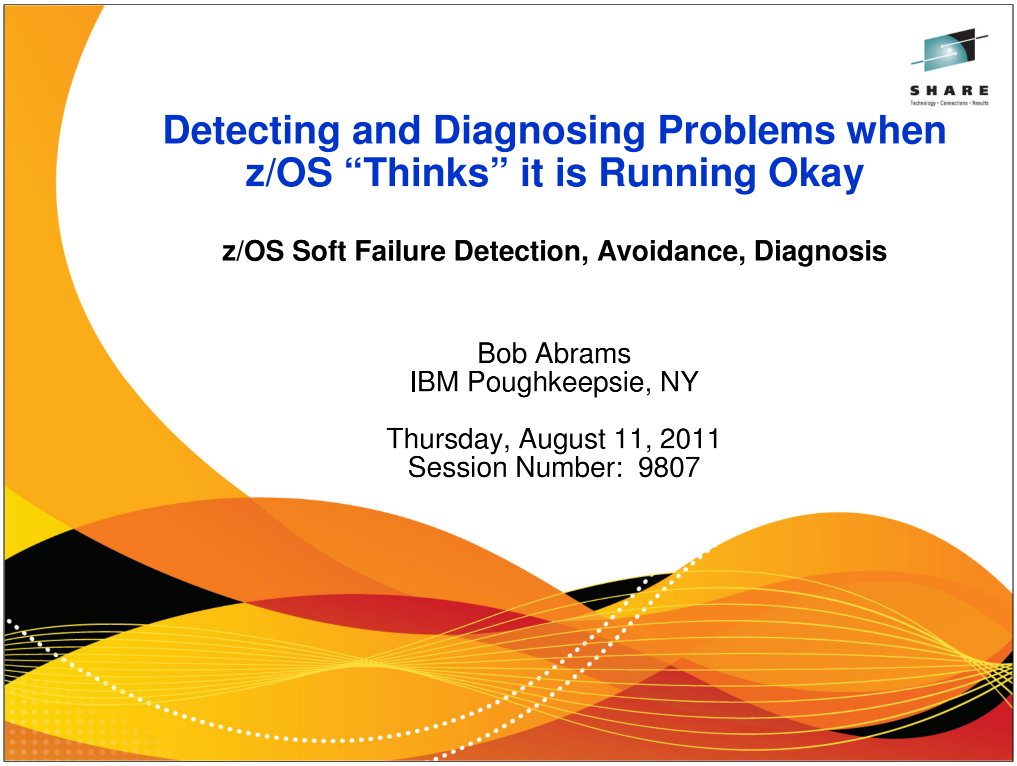
PFA : References

- Session 2866 IBM Experience Building Remote Checks for the IBM HEALTH CHECKER for z/OS presented at SHARE in Austin by James Caffrey
- Manual: z/OS Problem Management G325-2564-04 April 2009 not included in -02 on April 2009 z/OS V1R10 and Software Products DVD Collection need to obtain this from the web or a later DVD
- WSC Flash WP101454 by Riaz Ahmad can be found on the web, www.ibm.com/support/techdocs under the category of "White Papers."
- Session 2858 Health Checker: User Experience Beyond Installation presented at SHARE in Tampa by Sam Knutson and Dave Danner
- Session 2208 Bit Bucket x'26' presented at SHARE in Denver

SHARE in Orlando 2011

Acknowledgements (Knowing or unKnowing)

- James Caffrey (IBM)
- Vincent Bonanno (IBM)
- Bob Abrams (IBM)
- Karla Arndt (IBM)
- Ashvin Desai (IBM)
- Keith Martens (GEICO)



Welcome to this SHARE presentation on z/OS soft failure detection, avoidance and diagnosis.

The goal of this presentation is to show how many parts of the z/OS-based product stack work together to detect, avoid and diagnose Soft Failures.

Agenda:
Soft Failures: Detection, Prevention, Diagnosis

SHARE Technology • Connections • Results

- Soft Failure detection & avoidance
  - Provided at multiple levels of the stack
  - Types of problems handled by each type of soft failure detection
- Soft Failure Detect/Diagnose/Avoid Capabilities in z/OS
  - **Detection:** z/OS Components
  - **Avoidance:** Health Checks
  - **Detection & diagnosis:** PFA, Runtime Diagnostics
  - **Business Application view:** Systems Management products
- Lessons learned on reducing impact of soft failures

All elements work together for an integrated IBM solution approach

23

SHARE in Orlando 2011

Up to now, much of what you have heard about soft failure detection has been in the context of Predictive Failure Analysis (PFA).

PFA does a great job predicting problems related to “soft failures”, such as growth in resource usage and surfacing rapid error indicators representative of damaged system environments.

However, Soft Failure detection goes beyond PFA … soft failures have been, and continue to be, detected by base system components; many as a result of problems identified by customers.

Furthermore, many types of soft failures can be avoided by running health checks on a regular basis.

IBM has actually been working on this problem for a long time, just didn’t recognize the unifying theme until recently

PFA covers situations where it is difficult for a component to understand its impact to the system, and where there is no simple component fix.

And Runtime Diagnostics provides an efficient way to diagnose component issues that may result from soft failures, leading to a system that seems “sick but not dead”.

This presentation puts these different types of error detection in perspective and shows that IBM has an integrated solution approach where all of the solution elements work together to allow you to identify and react to these types of issues before they impact your system and cause an outage.

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

Symptoms of a Soft Failure

- 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

Manifested as

- Stalled / hung processes
  - Single system, sysplex members
  - Sympathy Sickness
- Resource Contention
- Storage growth
- CF, CDS growth
- I/O issues (channel paths, response time)
- Repetitive errors
- Queue growth
- Configuration
  - SPOF, thresholds,
    cache structure size,
    not enabling new features

24

SHARE in Orlando 2011

You can't put your finger on a specific symptom; the system looks alive, but performing poorly.

There are three general categories of software detected system failures: masked failure, hard failure, and failure caused by abnormal behavior. A masked failure is a software detected system failure which is detected by the software and corrected by the software. A hard failure is when the software fails completely, quickly and cleanly. For example, a hard failure occurs when an operating system kills a process.

A system failure caused by abnormal behavior is defined as unexpected, unusual, or abnormal behavior which causes the software solution to not provide the service requested. This abnormal behavior of the software combined with events that usually do not generate failures produce secondary effects that may eventually result in a system failure. These types of failures are known as soft failures.

Customers have told us that these soft failures are a small percentage of the problems when compared to masked failures and hard failures, but they cause most of the business impact.

They are hard to diagnose due to the fact that the failure likely does not occur in the address space causing the problem, but more likely occurs in another address space. This sympathy sickness has been observed when either hard failures or abnormal behavior generates a system failure which could not be isolated to a failing component or subcomponent. Failures caused by abnormal behavior often generate sympathy sickness where the problem escalates from a minor problem to the point that the service eventually stops working. Because they are difficult to detect, are very unique, can be triggered anywhere in either software or hardware, and occur infrequently, failure isolation is very difficult.

Hard failures are deterministic in nature. However, a failure caused by soft failures is difficult to recognize within the component and are probabilistic and depend on secondary effects to cause observable damage.

Typically, soft failures occur in 4 generic areas:

•Exhaustion of shared resources

•Recurring or recursive failures

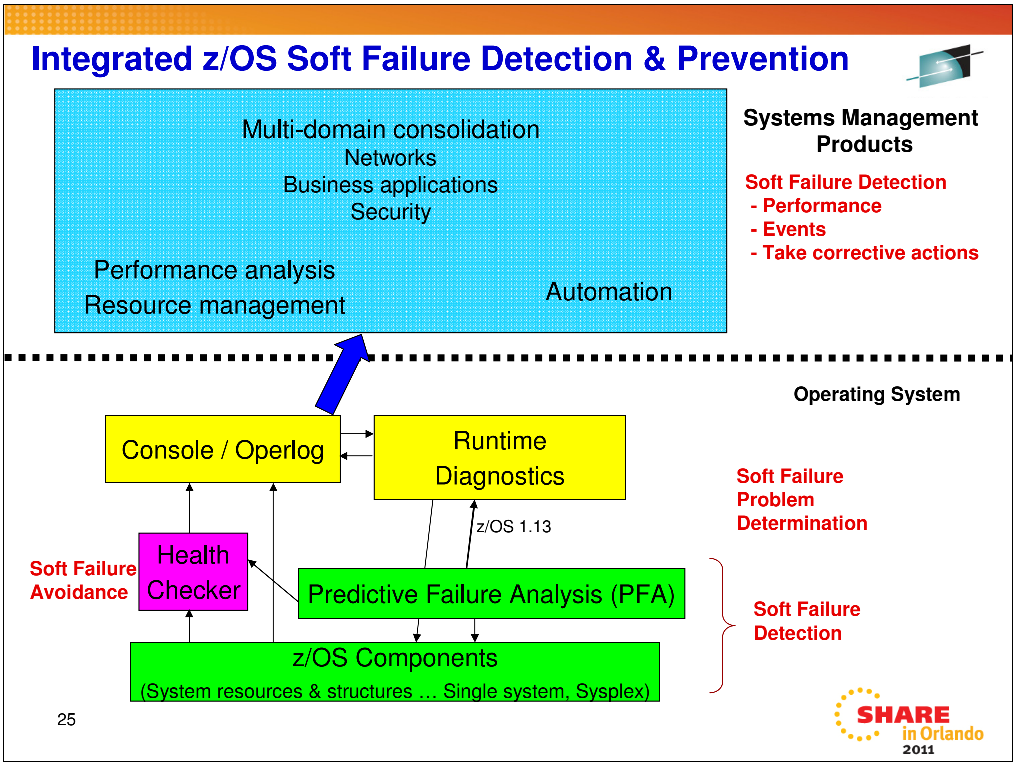
•Serialization problems such as classic deadlocks and priority inversions

•Unexpected state transition

Soft Failures ("sick but not dead" events) can vary from day to day, or be dependent on workloads, or can be a result of conditions that build up over time, and, as a result, often cannot be detected by preset or pre-programmed thresholds (as those often found in monitoring products).

Soft failures tend to manifest themselves in different ways. Configuration issues often appear as:

- Single points of failure / Cache structures too small / Log stream thresholds / Sufficient space for root file system / Not enabling newer features



This chart shows the collection of components that work together to deliver a solution focused on detecting, avoiding & diagnosing soft failures. We'll discuss each area in sections of this presentation.

•z/OS components

•Health Checks

•PFA & RTD

•Systems management products

PFA is built into the operating system. It is looking for a small number of generic events that could cause a soft failure. It is not looking for events or soft failures in specific address spaces unless they could cause a system crash or hang. PFA is operating system centric in that it works on z/OS. It learns the behavior of the individual behavior and creates predictions for that behavior. It detects soft failures by using complex algorithms imbedded in the component to compare the model behavior for that particular system to the current behavior. PFA is built using remote health check support and provides the information for the soft failure via IBM Health Checker for z/OS which issues the exception to the console (if so configured) as well as the exception and the report data to the health check output in SDSF. From the messages provided by PFA via the health checker support to the console, other products can be used to further analyze the situation.

When PFA detects a system or job with low resource usage, it invokes RTD on behalf of that system or job to determine whether there are factors that could be causing a problem.

