POWER8® Processor-Based Systems RAS

Introduction to Power Systems™ Reliability, Availability, and Serviceability

January 12th, 2016

IBM Server and Technology Group
Daniel Henderson
Senior Technical Staff Member, RAS

Trademarks, Copyrights, Notices and Acknowledgements

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Active Memory™	AIX®	POWER®	POWER Hypervisor™	Power Systems™	Power Systems Software™
POWER6®	POWER7®	POWER7+™	POWER8™	PowerHA®	PowerLinux™
PowerVM®	System x®	System z®		POWER Hypervisor™	

Additional Trademarks may be identified in the body of this document.

The following terms are trademarks of other companies:

Intel, Intel Xeon, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

LTO, Ultrium, the LTO Logo and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Microsoft, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

Other company, product, or service names may be trademarks or service marks of others.

Notices

The last page of this document contains copyright information, important notices and other information.

Acknowledgements

While this whitepaper has a single principal author/editor it is the culmination of the work of a number of different subject matter experts within IBM who contributed ideas, detailed technical information, and the occasional photograph and section of description.

These include the following:

Theodore Maeurer, Thoi Nguyen, Doug Baska, Michael Mueller, Jim O'Connor, Gary Andrews, K Paul Muller, Len Daniels, Ken Wright, Steven Gold, Marc Gollub, Pat Egan, Ravi A. Shankar, Craig Shempert, Kevin Reick, Naresh Nayar, Peter Heyrman, Resham Kulkarni, Kanwal Bahri, Christopher Sturgill, Audrey Romonosky, George Ahrens, Jim Mitchell, Dan Hurlimann and Rob Overton.

Table of Contents

Introduction

Announce History

Between April 2014 and October 2015, IBM has introduced a family of Power Systems® based on POWER8® processors. This family ranges from highly scalable enterprise-class servers supporting up to 16 processor modules to scale-out systems with one processor module.

The accompanying announcement material for each system speaks in detail about the performance characteristics of the processor and talks about its key reliability, availability and serviceability (RAS) attributes of each system.

These systems, leverage the POWER8 processor, enterprise-class memory, and the error detection and fault isolation characteristics afforded by IBM's flexible service processor. They therefore all share a certain set of reliability, availability and serviceability characteristics which are the subject of this whitepaper.

October 2015 Announcement

In October 2015 IBM announced, the Power System S812LC and Power System S822LC. These systems, while based on a POWER8 processor, take a somewhat different approach to error detection, fault isolation and error handling. Therefore a separate section in the whitepaper will discuss these servers while the remainder of the whitepaper will be applicable only to the previously announced servers.

Also announced at that time were enhancements to the fault tolerance and concurrent maintenance of the PCIe Gen3 I/O Drawer enabled in firmware level FW840, including:

1. Concurrent repair of an I/O Module in the I/O drawer
2. Concurrent add of PCIe3 Optical Cable Adapters to system nodes of IBM Power System E880 and E870
3. Concurrent add of PCIe Gen3 I/O Drawers to an IBM Power System E880 or E870 System.

Additional details and requirements are specified in Figure 14: PCIe Gen3 I/O Expansion Drawer RAS Feature Matrix.

Whitepaper Organization

The introduction to the first edition of this whitepaper noted that For POWER7® and POWER7+™ systems, the reliability, availability and serviceability characteristics of Power Systems were documented in detail in a POWER7 RAS whitepaper[1].

The server landscape has changed significantly since the first POWER7-based system was introduced in 2010. While what is written in the whitepaper is generally still valid for POWER8 processor-based systems, the relevance can be questioned in a cloud-centric, collaborative hardware and software environment.

The designers of the POWER8 processor-based systems believe that even in these new environments, server reliability, availability and serviceability will still be the key differentiators in how comparably priced systems perform in real-world deployments especially considering the impact of even partial service disruptions.

Therefore this whitepaper is organized into five sections:

Section 1: POWER8 RAS Summary

Introduces the POWER8 processor and systems based on the processor.

Readers very familiar with systems based on previous generations of POWER processors may use this section as a quick reference to what's new in POWER8.

[1] POWER7 System RAS: Key Aspects of System Reliability Availability and Serviceability, Henderson, Mitchell, et al. , 2010-2014, http://www-03.ibm.com/systems/power/hardware/whitepapers/ras7.html

Section 2: Hallmarks of Power Systems RAS

First summarizes the defining characteristics of Power Systems and how they may differ from other system design approaches.

Section 3: POWER8 Common RAS Design Details

Discusses in detail the RAS design of any IBM system based on a POWER8 processor, concentrating on processors and memory. This provides a more in-depth discussion of how what is distinctive in the POWER processor design and how improves reliability and availability of systems.

Section 4: Server RAS Design Details

Talks about different RAS characteristics expected of different families of hardware servers/systems, with a detailed discussion of server hardware infrastructure.

Section 5: Application Availability

Addresses in general terms what is required to achieve a high level of application availability in various system environments. It gives a numeric approach to evaluating the contributions to availability expected of various components of a system stack from the hardware, firmware/hypervisor layers, operating systems and applications.

It particularly illustrates the advantages of enterprise-class hardware in a variety of application environments.

Section 1: POWER8 RAS Summary

RAS Summary: Common System Features

POWER8 Processor RAS

POWER8 processor based systems using the IBM flexible service processor are capable of scaling to more than 4 sockets use a single chip module (SCM) that contains a single processor chip using 22nm technology with up to 12 functional cores.

Other systems use a dual-chip processor module (DCM) comprised of two processor chips, each with a maximum of 6 cores yielding a 12 core maximum processor socket.

The SCM compared to the DCM has greater fabric bus capacity, allowing systems to scale beyond four processor modules. Models based on the SCM may also operate at higher frequencies than models based on the DCM.

When comparing POWER8 to POWER7 and POWER7+, attention may first be drawn to the enhanced performance that POWER8 provides. In addition to supporting more cores, threading and cache capacity enhancements provide other performance improvements as highlighted in Figure 1. This increased capacity and integration by itself can improve overall server reliability by doing more work per processor socket.

Figure 1: POWER8 Processor Compared to POWER7+

Feature	POWER8	Equivalent POWER7+™ Processor Module
Cores	8-12 Depending on model	8 Maximum
Threads	8/core	4/core
L3 Cache	8MB/core	10MB/core
Memory Bandwidth (100% DRAM Utilization)	192 GBs per socket	68 GBs per socket
L4 Cache/ DIMM	16 MB	None

In addition a number of components that were external to the processor in POWER7 are now integrated into POWER8, providing additional reliability advantages compared to having separate modules. These features are illustrated by .
Figure 2: POWER8 DCM Module Integration.

Figure 2: POWER8 DCM Module Integration

POWER8 sockets have 12 cores maximum vs. 8 core max in POWER7

- Integration providing reliability enhancement

Incorporates an On Chip Controller

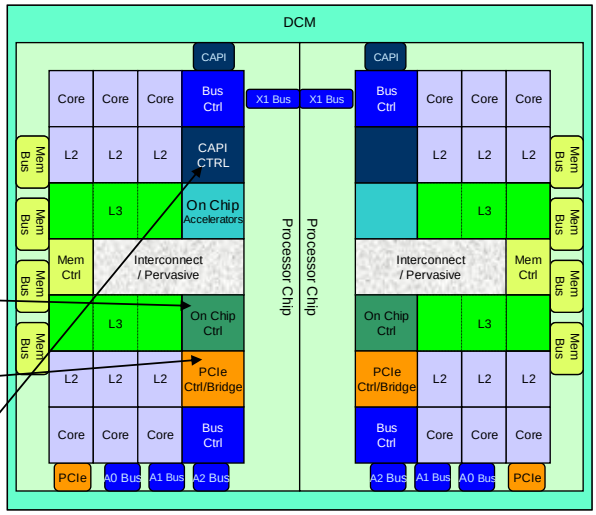
- Eliminating a need for a separate module to handle power management and thermal monitoring
  - And handles other similar tasks
    - ✓ Without need for Host code

Has Integrated PCIe Ctrl/PHB

- Simplifying integrated I/O attachment

Adds an Interface to Attach External Coherent Co-processors

- Which incorporates fault isolation/containment features similar to that used for PCIe and Internal Accelerators



DCM Module Used with POWER System S824

POWER processors from POWER7 through POWER7+ to POWER8 have continued to advance in RAS design and technology in areas of soft error avoidance, self-healing capabilities, error recovery and mitigation. Figure 3 provides a table of the key POWER7 processor RAS features and enhancements made in POWER7/POWER7+ and POWER8.

