Constructing Mission Critical Solutions using Superdome

Ken Pomaranski
Hewlett Packard Company
ken_pomaranski@hp.com

Delivering High Availability is much more than just providing a few token HA features in hardware. The entire end-to-end solution must be considered before a system can be labeled as acceptable for mission critical environments.

High Availability is…

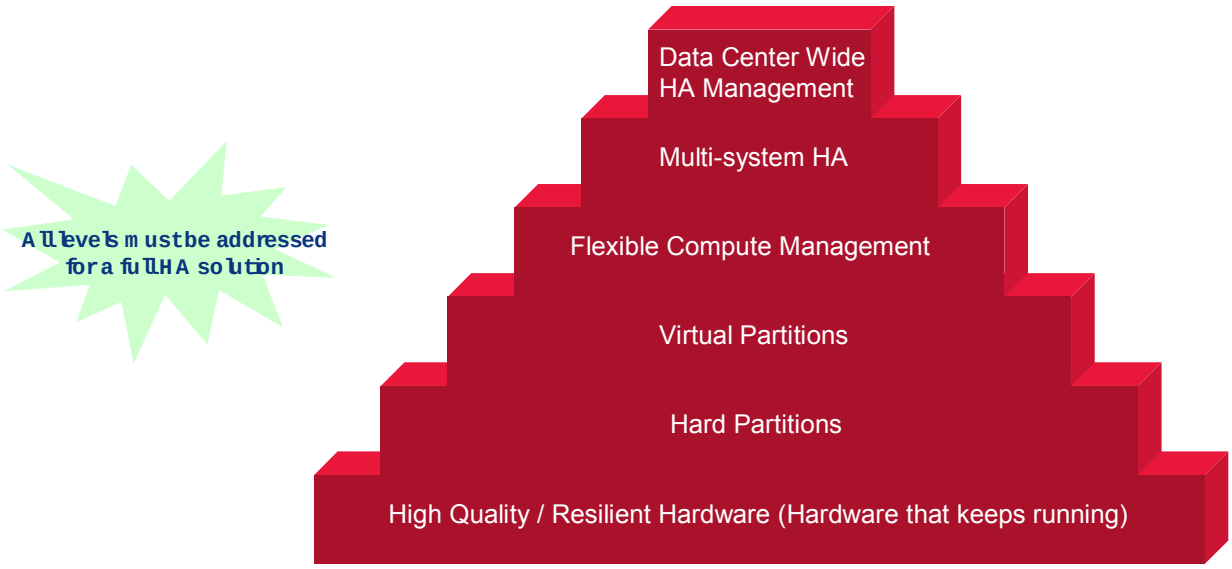
- **built, managed, and measured**
- **hardware, system software, applications & middleware, and IT processes designed to minimize both planned and unplanned downtime**

The ‘HA pyramid’ demonstrates this best.

Each level of the pyramid is nothing without the level below it. For instance, what good is hard partitioning if the system underneath it fails at an unacceptable rate? What good are virtual partitions if you cannot section them off into hard partitions when extra isolation is required? (For example, when HW isolation is desired between a production partition and a development partition.)

What good is any system without a management scheme (system and fault) that doesn’t seamlessly connect and integrate with all servers in a data center?

The High Availability Pyramid

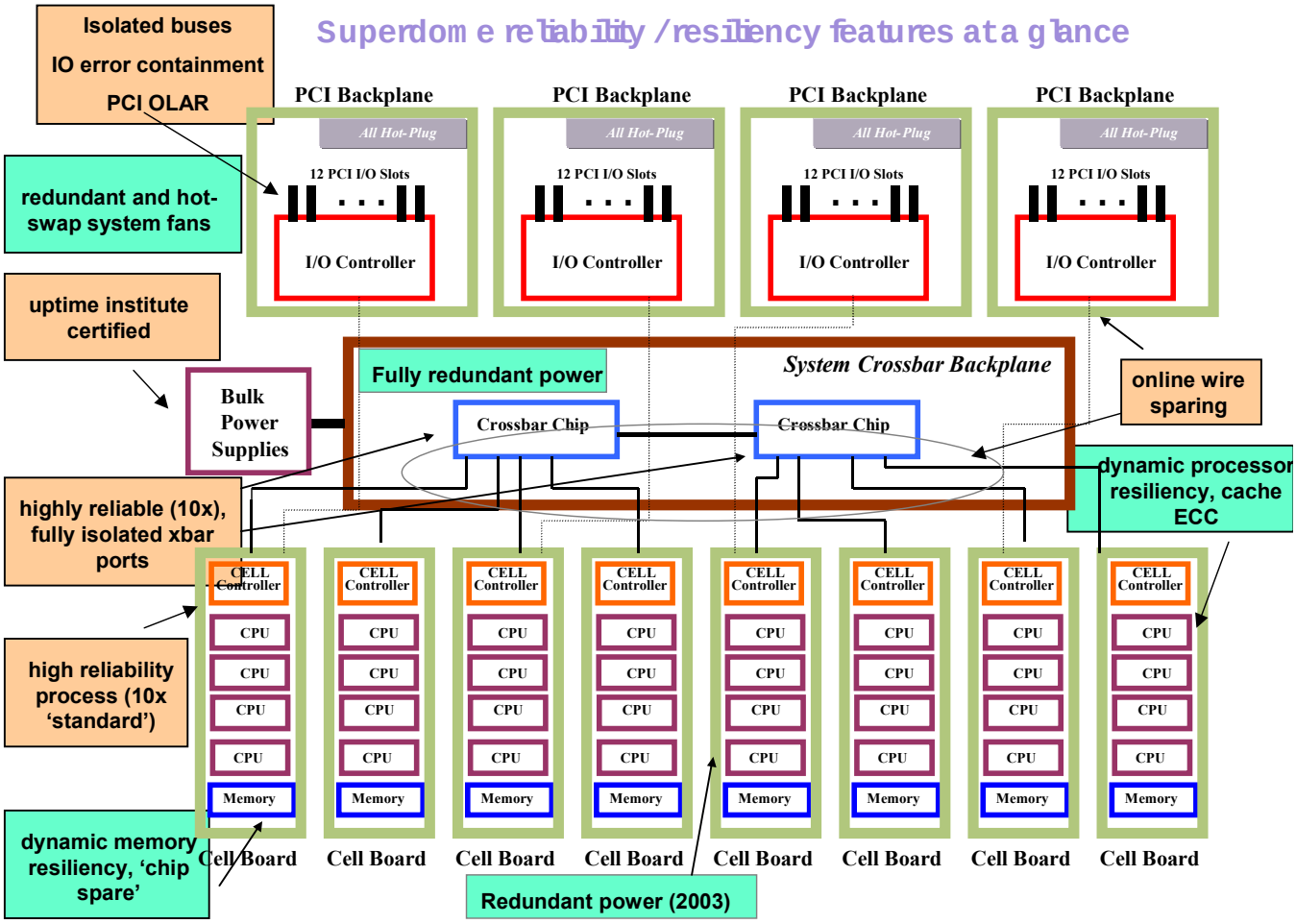


What we will discuss today:

I will focus on the very bottom and very top of the pyramid

- Setting up Superdome to deliver max Single System HA (SSHA)
- Reducing planned downtime & downtime due to user error across the data center
- Measurement of Availability

I will discuss HP's solutions in these areas, specifically EMS and HAO (high availability observatory). Specific tips / recommendations on configuring these tools will be presented.

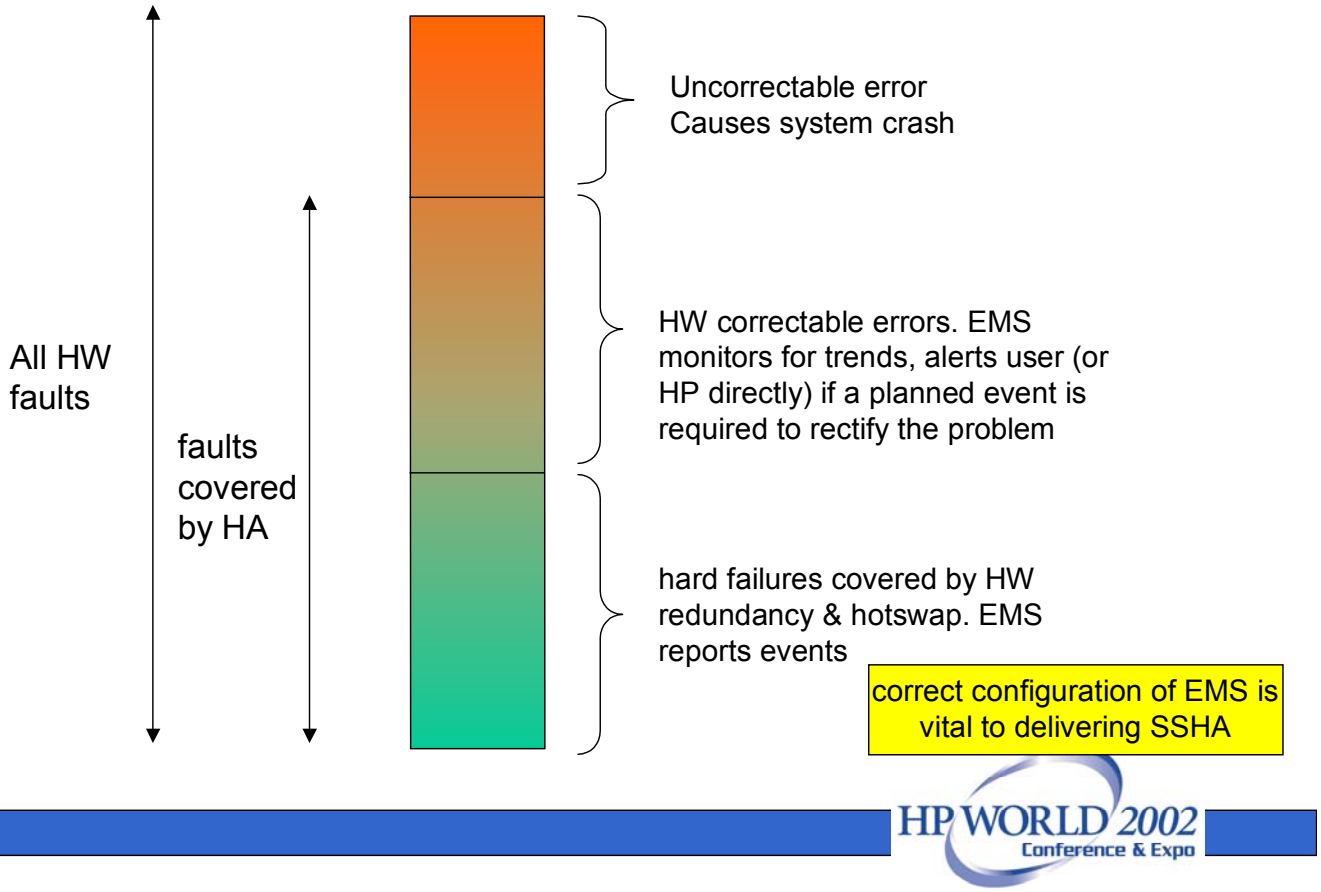


A computer can be divided into 4 basic subsystems:

1. CPU
2. Memory
3. IO
4. Cabinet infrastructure

Superdome has HA features that address each of these subsystems. Each has been specifically designed (using field data) to reduce the REAL causes of downtime.