In addition, the installation can invoke RTD directly via operator command. In either case, the output of PFA & RTD can be automated on to take further action. In addition, most z/OS management products offer performance analysis and various forms of resource management.

Later in this presentation, we discuss a set of Tivoli systems management products offered by IBM.

The OMEGAMON XE for Management Console will see all health check alerts including the PFA ones. You can build a situation that will alert you if a PFA check is raised and forward that event to other Tivoli event management products (like OMNIBUS or Tivoli Event Console). You can also use Runtime Diagnostics to further analyze PFA results. Runtime Diagnostics provides detailed analysis either the entire system or address space looking for soft failures. It uses lists of messages identified by specific components to review critical messages in the joblog. It also stores information about enqueues to analyze contention, evaluates local lock conditions, and queries a job that has a task in a TCB loop.

Runtime Diagnostics is designed to be used whenever the help desk or operations reports a problem on the system. You should use Runtime Diagnostics to get ready before calling service. Runtime Diagnostics should also be used to help identify the address space causing a PFA exception. PFA identifies a list of potential villains. Runtime Diagnostics can be used to further analyze that address space to detect if it is causing a real problem and to identify what action to take to resolve the problem.

Runtime Diagnostics and Omegamon XE provide additional lower level details than are provided by PFA. The documentation for Runtime Diagnostics as well as PFA can be found in the z/OS Problem Management guide. Starting with z/OS 1.13, PFA calls Runtime Diagnostics to verify abnormally low conditions.

Using customer policy, TSA (via Netview) can detect that a health checker message for PFA exceptions were issued and drive actions.

Some general considerations …

- The key to reducing the impact of soft failures is
  - Avoid them using z/OS Health Checker
  - Enable system checking where possible
  - Automate alerts
    - Display, take action
- z/OS can survive / recover from most soft failures
  - Take advantage of what the base operating system has to offer
    - Soft failure detection across many z/OS components
    - Start Health Checker, PFA, RTD (R13) at IPL (e.g., COMMNDxx)

26

SHARE in Orlando 2011

The key to reducing the impact of soft failures is

•Avoid them using health checker

•Enable system checking where possible

•Alerts can be acted upon. You can display them, automate on them and take action to address the detected problem(s)

It is true that z/OS can survive or recover from many forms of soft failures, as demonstrated by the different types of component checking.

•But you need to take advantage of what the base operating system has to offer; this requires enabling some function parameters, and starting health checker, PFA and RTD during the IPL, such as in the COMMNDxx parmlib member

Most metrics that are used to detect soft failures are very time sensitive, especially when predicting activity based on averages sampled over time.

Predictive trend analysis is not intended to find immediate problems that will bring down a system on a machine-time scale, as the sampling minimum is 1 minute … 15 minutes for some checks. With some checks (like the ENQ request rate), default collection and comparison every minute; so we could detect something within 2 minutes.

Detection of Soft Failures by z/OS Components

- z/OS attempts to detect soft failures as close to the source as possible
  - Uses the least amount of resources
  - Requires the smallest amount of the stack to do detection
- Detection of a soft failure requires ability to identify when something is wrong
  - Thresholds set by the installation
- In general, components try to avoid soft failures
  - “Throttles” may be used to manage internal requests
- Examples follow …

27

SHARE in Orlando 2011

Intro to Component detection of Soft Failures

Several examples of component soft failure detection follow on the next set of charts

Whenever possible, components try to avoid soft failures!

Component Examples: Detection, Identification of soft failures … Single system

Component	Features	Functions
GRS	Enhanced contention analysis for ENQ, Latch	Identify Blocker/Waiter, Deadly embraces, Job name, Creator ASID
	GRS Latch identity string	Associate name with Latch number
	WLM management of blocking units	Prevent deadlocks caused by starvation
	GRS ENF 51	Prevent exhaustion of common storage resulting from GRSQSCAN processing
UNIX System Services	Latch identity exploitation	Explanations for latch usage on D GRS
	XCF communication improvements (R13)	Detected lost messages in sysplex, via message ordering
	System Limits	Checks for buildup of processes, pages of shared storage (process & system level)
	D OMVS,WAITERS to diagnose file system latch contention (enhanced R13: file latch activity)	Identifies holders, waiters, latches, file device numbers, file inode numbers, latch set identifiers, file names, and owning file systems
JES2	JES2 Monitor	Assists in determining why JES2 is not responding to requests “Monitor” msgs issued for conditions that can seriously impact JES2 performance

Details in backup section

SHARE in Orlando 2011

Component Examples: Detection, Identification, recovery of soft failures … Single system

Component	Features	Functions
IOS	Missing Interrupt Handler	Detect incomplete I/O operations, within a policy driven time period (device, CU, fabric); recover, FFDC
	Identify systems sharing a reserve	Identify partner system sharing device D U,VOL= … D GRS,DEV= …
	Captured UCB protection	Prevent accidental overlays of real UCBs in SQA by Legacy applications
	I/O timing facility	Abnormally end I/O requests exceeding I/O timing limits for device; Hyperswap devices as well
	Detect & remove "Flapping Links"	Improved channel recovery (hardware)
	Dynamic Channel Path Management	WLM dynamically move channel paths from one CU to another, in response to workload changes
DFSMS	CAS contention detection	Identify, terminate service tasks beyond a monitored wait time
	VSAM RLS index traps	Checks the structure of all index CIs before writing them to DASD
	Media manager	Recover channel program error retry from I/O errors, using a lower level protocol

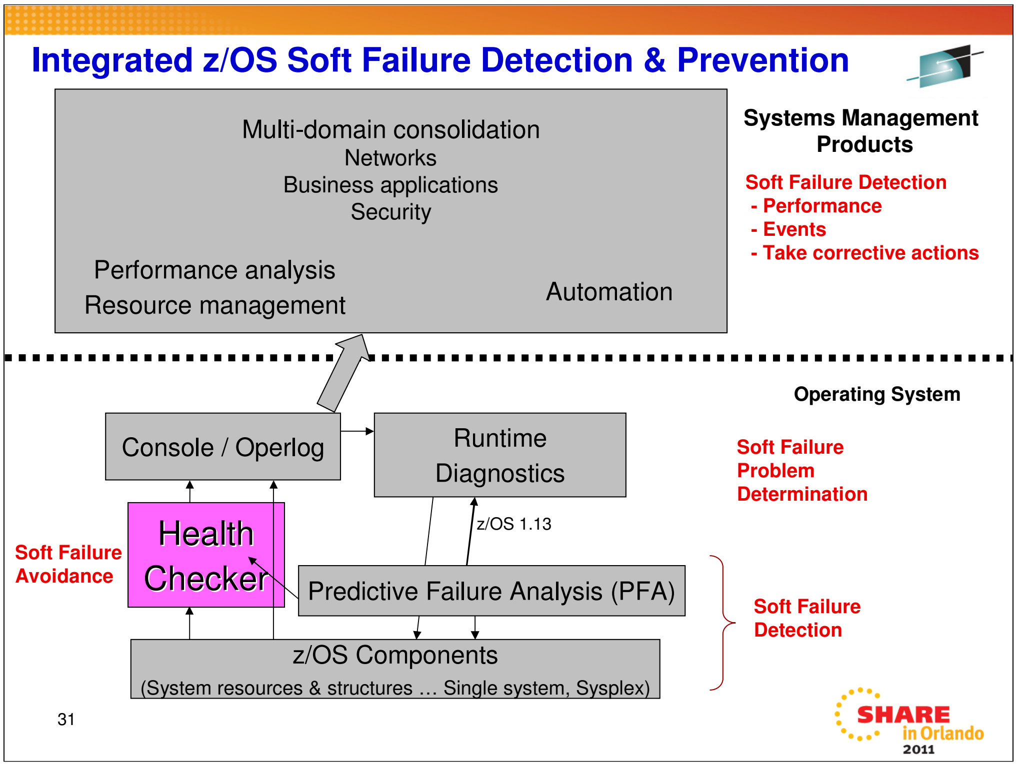
Details in backup section

SHARE in Orlando 2011

Component Examples: Detection of soft failures … Sysplex

Compone nt	Features	Functions
XCF / XES	Stalled member support	Identify unresponsive system, restore to normal operation OR remove it to avoid sympathy sickness
	Exploitation of BCPii to determine dead system more quickly	Avoid waiting the Failure Detection Interval (FDI) if the system is truly dead … detect & reset failed system, eliminate data corruption, avoid sympathy sickness.
	Sysplex Failure Management, scenarios ***How long to allow …***	• Not updating status, Not sending signals (ISOLATETIME(0): Fencing initiated n seconds after FDI exceeded) • System updating status, not sending signals (Loss of connectivity: CONNFAIL(YES): remove systems with low weights) • System Not Updating Status, But IS Sending Signals (SSUMLIMIT(900) … length of time system can remain not updating heartbeat (semi-sick), but sending signals) • Sysplex Member Stalled (MEMSTALLTIME … break out of of an XCF signaling jam by removing the largest build-up) • Structure Hang conditions … Take action when connector does not respond, avoiding user hangs (CFSTRHANGTIME) (R12)
	Critical Member support; GRS exploitation (R12)	If a critical member is “impaired” for long enough, XCF will eventually terminate the member; GRS: remove system

Details in backup section



Next we'll discuss Health checks, hosted by the z/OS Health Checker

IBM Health Checker for z/OS
Soft Failure Avoidance

- 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
  - …

- Also used to emit PFA alerts
  - Warnings of detected soft failures

- 187 z/OS Health Checks in z/OS R13

SHARE in Orlando 2011

Role of health checker is to avoid subtle configuration error from resulting in Soft Failures

Health Checker: Soft Failure avoidance
Important examples

Component	Health Check	Functions
XCF	XCF_CDS_SPOF	Evaluates primary & secondary CDS configuration to determine if Sysprog inadvertently created a single point of failure
	XCF_SFM_SUM_ACTION	Checks ISOLATETIME value, to allow SFM to fence and partition a system without operator intervention and without undue delay.
	XCF_SFM_SUMLIMIT	Checks status update missing (SUMLIMIT) value
	XCF_SFM_ACTIVE	Verifies SFM active, policy values
	XCF_SFM_CFSTRHANGTIME	Verifies CFSTRUCTURE hang time
	XCF_SFM_CONNFAIL	Threshold for loss of connectivity
RACF	RACF_GRS_RNL	Evaluates whether the RACF ENQ names are in a GRSRNL list: system exclusion resource name list (SERNL) or the system inclusion resource name list (SIRNL)

Details in backup section

Health Checker: Soft Failure avoidance examples

Component	Health Check	Functions
Serviceability	DAE_SUPPRESSING	DAE suppresses duplicate SVC dumps so that system resources (processor cycles and dump space) are not used for a dump which provides little or no additional diagnostic value
	SVA_AUTOIPL_DEFINED	Check whether Program-Directed IPL and not GDPS, and whether AUTOIPL policy is active
	SVA_AUTOIPL_DEV_VALIDATION	Validates SADMP, MVS IPL devices
UNIX System Services	USS_PARMLIB	Validate current system against parmlib IPL'd with Remind you to update parmlib (due to dynamic changes)
	USS_CLIENT_MOUNTS	With Sysplex, some file systems accessed locally, some of function shipped to the File system owner. Some are accessed locally, but are configured to function ship
	USS_FILESYS_CONFIG	Checks if mount attribute access is read only; whether HFS's in Sysplex root