Figure 3: POWER7/POWER7+ and POWER8 RAS Enhancements

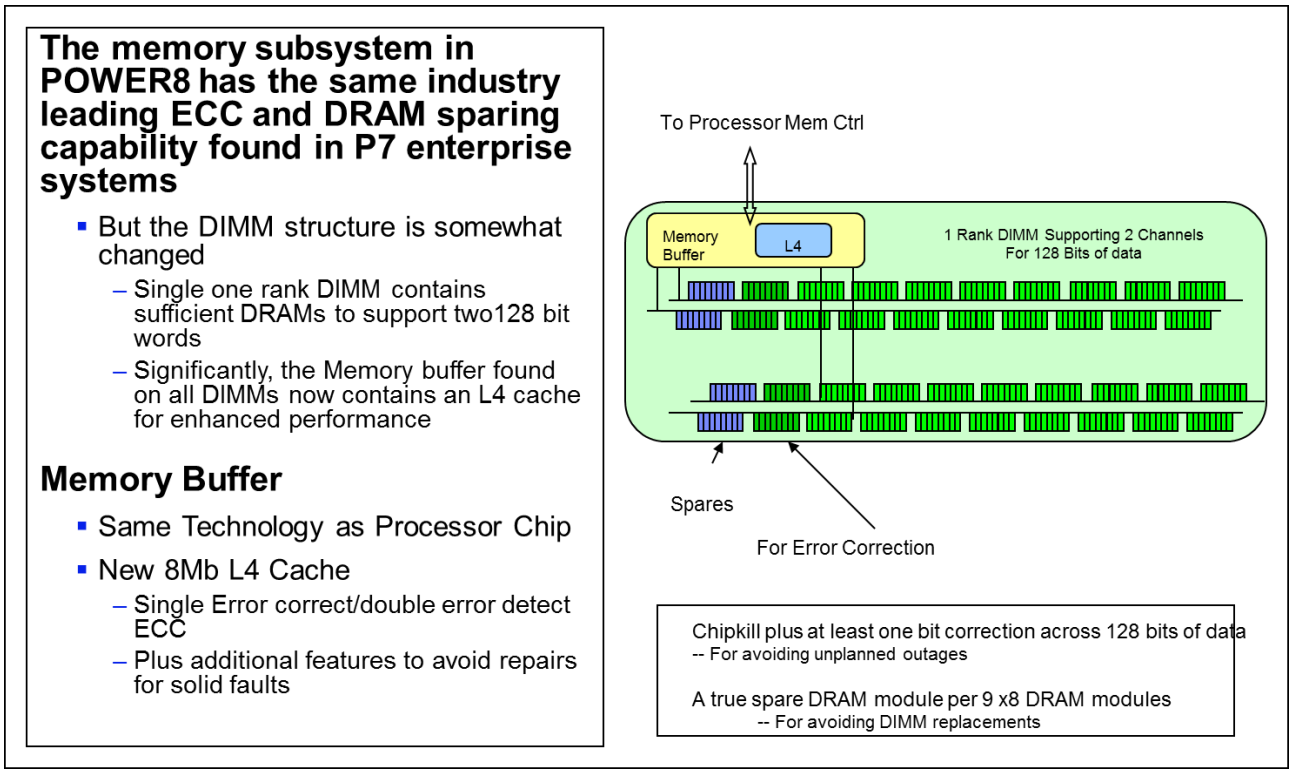
Area	POWER7 Enterprise Had	POWER7+ Enterprise Had	POWER8 Enterprise Added
Error Detection/Fault Isolation/New Function	▪ L2/L3 cache ECC ▪ Memory Chipkill and symbol error correction	▪ Error Handling for On Chip Accelerators	▪ Error handling for Off Chip accelerators etc. using CAPI interface ▪ Advanced error checking on fabric bus address generation ▪ Integrated On Chip Controller used for Power/Thermal Handling ▪ Host Boot ▪ Integrated PCIe function
Advanced technology for avoiding soft errors	▪ SOI Processor Modules ▪ eDRAM L3 cache ▪ Stacked Latches	▪ More comprehensive use of stacked latches	▪ eDRAM for L4 Cache

Recovery/Retry	▪ Processor Instruction Retry ▪ Memory buffer soft error retry	▪	▪ Memory instruction replay
Self-healing/Repair/Fault Avoidance	▪ L2/L3 cache line delete ▪ Dynamic Memory Bus data-lane repair ▪ Spare DRAM(s) in memory	▪ Dynamic inter-node fabric bus lane repair ▪ L3 cache column repair	▪ Dynamic inter-node fabric bus lane repair ▪ L4 cache persistent fault handling
Other Error Mitigation	▪ Alternate Processor Recovery ▪ CPU predictive deconfiguration ▪ Active Memory Mirroring of Hypervisor	▪ Use of Power On Reset Engine for dynamic processor re-initialization	▪ Dynamic substitution of unassigned memory for memory DIMMs called out for repair

Memory RAS

POWER8 processor-based systems maintain the same basic three part memory subsystem design consisting of two memory controllers in each processor module which communicate to buffer modules on memory DIMMS which in turn access the DRAM memory modules on DIMMs.

Figure 4: POWER8 Custom DIMM Design



POWER8 processor based systems using the IBM flexible service processor feature custom DIMMS containing IBM's next generation memory buffer chip. This custom buffer-chip is built using the same technology as the POWER8 processor chip incorporating the same technology to avoid soft errors and a design allowing retry for internally detected faults.

The error correction algorithm and layout on the custom DIMMs allow for multiple DRAM module failures per DIMM to be tolerated using x8 technology. This includes the use of spares to avoid replacing a DIMM on such a failure even when x8 DRAM modules are used.

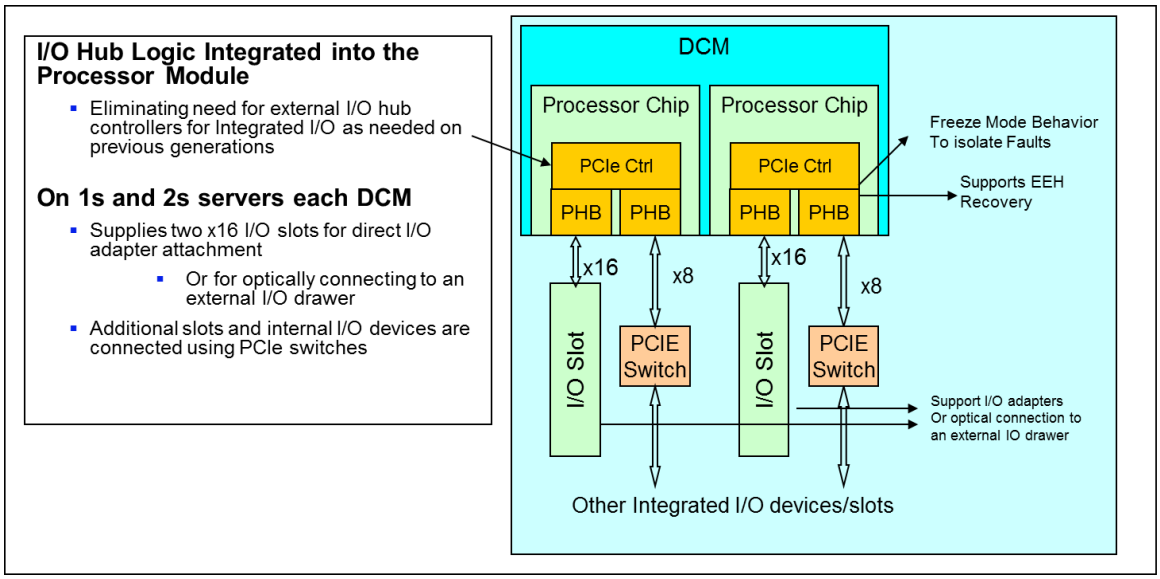
These DIMMS also differ from those used in POWER7 in that all of the data of every ECC word is completely contained on a single DIMM. This ensures that uncorrectable errors, as unlikely as they may be to occur, can be repaired by replacing just a single DIMM rather than a pair of DIMMS as was required in POWER7 based systems.

Finally, each IBM memory buffer on each DIMM contains additional function: An L4 cache. This L4 cache is designed from a hardware standpoint and ECC to detect and correct externally induced soft errors in a way similar to the L3 cache.

Since the data in the L4 cache is used in conjunction with memory rather than associated with a particular processor, the techniques used for repairing solid errors is somewhat different from the L3 design. But the L4 cache has advanced techniques to delete cache lines for persistent recoverable and non-recoverable fault scenarios as well as to deallocate portions of the cache spanning multiple cache lines.

I/O Subsystem

Figure 5: 1s and 2s POWER8 Processor-based Scale-out Systems



Improving reliability through greater integration is a theme present in the CEC I/O subsystem. The “I/O hub” function that used to be provided for CEC I/O by separate module in POWER7 systems has been integrated into the POWER8 processor as well.

Each processor module can directly drive two I/O slots or devices with the PCIe controllers in each processor without an external I/O Hub.

Capability is also provided for additional integrated I/O devices and slots using PCIe switches.

While the I/O hub has been integrated into the processor module it still retains a design that supports end-point error recovery as well as “freeze on fault” behavior and fault isolation, so that unrecoverable errors can be contained to a partition using the I/O.

RAS Summary: RAS Capabilities of Various System Models

Introduction

POWER8 processor based systems using the IBM flexible service processor can be grouped into several categories with similar RAS features aligning with the maximum number of processor sockets supported in each system.

The first group includes 1 and 2 socket systems (1s and 2s) that are designed for a scale-out environment. Other groups include 4 socket systems and systems with 4 socket building-blocks that can be interconnected to create systems with 8, 16, 24, 32 or more processor sockets.

From a RAS standpoint, all systems are built on a strong POWER processor base. The preceding sub-section summarized the RAS characteristics of these features and the POWER8 processor/memory improvements.

Enterprise-level RAS is more involved than just processor and memory RAS. The structure of the systems using these components is also critical in avoiding application outages.

Power Systems RAS characteristics are designed to be suitable to the size of the system and intended use. Figure 6 illustrates some of different Power Systems offered and their associated RAS characteristics.

Figure 6: Comparing Power System RAS Features

	POWER7 1s,2s Servers and IBM Power System 750/760	POWER8 1s and 2s Systems ^	POWER8 IBM Power System E850	POWER8 IBM Power System E870/E880
POWER7+ Processor soft error protection, recovery and self-healing	Yes	Yes *	Yes	Yes
New POWER8 Processor features including: ▪ Integrated PCIe controller ▪ Integrated Power/cooling monitor function using on chip controller ▪ Memory buffer replay	No	Yes *	Yes	Yes
PCIe hot-plug	750/760	Yes	Yes	Yes
PCIe controllers integrated into processor	No	Yes	Yes	Yes
Enterprise Memory with custom DIMMS, Chipkill and spare DRAM capabilities,memory buffer soft error handling features	No	Yes	Yes	Yes
Active Memory Mirroring for the Hypervisor	No	No	Supported as an Option	Yes in base configuration
Can take advantage of Capacity Update on Demand	No	No	Available Option	Yes
Dynamic Substitution of Unused memory for predictive memory faults	No	Yes+	Yes	Yes
Triple Redundant Ambient Temperature Sensors On Op Panel	No	Yes+	Yes	Yes
Redundant and spare voltage converter phases supplied to processors and to CDIMMS	No	No	Redundant or Spare	Both redundant and spare
Redundant global processor clocks	No	No	No	Yes
Redundant service processor	No	No	No	Yes
Can re-IPL other nodes when an entire node must be de-configured due to a fault	No	No	No	Yes on multiple node systems
Supports Enterprise System Pools	No	No	No	Yes

^1s and 2s systems in this table include:
AIX® IBM and Linux™: IBM Power S812, IBM Power S822 and IBM Power S824
Linux Only: IBM Power S812L, IBM Power S822L and IBM Power S824L
See Figure 14 for information about POWER7 S812LC and S812LC
*Not all features are supported with PowerKVM(TM)
+Support depends on firmware level installed in system

POWER8 processor-based 1s and 2s Systems IBM POWER System S814, S822, S824

The one processor module socket (1s) IBM Power System S814 server, and two processor module socket (2s) IBM Power System S822 and IBM Power System S824 servers are designed to run with the IBM POWER™ Hypervisor and support AIX®, Linux™ and IBM i operating systems.

Those familiar with POWER7 servers will see that these servers are aimed at replacing POWER7 processor-based 1s and 2s servers. Typically 1s and 2s systems may be thought of as “scale-out” systems designed to run a set of enterprise applications in concert with other similar systems in some form of clustered environment.