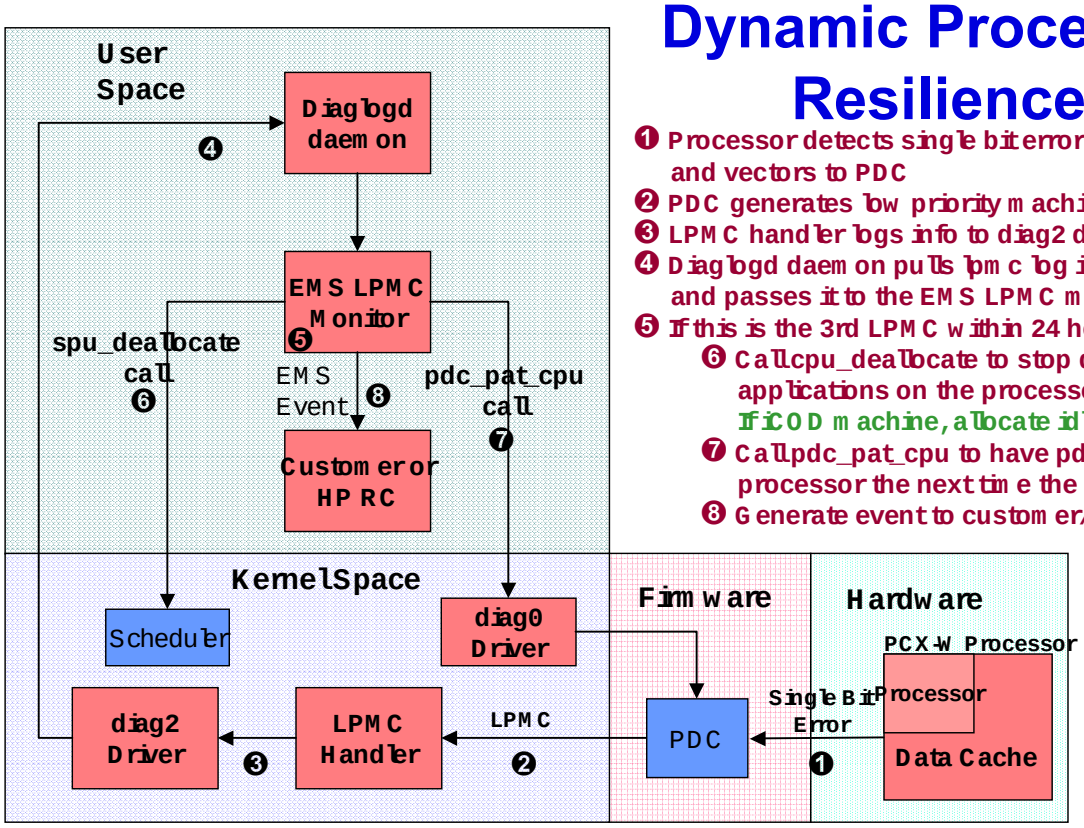
Fault Tolerant Bar graph



In HP systems, the hardware works in tandem with EMS to cover most system errors. It is estimated that over 90% of all system errors are covered by HA features. (Achieving 100% requires fault tolerant computers.) It is therefore imperative that EMS is setup and configured correctly.

This portion of the paper will discuss two specific cases in which EMS is an integral part of the solution, then discuss some EMS tricks / tips.

Dynamic Processor Resilience



❶ Processor detects single bit error in data cache and vectors to PDC
❷ PDC generates low priority machine check (LPMC)
❸ LPMC handler logs info to diag2 driver
❹ Diaglogd daemon pulls lpmc log info from diag2 and passes it to the EMS LPMC monitor
❺ If this is the 3rd LPMC within 24 hours:
  ❻ Call cpu_deallocate to stop dispatching applications on the processor
  If iCOD machine, allocate idle processor
  ❼ Call pdc_pat_cpu to have pdc disable the processor the next time the system boots
  ❽ Generate event to customer/HP

DPR makes the system fully resilient to CPU cache errors which is one of the greatest contributors to system downtime. Cache errors contribute 80% of total CPU hardware errors.

Dynamic Memory Resilience (DMR)

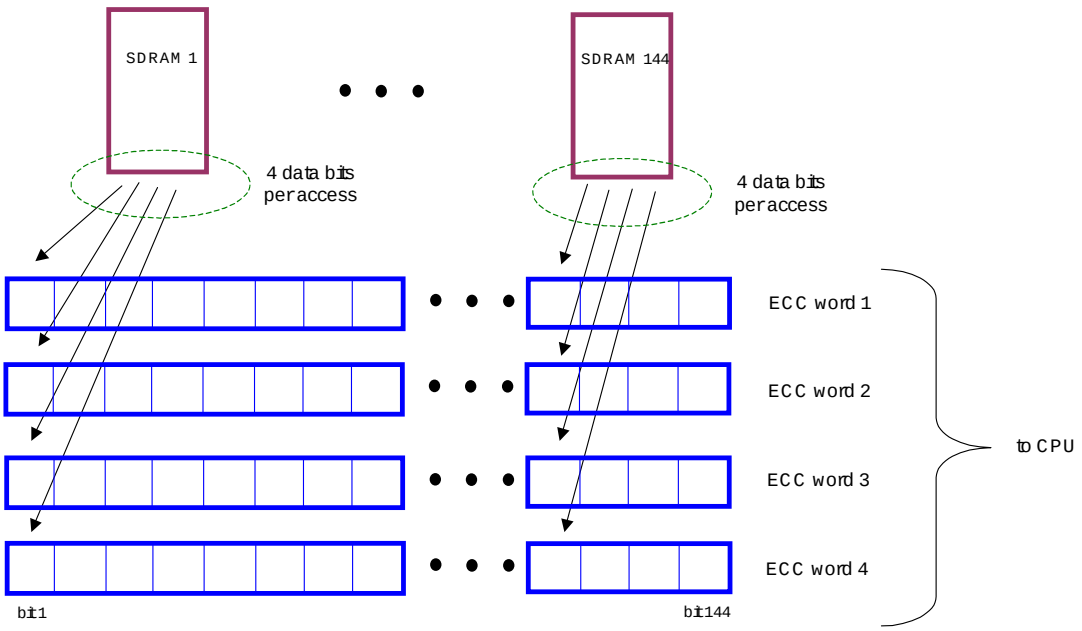
Main memory failures are demonstrated to be the second largest cause of customer downtime. Great care has been taken to address this failure mode in Superdome with these specific features:

- **Memory 'chip spare':** the ability of the system to continue to run in the face of any single or multi-bit chip error on a DRAM.
- **Dynamic memory resiliency (DMR)**: is the system's ability to de-allocate failed memory pages *online*. It works similar to Dynamic Processor Resiliency in that if a location in memory proves to be 'questionable' (i.e., exhibits persistent errors), that memory will be de-allocated online, with no customer visible impact.
- **HW memory scrubbing:** refers to the HW feature that automatically removes single bit errors (SBE) that reside in main memory.

The combination of these features have nearly eliminated memory as a cause of downtime in HP systems.

The memory subsystem design is such that a single SDRAM chip does not contribute more than 1 bit to each ECC word. The memory system is built with x4 SDRAMS. Each main memory access results in 576 bits being read, 4 bits each from 144 SDRAMS. The 576 bits are divided into 4 separate ECC domains, with each SDRAM contributing only *one bit* to each ECC domain. Each ECC domain can correct one bit in error. Therefore, the ONLY way to get a multiple bit memory error from SDRAMS is if more than one SDRAM failed at the same time, which statistically will not occur.

The system is also resilient to any cosmic ray or alpha particle strike because, at the most, these failure modes can only affect multiple bits in a *single* SDRAM. These types of failures are also avoided with this architecture, again since no SDRAM contributes more than one bit to an ECC domain. The memory is effectively RAID'd or stripped, similar to high-end disk storage systems. Data is protected!



This scheme is sometimes called 'chip-kill' in the industry. 'Chip-spare' or 'dynamic memory resiliency' are the terms used by HP, since it more accurately portrays the fact that the memory chips can be thought of as being redundant. N+1 per set of 4 DIMMS.