Details in backup section

Health Checker: Soft Failure avoidance examples

Component	Health Check	Functions
IOS	IOS_CAPTUCB_PROTECT	UCB capture protection is enabled, allowing UCBs to be temporarily copied to 24-bit storage for legacy software access
	IOS_CMRTIME_MONITOR	Detects if any control units in the system are reporting inconsistent average initial command response (CMR) time (round trip delay) for their attached channel paths. Exception issued when a CU has a path with highest avg CMR time greater than a threshold/ratio
System Logger	IXGLOGR_STRUCTUREFULL	Primary structure full; need to offload
	IXGLOGR_ENTRYTHRESHOLD	High number of entries in element pools
	IXGLOGR_STAGINGDSFULL	Full staging data space

Details in backup section

z/OS Health Check: *Example Categories*

Category	Examples
Detect Single points of failure	VSAMRLS_SINGLE_POINT_FAILURE (SHCDS data sets) XCF_CDS_SPOF (XCF Couple Data Sets) XCF_CF_CONNECTIVITY (CF links, SPOF)
Security	RACF_GRS_RNL (for RACF datasets) SDSF_CLASS_SDSF_ACTIVE (SDSF settings)
Address space checks	IEA_ASIDS (number of ASIDs remaining) IEA_LXS (number of LX's remaining) SUP_LCCA_ABOVE_16M
GRS	GRS_MODE (system configured in STAR mode) GRS_SYNCHRES (GRS synchronous reserve processing enabled) GRS_CONVERT_RESERVES (reserves converted to ENQs)
I/O	IOS_CAPTUCB_PROTECT IOS_CMRTIME_MONITOR (Check for inconsistent average initial command response (CMR)) IOS_MIDAW (MIDAW enabled)

36

SHARE in Orlando 2011

z/OS Health Checks categorized by types of areas they examine

z/OS Health Check: *Example Categories*

Category	Examples
Optimal component settings	ALLOC_* (Allocation) CNZ_* (Consoles) CSRES (Comm Server), CSTCP_* (TCP/IP) SDSF_*, …
Sysplex configuration	XCF_* XCF_CF_* CSTCB_* RRS_* IXGLOGR_* VSAMRLS_* XCF_SFM_* CNZ_* Etc.

z/OS Health Checks categorized by types of areas they examine

z/OS Health Check: *Example Categories*

Category	Examples
Serviceability (Dump, Trace options)	SDUMP_AVAILABLE SDUMP_AUTO_ALLOCATION (auto-alloc SDUMP data sets) CSTCP_SYSTCPIP_CTRACE (CTRACE active, options) CSVTAM_VIT_SIZE (VTAM Internal Trace table size) CSVTAM_VIT_DSPSIZE (VTAM Internal Trace) SVA_AUTOIPL_DEFINED SVA_AUTOIPL_DEV_VALIDATION DAE_SHAREDSN DAE_SUPPESSING
Buffer sizes, storage limits	CSTCP_TCPMAXRCVBUFRSIZE CSVTAM_CSM_STG_LIMIT VSAMRLS_CFCACHE_MINIMUM_SIZE XCF_MAXMSG_NUMBUF_RATIO RSM_MEMLIMIT RSM_MAXCADS RSM_AFQ RSM_REAL RSM_RSU VSM_*

38

SHARE in Orlando 2011

z/OS Health Checks categorized by types of areas they examine

z/OS Health Check: *Example Categories*

Category	Examples
Hardware	SUP_HIPERDISPATCH (Verify Hiperdispatch enabled) SUP_HiperdispatchCPUConfig (monitors the number of CPUs installed and Hiperdispatch state of the system)
Other component specifics	Console configuration HSM control data set backups JES2 ready to upgrade Reconfiguration SMS CDS configuration System logger Staging data sets full, entry thresholds, structure full USS/ zFS: File system issues VSAM RLS: false contention, monitor contention, monitor unresponsive CICS regions, TVS enabled
Migration checks	

39

SHARE in Orlando 2011

z/OS Health Checks categorized by types of areas they examine

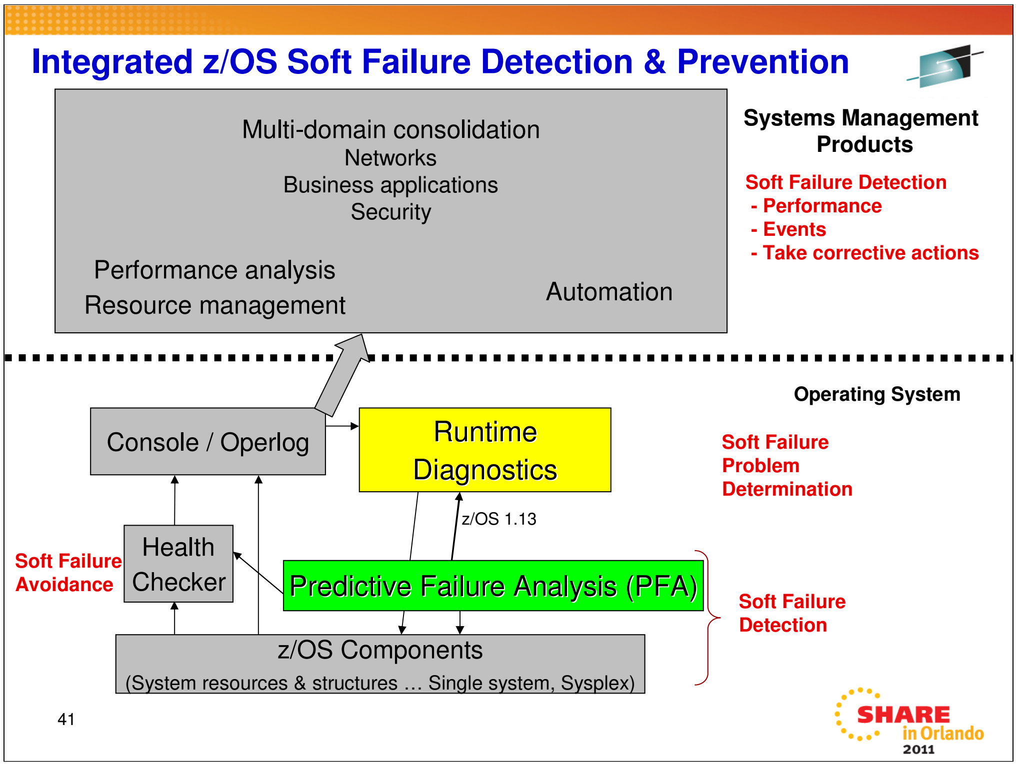
Important considerations when enabling z/OS Health Checks

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
   a. Start Health Checker and try to resolve all exceptions
   b. Activate all health checks, resolve all exceptions
   c. *Don't* think that you must activate all health checks at once to get benefit
   d. Goal should be to remove all exceptions
      - by fixing the condition
      - by tuning the check so that it looks for what you need it to look for
      - (as a last resort) by deactivating the check
   e. Once you can run cleanly, you will be in the ideal position to know that an exception indicates something has changed
   f. 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
   a. Some z/OS Health Check recommendations conflict with GDPS function, so follow GDPS guidelines

40

SHARE in Orlando 2011

Turn it on. You might find so many exceptions that you feel overwhelmed. But they're all probably things you ought to check out. Since you've been running this way for some time, they're not likely things that you absolutely need to deal with immediately. Your goal should be to get rid of all the exceptions, whether by fixing the condition, or by tuning the check so that it looks for what you need it to look for, or as a last resort by deactivating or even deleting the check. Once you can run cleanly, you will be in the ideal position of knowing that when an exception shows up it is definitely something you want to look at, as something has changed.



Next, let's transition to the Soft Failure detection & PD segment … Predictive Failure Analysis (PFA) and Runtime Diagnostics (RTD)

Soft Failure Detection: Predictive Failure Analysis

- Monitors trends in certain z/OS system resources & metrics
- Predicts expected, normal behavior as well as future behavior; identifies exceptions as Soft Failures
  - … when the resources will run out, or when metrics become abnormal when compared to expected prediction
  - Machine-learning technology used to determine what's normal
  - Statistical analysis used to identify exceptions
  - Focuses on metrics affecting different layers of the software stack
  - Exceptions alerted and reports written using Health Checker for z/OS
  - Identifies areas related to
    - resource exhaustion
    - damaged address spaces and damaged systems
  - Tune comparison algorithms using configurable parameters such as STDDEV; defaults selected based on IBM test systems
  - Tunable configuration parameters per check
  - Invokes Runtime Diagnostics to check for hung address spaces (R13); RTD validates and suggests next steps

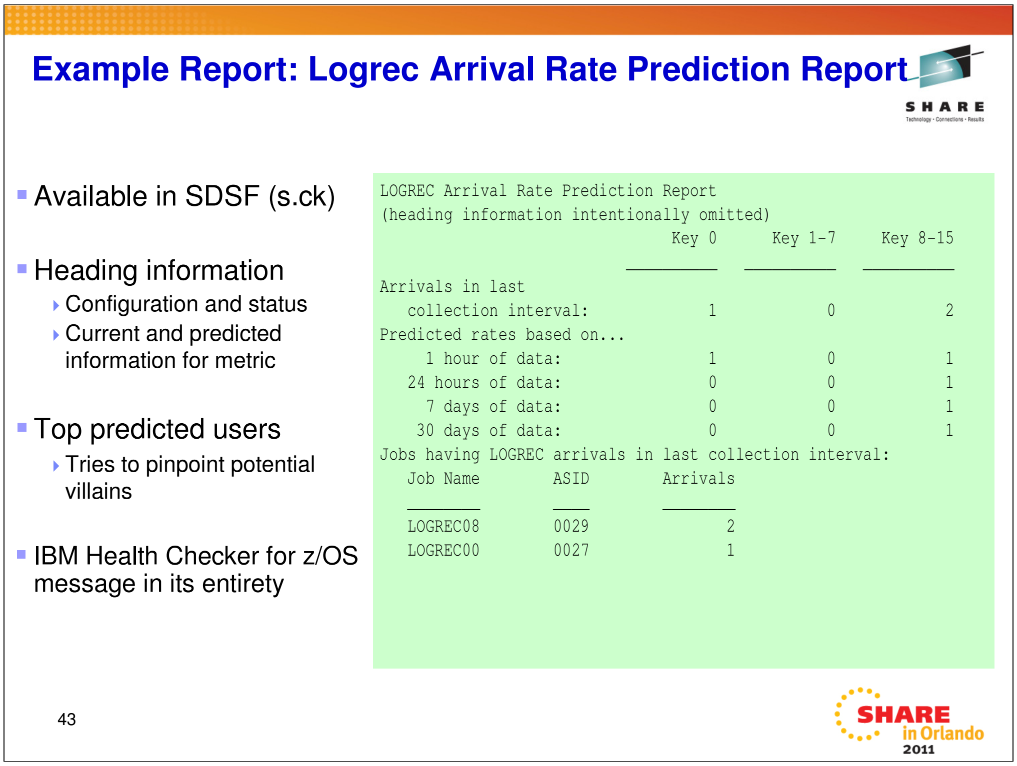
42

SHARE in Orlando 2011

Predict expected, normal behavior based on modeling of past behavior over the past 24 hours, week, month and statistical analysis of current activity.