Responding to the increased performance and capacity, these 1s and 2s systems were designed with enhancements to both Processor and System platform level RAS characteristics compared to predecessor POWER7 and POWER7+ processor-based systems.

Outside of the processor itself, perhaps the two most noticeable system enhancements are:

- The ability to remove/replace PCIe adapters without the need to shut down the system or terminate partitions
- The use of what was previously considered to be Enterprise Memory using custom DIMMs with an IBM memory-buffer chip on-board each DIMM and featuring spare DRAM modules

The first feature reduces planned outages for repair of I/O adapters. The second is intended to reduce planned outages for repair of DIMMs as well as unplanned outages due to DIMM faults.

POWER8 processor-based 1s and 2s Systems IBM POWER System S812L, S822L and S824L,

From a hardware standpoint The IBM Power System S812L, S822L and IBM Power System S824L are similar to the servers described in the previous section, though they are designed to run Linux exclusively.

As an alternative to the POWER Hypervisor and PowerVM® these systems can be configured for open virtualization using the IBM PowerKVM™ Power Kernel-Based Virtual Machine to provide a hypervisor function. PowerKVM in turn uses the OpenPower Abstraction Layer (OPAL) for certain hardware abstraction functions.

Systems running PowerKVM do not have identical virtualization features as those running PowerVM.

Generally speaking processor and memory error detection, fault isolation and certain self-healing features are handled entirely within the hardware and dedicated service processor and are available in Power Systems regardless of the hypervisor deployed.

Certain functions that require notification of the hypervisor but do not require interaction with operating systems are also implemented in systems running with PowerKVM as well. Such features include processor instruction retry.

Some functions are not currently supported in the PowerKVM environment may be added over time. These currently include L2/L3 cache line self-healing features.

Other functions such as Alternate Processor Recovery depend heavily on the virtualization features of the POWER Hypervisor and are unlikely to become part of the PowerKVM environment.

These differences are highlighted in Figure 7: Comparing Power System S812L and S822L to S812LC and S822LC.

It should also be noted that different hypervisors and operating systems themselves may have differing capabilities concerning avoiding code faults, security, patching, boot time, update strategies and so forth. These can be important but are not discussed in this whitepaper.

POWER8 processor-based IBM Power System S812LC and S822LC

Introduction

These systems make use of the OpenPower Abstraction Layer (OPAL) software to provide essential abstraction of the hardware either directly to an operating system or through the PowerKVM hypervisor.

These systems differ from other Power Systems in that they are intended to be managed by the Intelligent Platform Management Interface (IPMI) providing an alternative approach to system management.

Because of this, they do not make use of the IBM flexible service processor for handling errors. Instead they make use of a Baseboard Management Controller (BMC).

Figure 7 Gives an overview of how these systems compare to IBM Power System S812L and S822L configured to run with the PowerKVM hypervisor.

Figure 7: Comparing Power System S812L and S822L to S812LC and S822LC

Legend: ● supported, ▬ not supported

RAS Item	Power System S812LC/S822LC	Power System S812L/S822L running PowerKVM	Power System S812L/S822L running PowerVM
Processor/Memory			
Base Processor Features Including: ▪ L1 Cache Set Delete ▪ L2/L3 Cache ECC ▪ Processor Fabric Bus ECC ▪ Memory Bus CRC with Retry/Sparing	●	●	●
Processor Instruction Retry	●	●	●
Advanced Processor Self-Healing and fault handling: ▪ L2/L3 Cache Line Delete ▪ L3 Cache Column eRepair ▪ Core Contained Checkstops ▪ Alternate Processor Recovery	▬	▬	●
Enterprise Memory with Spare DRAMS	▬	●	●
Power/Cooling			
Redundant Power Supplies	S822LC – Configuration Dependent S812LC -Standard	●	●
OCC error handling w/ power safe mode	●	●	●
Redundant Fans	●	●	●
Hot Swap Fans	S822LC	●	●
Hot Swap DASD / Media	● (DASD only)	●	●

I/O			
Dual disk controllers (split backplane)	—	●	●
Hot Swap PCI Adapter	—	●	●
Concurrent Op Panel Repair	N/A	●	●
Cable Management	Arm with slide rail S822LC (8335 only)	Arm	Arm
Service Infrastructure			
Service Processor Type	BMC	FSP	FSP
Service Indicators	Limited Lightpath	Partial Lightpath	Partial Lightpath
Operator Panel	LED/Button	LCD Display	LCD Display
Hardware Abstraction Software	OPAL	OPAL	POWER Hypervisor
Supported Hypervisor	PowerKVM	PowerKVM	PowerVM
Error Notifications	SEL events in BMC Logs in OS Syslog	Logs in Syslog & ASMI	Logs in Syslog & ASMI
Service Videos	No	Yes	Yes
Redundant VPD	—	●	●
Warranty			
Standard Installation	Customer	Customer	Customer
Standard Service	Customer	Customer	Customer
Standard Feature Installation	Customer	Customer	Customer
Service Indicators	Limited Lightpath	Partial Lightpath	Partial Lightpath
Service Videos	No	Yes	Yes
Warranty	3 Yr	3 Yr	3 Yr
Standard Service Delivery	100% CRU Kiosk parts	9x5 Next Business Day	9x5 Next Business Day

Error Handling

In implementing the described error reporting and management interface, these servers make use of a Baseboard Management Controller (BMC) based service processor.

User level error reporting is done through the IPMI interface to the BMC including thermal and power sensor reports.

The BMC service processor has different capabilities than the IBM designed Flexible Service Processor (FSP) used in other Power Systems.

For example, the FSP can monitor the processor and memory sub-system for recoverable errors that occur during run-time. The same service processor is also capable of doing analysis of faults at IPL-time or after a system checkstop.

The BMC-based service processor structure does not allow for the service processor to fully perform those activities.

Therefore, to maintain essential Power System First Failure Data Capture capabilities for the processor and memory, much of the run-time diagnostic code that runs on the FSP in other Power Systems has been written as an application that can run as code on a host processor.

This code can monitor recoverable errors and make service related callouts when an error threshold is reached. Compared to the FSP based system, however, this host-based code will generally not make requests for resource de-configuration on recoverable errors.

Some unrecoverable errors that might be handled by the hypervisor, such as uncorrectable errors in user data can still be signaled to the hypervisor.

Unrecoverable errors not handled by the hardware or hypervisor will result in a platform checkstop. On a platform checkstop, the service processor will gather fault register information. On a subsequent system reboot, that fault register information is analyzed by code similar to the run-time diagnostics code, running on the processor module the system booted from.

If uncorrectable errors are found in hardware during the boot process, it might be possible for a system design to run through a continuous process of booting, encountering an unrecoverable error, then unsuccessfully attempt boot again.

To prevent such repeated crashes, or systems failing to IPL due to uncorrectable resources, uncorrectable errors detected during the diagnostic process at boot-time can be deconfigured (guarded) to allow the system to IPL with the remaining resources, where possible.

IPL can only occur from the first processor in a multi-socket system, however.

In addition the BMC-based service processor is also essential to boot. The BMC design allows for a normal and a “golden” boot image to be maintained. If difficulties occur during normal IPL, the system can fall back to the “golden” boot image to recover from certain code corruptions as well as to handle issues with guarded resources.

POWER8 processor-based IBM POWER E850 System

Introduction

The IBM Power System E850 has 4 processor sockets, with a minimum of two processor sockets each capable of supporting a POWER8 dual chip modules (DCM). These DCMs are the same type as used in 1s and 2s systems. However, because of the increased capacity this model was designed with additional enterprise-class availability characteristics.

Significantly, compared to POWER7 processor based IBM Power System 750 or 760, the power distribution has been enhanced to provide voltage converter phase redundancy for processor core and cache voltage levels to avoid unplanned outages and voltage converter phase sparing for other voltages levels (such as used by memory) to avoid unplanned repairs.

The systems are capable of taking advantage of such enterprise features as Capacity Update on Demand and RAS features which leverage this capability are included where spare capacity is available. In addition, systems have the option of mirroring the memory used by the hypervisor.

System Structure

Figure 8 below gives a reasonable representation of the components of a fully populated E850 server.

The figure illustrates the unique redundancy characteristics of the system. Power supplies are configured with a N+1 redundant capability that maintains line-cord redundancy when power supplies are properly connected to redundant power sources.

Fan rotors within the system also maintain at least an N+1 redundancy, meaning the failure of any one fan rotor, absent any other fault, does not cause a problem with system overheating when operating according to specification.

Fans used to cool the CEC planar and components associated with it are also concurrently maintanable.

There are additional fans in the lower portion of the system used to cool components associated with the RAID controller/storage section. These fans can not be repaired concurrently. Therefore the system is supplied with sufficient fan rotors to ensure N+1 redundancy, and in addition two additional fan rotors are provided. The additional fan rotors are considered as integrated spares that do not require replacing.

This configuration allows the system to run with two of these fan rotors failed, without requiring a repair, and the system still maintains n+1 fan rotor redundancy.

Given the amount of sparing built in, there is a very low expecation of needing to take a system down to repair any of these fans. Should such a repair be necessary, however, the entire set of 8 fans would be replaced in a single repair action.

As in other systems, the power supplies take AC power inputs and convert to a single 12 volt dc level. Voltage regulator modules (or VRMs) are then used to convert to the various voltage levels that different components need (1.2 volts, 3.3 volts. etc.)