Some in the industry deal with multi-bit SDRAM failures by *accepting* the fact that they will occur. That is, a scheme is used that supports failure detection, but not correction. This scheme, while it may be acceptable in low-end markets, is a dangerous choice for those customers that are 'betting their business' on not having any downtime. *Systems based upon this 'corner cutting' are at high risk to fail due to memory problem.*

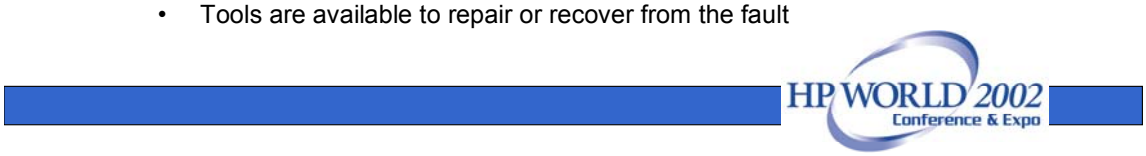
Dynamic page de-allocation (DMR) works by 'sparing-out' memory pages that have hard single bit errors. This prevents possible data corruption / system crashes due to a cosmic ray event that just happens to occur in the same memory word as the hard error. There are 50 - 200 spare pages available. This compares favorably with other schemes in the industry, which use hardware based schemes that are costly and are limited by the number of spares DRAMS.

Further aiding in this process is the 'HW memory scrubber', which automatically corrects and clears single-bit errors in every memory line that is read by the CPUs. This is an advantage over most SW based scrubbers, which are limited to 'scrubbing' only that memory which is not 'locked down' by the OS or an application. In competitive systems, a single bit memory error can be a ticking time bomb that can blow if a subsequent single bit memory error occurs in the same memory rank.

HP Fault Management

Increase system availability by moving from **reactive** fault detection, diagnosis and repair to **proactive** fault detection, diagnosis and repair.

- Detect problems automatically as close as possible to when they actually occur
- Diagnose problems automatically at the time of detection
- Automatically report in understandable text:
  - A description of the problem
  - The likely cause(s) of the problem
  - The recommended action(s) to resolve the problem
  - Detailed information about the problem
- Tools are available to repair or recover from the fault

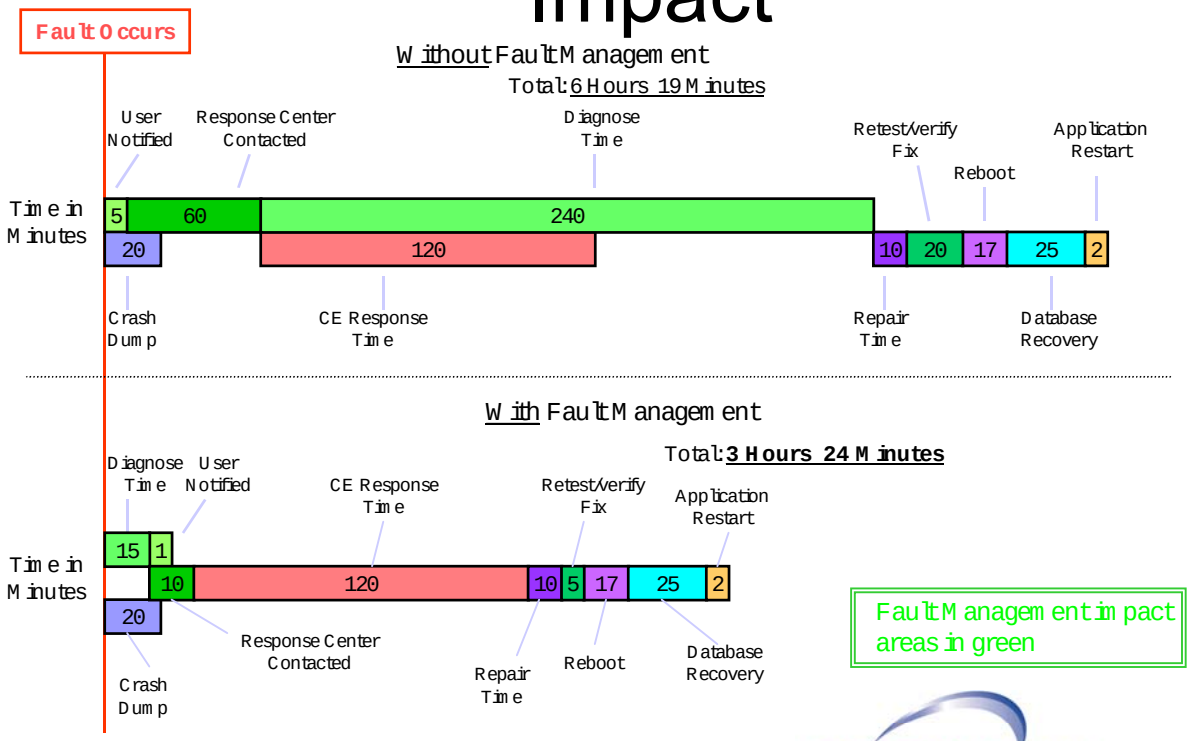


Fault Management provides immediate alerts of potential problems, as well as real problems as soon as they are detected. Customers are then able to take corrective action. In some cases fault monitors are smart enough to repair or prevent future faults from occurring. Fault Management currently uses the EMS (Event Management System) infrastructure for its notification methodology. EMS enables a wide variety of notification methods (pager, email, SNMP traps, system console, system log, text logfile TCP/UDP, OpenView OPC messaging).

Fault Management events can themselves be viewed and browsed directly by the customer on the actual system, or the customer can install HP's TopTools Management server and aggregate information from multiple systems in the data center.

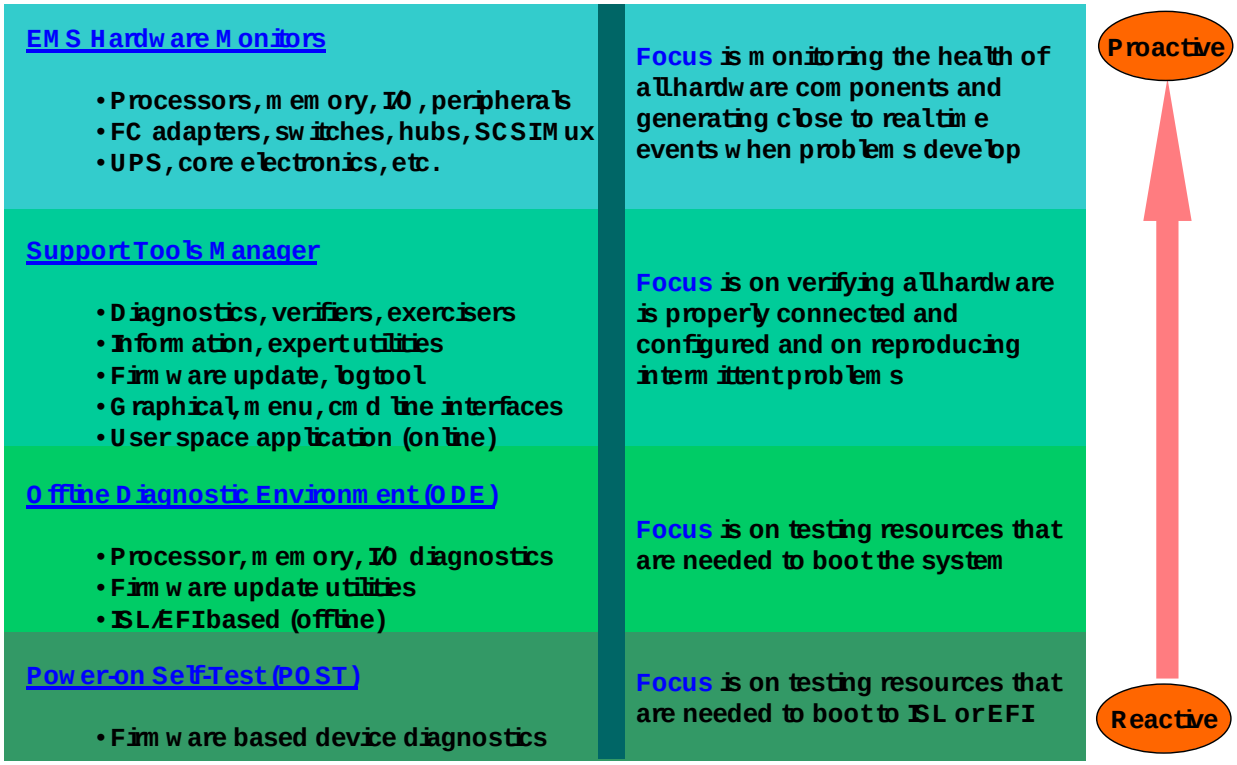
Customers also have the option to integrate Fault Management events with Enterprise Management software like HP's OpenView or other Enterprise Management software from BMC, Tivoli, Computer Associates or MicroMuse.

Fault Management Impact

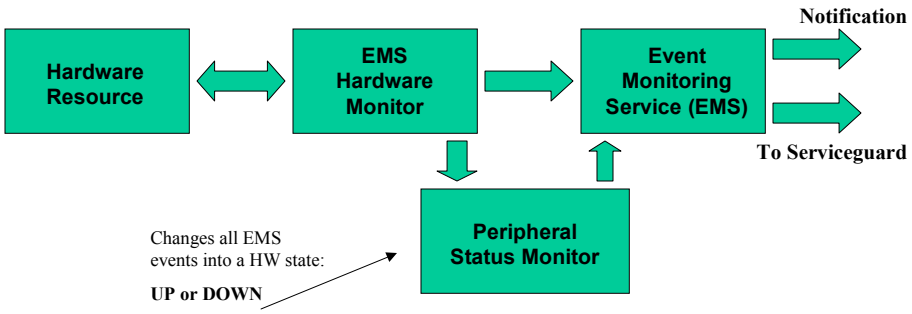


Fault Management impact areas in green

Hardware Troubleshooting Tools



Hardware monitoring



The typical hardware monitoring process works as follows:

1. While monitoring its hardware resources, the hardware event monitor detects some type of abnormal behavior on one of the resources.