Tunable algorithms, other parameters

Invokes RTD to check for hung address spaces when PFA considers address space when detection rates are “too low”



When PFA detects there is no problem, a prediction is produced in the health check option of SDSF. When PFA detects there is a problem, an exception report is printed. All PFA reports are available in SDSF. There is heading information which contains configuration and status for the collections and models.

Each check has its own check-specific information. All checks will display potential villains when an exception occurs. Most of the checks will also list the top address spaces or other important information even when there isn't an exception.

The exception message is configured by default to be issued as a WTO. That message is also included in the exception report along with its detailed response information.

The numbers are in this example are not from a real exception.

The PFA Health Checks

- z/OS 1.10 SPE
  - Common storage exhaustion check
    - CSA + SQA below the line
    - ECSA + ESQA above the line
  - LOGREC arrival rate check
    - Groups arrivals by key
    - Four time ranges
- z/OS 1.11
  - Frames and slots usage check
    - Tracks all address spaces that start within an hour after IPL (persistent)
  - Message arrival rate (WTO/WTOR) check
    - Chatty, persistent address spaces
    - Non-chatty, persistent address spaces
    - Non-persistent address spaces
    - Total system
- z/OS 1.12
  - SMF arrival rate check
    - Same categories as message arrival rate check
  - Modeling improvements
    - More granular common storage check
    - Supervised learning (exclude jobs)
    - Dynamic modeling
  - Performance and serviceability
- z/OS 1.13
  - JES spool usage check
    - Tracks all persistent address spaces
    - JES2 only
  - Enqueue request rate check
    - Chatty, persistent address spaces
    - Total system
  - Integration with Runtime Diagnostics to detect rates that are "too low"

How often do you see soft failures on your systems?

Runtime Diagnostics

- Analyzes a “sick, but not dead” system in a timely manner
- Performs analysis similar to a very experienced system programmer
  - But faster – goal of 60 seconds or less
  - More comprehensive
  - Looks for specific evidence of “soft failures”
  - Provides suggested next steps
- Runtime Diagnostics
  - Is not automation or a monitor
  - Takes no corrective action, but recommends next steps
  - Has no background processing and minimal dependencies on system services

Component Analysis → Analyzes Operlog for specific component msgs

Global Resource Contention → Detects contention in ENQs in system address spaces, GRS latches, and the z/OS UNIX file system

Address Space Execution → Detects important execution information: CPU usage, local lock usage, and TCB loops

45

SHARE in Orlando 2011

Looking at the operating system only!

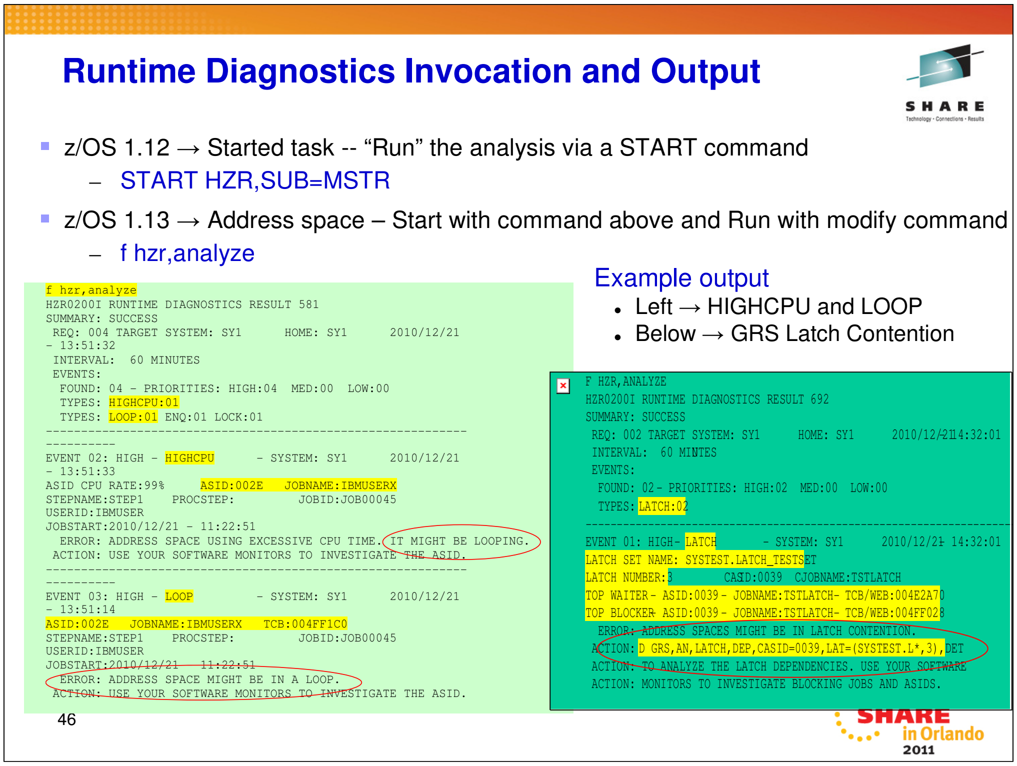
Diagnose sick system by identifying symptoms that could lead to identifying the culprit, and offering next steps to take.

3 areas:

•Component analysis (messages)

•Global resources (ENQs)

•Local address space characteristics



Loop detection: Runtime Diagnostics looks through all tasks in all address spaces to determine if a task appears to be looping. Runtime Diagnostics does this by examining various system information for indicators of consistent repetitive activity that typically appears when a task is in a loop. When both a HIGHCPU event and a LOOP event (shown in the example) 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.

Runtime Diagnostics Symptoms Detected

- 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 system 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 TCP 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

Use it when getting ready for a bridge call.

Discreet symptoms

z/OS 1.13 PFA Integration with Runtime Diagnostics

- **Detects damaged or hung system or address space based on rates being "too low"**
  - When PFA detects too low, Runtime Diagnostics is executed
- **Output**
  - **"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
- **Supported for** Message Arrival Rate, SMF Arrival Rate, Enqueue Request Rate

```
                    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.)
```

48

SHARE in Orlando 2011

When an exception for an abnormally low condition is found, a health check exception will be issued explaining the problem. The PFA report will include the current rates and predicted rates for the category that was failing. In addition it will include the Runtime Diagnostics output received when PFA called Runtime Diagnostics to verify the problem.

Note that in this example, PFA indicated that jobs JOBS4 and JOBS5 had a Message Arrival Rate that was too low when compared to their expected rates for any of the time ranges. Runtime Diagnostics verified that there could be a problem by detecting both a HIGHCPU and a LOOP event for JOBS4. Therefore, the abnormally low message arrival rate coupled with the results of Runtime Diagnostics show that JOBS4 is very likely looping. The Runtime Diagnostics output for JOBS5 were similar, but were purposely omitted from this display due to lack of space.

Just like the other PFA prediction reports, the PFA prediction reports for abnormally low conditions are available in SDSF.

Extending to Systems Management Products

- Many (ISV) Systems Management products support
  - Actions based on WTO message events
  - Automation of Health Check events
    - PFA Health Check events = soft failures
  - Performance analysis
  - Integration of Alert displays, performance exceptions, event based actions

49

SHARE in Orlando 2011

Thus far we have discussed functions in the z/OS stack that perform detection, avoidance and PD for Soft Failures, and exceptions are emitted via WTO messages.

Let’s now turn to the Systems Management stack.

Most management stacks provide performance analysis and Resource management, and offer automation of WTO message events to translate the base event to a business action.

In addition, some systems management vendors offer consolidation points for handling OS events, network issues, security, etc.

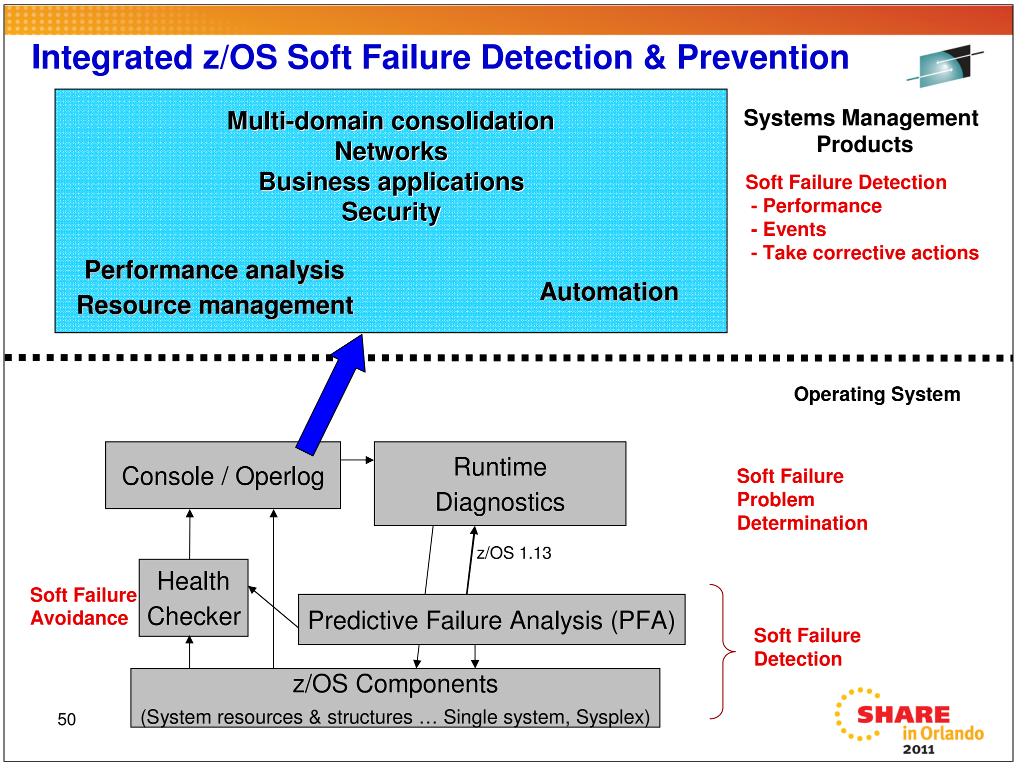
Tivoli products can integrate a variety of soft failure alert types