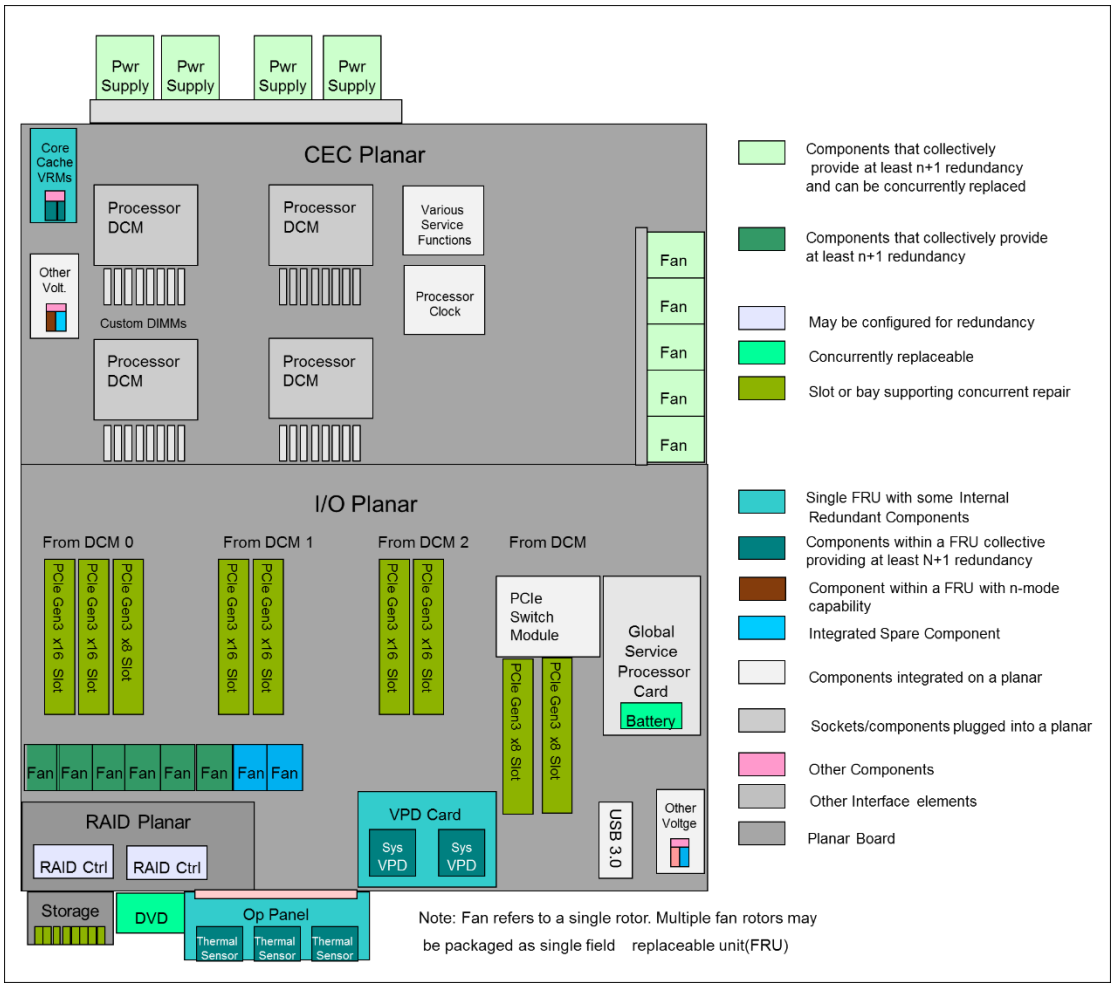
As the general rule the design for such a VRM can be described as consisting of some common logic plus one or more converters, channels, or phases. The IBM Power System E850 provides one more such phase than is needed for any voltage output that is distributed to critical elements such as a processor or memory DIMM.

The module(s) supplying the processor core and cache elements are replaceable separate from the CEC planar. If one of the phases within the module fails, a call for repair will be made, allowing for redundancy of these components. For other cases, such as the module cards supplying memory DIMMS, the extra phase is used as an integrated spare. On the first phase failure in such a VRM, the system continues operation without requiring a repair, thus avoiding outage time associated with the repair.

It should be noted that, when a component such as a processor module or a custom memory DIMM is supplied voltage levels with redundant phases, redunancy typical ends at the processor or CDIMM. On

some devices, such as for a memory DIMM the can be some sort of voltage converter to further divide a voltage level for puposes such as providing a reference voltage or signal termination. Such on-component voltage division/regulation is typically not as demanding an activitiy as previously discussed and is not included in that discussion of voltage regulation.

Figure 8: A Schematic of a IBM Power System E850



Comparing the IBM Power System E850 to POWER7 based IBM Power System 750 and 760

Figure 6: Comparing Power System RAS Features, gives a quick reference comparing the main features of various POWER7 and POWER8 processor based systems. The table illustrates significant areas where the availability characteristics of the E860 have advanced over what POWER7 provided.

These include processor and memory enhancements such as the integration of the I/O hub controller into the processor chip, the use of enterprise memory. The infrastructure was also improved in key areas especially by providing redundancy and sparing for critical voltage converter phases. Other such improvements included more redundancy in the op panel function, and the ability to dynamically replace the battery used to back-up time of day information when the system is powered off.

These systems are now enabled with options to use enterprise features of the PowerVM such as active memory mirroring of the hypervisor and Capacity Update on Demand. These functions, described in detail later in this document, also can enhance availability when compared to scale-out systems or the prior generation IBM Power System 750/760.

Comparing the IBM Power System E850 to IBM Power System E870 and IBM Power System E880

There are still some significant differences between the IBM Power System E850 and the E870/E880 Systems that can impact availability.

The latter systems have a redundant clock and service processor infrastructure which the E850 lacks.

As will be discussed in the next sub-section, the IBM Power E870 and E880 systems can expand beyond 4 sockets by adding additional 4 socket system node drawers. When one of these systems is configured with more than one such drawer, there is the added RAS advantage that a system can IPL with one of the drawers disabled, using resources of the remaining drawers in the system. This means that even if there was a fault in one of the system node drawers so severe that it took the system down and none of the resources in that drawer could subsequenlty be used, the other drawers in the system could still be brought back up and put in to service.

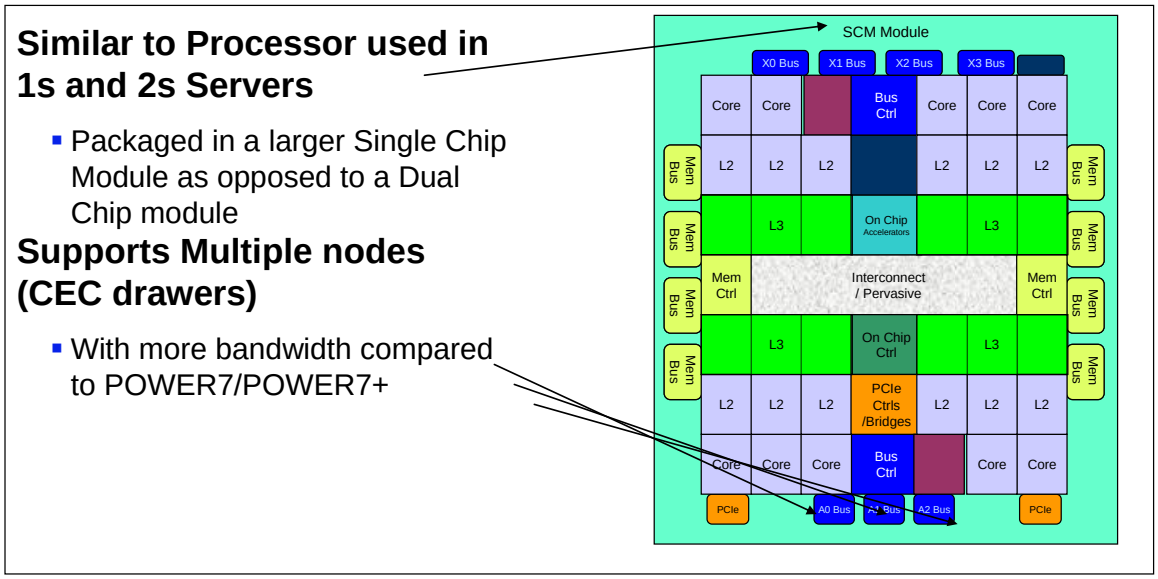
This would not be true of an E850 system.

POWER8 processor-based IBM Power System E870 and IBM Power System E880

Introduction

On October 6th, 2014, IBM introduced two new enterprise-class Power Servers based on a single core module (SCM) version of the POWER8 processor. The SCM, provides additional fabric busses to allow processors to communicate with multiple 4-socket drawers allowing system scaling beyond 4 sockets supported by the DCM module.

Figure 9: POWER8 SCM

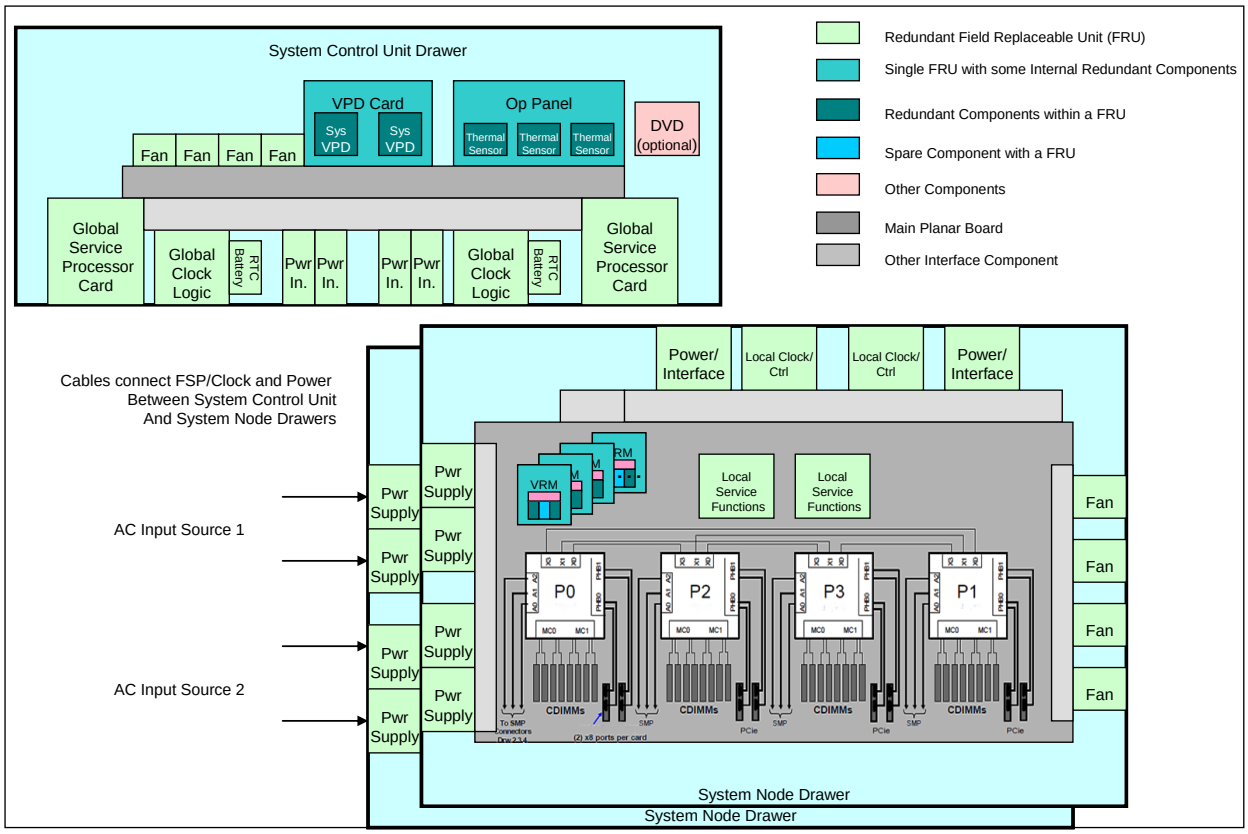


Accordingly, the basic building-block of an IBM Power E870 or IBM Power E880 system is a 4 process-socket system node drawer which fits into a 19” rack. Each node drawer also includes 32 DIMM sockets and 8 PCIe I/O adapter slots. Announced systems support 1 or 2 such system nodes.