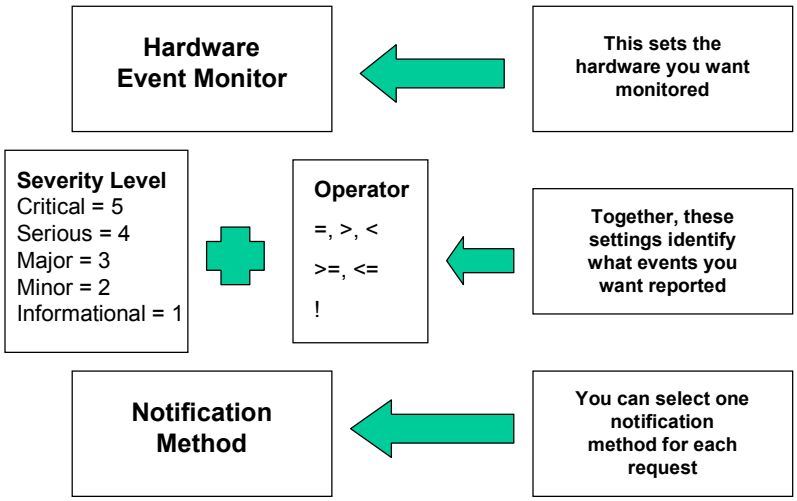
2. The hardware event monitor creates the appropriate event message, which includes suggested corrective action, and passes it to the Event Monitoring Service (EMS).

3. EMS sends the event message to the system administrator using the notification method specified in the monitoring request.

4. The system administrator (or Hewlett-Packard service provider) receives the messages, corrects the problem, and returns the hardware to its normal operating condition.

5. If the Peripheral Status Monitor (PSM) has been properly configured, events are also processed by the PSM. The PSM changes the device status to DOWN if the event is serious enough. The change in device status is passed to EMS, which in turn alerts MC/ServiceGuard. The DOWN status will cause MC/ServiceGuard to failover any package associated with the failed hardware resource.

Building a Monitoring Request



Example of Adding a Monitoring Request

The following example illustrates the process of adding a monitoring request. In this example a request is added that will send all CRITICAL events detected by the AutoRAID disk array monitor to an email address of admin@hp.com.

```
================================================================
========== Monitoring Configuration Main Menu =============

Select:
(S)how current monitoring requests configured via monconfig
(C)heck detailed monitoring status
(L)ist descriptions of available monitors
(A)dd a monitoring request
(D)elete a monitoring request
(M)odify an existing monitoring request
(E)nable Monitoring
(K)ill (disable) monitoring
(H)elp
(Q)uit
Enter selection: [s] a <== SELECT ADD OPTION

================================================================================
==================== Add Monitoring Request ====================

Start of edit configuration:


A monitoring request consists of:
- A list of monitors to which it applies
- A severity range (A relational expression and a severity. For example,
%< "MAJOR WARNING" means events with a severity "INFORMATION" and
"MINOR WARNING")
```

```
- A notification method

Please answer the following questions to specify a monitoring request.

Monitors to which this configuration can apply:
       1) /storage/events/disk_arrays/AutoRAID
       2) /storage/events/disks/default
       3) /adapters/events/FC_adapter
       4) /connectivity/events/multiplexors/FC_SCSI_mux
       5) /storage/events/enclosures/ses_enclosure
       6) /storage/events/tapes/SCSI_tape
       7) /storage/events/disk_arrays/FW_SCSI
       8) /storage/events/disk_arrays/High_Availability

Enter monitor numbers separated by commas
{or (A)ll monitors, (Q)uit, (H)elp} [a] 1 <== SELECT AUTORAID MONITOR

Criteria Thresholds:
       1) Informational 2) Minor Warning 3) Major Warning
       4) Serious 5) Critical
Enter selection {or (Q)uit,(H)elp} [4] 5 <== SELECT ONLY CRITICAL EVENTS

Criteria Operator:
       1) %< 2) %<= 3) > 4) >= 5) = 6) !
Enter selection {or (Q)uit,(H)elp} [4] 5 <== (=CRITICAL)

Notification Method:
       1) UDP 2) TCP 3) OPC 4) SNMP
       5) TEXTLOG 6) SYSLOG 7) EMAIL 8) CONSOLE
Enter selection {or (Q)uit,(H)elp} [7] <== SELECT EMAIL ADDRESS

Enter Email Address: [root] admin@hp.com admin@hp.com

User Comment:
       (C)lear (A)dd
Enter selection {or (Q)uit,(H)elp} [c] a <== ADD COMMENT

Enter comment: [] This is a test message. IF DESIRED

Client Configuration File:
       (C)lear (A)dd
       Use Clear to use the default file.
Enter selection {or (Q)uit,(H)elp} [c] c <== SPECIFY CLCFG FILE IF DESIRED (USUALLY
CHOOSE DEFAULT)

New entry:

Send events generated by all monitors
/storage/events/disk_arrays/AutoRAID <== NEW MONITORING with severity = CRITICAL to EMAIL
admin@hp.com REQUEST with comment: “This is a test message”

Are you sure you want to keep these changes?
{(Y)es,(N)o,(H)elp} [n] y
```

Notification Methods

- **EMAIL*** - sends notification to the specified email address
- **TEXTLOG*** - sends notification to specified file
- **SNMP** - sends notification using SNMP traps
- **CONSOLE** - sends notification to the system console
- **TCP** - sends notification to the specified target host and port
- **UDP** - sends notification to the specified target host and port
- **OPC** - sends notification to OpenView ITO applications (available only on systems with OpenView installed).
- **SYSLOG** - sends notification to the system log

Only one notification method can be selected for each monitor request, consequently you will need to create multiple requests to direct event notification to different targets. Those notification methods denoted by a '* ' are the only methods that deliver the entire content of the event message.

Modifying Monitoring Requests

Modifying an existing monitoring request is a convenient way to alter one of the settings used in the request. Simply select a monitoring request and then change the desired setting. All other aspects of the request remain unchanged.

To modify a monitoring request:

1. Run the Hardware Monitoring Request Manager by typing:
   `/etc/opt/resmon/lbin/monconfig`

2. From the main menu selection prompt, enter **M**
   All current monitoring requests are displayed.
3. From the list of current monitoring requests, enter the number of the request you want to modify.

4. As you are prompted for each monitoring request setting, change the settings to achieve the desired results.

5. Save the request when prompted.

Retrieving and Interpreting Event Messages

Event messages generated by hardware monitoring can be delivered using a variety of notification methods.
To simplify receiving event messages you may want to use the email and/or textfile notification methods.

Both of these methods, which are included in the default monitoring, receive the entire content of the message so you can read it immediately.

Methods such as console, syslog, and SNMP alert you to the occurrence of an event but do not deliver the
entire message. You are required to retrieve it using the resdata utility. For these methods, the event
notification will include a message similar to the following:

```
Execute the following command to obtain event details: /opt/resmon/bin/resdata
-R 392036357 -r /storage/events/tapes/SCSI_tape/10_12_5.0.0 -n 392036353 -a
```

It is important that you execute the command exactly as indicated, including the two critical number fields that are indexes for the resdata entries.

Sample Event Message

The following is a portion of a sample event message:

```
> Event Monitoring Service Event Notification %<
Notification Time: Wed Sep 9 10:48:30 1998

hpbs8684 sent Event Monitor notification information:
/storage/events/disks/default/10_4_4.0.0 is >= 1.
Its current value is CRITICAL(5).

Event data from monitor:
        Event Time : Wed Sep 9 10:48:30 1998
        Hostname : hpbs8684.boi.hp.com IP Address : 15.62.120.25
        Event Id : 0x0035f6b15e00000000 Monitor : disk_em
        Event # : 100037 Event Class : I/O
        Severity : CRITICAL
        Disk at hardware path 10/4/4.0.0 : Media failure

Associated OS error log entry id(s):
000000000000000000

Description of Error:
The device was unsuccessful in reading data for the current I/O request due to an error
on the medium. The data could not be recovered. The request was likely processed in a way
which could cause damage to or loss of data.

Probable Cause / Recommended Action:
The medium in the device is flawed. If the medium is removable, replace
the medium with a fresh one. Alternatively, if the medium is not
removable, the device has experienced a hardware failure.
Repair or replace the device, as necessary.
```

EMS file locations

`/usr/sbin/stm/uut/bin/tools/monitor/monitor_name`	Monitor executable files.
`/var/stm/config/tools/monitor/Global.cfg`	Default monitor configuration file.
`/var/stm/config/tools/monitor/monitor_name.cfg`	Monitor-specific configuration files.
`/var/stm/config/tools/monitor/default_monitor_name.clcfg`	Monitor client configuration file. Only for hardware monitors converted to multiple-view (Predictive-enabled). New as of June 2000 release.
`/var/stm/config/tools/monitor/monitor_name.sapcfg`	Monitor startup configuration files.
`/var/stm/config/tools/monitor/monitor_name.psmcfg`	PSM configuration files.
`/etc/opt/resmon/lbin/monconfig`	Hardware Monitoring Request Manager file
`/etc/opt/resmon/lbin/startcfg_client`	Startup client file
`/etc/opt/resmon/lbin/set_fixed`	PSM set_fixed utility file (Manually returns the operational state of a HW component to 'UP')
`/etc/opt/resmon/dictionary/monitor_name.dict`	Monitor dictionary files

Tips for Hardware Monitoring

Here are some tips for using hardware monitoring.

✓ **Keep hardware monitoring enabled to protect your system from undetected failures.** Hardware monitoring is an important tool for maintaining high-availability on your system. In a high-availability environment, the failure of a hardware resource makes the system vulnerable to another failure. Until the failed hardware is repaired, the backup hardware resource represents a single-point of failure. Without hardware monitoring you may not be aware of the failure. But if you are using hardware monitoring, you are alerted to the failure. This allows you to repair the failure and restore high-availability as quickly as possible.

✓ **Integrate the peripheral status monitor (PSM) into your MC/ServiceGuard strategy.** An important feature of hardware monitoring is its ability to communicate with applications responsible for maintaining system availability, such as MC/ServiceGuard. The peripheral status monitor (PSM) allows you to integrate hardware monitoring into MC/ServiceGuard. The PSM gives you the ability to failover a package based on an event detected by hardware monitoring. If you are using MC/ServiceGuard, you should consider using the PSM to include your system hardware resources in the MC/ServiceGuard strategy. In addition, the necessary notification methods are provided for communicating with network management application such as HP OpenView.