•PFA alerts

•Performance issues

•Other message automation

•Policy to control corrective actions



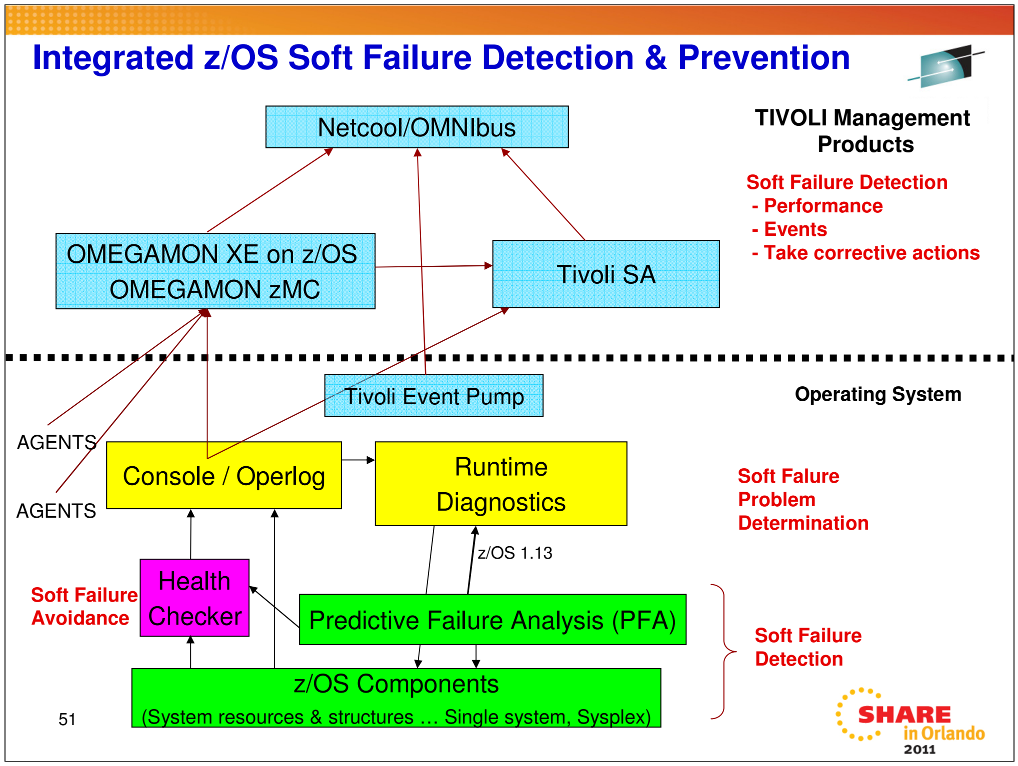
Thus far we have discussed functions in the z/OS stack that perform detection, avoidance and PD for Soft Failures, and exceptions are emitted via WTO messages.

Let's now turn to the Systems Management stack.

Most management stacks provide performance analysis and Resource management, and offer automation of WTO message events to translate the base event to a business action.

In addition, some systems management vendors offer consolidation points for handling OS events, network issues, security, etc.

Transition to next chart … Systems Management Product box becomes Tivoli products example

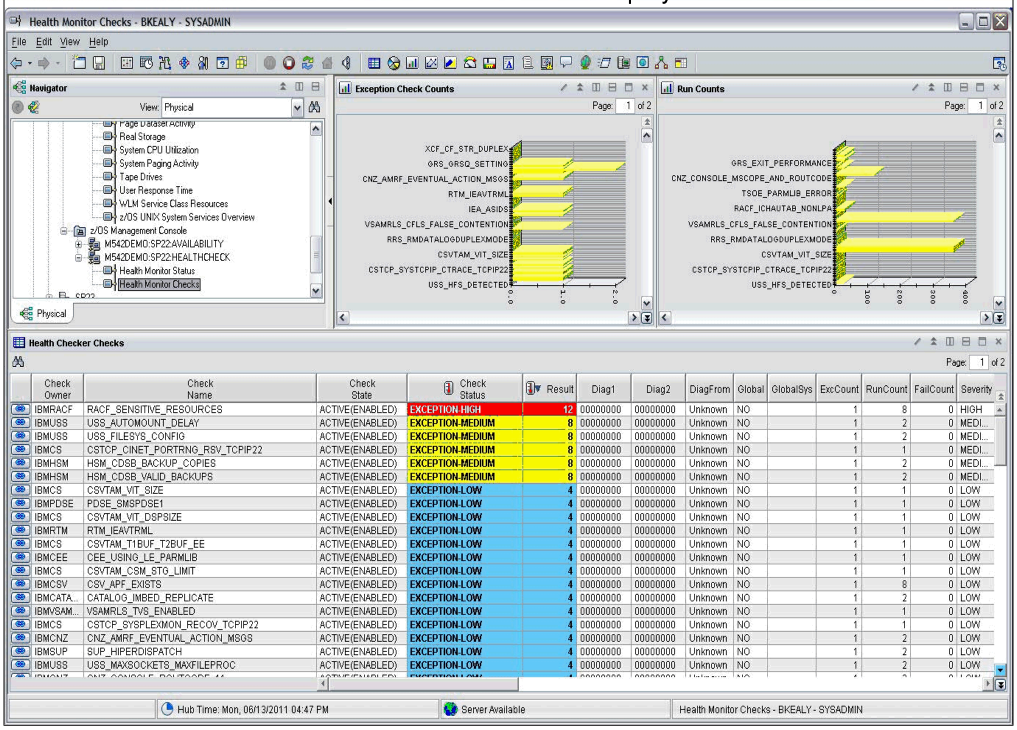


Messages provided by PFA via the health checker support, as well as outputs from other health checks and z/OS components, can be used to further analyze the situation.

The OMEGAMON XE for Management Console will see all health check alerts including the PFA ones. You can build a situation that will alert you if a PFA check is raised and forward that event to other Tivoli event management products (like OMNIBUS or Tivoli Event Console). You can also use Runtime Diagnostics to further analyze PFA results. Runtime Diagnostics provides detailed analysis either of either the entire system or address space looking for soft failures. It uses lists of messages identified by specific components to review critical messages in the joblog. It also stores information about enqueues to analyze contention, evaluates local lock conditions, and queries a job that has a task in a TCB loop.

Using customer policy, TSA (via Netview) can detect that a health checker message for PFA exceptions were issued and drive actions.

zMC Health Check display



Example of Health Check display on the zManagement Console (z/MC)

Exception check counts and counts of how often checks are run are shown in the top half of the screen.

Health check status is shown in the bottom half of the screen image

Overall: Reducing Impact of Soft Failures

SHARE
Technology • Connections • Results

- Automation of alerts is key
  - Display, take action
- z/OS can survive / recover from most soft failures
  - But, take advantage of what the base operating system has to offer
    - Soft failure detection across many z/OS components
    - Start Health Checker, PFA, RTD (R13) at IPL (e.g., COMMNDxx)
- Most metrics are very time sensitive
  - Defaults selected based on z/OS test environments; should be good for most
- Predictive trend analysis typically not done on a Machine-time scale
  - PFA not designed to detect anomalies that could terminate a system on machine-time scale
  - Shortest data comparison is once a minute; identification of a program consuming CSA make take a couple minutes
  - PFA has tuned comparison algorithms using what is learned from your system
    - Configuration parameters are tunable to make the algorithms more accurate for your workloads
    - All checks have configurable parameters, e.g. STDDEV (Lower → more sensitive)

53

SHARE in Orlando 2011

The key to reducing the impact of soft failures is

•Avoid them using health checker

•Enable system checking where possible

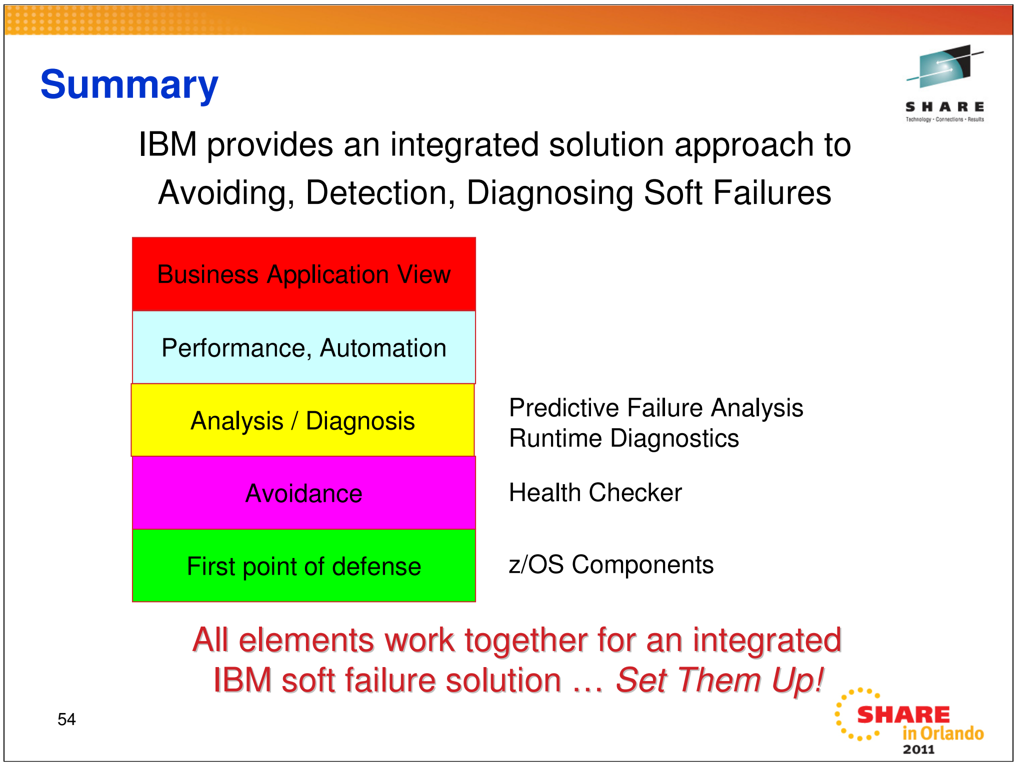
•Alerts can be acted upon. You can display them, automate on them and take action to address the detected problem(s)

It is true that z/OS can survive or recover from many forms of soft failures, as demonstrated by the different types of component checking.

•But you need to take advantage of what the base operating system has to offer; this requires enabling some function parameters, and starting health checker, PFA and RTD during the IPL, such as in the COMMNDxx parmlib member

Most metrics that are used to detect soft failures are very time sensitive, especially when predicting activity based on averages sampled over time.

Predictive trend analysis is not intended to find immediate problems that will bring down a system on a machine-time scale, as the sampling minimum is 1 minute … 15 minutes for some checks. With some checks (like the ENQ request rate), default collection and comparison every minute; so we could detect something within 2 minutes.



All of the elements work together for an integrated IBM soft failure solution … Set Them Up!

z/OS components
Health checks
PFA, RTD
Systems Management … Tivoli products

Acknowledgements

- Thank you to all who contributed information for this presentation

Jim Caffrey	z/OS Predictive Technologies
Karla Arndt	PFA / RTD
Scott Bender	USS Kernel
Ron Bretschneider	DFSMS - Media Manager
Mark Brooks	XCF, XES, CF
John Case	USS File System
Brian Kealy	Omegamon
Nick Matsakis	GRS, Availability
Terri Menendez	DFSMS - RLS
Peter Relson	Health Checker
Dale Riedy	IOS
Wayne Rhoten	DFSMS
Dave Surman	z/OS Architect
Tom Wasik	JES2
Doug Zobre	System Logger

55

SHARE in Orlando 2011

Backup Reference –
Component Soft Failure detection

56

SHARE in Orlando 2011

Detection of Soft Failures on a z/OS image: GRS serialization

- Enhanced contention analysis for ENQ / Latch
  - D GRS,ANALYZE,BLOCKER / WAITER / DEPENDENCY
  - D GRS,ANALYZE,LATCH,BLOCKER / WAITER / DEPENDENCY
    - Blocker/Waiter, Deadly embraces, Job name, Creator ASID, etc.
- GRS Latch identity string
  - Associate name with latch number
  - Included in D GRS latch analysis responses
  - Exploited by USS, RRS, Logger, RACF
- GRS interacts with WLM to manage priority of blocking units of work
  - Prevent deadlocks causing starvation
  - WLM's "trickle" support ensures that critical work is given cycles gradually to resolve any deadlocks
- GRS monitor
  - ENF 51 generates blocks in common storage (SQA)
  - SRBs suspended due to stuck receiver (e.g., RMF)
    - Therefore too many requests can cause common storage outage
  - GRS piped the requests elsewhere to avoid exhausting common storage
- Exploits XCF Critical member support (see XCF critical member support)

57

SHARE in Orlando 2011

GRS provided enhanced contention analysis to identify ENQ blockers and waiters a number of releases ago,

And recently implemented similar support for latches, also displayed using the D GRS,ANALYZE command.