In addition to the system nodes, each Power E870 or E880 system includes a separate system control unit also installed into a 19” rack and connected by cables to each system node. The system control node contains redundant global service processors, clock cards, system VPD and other such system-wide resources.

Figure 10 gives a logical abstraction of such a system with an emphasis on illustrating the system control structure and redundancy within the design.

Figure 10: Logical View of Power System E870/E80 System Structure



The figure illustrates a high level of redundancy of the system control structure (service processor, clocking, and power control) as well as power and cooling.

Most redundant components reside on redundant cards or other field replaceable units that are powered and addressed independently of their redundant pairs. This includes the redundant global and local service processor functions and clocks, the power interface that provides power to from the system nodes to the System control unit, as well as system fans.

There is only one system VPD card and one Op Panel, but the system VPD is stored redundantly on the system VPD card, and there are three thermal sensors on the Op panel ensuring no loss of capability due to the malfunction of any single sensor.

At the CEC-level there are voltage regulator modules (VRMs) packaged in Voltage Regulator Module (VRM) cards that are individually pluggable. Each such VRM can be described as having some common elements plus at least three components (known as converters, channels or phases) working in tandem. Proper function can be maintained even if two of these phases fail, so by definition the design allows for phase redundancy and in addition contains an additional spare phase. Using the spare phase prevents the need to replace a VRM card due to a failure of a single phase.

New in POWER8, the battery used to maintain calendar time when a Flexible Service Processor (FSP) has no power is now maintained on a separate component. This allows the battery to be replaced concurrently separate from the service processor.

In the design, fans and power supplies are specified to operate in N+1 redundancy mode meaning that the failure of a single power supply, or single fan by itself will not cause an outage. In practice, the actual redundancy provided may be greater than what is specified. This may depend on the system and configuration. For example, there may be circumstances where the system can run with only two of four

power supplies active, but depending on which power supplies are faulty, may not be able to survive the loss of one AC input in a dual AC supply data-center configuration.

Compared to POWER7 based IBM Power System 795

Individuals familiar with the POWER7 based Power 795 system will find similarities in the IBM Power E870/E880 design. Among the differences, however, is that the E870/E880 systems contain the global control functions with a separate drawer using cables to connect to each system node drawer, rather than making use of a large central backplane.

Also significantly, taking advantage of the integration of the PCIe controller into the processor, the E870 and E880 systems have 8 PCIe slots, each capable of supporting a PCIe adapter within the system node drawer, or connection to an external I/O expansion drawer. In contrast the POWER7 processor based 795 systems only supported I/O expansion drawers.

Figure 11 illustrates similarities and differences as well as some of the significant improvements in the system structure between the two systems.

Figure 11: Some Power System Featuresa E870/E880 Compared to 795

Area	POWER7 795 System	POWER8 4+ Socket Enterprise Systems
System Structure Similarities	▪ Global Service Processor/Clocks distributed to all processor books so that redundancy available in all configurations. ▪ N+1 Redundant power and cooling ▪ Redundant External Thermal/Power Monitoring Control Module	▪ Global Service Processor/Clocks so that redundancy available in all configurations ▪ N+1 Redundant power and cooling ▪ Thermal Power/Monitoring Control integrated into each Processor Module
System Structure Differences	▪ 4 Socket Processor Books ▪ 24” Rack ▪ Single Central Backplane interconnected ▪ I/O hub function external to processor required for I/O ▪ I/O drawers required for I/O	▪ 4 Socket System Node Drawers ▪ 19” Rack ▪ Drawers Cable Interconnected with more bandwidth capability using positive latching connectors ▪ I/O hub function integrated into processor (no external I/O hub required) ▪ Integrated I/O Slots available ▪ Optional I/O drawers ▪ Triple Redundant Ambient Temperature Sensors
Planned Outage Avoidance	▪ Concurrent repair of fans/power supplies, FSP ▪ Concurrent repair of PCIe adapters, DASD when configured for redundancy ▪ CHARM for repair of systems with 2 or more nodes required removing resources from one node during node repair	▪ Concurrent repair of fans/power supplies, clock batteries as component separate from the FSP ▪ Concurrent repair of PCIe adapters, DASD when configured for redundancy ▪ Emphasis on Live Partition Mobility to ensure application availability during planned outages regardless of the number of nodes

Compared to POWER7 based IBM Power System 770/780

Readers familiar with the POWER7 processor-based 770 and 780 servers will also see some similarities in the basic system structure, but will also see a significant number of differences and improvements. These include that the global service processors and clocks are made redundant in the system design even when a single system node drawer is used. That was not the case with the 770 or 780 single node systems.

Figure 12: Power System E880 and E870 Compared to 770/780

Area	POWER7+ 770/780 System	POWER8 4+ Socket Enterprise Systems
System Structure Similarities	▪ 1-4 CEC drawers (Drawers, 4 sockets max each drawer) ▪ 19” Rack ▪ Drawers Cable Interconnected ▪ Integrated I/O and I/O Slots ▪ N+1 Redundant power and cooling ▪ Optional I/O drawers	▪ 1-4 CEC Drawers (Drawers, 4 sockets max each drawer) ▪ 19” Rack ▪ Drawers Cable Interconnected ▪ Integrated I/O Slots ▪ N+1 Redundant power and cooling ▪ Optional I/O drawers
System Structure Differences	▪ Redundant FSP/Clocks only available in systems with 2 or more CEC drawers ▪ Some critical voltage regulators integrated on planars ▪ Single ambient temperature sensor ▪ Single power/thermal control module ▪ I/O hub controllers external to processor for integrated and attached I/O required ▪ Separate I/O planar required for integrated I/O ▪ Integrated DASD Options	▪ Global FSP/Clock in separate drawer provides redundancy for 1 drawer (4 socket) systems ▪ Extensive improvements in power/cooling design/reliability including: ▪ Critical CEC-level Voltage Regulation on pluggable modules with N+1 phase design with integrated spare phase ▪ Triple redundant ambient temperature sensors ▪ Power/Thermal management integrated into each processor ▪ No external I/O hub required ▪ No separate I/O planar for integrated I/O needed ▪ No integrated DASD options
Planned Outage Avoidance	▪ Concurrent repair of fans/power supplies ▪ Concurrent repair of PCIe adapters, DASD when configured for redundancy ▪ Clock Batteries integrated into FSP function (not individually concurrently maintainable)	▪ Concurrent repair of fans/power supplies ▪ Concurrent repair of PCIe adapters, DASD when configured for redundancy ▪ Separately concurrently clock batteries

	▪ CHARM for repair of systems with 2 or mode nodes required removing resources of one node during node repairs	▪ Emphasis on Live Partition Mobility to ensure application availability during planned outages

And again, integration of the PCIe controller into the processor allows the E870/E880 systems to support integrated I/O slots without the need for a separate I/O planar.

CEC Hot Add Repair Maintenance (CHARM)

The bulk of all hardware replacements in an E870/E880 system are expected to be of components for which individual concurrent maintenance can be performed: power supplies, fans, I/O adapters and devices, clock batteries, and so forth.

However, some repairs cannot be made concurrently with system operation.

In POWER7, PowerVM Enterprise Edition supported the concept of Live Partition Mobility (LPM). LPM, which is discussed in detail elsewhere in this whitepaper, allows a properly configured partition virtualized under PowerVM to be migrated from one system to another without partition termination.

Migrating partition from one system to another allows for application continuity in the event a system requires an outage for repair.

In addition, Power Systems 795, 770 and 780 also support the concept of allowing one node (processor book or CEC drawer) to be removed from a running system configuration for repair while the other processor/books or drawers would continue to be powered on and run work-load.

This feature requires that all of the workload running in the load would have to be “evacuated” from the node before such a repair could be made. Processor and memory in Power Systems using PowerVM are virtualized. Therefore if sufficient resources could be found within other nodes in the system to take up the workload to be evacuated, and I/O was redundant and properly virtualized, it would be possible to evacuate a node and perform this repair without application outage.

LPM (important for not only service, but overall workload management) continues to be enhanced for Power Systems, including advancements in terms of performance and ease of use.

Anticipating the use of the more flexible LPM, CHARM is not a feature provided in IBM Power E870/E880 systems.

There are a number of reasons for this:

Dynamic node repair could not be supported on systems with single nodes. In practice systems with just two nodes may also find it difficult to find enough resources for a successful node evacuation.

CHARM only handles failures within a node and with some limitations. Repair of some global resources, even with CHARM still require a planned system-wide outage.