✓ **Utilize the many notification methods available.** The notification methods provided by hardware monitoring provide a great deal of flexibility in designing a strategy to keep you informed of how well your system hardware is working. The default monitoring configuration was selected to provide a variety of notification for all supported hardware resources. As you become familiar with hardware monitoring, you may want to customize the monitoring to meet your individual requirements.

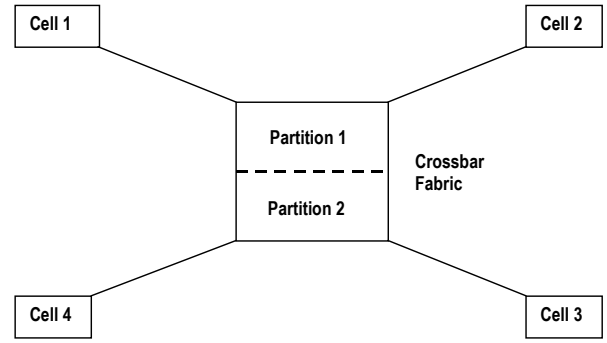
✓ **Use email and/or textfile notification methods for all your requests.** Both of these methods, which are included in the default monitoring, receive the entire content of the message so you can read it immediately. Methods such as console and syslog alert you to the occurrence of an event but do not deliver the entire message. You are required to retrieve the message using the resdata utility, which requires an additional step.

✓ **Use the `All monitors' option when creating a monitoring request.** This applies the monitoring request to all monitors. This has the benefit of ensuring a new class of hardware resource added to your system will automatically be monitored. This means that19new hardware is protected from undetected hardware failure with no effort on your part.

✓ **Easily replicate your hardware monitoring on all your systems.** Once you have implemented a hardware monitoring strategy on one of your system, you can replicate that same monitoring on other systems. Simply copy all of the hardware monitor configuration files to each system that will use the same monitoring. The monitor configuration files live in /var/stm/config/tools/monitor. Of course, you must have installed hardware event monitoring on each system before you copy the configuration files to it. Be sure to enable monitoring on all systems.

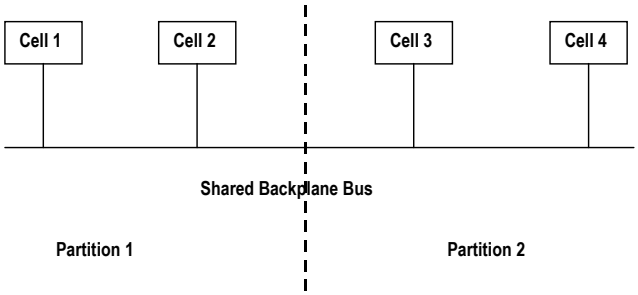
Hard Partition Isolation

HP Architecture



On the HP system, the crossbar logically separates the two physical partitions to provide performance and isolation.

Competitor Architecture



The competitor's shared backplane has all its cells competing for the same electrical bus. In this design, a snoopy bus-coherency scheme requires all transactions to be broadcast to and processed by all system cells. The high-queuing delays and saturation of the shared backplane bus can limit performance scaling

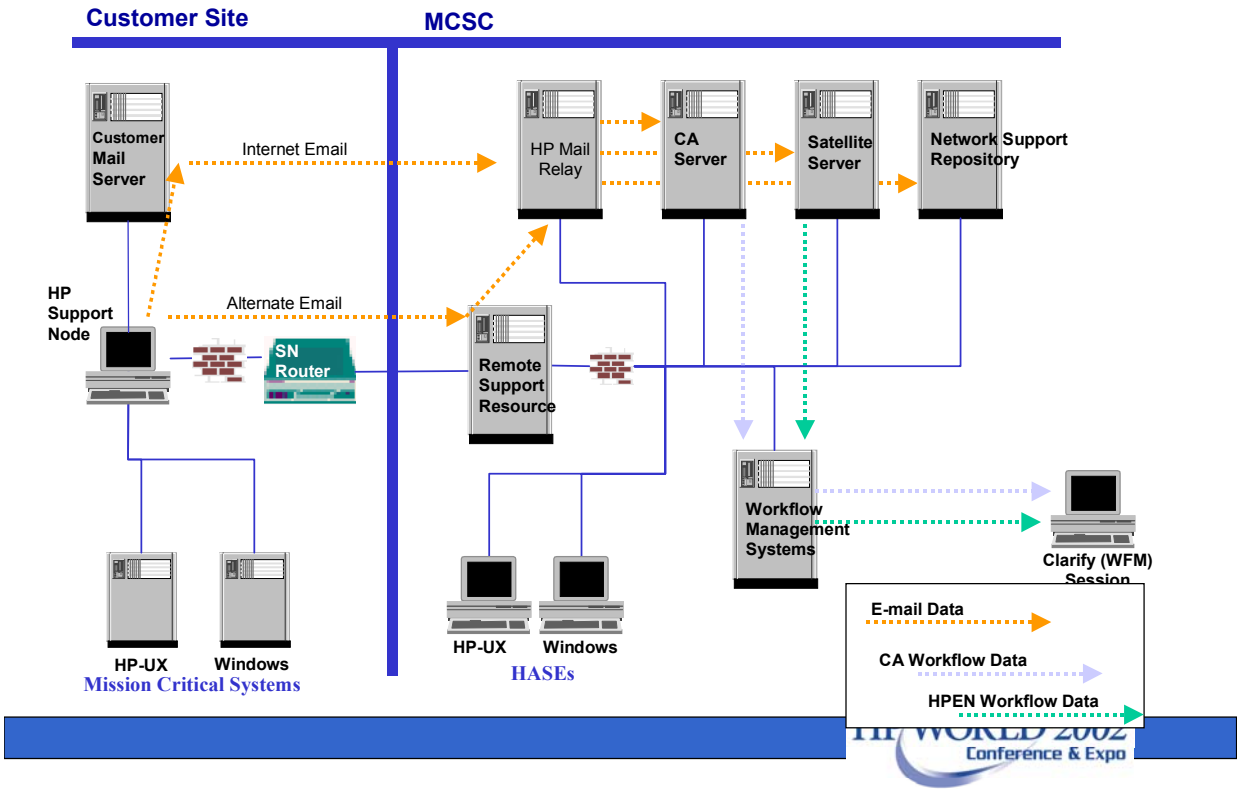
Each hard partition runs its own independent instance of the operating system. Applications cannot span hard partitions since each partition runs its own instance of the OS, essentially functioning as a stand-alone server. However, different partitions may be executing the same or different revisions of an operating system, or they may be executing different operating systems altogether . (In later versions of hard partitions).

Unlike other systems with domains, HP's hard partitions have hardware dedicated to 'guarding' partitions from errant transactions generated on failing partitions. A failure in one domain will not affect any other domains

These features result in a much lower shared failure (Single points of failure) rate between hard partitions than in other systems.

Par-Manager is the tool that is used to setup and configure hard partitions.

HAO Components and Use Model



1. The HAO helps HP provide fast, dependable proactive and reactive support services, which reduces costly downtime in your mission critical environment

2. The HAO significantly reduces unplanned downtime by providing critical information on systems and network inter-connect devices to HP more quickly

3. The HAO helps you to increase the productivity of your network, systems and applications to yield a higher return on your IT investments

4. The HAO raises system availability by proactively addressing potential problem areas

HP Event Types and Information

HP Event Notifier sends the following information to the MCSC:

Type	Timeframe	Transmission Size
Fault Events	Realtime – as occurred or polled interval	5 – 10 Kb
Chassis Code	Polled every ten minutes	Up to 100 Kb (normally smaller)
EMS logs	Polled once per day from client's Mission Critical systems	Up to 500 Kb (normally smaller)

HP Configuration Tracker

Tracker automatically collects configuration data for systems and network interconnect devices. It identifies configuration differences to answer the critical question: "What's Changed?"

Tracker performs the following tasks:

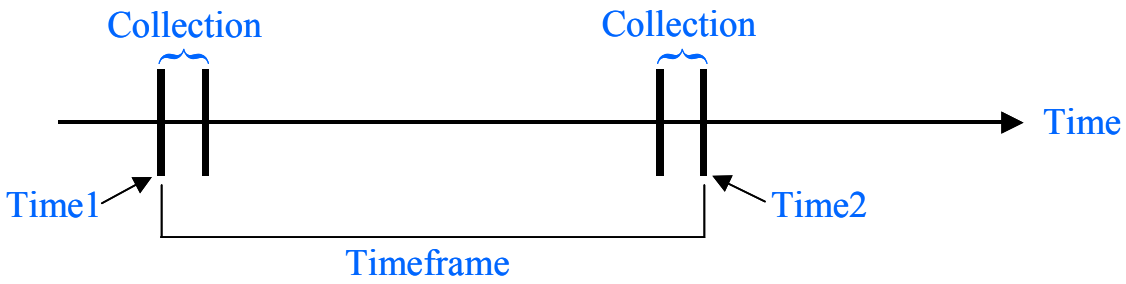
- Automatically collects data daily or weekly.
- Significantly reduces time to gather critical information .
- Allows HP System Recovery Specialist and IT Administrator to view the **same** critical information.
- Transmits hardware, O/S, network interconnect configuration information to MCSC for proactive analysis.
- Creates "user-defined" collectibles to expand collection items.
- Transports configuration data, alarms and log files to the MCSC daily.

HP WORLD 2002 Conference & Expo

- A collection is a snapshot of configuration data for the specified devices. Although a collection generates only one set of data, it can take from a few seconds to a few hours to build this snapshot.

- The Time1 timestamp that appears above the Tracking Tree denotes the time that the Time1 collection began. The Time2 timestamp denotes the time that the Time2 collection ended.

- The timeframe is from the beginning of Time1 to the end of Time2.