In R11, GRS delivered the ability to identify latch usage, which is included in the D GRS response. The Latch Identity string is exploited by USS, RRS, Logger and RACF for their latches.

GRS encountered a situation where its ENF 51 events schedule an SRB, which gets suspended due to contention events, keeping its control blocks in common storage. If there are too many requests, SRBs exhaust common storage, causing a system outage..

•The 64 bit SSRBs support helped with the ENF 51 case where the consumer could not keep up with spikes. The suspended SRBs (SSRBs) which were in common could previously exhaust common storage. GRS didn't change to use Pause/Release for this in any way.

•GRS QSCAN/ISGQUERY did change to use Pause/Release rather than schedule a "resume" SRB for its requests as there were cases where the requester space had storage issues which prevented the "resume" SRB from getting it dynamic area in the target address space. This in turn resulted in the QSCAN/ISGQUERY invoker never getting resumed even after the storage issue cleared up. By using Pause/Release, the possibly temporary storage problem was circumvented such that the QSCAN/ISGQUERY invoker was not left hung out to dry.

GRS/WLM support to identify ENQs being held by units of work with lower priorities, thus possibly causing deadlocks. Priority is improved to complete processing and release the serialization.

•WLM/SRM added support to identify critical resources, which allows a resource owner to identify a case where a critical resource is blocked by a holder/holders and as such it should take more action to get the holder moving, above and beyond the ENQHOLD/ENQRelease sysevent services that are used. Only DB2 uses this interface.

•WLM introduced the "trickle" support which insures that all dispatchable work, including discretionary, gets some cycles every so often in order to help alleviate serialization bottle necks that were not resolved by ENQHOLD/ENQRELEASE or "critical", or the resource serialization provider did not use the "promotion" services. This is believed to have helped in these types of cases.

Too many XMPOSTs for same ECB

Looks @ ECB to see if already posted & ensure that the post is not done

Detection of Soft Failures on a z/OS image: UNIX System Services serialization

- Latch identity explanations for the latches used by USS (R13)
  - FS: <fs name> … MOUNT … MessageQ ID=<msg-ID in decimal>
    - System traversing or modifying structures related to the message queue
- XCF communication improvements
  - Lost XCF message detection (R13)
    - Utilizes XCF message ordering to detect lost messages
    - Activate with parmlib option, SETOMVS LOSTMSG=ON/OFF
  - Member Gone detects stall, attempts fix; if takeover fails, initiates sysplex-wide dump
- USS System Limits (R10)
  - Checks for buildup of processes, pages of shared storage (process & system level)
  - When 85% process utilization is reached, WTO messages are issued
  - For example: MAXASSIZE, MAXCPUTIME, MAXFILEPROC, MAXPROCUSER, MAXQUEDSIGS, MAXTHREADS
  - Displayed via D OMVS,LIMITS
- DISPLAY OMVS,WAITERS to diagnose file system latch contention problems
  - Enhanced in R13 to show a table for file latch activity
  - Holders, waiters, latches, file device numbers, file inode numbers, latch set identifiers, file names, and owning file systems

58

SHARE in Orlando 2011

USS examples

GRS Latch identity service is exploited to identify latch usage for file systems and other latch usage.

Example:

SY2 D GRS,ANALYZE,LATCH,WAITER

SY2 ISG374I 16.15.24 GRS ANALYSIS 734

LONG WAITER ANALYSIS: ENTIRE SYSTEM

----- LONG WAITER #1

WAITTIME JOBNAME E/S CASID LSETNAME/LATCHID

00:01:01 TC0 *E* 000E SYS.BPX.A000.FSLIT.FILESYS.LSN

20:FS: HOST12.AJAX.DIRECTORY

FS: <fs name>: If the LSETNAME is SYS.BPX.A000.FSLIT.FILESYS.LSN, the latch is used to serialize operations on the file system named in the latch identity string.

● MOUNT: This latch is used by the file system to serialize operations such as file system mount, unmount, move, and automount and others.

● MessageQ ID=<msg-ID in decimal>: This latch is used when the system is traversing or modifying structures related to the message queue whose identifier is shown in the latch identity string.

Lost XCF message detection … incurs a performance penalty in high UNIX traffic environments; better reliability

USS built a set of System Limits to identify storage creep

Dynamic socket limit

Detection of Soft Failures on a z/OS image: IOS examples

SHARE Technology · Connections · Results

Missing Interrupt Handler

- Incomplete I/O: Prevents an application or system outage due to an error in any one of the following places:
  - Device
  - Control Unit
  - Fabric
  - Operator/CE error (IML, cable pulls, etc…)
- Outage is prevented by:
  - Detecting when an I/O operation has not completed within a policy driven time period
  - Invoking system diagnostic routines to understand the scope of the error
  - Driving hardware and software recovery mechanisms
  - First Failure Data Capture

Identify sharing systems holding a reserve

- Start-pending MIH condition → D U,VOL= to identify device number
- D GRS,DEV=dddd to determine reserve status
- Identify other system with reserve, in message (IOS431I device reserve to CPU …)

Captured UCB protection

- Creates a temporary copy of UCBs for Legacy applications
- Prevents accidental overlays of real UCBs in SQA

59

SHARE in Orlando 2011

MIH intercepts incomplete I/O operations to prevent an application or system outage due to a device, control unit, fabric or hardware (cabling) error.
Once the scope of the problem is understood, hardware & software recovery mechanisms are invoked and diagnostic data is captured.

Identify sharing systems holding a reserve:
IOS071I dddd,**,jobname, START PENDING

Normally, due to a reserve being held on another system

On Sharing systems

Use D U,VOL=volser to identify device number

Use D GRS,DEV=dddd to identify reserve status

IOS431I will identify systems holding reserves

IOS431I DEVICE dddd RESERVED TO CPU=serialmodn,LPAR ID=ii SYSTEM=sysname

Captured UCB protection … prevent Legacy components from impacting IOS by modifying the UCB; Solution for 24-bit programs

I/O Timing Facility – Identify slow I/O response time

- Times the entire I/O request
  - If exceeds timing limit …
  - Abnormally ends I/O requests exceeding I/O timing limits for device
  - Application posted with permanent error, error logged to Logrec
- Facility can trigger a Hyperswap when I/O timeout occurs for a device monitored
  - Whether I/O operation should be terminated or started on the "swap TO" device

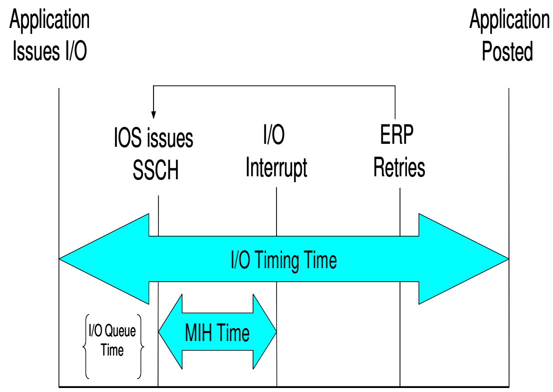


Figure 1 - MIH vs. I/O Timing

60

The system invokes the I/O timing facility to monitor I/O requests. If an active I/O request has exceeded the I/O timing limit, the system abnormally ends the request and does the following:

•Clears the subchannel of all active, start pending, or halt pending I/O requests.

•Issues a message to the system operator.

•Obtains information about the terminated request (such as whether the request was queued or started) to build an MIH record.

If a queued I/O request has exceeded the I/O timing limit, the system abnormally ends the request and does the following:

•Issues a message to the system hardcopy log

•Obtains information about the terminated request (such as whether the request was queued or started) to build an MIH record.

The I/O timing facility can be enabled to trigger a HyperSwap when an I/O timeout occurs for a device that is monitored for HyperSwap. Optionally, the user can specify whether a timed-out I/O operation that initiates a HyperSwap is to be terminated or allowed to be started on the swap 'TO' device.

For any I/O requests that exceeds the I/O timing limit, the system performs the following actions:

When the I/O timing trigger is not enabled for HyperSwap, or is enabled and the IOTTERM option is also enabled:

> •Abnormally ends the I/O request that has exceeded the time limit, and does not requeue the request for execution.
>
> •Issues a message.
>
> •Writes an entry in the SYS1.LOGREC data set for the abnormally ended I/O request.

When the I/O timing trigger is enabled for HyperSwap and the IOTTERM option is disabled:

> •Abnormally ends the I/O request that has exceeded the time limit, and requeues the request for later execution on the swap 'TO' device at the completion of the HyperSwap.
>
> •Issues a message for the first timeout condition that triggers a HyperSwap on the associated DASD subsystem.
>
> •Writes an entry in the SYS1.LOGREC data set for the abnormally ended I/O request.

Improved Channel Recovery

- For frequently-occurring path errors, better to have hardware problem cause path taken offline than continue to cause problems
  - IOS recovery delays application I/O even when there are other paths

Proactively Removing Paths – Flapping Links

- Logic path between channel & control unit becomes available, unavailable multiple times over a short period
- Drives IOS recovery for all devices on the affected link
- May cause application I/O delays
- When channel detects that link has “flapped” 5-9 times in 5 minutes, it stops attempting to establish a logical path

Dynamic Channel Path Management

- Simplify I/O configuration definition task
- Static channel path definitions needed to be re-evaluated when workloads shift
- DCM lets WLM dynamically move channel paths from 1 CU to another, in response to workload changes
- Improve workload management
- DASD I/O resources are used more efficiently
- Improves Availability
- Foundation for auto-configuration
- Balance mode, Goal mode

61

SHARE in Orlando 2011

Customers have said that when errors occur frequently enough on a path that they would rather see the path taken offline rather than having the hardware or z/OS repeatedly try to recovery the path. An example where support was added is the flapping links support that was introduced in the z9 processor. Flapping links is a condition where the logical path between the channel and control unit becomes available and unavailable (e.g., loss of light) multiple times within a short period of time. This causes IOS recovery processing to be initiated multiple times for all devices on the affected link, which may delay application I/O for long periods of time, even though there are other paths available. When the channel detects that the link has “flapped” 5-9 times in 5 minutes, it stops attempting to establish a logical path.