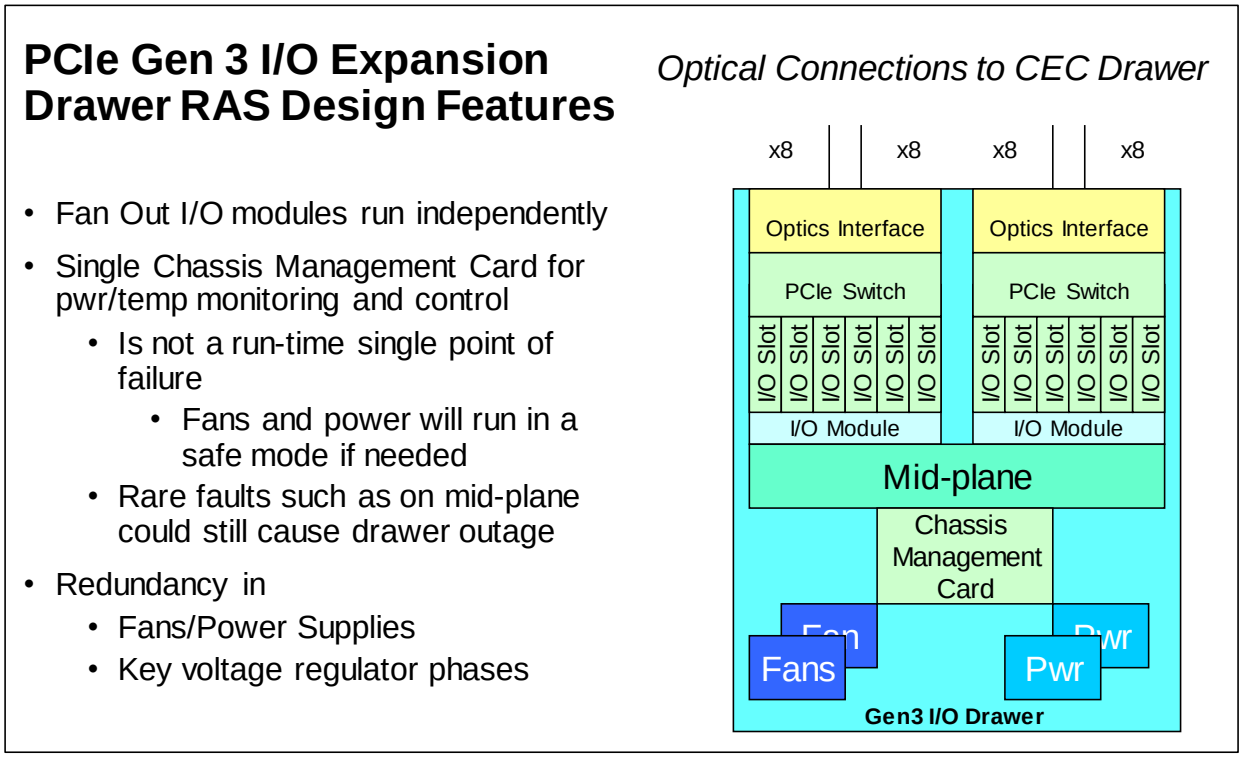
Use of CHARM for planned outage avoidance is limited to hardware failures. It doesn’t assist in keeping applications running during an upgrade of firmware to one level to another, during load re-balancing across systems or other software related reasons for partition planned outages.

CHARM does not mitigate the impact of unplanned partition reboots. Minimizing the impact of unplanned partition outages is often achieved through system clustering (PowerHA System-Mirror for example). In such clustered environments planned outages will often be minimized by LPM or other application failover regardless of CHARM capabilities.

PCIe Gen 3 I/O Expansion Drawer

PCIe Gen3 I/O Expansion Drawers can be used in systems to increase I/O capacity.

Figure 13: PCIe Gen3 I/O Drawer RAS Features



These I/O drawers are attached using a connecting card called a PCIe3 cable adapter that plugs in to a PCIe slot of the main server.

Each I/O drawer contains up to two I/O drawer modules. An I/O module uses 16 PCIe lanes controlled from a processor in a system. Currently supported is an I/O module that uses a PCIe switch to supply six PCIe slots.

Two different active optical cable are used to connect a PCIe3 cable adapter to the equivalent card in the I/O drawer module. While these cables are not redundant, as of FW830 firmware or later; the loss of one cable will simply reduce the I/O bandwidth (number of available lanes available to the I/O module) by 50%.

Infrastructure RAS features for the I/O drawer include redundant power supplies, fans, and DC outputs of voltage regulators (phases).

The impact of the failure of an I/O drawer component can be summarized for most cases by the table below.

Figure 14: PCIe Gen3 I/O Expansion Drawer RAS Feature Matrix

Faulty Component	Impact of Failure	Impact of Repair	Prerequisites
I/O adapter in an I/O slot	Loss of function of the I/O adapter	I/O adapter can be repaired while rest of the system continues to operate.	Multipathing/ I/O adapter redundancy, where implemented, can be used to prevent application outages
First Fault on a Data lane (in the optics between the PCIe3 cable adapter card in the system and the I/O module)	None: Spare Used	No Repair Needed: Integrated Sparing Feature	Full functionality implemented in FW830
A Second Data Lane fault or other failure of one active optics cable	System continues to run but numer of active lanes available to I/O module will be reduced	Associated I/O module must be taken down for repair; rest of the system can remain active.	Described functionality implemented in FW840 firmware*. Earlier FW levels may lose all of the I/O in the associated I/O module
Other Failure of PCIe3 cable adapter card In System or I/O module	Loss of access to all the I/O of the connected I/O Module	Associated I/O module must be taken down for repair; rest of the system can remain active.	Described functionality implemented in FW840* on systems with an HMC
One Fan	System continues to run with remaining fan	Concurrently repairable	
One Power supply	System continues to run with remaining fan	Concurrently repairable	
Voltage Regulator Module associated with an I/O module	System continue to run for a phase failure transition to n mode. Other faults will impact all the I/O in the Module	Associated I/O module can not be active during repair; rest of the system can remain active.	Described functionality implemented in FW840* on systems with an HMC
Chassis Management Card (CMC)	No Impact to running system, but once powered off, the I/O drawer can not be re-integrated until CMC is repair	I/O drawer must be powered off to repair (loss of use of all I/O in the drawer)	Described functionality for systems with an HMC
Midplane	Depends on source of failure, may take down entire I/O drawer	I/O drawer must be powered off to repair (loss of use of all I/O in the drawer)	Described functionality for systems with an HMC

*FW840 is not currently available for IBM Power System E850. Systems running FW levels prior to FW840 must have the entire I/O drawer powered off for repair of these components.

It should be noted that in addition to the above, the E880/E870 Power Systems do support the concurrent add of a cable card in support of adding an I/O drawer to a system starting with FW840 firmware. They also support adding an I/O drawer to the system. Adding an I/O module to an existing I/O drawer concurrently is not supported.

Section 2: Hallmarks of Power Systems RAS

This section will give a detailed discussion of the common RAS features of Power Systems based on the POWER8 processor.

Before discussing the details of the design, however, this subsection will summarize some of the key characteristics that tend to distinguish Power Systems compared to other possible system designs.

Integrated System Design

IBM Power Systems all use a processor architected, designed and manufactured by IBM. IBM systems contain other hardware content also designed and manufactured by IBM including memory buffer components on DIMMs, service processors and so forth.

Additional components not designed or manufactured by IBM, are chosen and specified by IBM to meet system requirements, and procured for use by IBM using a rigorous procurement process that delivers reliability and design quality expectations.

The systems that IBM design are manufactured by IBM to IBM's quality standards.

The systems incorporate software layers (firmware) for error detection/fault isolation and support as well as virtualization in a multi-partitioned environment.

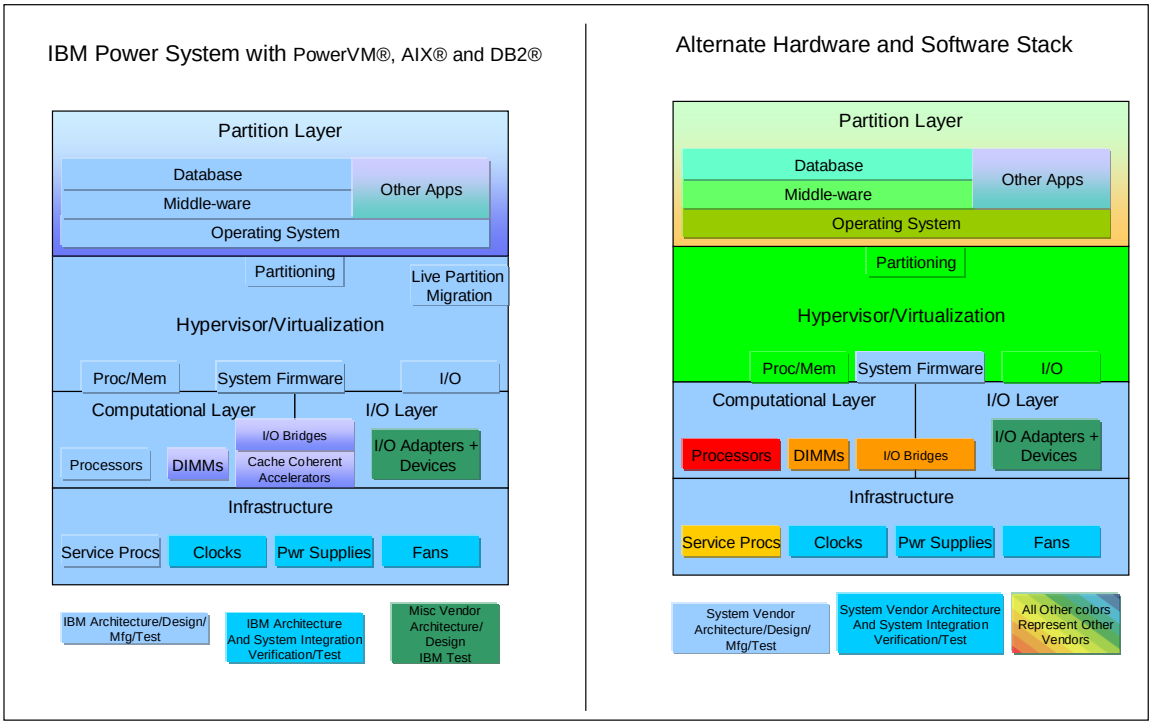
These include IBM designed and developed service firmware for the service processor. IBM's PowerVM hypervisor is also IBM designed and supported.

In addition, IBM offers two operating systems developed by IBM: AIX and IBM i. Both operating systems come from a code base with a rich history of design for reliable operation.

IBM also provides middle-ware and application software that are widely used such as IBM WebSphere® and DB2® pureScale™ as well as software used for multi-system clustering, such as various IBM PowerHA™ SystemMirror™ offerings.

All of these components are designed with application availability in mind, including the software layers, which are also capable of taking advantage of hardware features such as storage keys that enhance software reliability.

Figure 15: IBM Enterprise System Stacks



Where IBM provides the primary design, manufacture and support for these elements, IBM as a single company can be responsible for integrating all of the components into a coherently performing system, and verifying the stack during design verification testing.

In the end-user environment, IBM likewise becomes responsible for resolving problems that may occur relative to design, performance, failing components and so forth, regardless of which elements are involved.

Being responsible for much of the system, IBM puts in place a rigorous structure to identify issues that may occur in deployed systems, and identify solutions for any pervasive issue. Having support for the design and manufacture of many of these components, IBM is best positioned to fix the root cause of problems, whether changes in design, manufacturing, service strategy, firmware or other code is needed.

Alternate System Approach

In contrast, other systems may be designed in a less integrated fashion. The company integrating the system hardware may not have designed the processor or any of the other components integrated into the system besides the system boards and mechanical hardware.

That vendor may be responsible for the system manufacture and test, but may have to do so without the benefit of the detailed knowledge of the internal design of the components being integrated.