HP Configuration Analyzer

The HP Configuration Analyzer (CA) automatically analyzes customer configurations using patch analyzers and notifies the MCSC of potential problems.

CA benefits include:

- Proactive analysis of Application Patch Sets.
- Flexible analysis scheduling for all analyzers.
- Automatic generation of in workflow management cases that notify HP Support Personnel of potential problems.
- Access to customer configuration data at the MCSC.

- Application Patch Analyzers analyze systems and recommend patches based on a specific HP-UX application or group of HP-UX products.
- Use known Good Models (KGMs) to compare against customer configuration information

- Generate cases in Clarify (WFM) and report differences

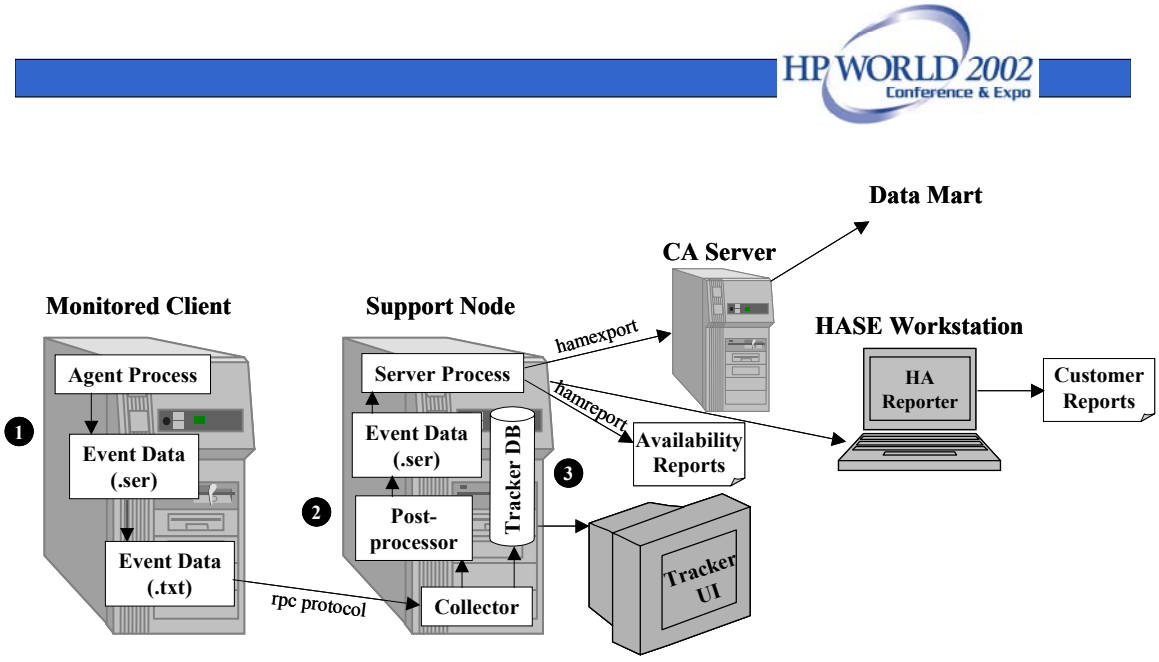
- Review patch summary reports using MCSC Monitor

Defining Availability

HA Meter documents all downtime (planned and unplanned) and quantifies **availability** of a system, cluster, package, or node.

$$\text{Availability} = \frac{\text{(total elapsed time - sum of down times)}}{\text{total elapsed time}} * 100$$

Availability calculations are based on internal timestamps reported in milliseconds, UTC (Universal Time Coordinate).



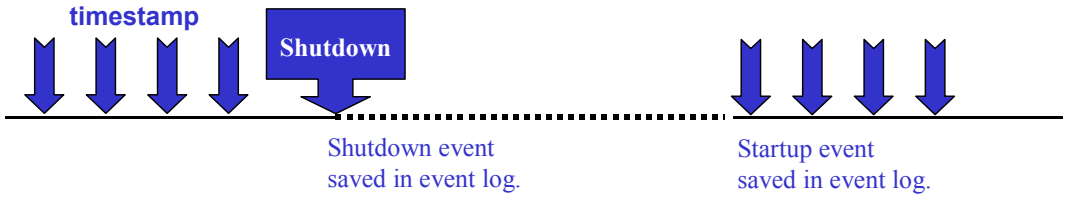
1. When an event occurs, the Agent process creates an event file in .ser format. It then converts that file into .txt format so Tracker can collect it. (The .txt version is deleted after each collection.)

2. The Tracker Collector pulls only new event data from the client via the rpc protocol during a Tracker collection using the HA Meter collectible. The data is then stored in the Tracker database and sent to the HA Meter post-processor, where it is converted back into the .ser format so the Server process can read it.

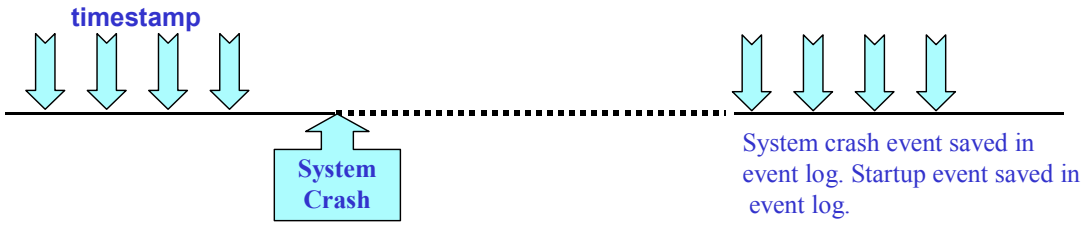
3. Event data in the Tracker database is viewable via the Tracker UI. Event data in the Server process can appear in Availability Reports on the HPSN (for ASEs) and in customer reports via the CA Server at the MCSC using HA Reporter on your workstation.

Shutdown Versus System Crash

Normal Operation (shutdown):



Crash Events:



Execute a Planned Shutdown

- To shut down a system and enter a shutdown cause, follow these basic steps:
  1. Shut down the HA Meter Agent using the `shutdown_ham` command.
  2. Enter the cause code for the system shutdown (cause codes are listed on the following page).
- **NOTE:** This is the only HA Meter procedure that customers may execute on their own.

Select a Shutdown Cause

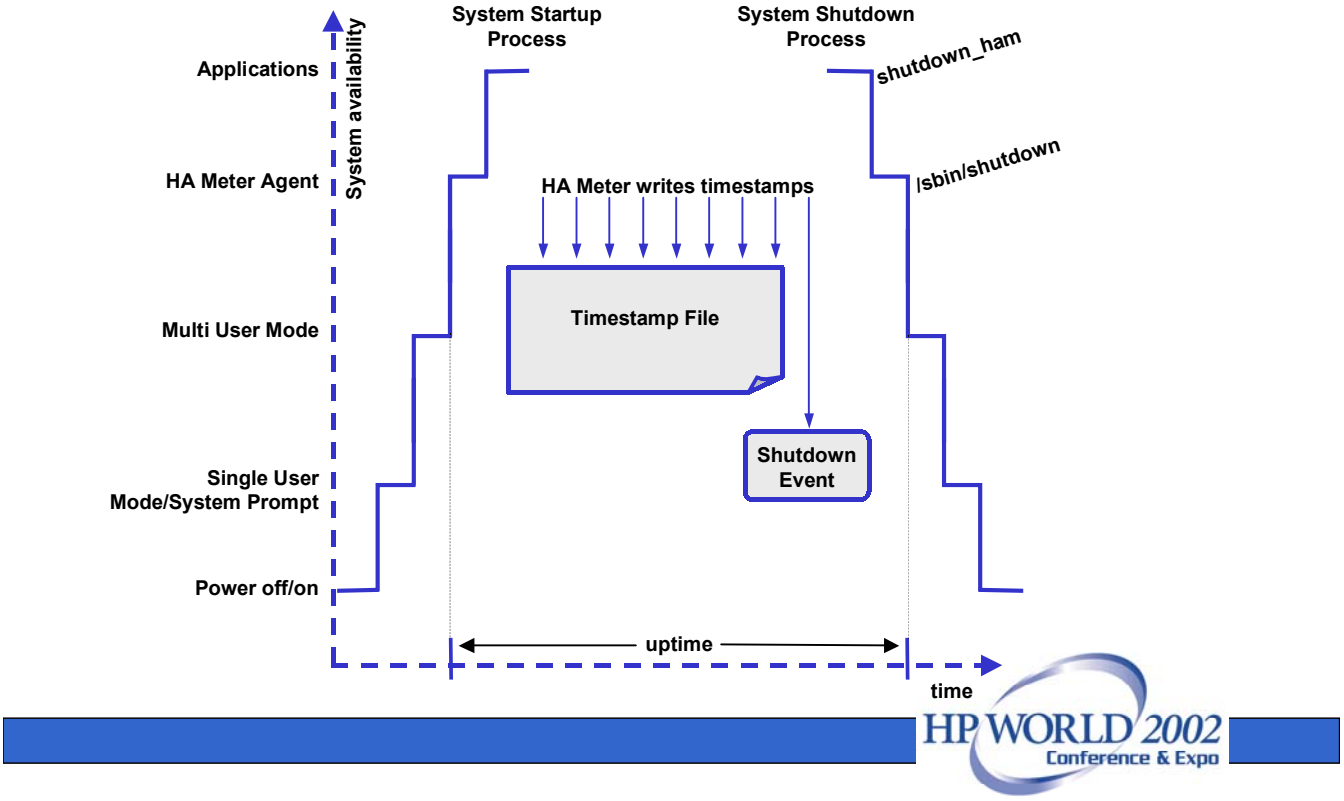
1. Hardware Failure
2. OS Failure
3. Application Failure
4. Middleware Failure
5. Patch/Software Installation
6. Kernel Reconfiguration
7. Hardware Upgrade/Installation
8. Hardware Reconfiguration
9. Scheduled Reboot
10. Other Scheduled Maintenance
11. System Backup
12. Environmental Failure
13. Other (Please Specify)

Defining Downtime

- HA Meter distinguishes planned versus unplanned downtime solely on the basis of whether the `shutdown_ham` (or any standard `shutdown`) command was used to halt the system. If any of these commands are used, the downtime is marked as planned; otherwise it is marked as unplanned. The user may record the cause of the shutdown only by using the `shutdown_ham` command. Planned downtime events also are generated when the user stops the HA Meter Agent process using the `HAMagent` script located in `/sbin/init.d`.