In addition, customers have also said that they’d like to see z/OS be more proactive about removing failing paths from devices. That is, instead of waiting for each device to trip over the error and take the required recovery action, they’d like to see z/OS remove the path from all devices in an LCU when an error causes the path to be removed from the first device. This will significantly reduce recovery time and improve application performance when an error occurs.

Dynamic Channel Path management

Prior to Dynamic Channel Path Management, all channel paths to I/O control units had to be statically defined. In the event of a significant shift in workload, the channel path definitions would have to be reevaluated, manually updated via HCD, and activated or POR'ed into the configuration. Dynamic Channel Path Management lets Workload Management dynamically move channel paths through the ESCON Director from one I/O control unit to another, in response to changes in the workload requirements. By defining a number of channel paths as "managed," they become eligible for this dynamic assignment. By moving more bandwidth to the important work that needs it, your DASD I/O resources are used more efficiently. This may decrease the number of channel paths you need in the first place, and could improve availability -- in the event of a hardware failure, another channel could be dynamically moved over to handle the work requests.

Dynamic Channel Path Management operates in two modes: balance mode and goal mode. In balance mode, Dynamic Channel Path Management will attempt to equalize performance across all of the managed control units. In goal mode, which is available only when WLM is operating in goal mode on all systems in an LPAR cluster, WLM will still attempt to equalize performance, as in balance mode. In addition, when work is failing to meet its performance goals due to I/O delays, WLM will take additional steps to manage the channel bandwidth accordingly, so that important work meets its goals.

Detection of Soft Failures on a z/OS image: DFSMS examples

- **CAS Contention Detection**
  - Runs as part of the CAS analysis task
  - Periodically checks the Catalog Address Space (CAS) service tasks list (every 30 seconds or upon request)
    - Based on a set wait time and reason class, determines those tasks which are beyond the wait time.
      - *Checks for service tasks that are active and waiting on the SYSZTIOT enqueue. It sets timer for each waiting task (10 min)*
    - Creates a symptom record for each task past the limit
    - Terminates some of the violating tasks, which were considered safe to terminate
- **VSAM RLS index traps**
  - Set the trap using a V SMS,MONDS command
  - Checks the structure of all index CIs before writing them to DASD.
    - If problem, abend is issued and write is avoided
- **Media Manager**
  - Channel program error retry from I/O errors, using a lower level protocol supported by the device
    - zHPF transport mode channel program
    - Command mode channel program with MIDAWs
    - Command mode channel program with IDAWs
  - Media Manager will retry the I/O with one of the lower level protocols

62

SHARE in Orlando 2011

Catalog contention detection

•The possibility exists that while catalog service tasks are waiting on an event, the event may not finish or complete. It could take the event an unreasonably long time to return, leaving the Catalog Address Space (CAS) service task waiting, possibly indefinitely. Contention support identifies those CAS tasks which appear to be stuck while waiting on an event. When these tasks are identified as having passed a threshold, in all cases, a symptom record will be created. Some CAS tasks past the threshold, which are identified as reasonably safe to terminate, are terminated, freeing the available CAS service task for additional work. The wait threshold is initially based on a time defaulted by the system (30 minutes). The wait threshold is also user selectable with a minimal wait-time of 30 minutes.

VSAM RLS index traps

•Set the trap via a VARY SMS “monitor data set” command (V SMS,MONDS(IGWVSAM.BASE.INDEX.TRAP),ON)

•VSAM checks index Control Intervals for problems

Media Manager

• Channel program error retry from I/O errors, using a lower level protocol supported by the device

•zHPF transport mode channel program

•Command mode channel program with MIDAWs

•Command mode channel program with IDAWs

•Media Manager will retry the I/O with one of the lower level protocols

Detection of Soft Failures on a z/OS image: JES2 Monitor

- Assists in determining why JES2 is not responding to requests (single system)
- “Monitor” messages issued when conditions exist that can seriously impact JES2 performance (z/OS or JES2 issues)
- Automatically started when JES2 is started
- Results displayed via $JD STATUS command
  - Any conditions the monitor detected that could impact JES2
- Corresponding monitor address space for each JES2 address space
  - $JD MONITOR displays status info for each monitor task
  - Samples values at regular intervals
- Incident categories:
  - Normal processing
  - Tracking: processing time exceeds threshold
  - Alerts: Incident being tracked crosses a second (sampling) threshold
    - Exclusive incidents focus attention on primary incident
- Resource utilization
  - Low, high, average, current utilization
- $JD HISTORY displays up to 72 hours of resource utilization & CPU sample statistics

For more information, see JES2 Diagnosis book, GA22-7531

63

Detection of Soft Failures in a Sysplex: XCF stalled member support

- A system may appear to be healthy with respect to XCF system status monitoring:
  - Updating status in the sysplex CDS
  - Sending signals
- But is the system actually performing useful work?
  - There may be critical functions that are non-operational, making the system unusable
    - Could induce sympathy sickness elsewhere in the sysplex
    - Waiting for a response; waiting to get an ENQ, latch, lock
  - Causes include
    - Dead system
    - Loops (spin, SRB)
    - Low weighted LPAR
    - Loss of a Coupling Facility
- *Long periods of sympathy sickness may have a greater negative impact on the sysplex than termination of an XCF group member, address space, structure connector, or even a system*
- Action should be taken to restore the system to normal operation OR remove it to avoid sympathy sickness
  - Helps reduce the incidence of sysplex-wide problems that can result from unresponsive critical components

64

SHARE in Orlando 2011

XCF and Sysplex sympathy sickness

Intro to XCF/SFM support for termination of stalled XCF group members, related to avoiding sympathy sickness.

Many of the issues detected by XCF & SFM are causes of soft fialures … stalled members, underlying system issues. The stall of an XCF group member is often the result of some single or cascaded set of problems that will ultimately affect the ability of the sysplex to service business applications.

To present these types of soft failures, action should be taken to restore the impacted system to normal operation OR remove it to avoid sympathy sickness.

Detection of Soft Failures in a Sysplex: Sysplex Failure Management (SFM)

- Single system "Sick but not dead" issues can escalate to cause sysplex-wide problems
  - Typically holds resources needed by other systems in the sysplex
- Implements best practices of a resilient sysplex
- Enables automatic, timely, corrective action to be taken when applications or systems appear to be causing sympathy sickness
- Protects your sysplex when your operators and/or your automation are inattentive, unable, or incapable of resolving the problem
- Define an SFM policy to help meet availability and recovery objectives
  - Applications or systems are not permitted to linger in an extremely sick state such that they adversely impact other systems in the sysplex
  - Applications or systems are not terminated prematurely
  - Failure Detection Interval (FDI): amount of time a system is permitted to appear unresponsive (Not updating heartbeat, Not sending signals)
- Use of BCPii to determine a system is down dramatically improves this detection (over use of heartbeat) (see BCPii topic)

65

SFM deals with the detection and resolution of soft failures that could cause sympathy sickness conditions when a system or sysplex application is unresponsive.

Single-system "sick but not dead" issues can and do escalate to cause sysplex-wide problems.

A sick system typically holds resources needed by other systems a unable to participate in sysplex wide processes

Thus other systems become impacted.

Root cause of the sickness is a single system problem … contention, dispatching delays, spin loops, overlays, queue/data corruption, etc.

(These are soft failure symptoms as well.)

Routing work away from the troubled system does not necessarily guarantee that other systems will not be impacted.

However, allowing non-terminating problems, where something simply becomes unresponsive, to persist *typically compounds the problem*.

By the time manual intervention is attempted, it is often very difficult to identify the appropriate corrective action.

Appropriate SFM specifications enable systems in the sysplex to take corrective action automatically.

In general, each parameter arises out of real world situation that led to some sort of (usually quite ugly) outage.

The next few charts outline features that are important to detecting soft failure situations related to cluster processing.

XCF_SFM_ACTIVE health check

Detection of Soft Failures in a Sysplex: SFM

- System Not Updating Status, Not Sending Signals
  - ISOLATETIME(0)
    - n seconds after the FDI exceeded fencing is initiated by all systems
    - Commands are sent across the coupling facility to the target system and I/O is isolated
    - After fencing completes successfully, sysplex partitioning continues
- System updating status, not sending signals
  - Loss of connectivity: CONNFAIL(YES)
    - SFM determines sets of systems that do have full signal connectivity
    - Selects a set with largest combined system weights
    - Systems in that set survive, others are removed
    - Ensure the weights assigned to each z/OS system adequately reflect the relative importance of the system
- System Not Updating Status, But IS Sending Signals
  - SSUMLIMIT(900)
    - Indicates the length of time a system can remain in the state of not updating the heartbeat and sending signals
    - *This is the amount of time a system will remain in a "semi-sick" state.*
    - Once the SSUMLIMIT has been reached the specified action will be initiated against the system
- Sysplex Member Stalled
  - MEMSTALLTIME (600-900)
    - Enable XCF to break out of an XCF signaling traffic jam
    - SFM automatically starts removing the largest build-up, adversely impacting other systems in the sysplex
    - Action XCF will take: terminate the stalled member with the highest quantity of signals backed up

66

SHARE in Orlando 2011

XCF_FDI health check
XCF_SFM_SUM_ACTION health check
XCF_SFM_SSUMLIMIT health check

FDI = Failure Detection Interval (XCF)

•Amount of time a system is permitted to appear unresponsive (not updating heartbeat, not sending signals)

•If the specified FDI value is too short, it might trigger unnecessary actions by SFM;

•If FDI is set too long, it could elongate the detection window for soft failures related to sympathy sickness

•It's best to let FDI default to being based on internal "spin time" (rather than specifying it as a "user FDI" value)

This chart outlines a number of common sysplex customer situations detected by SFM.

MEMSTALLTIME enables system to break out of an XCF signaling traffic jam. SFM will automatically start removing the largest build up. In the picture above, imagine all the blue cars were instantly removed.

(SSUM = Status Update Missing condition) … SSUMLIMIT(#seconds)

When you have specified or defaulted to SSUMLIMIT(NONE), and a system has not updated its status within the failure detection interval but continues to produce XCF signaling traffic, SFM prompts the operator to optionally force the removal of the system. The fact that XCF signalling continues indicates that the system is functional but may be experiencing a temporary condition that does not allow the system to update its status. If the operator decides that removal of the system is necessary, message IXC426D provides the prompt to isolate the system and remove it from the sysplex. In this case, the ISOLATETIME interval specified in the SFM policy is ignored.

If XCF signaling also stops, SFM will start to isolate the failing system at the expiration of the ISOLATETIME interval. With a value other than none specified for the SSUMLIMIT SFM administrative data utility parameter, SFM will start to isolate the system when the time specified for the SSUMLIMIT parameter has expired for a system that is in status update missing condition but still producing XCF signalling traffic.

If the system stops producing XCF signalling traffic, SFM may start to isolate the failing system before the SSUMLIMIT time expires, at the expiration of the ISOLATETIME interval.