The systems sold by the vendor may run using other vendor's designs for the firmware layers such as the hypervisor, as well as the operating system, other virtualization software, and typically also middle-ware and applications.

When it comes to resolving issues in deployed systems, the system vendor may be responsible for replacing failed hardware, but be unable to effectively resolve issues in components the vendor does not design.

When pervasive issues are discovered in components not manufactured or developed by the vendor, the ability to address the issue may be severely restricted.

Hardware That Takes Care of the Hardware

The need for hardware to largely handle hardware faults is fundamental to the Power System design. This means that the error detection, fault isolation and fault avoidance (through recovery techniques, self-healing and etc.) of processors, memory, and hardware infrastructure, are primarily the responsibility of the hardware. A dedicated service processor or specific host code may be used as part of the error reporting or handling process, but these firmware layers, as supplied by IBM, are considered part of the hardware as they are designed, tested and supported along with the underlying hardware system.

Figure 16 below indicates the way in which this handling can be contrasted to an alternative design approach. In the alternative approach, the hardware is considered "good enough" if it is able to transmit error information to software layers above it. The software layers may be capable of mitigating the impact of the faults presented, but without being able to address or repair the underlying hardware issue.

Figure 16: Two Concepts of Server RAS

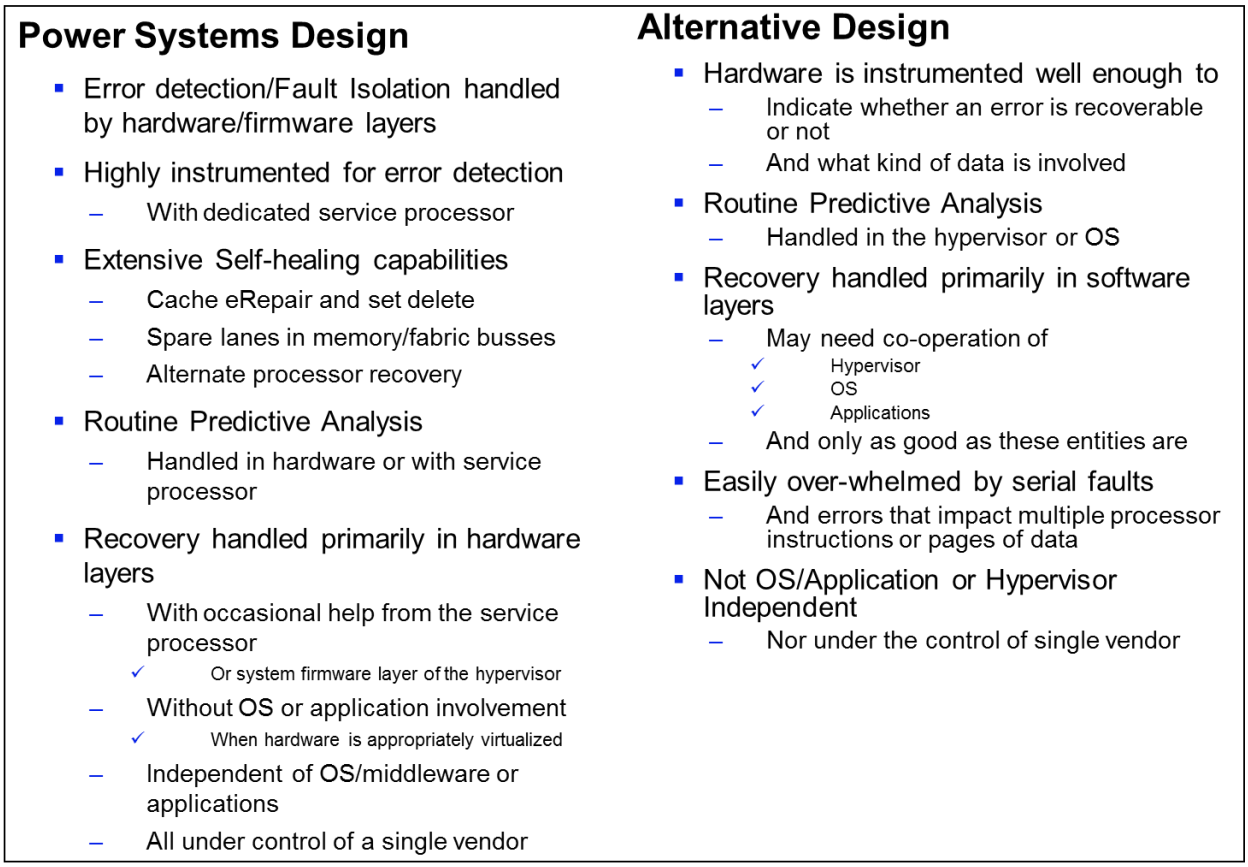


Figure 17: Two Approaches to Handling an Uncorrectable Cache Fault (Unmodified Data)

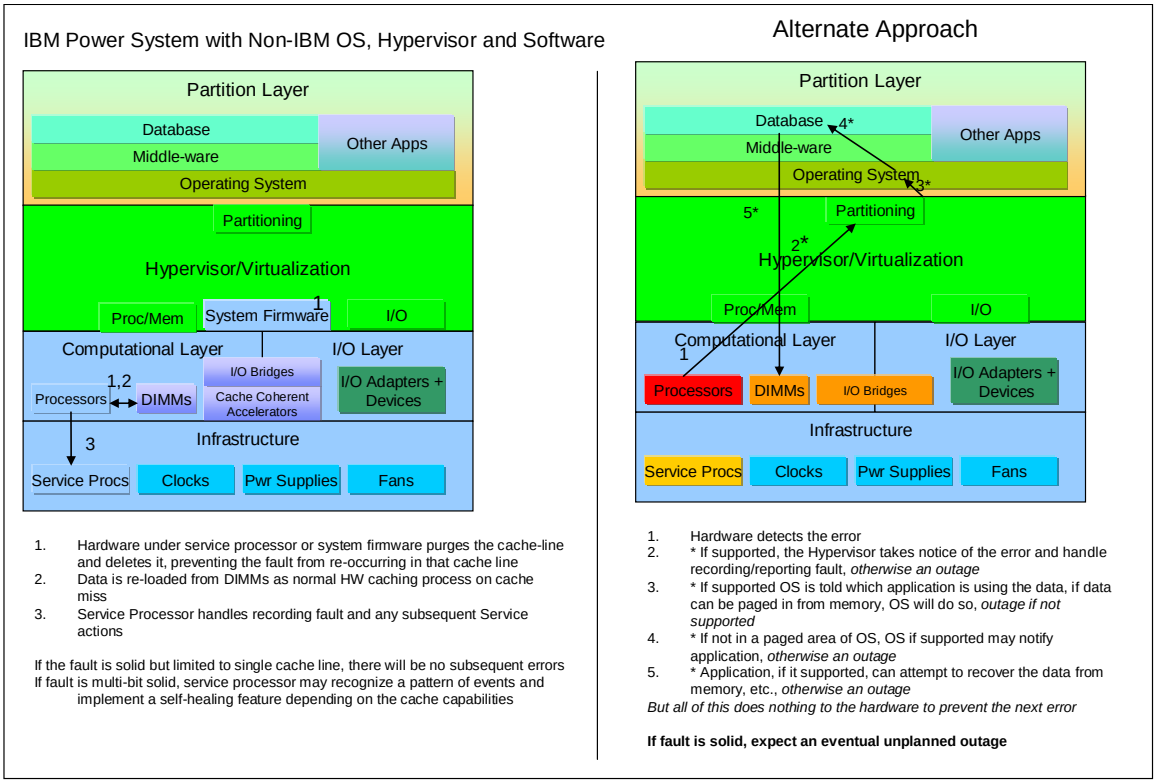


Figure 17 gives an example how even on a software stack not supplied by IBM, an uncorrectable fault in unmodified data in a cache can be detected, isolated, and permanently addressed entirely within the context of IBM supplied hardware and firmware.

This is contrasted with a potential alternate design in which the fault condition is reported to software layers. Co-operation of a hypervisor, operating system, and possibly even a specific application may be necessary to immediately avoid a crash and survive the initial problem.

However, in such environments the software layers do not have access to any self-healing features within the hardware. They can attempt to stop using a processor core associated with a cache, but if a cache line were beginning to experience a permanent error during access, the sheer number of errors handled would likely soon overwhelm any ability in the software layers to handle them. An outage would likely result before any processor migration would be possible.

Simply removing the faulty cache line from use after the initial detected incident would be sufficient to avoid that scenario. But hardware that relies on software for error handling and recovery would not have such “self-healing” capabilities.

Detailed Level 3 cache error handling will be discussed later.

The key here is to understand that the Power Systems approach is in stark contrast to a software based RAS environment where the hardware does little more than report errors to software layers and where a hypervisor, OS or even applications must be primarily responsible for handling faults and recovery.

Based on what was possible in early generations of POWER, Power Systems do also allow for some of the same kinds of software behaviors found in the alternate approach. But, because of the hardware retry and repair capabilities in current POWER processor-based systems, these can be relegated to a second or third line of defense.

As a result, the hardware availability is less impacted by the choice of operating systems or applications in Power Systems.

Leveraging Technology and Design for Soft Error Management

Soft errors are faults that occur in a system and are either occasional events inherent in the design or temporary faults that are due to an external cause.

For data cells in caches and memory and the like, an external event such as caused by a cosmic ray generated particle may temporarily upset the data in memory. Such external origins of soft errors are discussed in detail in Section Six.

Busses transmitting data may experience soft errors due to a clock drift or electronic noise.

Logic in processors cores can also be subject to soft errors where a latch may also flip due to a particle strike or like event.

Since these faults occur randomly and replacing a part with another doesn't make the system any more or less susceptible to them, it makes sense to design the system to correct such faults without interrupting applications, and to understand the cause of such temporary events so that parts are never replaced due to them.

Protecting caches and memory against such faults primarily involve using technology that is reasonably resistant to upsets and also by implementing an error correction code that corrects such errors.

Protecting data busses typically involves a design that detects such faults, allowing for retrying of the operation and, when necessary, resetting the operating parameters on the bus (retraining) all without causing any applications to fail.

Handling faults in logic is more difficult to accomplish, but POWER processors since POWER6® have been designed with sufficient error detection to not only notice key typical software upsets impacting logic, but to notice quickly enough to allow processor operations to be retried. Where retry is successful, as would be expected for temporary events, system operation continues without application outages.

Alternate System Designs

It can be difficult to discover the degree to which an alternate system design protects against these soft error events.