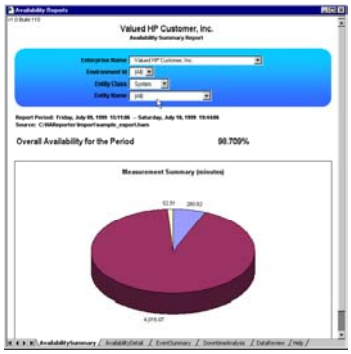
- To produce a customer report, it may be necessary to assign a cause to each downtime event through consultation with the customer. Some downtime may be excluded from the customer report, such as scheduled downtime or downtime resulting from customer error.

Standalone Agent Availability



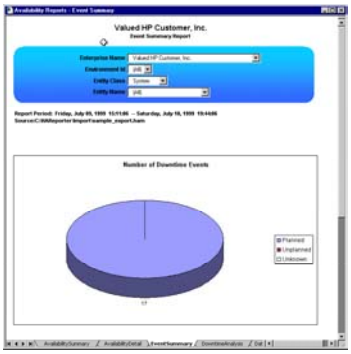
HA Reporter: Summary Data

Three types of summary reports are generated by HA Reporter:



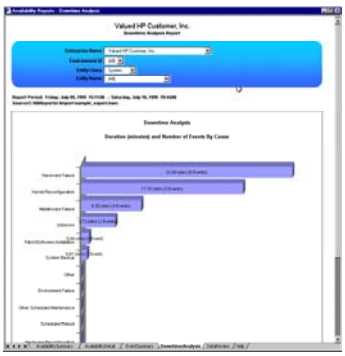
Availability Summary

Displays the aggregate availability, in terms of uptime and downtime.



Event Summary

Displays the number of downtime events and the total amount of downtime, in terms of planned or unplanned.

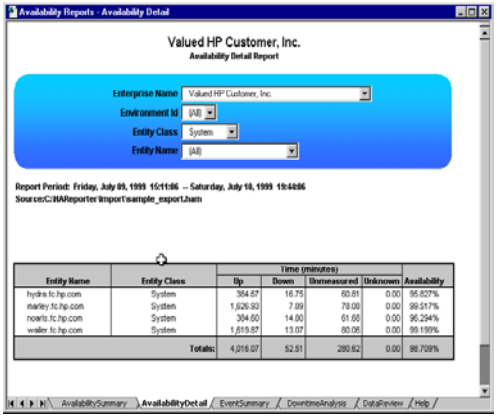


Downtime Analysis

Displays the downtime frequency and duration classified by cause.

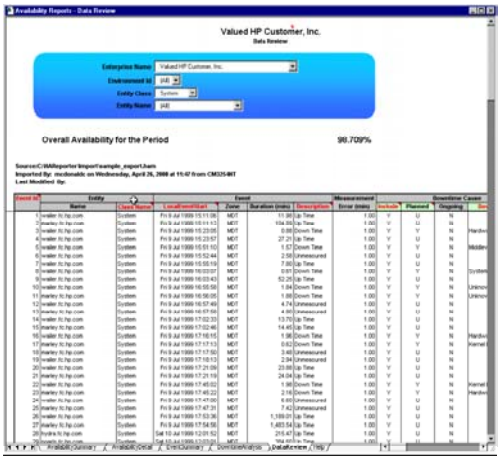
HA Reporter: Detailed Data

Two types of detailed reports are generated by HA Reporter:



Availability Detail

Displays availability, in terms of uptime and downtime, for each entity.



Data Review

Displays all data—start time, duration, type, attributes, root cause—associated with each availability event.

From Wed Dec 31 17:00:00 MST 1969 (HA Meter installation) to Thu Sep 27 14:57:12 MDT 2001.
Printed on Thu Sep 27 14:57:12 MDT 2001 at lonewolf.fc.hp.com
HAMeter revision A.02.50.003

HP-UX System Event History

Event	Date	Entity	Class	Downtime	+?	P?	Notes
1	Tue Aug 14 18:33:27 CEST 2001	ares.olympus.hp.com	system				Began measurement.
2	Tue Aug 14 18:33:47 CEST 2001	zeus.olympus.hp.com	system				Began measurement.
3	Tue Aug 14 18:37:05 CEST 2001	apollo.olympus.hp.com	system		+		Began measurement.
4	Wed Aug 15 06:30:07 CEST 2001	ares.olympus.hp.com	system	2480		N	went down
5	Thu Aug 16 19:03:51 CEST 2001	zeus.olympus.hp.com	system	33509		N	went down
6	Sun Aug 19 21:36:47 CEST 2001	ares.olympus.hp.com	system	5784		N	went down
7	Thu Aug 23 13:19:35 MDT 2001	athena.olympus.hp.com	system				Measurement terminated.
8	Thu Aug 23 13:23:13 MDT 2001	athena.olympus.hp.com	system				Began measurement.
9	Thu Aug 23 13:42:38 MDT 2001	hermes.olympus.hp.com	system				Measurement terminated.
10	Thu Aug 23 13:44:16 MDT 2001	hermes.olympus.hp.com	system				Began measurement.
11	Thu Sep 27 13:41:28 MDT 2001	hermes.olympus.hp.com	system	6		N	went down
12	Thu Sep 27 13:45:17 MDT 2001	athena.olympus.hp.com	system	12		Y	halted: 6

All times in minutes. See note at end about rounding. + = event ongoing at end of reporting period.

Aggregate HP-UX System Availability Report

Number of downtime events:	5
Number of planned downtime events:	1
Number of unplanned downtime events:	4
Aggregate measurement time:	291835
Total downtime:	41790
Total planned downtime:	12
Total unplanned downtime:	41778
HP-UX System availability for the period:	85.680%

Detailed HP-UX System Availability Report

Entity	Accuracy	Entity info	Availability	R?
apollo.olympus.hp.com	+/- 0.5 min	9000/871 500776557 HP-UX B.11.11	100.000%	
ares.olympus.hp.com	+/- 0.5 min	9000/889 529186261 HP-UX B.11.00	87.012%	
athena.olympus.hp.com	+/- 0.5 min	9000/871 2006045407 HP-UX B.10.20	99.976%	
hermes.olympus.hp.com	+/- 0.5 min	9000/871 2003231967 HP-UX B.10.20	99.989%	
zeus.olympus.hp.com	+/- 0.5 min	9000/800 524786597 HP-UX B.11.11	47.332%	

MC/ServiceGuard Cluster Event History

Event	Date	Entity	Class	Downtime	+?	P?	Notes
1	Fri Aug 17 00:06:40 CEST 2001	webserver	cluster				Began measurement.
2	Fri Aug 17 00:20:40 CEST 2001	webserver	cluster				Measurement terminated.
3	Fri Aug 17 00:32:22 CEST 2001	webserver	cluster		+		Began measurement.
4	Thu Aug 23 13:30:10 MDT 2001	cluster10	cluster				Began measurement.
5	Thu Aug 30 17:17:40 MDT 2001	cluster10	cluster	1			Down
6	Thu Sep 27 13:45:17 MDT 2001	cluster10	cluster	72	+		Down

All times in minutes. See note at end about rounding. + = event ongoing at end of reporting period.

Aggregate MC/ServiceGuard Cluster Availability Report

Number of downtime events: 2
Aggregate measurement time: 110886
Total downtime: 73
MC/ServiceGuard Cluster availability for the period: 99.934%

Detailed MC/ServiceGuard Cluster Availability Report

Entity	Accuracy	Entity info	Availability	R?
cluster10	+/- 0.5 min	MC/ServiceGuard A.10.12 2006045407 EMS HA Monitors A.03.20	99.856%	
webserver	+/- 0.5 min	MC/ServiceGuard A.11.09 524786597 EMS HA Monitors A.03.20.01	100.000%	

MC/ServiceGuard Localnode Event History

Event	Date	Entity	Class	Downtime	+?	P?	Notes
1	Fri Aug 17 00:06:40 CEST 2001	webserver/apollo	localnode				Began measurement.
2	Fri Aug 17 00:06:46 CEST 2001	webserver/zeus	localnode				Began measurement.
3	Fri Aug 17 00:20:40 CEST 2001	webserver/apollo	localnode				Measurement terminated.
4	Fri Aug 17 00:20:46 CEST 2001	webserver/zeus	localnode				Measurement terminated.
5	Fri Aug 17 00:32:22 CEST 2001	webserver/apollo	localnode		+		Began measurement.
6	Fri Aug 17 00:32:28 CEST 2001	webserver/zeus	localnode		+		Began measurement.
7	Thu Aug 23 13:30:10 MDT 2001	cluster10/athena	localnode				Began measurement.
8	Thu Aug 23 13:45:38 MDT 2001	cluster10/hermes	localnode				Began measurement.
9	Thu Aug 30 15:39:42 MDT 2001	cluster10/athena	localnode	52			Down
10	Thu Aug 30 17:00:41 MDT 2001	cluster10/athena	localnode	18			Down
11	Thu Aug 30 17:17:40 MDT 2001	cluster10/hermes	localnode	1			Down
12	Thu Aug 30 17:44:41 MDT 2001	cluster10/athena	localnode	3			Down
13	Thu Sep 27 13:41:28 MDT 2001	cluster10/hermes	localnode	76	+		Down
14	Thu Sep 27 13:45:17 MDT 2001	cluster10/athena	localnode	72	+		Down

All times in minutes. See note at end about rounding. + = event ongoing at end of reporting period.