Detection of Soft Failures in a Sysplex: SFM

Taking Action When a Connector Does Not Respond

- Connectors to CF structures participate in processes, respond to relevant events
  - XES monitors the connectors, reports unresponsive connectors
  - Users of the structure may hang until offending connector responds or is terminated
- CFSTRHANGTIME (z/OS R12)
  - How long the system should allow a structure hang condition to persist before taking action
  - Enables XES to automatically take action if a connector does not respond to a structure event in a timely fashion
- XES corrective actions:
  - Stop rebuild
  - Force user to disconnect
  - Terminate connector task, address space or system
  - RAS: ABEND026 dumps collected
  - CFSTRHANGTIME(900-1200)

67

SHARE in Orlando 2011

Connectors to CF structures need to participate in various processes and respond to relevant events

XES monitors the connectors to ensure that they are responding in a timely fashion

If not, XES issues messages (IXL040E or IXL041E) to report the unresponsive connector (outstanding responses)

Users of the structure may hang until the offending connector responds or is terminated

Installations often fail to react to these messages, or worse, react by terminating the wrong connector

CFSTRHANGTIME indicates how long the system should allow a structure hang condition to persist before taking corrective action(s) to remedy the situation

Corrective actions may include:

Stopping rebuild

Forcing the user to disconnect

Terminating the connector task, address space, or system

Each system acts upon its own connectors

IXL040E CONNECTOR NAME: connector-name, JOBNAME: jobname, ASID: asid HAS *text. process* FOR STRUCTURE *structure-name* CANNOT CONTINUE. | MONITORING FOR RESPONSE STARTED: *mondate montime*. DIAG: *x*

IXL049E HANG RESOLUTION ACTION FOR CONNECTOR NAME: *conname* TO STRUCTURE | *strname*, JOBNAME: *jobname*, ASID: *asid*: *actiontext*

IXL041E CONNECTOR NAME: connector-name, JOBNAME: jobname, ASID: asid HAS NOT RESPONDED TO THE event FOR SUBJECT CONNECTION: subject-connector-name. process FOR STRUCTURE structure-name | CANNOT CONTINUE. MONITORING FOR RESPONSE STARTED: mondate | montime. DIAG: x

IXL050I CONNECTOR NAME: *conname* TO STRUCTURE *strname*, JOBNAME: *jobname*, | ASID: *asid* HAS NOT PROVIDED A REQUIRED RESPONSE AFTER | *noresponsetime* SECONDS. TERMINATING *termtarget* TO RELIEVE THE | HANG.

Detection of Soft Failures in a Sysplex: SFM

BCPii: Avoid waiting the FDI+ if the system is truly dead !

- BCPii allows XCF to query the state of other systems via authorized interfaces through the support element and HMC network
- Benefits:
  - XCF can detect and/or reset failed systems more quickly
  - Works in scenarios where fencing cannot work
    - CEC checkstop or powered down
    - Image reset, deactivated, or re-IPLed
    - No CF
  - Eliminates the need for manual intervention, which may lead to data corruption problems
  - **Reduction in sympathy sickness time**
  - ***Set this up. It is a critical component of Resiliency AND Soft Failure Avoidance***

68

SHARE in Orlando 2011

SFM will automatically exploit BCPii and as soon as the required configuration is established. (a) Pairs of systems running z/OS 1.11 or higher (b) BCPii configured, installed, and available (c) XCF has security authorization to access BCPii defined FACILITY class resources (d) z10 GA2 with appropriate MCL's, or z196 (e) New version of the sypslex CDS is primary in the sysplex (f) toleration APAR OA26037 for z/OS 1.9 & 1.10 (g) SYSSTATE DETECT function is not enabled.

Detection & Prevention of Soft Failures in a Sysplex: Critical Member support

- A Critical Member is a member of an XCF group that identifies itself as "critical" when joining its group
- If a critical member is "impaired" for long enough, XCF will eventually terminate the member
  - Per the member's specification: task, space, or system
  - SFM parameter MEMSTALLTIME determines "long enough" before terminating the stalled member with the highest quantity of backed up signals
- **GRS declares itself a "critical member"**
  - If GRS cannot perform work for as long as the FDI, GRS is said to be "impaired"
  - XCF will remove a system from the sysplex if GRS on that system becomes "impaired" (key tasks not operating) to avoid sympathy sickness
    - Based on SFM MEMSTALLTIME(n)
    - For MEMSTALLTIME(NO), N=MAX(FDI, 120 seconds)

69

SHARE in Orlando 2011

Critical member is a member (component) of an XCF group identified to be "critical"

If GRS cannot perform its work for as long as the failure interval, it is marked "impaired"; GRS indicated that it is to be removed from the sysplex to avoid sympathy sickness. The monitoring includes work queues, units of working being able to be dispatched, and XCF messages that appears hung due to GRS not processing messages. The messaging support also include GRS STAR lock structure signal required for contention management.

SFM MEMSTALLTIME

MEMSTALLTIME enables the system to break out of an XCF signalling jam; specifies action XCF will take: terminate the stalled member with the highest quantity of signals backed up

"Back stop" to allow the system to take automatic action to alleviate a problem if it cannot be resolved in a timely manner

Health Check details

Health Checker: Soft Failure avoidance examples

- **DAE_SUPPRESSING**
  - DAE suppresses duplicate SVC dumps so that system resources (processor cycles and dump space) are not used for a dump which provides little or no additional diagnostic value
  - IBM recommendation is to activate this function.
    - If turned off, then health checker will issue an exception to alert the team to this sub optimal configuration.
- **XCF_CDS_SPOF**
  - z/OS uses two coupling data sets (CDS) to manage a parallel sysplex, primary and alternative.
  - This check evaluates the I/O configuration to determine if the I/O configuration has inadvertently created a single point of failure (SPOF) when accessing the data on the primary and alternative CDS.
  - Alternative CDS created to handle a problem with a switch or a storage device.
- **SVA_AUTOIPL_DEFINED, SVA_AUTOIPL_DEV_VALIDATION**
  - Check whether environment can support AUTOIPL, whether active
  - Validates SADMP, MVS IPL devices

SHARE in Orlando 2011

Examples of Soft Failure avoidance

•DAE

•CDS Single points of failure

•AutoIPL defined and valid DASD devices specified for Stand Alone Dump & MVS IPLs

Health Checker: Soft Failure avoidance examples

- **System Logger**
  - IXGLOGR_STRUCTUREFULL
    - Primary structure full; need to offload
  - IXGLOGR_ENTRYTHRESHOLD
    - High number of entries in element pools
  - IXGLOGR_STAGINGDSFULL
    - Full staging data space
- **UNIX System Services**
  - USS_PARMLIB
    - Validate current system against parmlib IPL'd with
    - Remind you to update parmlib (due to dynamic changes)
  - USS_CLIENT_MOUNTS
    - With Sysplex, some file systems accessed locally, some of function shipped to the File system owner. Some are accessed locally, but are configured to function ship
    - Check if function ship but could be done locally (performance awareness)
  - USS_FILESYS_CONFIG
    - Checks if mount attribute access is read only
    - HFS's in Sysplex root
- **Sysplex Failure management**
  - Examines / validates SFM values
    - XCF_SFM_ACTIVE
    - XCF_SFM_CFSTRHANGTIME
    - XCF_SFM_CONNFAIL
    - XCF_SFM_SSUMLIMIT
    - XCF_SFM_SUM_ACTION

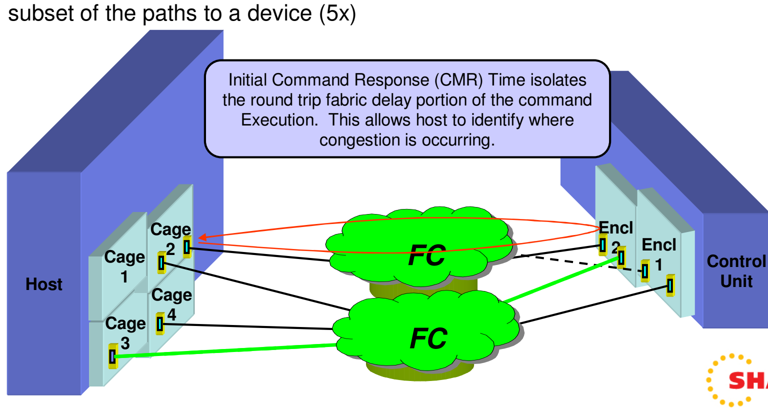
72

SHARE in Orlando 2011

Examples of Soft Failure avoidance

Health Checker: Soft Failure avoidance examples

- IOS_CAPTUCB_PROTECT
  - UCB capture protection is enabled, allowing UCBs to be temporarily copied to 24-bit storage for legacy software access
- IOS_CMRTIME_MONITOR
  - Fabric issues have resulted in unacceptable I/O service times
    - RMF device activity reports show average service times to be higher than normal
    - I/O queuing reports show abnormally high “initial command response” times on a subset of the paths to a device (5x)



73

SHARE in Orlando 2011

Examples of Soft Failure avoidance

IOS Captured UCB protection verifies that Captured UCB protection is enabled on this system. This allows UCBs to be temporarily copied to 24-bit addressable storage to allow access by Legacy software in the first 16 Mb of storage. This ensures that legacy software cannot interfere with the state of the devices

CMR time monitor detects if any control units in the system are reporting inconsistent average initial “command response” (CMR) time for their attached channel paths. An exception is raised if at least one control unit in the system has a path with an average CMR time that is highest among the other paths to the control unit, greater than a specified thresholds.

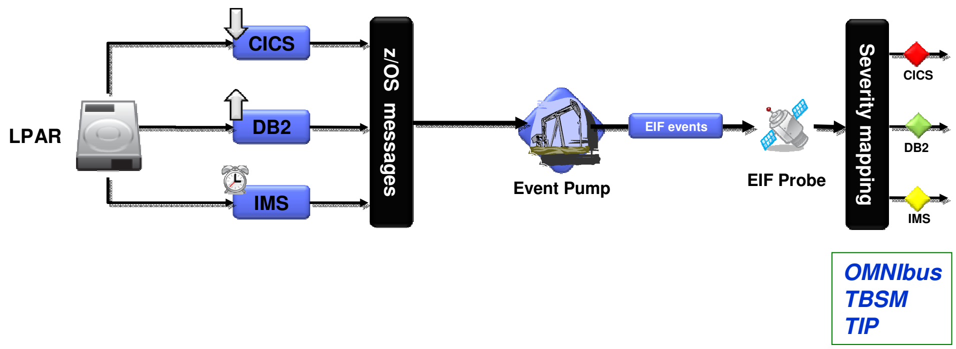
Tivoli Management Products

- Tivoli Management Products integrate
  - Soft Failures detected by PFA
    - Health Check exceptions surfaced by zMC (to be supported on *Omegamon XE*)
    - *Tivoli System Automation* policy to control of corrective actions
  - Performance issues detected by *Omegamon*
    - Evaluate entire software stack
    - Customer-defined model, selection of critical events
  - *Netcool/OMNIbus* provide centralized monitoring of Health Check Alerts, Performance, Situations, Network activity, etc.

Event Pump for z/OS

Each subsystem writes <u>messages</u> to z/OS

- These messages may contain **<u>state</u>** and **<u>status</u>** information
- The Event Pump parses the message, interprets the resource information, and converts the message to an EIF event



75

SHARE in Orlando 2011

Event Pump is a fairly new z/OS agent that extracts information based on messages, System Automation events, Netview PPI, Health Checker … and represents the events in terms of state & status, via EIF (event information FW).

EIF (Event Integration Facility) communicates with other (distributed) Tivoli products, as well as other vendor products.