Aggregate MC/ServiceGuard Localnode Availability Report

Number of downtime events: 6
Aggregate measurement time: 221757
Total downtime: 220
MC/ServiceGuard Localnode availability for the period: 99.900%

Detailed MC/ServiceGuard Localnode Availability Report

Entity	Accuracy	Entity info	Availability	R?
cluster10/athena	+/- 0.5 min	MC/ServiceGuard A.10.12 2006045407 EMS HA Monitors A.03.20	99.715%	
cluster10/hermes	+/- 0.5 min	MC/ServiceGuard A.10.12 2003231967 EMS HA Monitors A.03.10	99.848%	
webserver/apollo	+/- 0.5 min	MC/ServiceGuard A.11.09 500776557 EMS HA Monitors A.03.20.01	100.000%	
webserver/zeus	+/- 0.5 min	MC/ServiceGuard A.11.09 524786597 EMS HA Monitors A.03.20.01	100.000%	

MC/ServiceGuard Package Event History

Event	Date	Entity	Class	Downtime	+?	P?	Notes
1	Fri Aug 17 00:06:40 CEST 2001	webserver/webserver	package				Began measurement.
2	Fri Aug 17 00:06:40 CEST 2001	webserver/nfssamba	package				Began measurement.
3	Fri Aug 17 00:20:40 CEST 2001	webserver/webserver	package				Measurement terminated.
4	Fri Aug 17 00:20:40 CEST 2001	webserver/nfssamba	package				Measurement terminated.
5	Fri Aug 17 00:32:22 CEST 2001	webserver/webserver	package				Began measurement.
6	Fri Aug 17 00:32:22 CEST 2001	webserver/nfssamba	package		+		Began measurement.
7	Thu Aug 23 13:30:10 MDT 2001	cluster10/wsleeper	package				Began measurement.
8	Thu Aug 23 13:30:10 MDT 2001	cluster10/msleeper	package				Began measurement.
9	Fri Aug 24 01:20:56 CEST 2001	webserver/webserver	package	1			Down
10	Mon Aug 27 17:23:40 MDT 2001	cluster10/wsleeper	package	44494	+		Down
11	Mon Aug 27 17:23:41 MDT 2001	cluster10/msleeper	package	44494	+		Down
12	Mon Aug 27 17:23:42 MDT 2001	cluster10/FailFast	package				Began measurement.
13	Mon Aug 27 17:23:42 MDT 2001	cluster10/11XClock.1	package				Began measurement.
14	Mon Aug 27 17:23:42 MDT 2001	cluster10/sleepingBeauty	package				Began measurement.
15	Mon Aug 27 17:27:42 MDT 2001	cluster10/FailFast	package	2			Down
16	Tue Aug 28 12:44:13 MDT 2001	cluster10/sleepingBeauty	package	1			Down
17	Tue Aug 28 12:44:23 MDT 2001	cluster10/FailFast	package	2			Down
18	Tue Aug 28 12:44:42 MDT 2001	cluster10/11XClock.1	package	1			Down
19	Thu Aug 30 13:25:43 MDT 2001	cluster10/sleepingBeauty	package	1			Down
20	Thu Aug 30 13:35:43 MDT 2001	cluster10/11XClock.1	package	1			Down
21	Thu Aug 30 13:39:23 MDT 2001	cluster10/11XClock.1	package	6			Down
22	Thu Aug 30 13:39:53 MDT 2001	cluster10/sleepingBeauty	package	6			Down
23	Thu Aug 30 15:04:54 MDT 2001	cluster10/sleepingBeauty	package	13			Down
24	Thu Aug 30 15:17:53 MDT 2001	cluster10/sleepingBeauty	package	21			Down
25	Thu Aug 30 15:39:13 MDT 2001	cluster10/sleepingBeauty	package	74			Down
26	Thu Aug 30 16:33:54 MDT 2001	cluster10/11XClock.1	package	1			Down
27	Thu Aug 30 16:35:23 MDT 2001	cluster10/11XClock.1	package	1			Down
28	Thu Aug 30 17:00:12 MDT 2001	cluster10/11XClock.1	package	19			Down
29	Thu Aug 30 17:00:13 MDT 2001	cluster10/sleepingBeauty	package	20			Down
30	Thu Aug 30 17:17:24 MDT 2001	cluster10/FailFast	package	1			Down
31	Thu Aug 30 17:20:13 MDT 2001	cluster10/sleepingBeauty	package	1			Down

32	Thu Aug 30 17:21:42 MDT 2001	cluster10/11XClock.1	package	1		Down	
33	Thu Aug 30 17:25:24 MDT 2001	cluster10/sleepingBeauty	package	1		Down	
34	Thu Aug 30 17:44:12 MDT 2001	cluster10/11XClock.1	package	3		Down	
35	Thu Aug 30 17:44:13 MDT 2001	cluster10/sleepingBeauty	package	40153	+	Down	
36	Wed Sep 05 01:21:25 CEST 2001	webserver/webserver	package	1		Down	
37	Sun Sep 09 17:46:42 MDT 2001	cluster10/11XClock.1	package	25751	+	Down	
38	Tue Sep 18 11:10:23 MDT 2001	cluster10/FailFast	package	1		Down	
39	Tue Sep 18 13:34:34 MDT 2001	cluster10/FailFast	package	1		Down	
40	Tue Sep 18 14:07:03 MDT 2001	cluster10/FailFast	package	1		Down	
41	Thu Sep 20 01:21:55 CEST 2001	webserver/webserver	package	1		Down	
42	Thu Sep 27 13:41:28 MDT 2001	cluster10/FailFast	package	2		Down	
43	Thu Sep 27 13:45:13 MDT 2001	cluster10/FailFast	package	72	+	Down	

All times in minutes. See note at end about rounding. + = event ongoing at end of reporting period.

Aggregate MC/ServiceGuard Package Availability Report

Number of downtime events: 32
Aggregate measurement time: 355253
Total downtime: 155132
MC/ServiceGuard Package availability for the period: 56.332%

Detailed MC/ServiceGuard Package Availability Report

Entity	Accuracy	Entity info	Availability	R?
cluster10/11XClock.1	+/- 0.5 min	MC/ServiceGuard A.10.12 2006045407 EMS HA Monitors A.03.20	42.061%	
cluster10/FailFast	+/- 0.5 min	MC/ServiceGuard A.10.12 2006045407 EMS HA Monitors A.03.20	99.827%	
cluster10/msleeper	+/- 0.5 min	MC/ServiceGuard A.10.12 2006045407 EMS HA Monitors A.03.20	11.871%	
cluster10/sleepingBeauty	+/- 0.5 min	MC/ServiceGuard A.10.12 2006045407 EMS HA Monitors A.03.20	9.454%	
cluster10/wsleeper	+/- 0.5 min	MC/ServiceGuard A.10.12 2003231967 EMS HA Monitors A.03.10	11.871%	
webserver/nfssamba	+/- 0.5 min	MC/ServiceGuard A.11.09 500776557 EMS HA Monitors A.03.20.01	100.000%	
webserver/webserver	+/- 0.5 min	MC/ServiceGuard A.11.09 524786597 EMS HA Monitors A.03.20.01	99.997%	

Error Report

Entity name	Messages
HP-UX Systems	No errors recorded.
MC/ServiceGuard Clusters	No errors recorded.
MC/ServiceGuard Localnodes	No errors recorded.
MC/ServiceGuard Packages	No errors recorded.

Notes

Rounding:

1. In the report history downtime values are rounded to the next higher reporting unit (for example, 10 seconds is rounded up to 1 minute).

Report Fields:

Event - The availability event number.
Date - Local time the availability event.

Thus no downtime is ever reported as having a duration of zero.
2. Aggregate values are truncated to the report's significance.
For example, 89.9999% is rounded to 89.999% for "5 nines" significance.
Thus no availability is ever stated as 100% if there was downtime during the reporting period.
3. The aggregate figures are computed using internal precise duration measurements. Thus aggregate figures may not reflect the sum of duration values as seen in the history section due to rounding.
Aggregate measures should always be used as the preferred measure of total entity and shop availability.

Accuracy:
4. The measurements for entities in this report are constrained by the polling frequency of the measurement instrument. For example, if the entity is polled twice a minute, then each downtime period can be in error by as much as 1 minute (2 * 0.5 minutes, or +/- 0.5 min). For this reason downtime period values are stated in units no greater than the polling accuracy.
5. Each entity is determined to be reachable at the time of report generation. If the system cannot be pinged or otherwise reached from the HA Meter server the (R?) is set to suspect (?) and the computed availability for that system may not be meaningful. The availability of any MC/ServiceGuard element that depends on the unreachable system is also flagged as suspect.

Reporting
6. Run hamreport -h for complete option details.
7. The results generated by this report are current to the time of the report. A later run of the report may produce different results because the entities under measurement have been measured for that much longer. In addition, systems that were previously unreachable may now be reachable and their availability data may consequently be more accurate.
8. Error messages indicating problems in collecting HA Meter data may appear in the Error Report. Such messages appear only so long as the problem persists.
9. The Event History reports may contain no data. This would happen if the entity experienced no availability events during the report's query period, in which case the entity was up, down, or unmeasured during the period.

Entity - The name of the measured entity.
Class - Entity class under measurement.
Downtime - The duration of downtime.
+? - The event is ongoing at report time.
P? - Is the event planned (blank if n/a).
Notes - Shutdown cause string if supplied.
Entity info - Entity-specific information:
  system - Model name, model number, os name, os revision.
  cluster - Cluster type, revision, system id, EMS product, revision.
  localnode - Cluster type, revision, system id, EMS product, revision.
  package - Cluster type, revision, system id, EMS product, revision.
  Fields may be blank if no data exists.
Availability - Entity availability, percent.
R? - ? signifies the entity was unreachable at the time of report generation.

Cause Entry Codes:
1. Hardware failure
2. OS failure
3. Application failure
4. Middleware failure
5. Patch/Software installation
6. Kernel reconfiguration
7. Hardware upgrade/installation
8. Hardware reconfiguration
9. Scheduled reboot
10. Other scheduled maintenance
11. System backup
12. Environment Failure
13. Enter cause