Certain arrays such as caches may be protected with an error correction code capable of fixing a single bit upset by an external source, but not able to handle the occasional double-bit upset which might occur when a cosmic ray particle releases more energy than can be confined to a single bit.

Processors, memory buffers and other logic structures may not be particularly protected and vendors not advertising advanced features such as processor instruction retry, memory buffer replay, and so forth may simply take the approach that detection, but not recovery, is sufficient soft error protection.

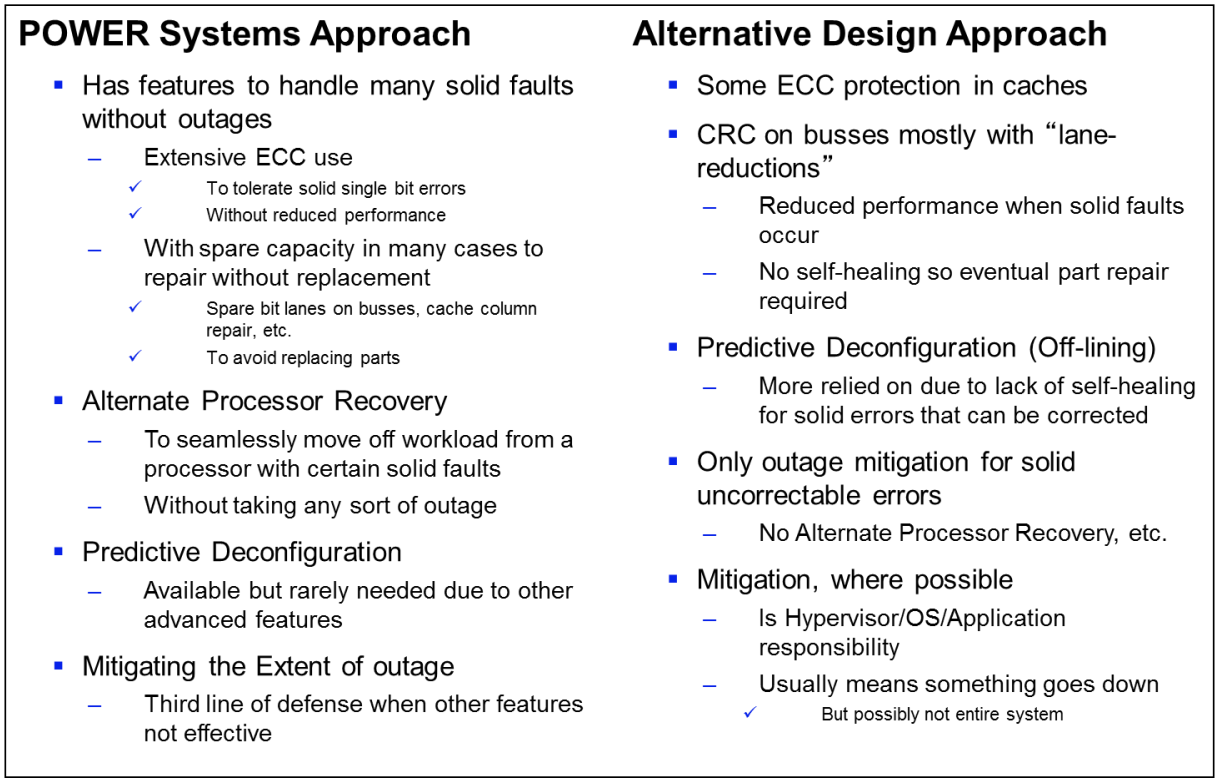
In environments without good fault isolation, it may not even be possible to distinguish between a randomly occurring crash due to a hardware soft error event, and a random code bug causing an outage.

Strategic Spare Capacity Leveraged to Self-Heal Hardware

POWER processors and memory used in systems have built in spare capacity to allow for something that fails to be substituted during run-time with a spare in order to avoid any outage due to the fault or for a repair.

Examples include a spare data lane on a bus, a spare bit-line in a cache, or a spare DRAM module on a DIMM.

Figure 18: Approaches to Handling Solid Hardware Fault



For certain more severe events solid events, such as when a processor core fails in such a way that it cannot even complete the instruction that it is working on, Alternate Processor Recovery allows another processor to pick up the workload on the instruction where the previous one left off.

Alternate Processor Recovery does depend on the hypervisor cooperating in the substitution. Advanced hypervisors such as found in PowerVM are able to virtualize processor cores so long as there is any spare capacity anywhere in the system, no application or operating system awareness is required.

At a system design level, various infrastructure components, power supplies, fans, and so forth may use redundancy to avoid outages when the components fail, and concurrent repair is then used to avoid outages when the hardware is replaced.

System Level RAS Rather Than Just Processor and Memory RAS

IBM builds Power systems with the understanding that every item that can fail in a system is a potential source of outage.

While building a strong base of availability for the computational elements such as the processors and memory is important, it is hardly sufficient to ensure application availability.

The failure of a fan, a power supply, a voltage regulator or a clock oscillator or I/O adapter may actually be more likely than the failure of a processor module designed and manufactured for reliability.

IBM designs systems from the start with the expectation that the system must be generally shielded from the failure of these other components where feasible and likewise that the components themselves are highly reliable and meant to last. Moreover when running without redundancy the expectation is that under normal operation the non-redundant component is capable of carrying the workload indefinitely without significantly decreasing the expected reliability of the component.

Greater than four socket Enterprise Power systems extend redundancy to include clock oscillators, service processors, and even the CEC-level voltage regulator phases within a drawer.

Section 3: POWER8 Common RAS Design Details

POWER8 Common: Introduction

The previous section notes that system RAS involves more than just processor and memory RAS, and more than just hardware RAS.

Power Systems all do share a POWER based processor in common, however, and the RAS features associated with the processor and by extension the memory and I/O paths are keys elements common to application availability in any Power System design.

These design elements, referred to more generally as the computational elements can generally be discussed independent of the design attributes of any particular Power System.

POWER8 Common: Architecture and Technology

Error Reporting and Handling

First Failure Data Capture Architecture

POWER processor-based systems are designed to handle multiple software environments including a variety of operating systems. This motivates a design where the reliability and response to faults RAS in the computational elements is largely independent of the software layers. Power Systems that include the dedicated service processor primarily use code running on the service processor for detecting, managing and resolving such faults, and ensuring that system operations continue or are recovered afterwards.

This approach is beneficial to systems as they are deployed by end-users, but also has benefits in the design, simulation and manufacturing test of systems as well.

Putting this level of RAS into the hardware cannot be an after-thought. It must be integral to the design from the beginning, as part of an overall system architecture for managing errors. Therefore, during the architecture and design of a processor, IBM places a considerable emphasis on developing structures within it specifically for error detection and fault isolation.

Each subsystem in the processor hardware has registers devoted to collecting and reporting fault information as they occur. The design for error checking is rigorous and detailed. The value of data is checked generally wherever it is stored. This is true, of course for data used in computations, but also nearly any other data structure, including arrays used only to store performance and debug data.

Error checkers are derived for logic structures using various different techniques such as checking the validity of state-machine transitions, defining and checking protocols for generated commands, doing residue checking for certain computational instructions and by other means in an effort to detect faults before the resulting impact propagates beyond the detecting sub-system.

The exact number of checkers and type of mechanisms isn't as important as is the point that the processor is designed for very detailed error checking; much more than is required simply to report during run-time that a fault has occurred.

All of these errors feed a data reporting structure within the processor. There are registers that collect the error information. When an error occurs, that event typically results in the generation of an interrupt which is handled by the system's dedicated service processor where available – an IBM designed processor independent of the system processors.

Code within a hypervisor does have control over certain system virtualized functions, particularly as it relates to I/O including the PCIe controller and certain shared processor accelerators. Generally areas in these areas are signaled to the hypervisor.

In addition, there is still a reporting mechanism for what amounts to the more traditional machine-check or checkstop handling.

First Failure Data Capture Advantages

This detailed error detection and isolation capability has a number of key advantages over alternate designs.

Being able to detect faults at the source enables superior fault isolation which translates to fewer parts called out for replacement compared to alternative designs. IBM's history indicates a real advantage in calling out fewer parts on average for faults compared to other systems without these capabilities.

The POWER design also plays a key role in allowing hardware to handle faults more independently from the software stacks as compared to alternatives.

Other advantages in fault handling, design verification and manufacturing test are discussed in greater detail below.

Dedicated Service Processor Recoverable and Unrecoverable Error Handling

As mentioned, Power Systems provide a dedicated, IBM designed, service processor meant to access the fault information of IBM designed modules such as the POWER8 processor.

Ideally the dedicated service processor primarily handles recoverable errors including orchestrating the implementation of certain "self-healing" features such as use of spare DRAM modules in memory, purge and delete of cache lines, using spare processor fabric bus lanes, and so forth.

However, the service processor, when handling an error, has access not only to the basic error information stating what kind of fault occurred, but also access to considerable information about the state of the system hardware – the arrays and data structures that represent the state of each processing unit in the system, and also additional debug and trace arrays that are built into the hardware for performance and error analysis purposes.

In the rare event that a system outage should result from a problem, the service processor has access to this information as well as to the fault isolation registers normally deployed to manage error detection and fault isolation.

Even if a severe fault causes system termination, this access can allow the service processor to determine root cause of a problem, deallocate the failed components when so isolated, and allow the system to restart with confidence that the problem has been resolved.

Should a failure point to a code design issue or other such problem, information collected by the service processor, in conjunction with a hypervisor for code related problems should often be sufficient to identify root cause of the problem, without the need for any sort of recreation. This expectation is based on experience with several generations of POWER processor-based systems with similar capabilities.

Alternate Design

A processor could be designed with less error detection/fault isolation and recovery capabilities. Such an alternate design might focus on sufficient error detection capabilities to note that a failure has occurred and pass fault information to software layers to handle.

For errors that are "correctable" in the hardware – persistent single bit errors, for example, in a cache corrected with an ECC algorithm, software layers can be responsible for predictively migrating off the failing hardware (typically at the granularity of deallocating an entire processor core.)

For faults that are not correctable in hardware – such as double bit errors in a cache corrected with a typical ECC algorithm, software layers are also responsible for "recovery."

Recovery could mean "re-fetching" data from another place that it might be stored (DASD or memory) if it was discovered badly in a cache or main memory. "Recovery" could mean asking a database to rebuild itself when bad data stored in memory is found; or "recovery" could simply mean terminating an application, partition or hypervisor in use at the time of the fault.

It should be understood that the techniques described for a “software” approach to handling errors depends on the software layers running on a system understanding how the hardware reports these errors and implementing recovery.

When one company has ownership of all the stack elements, it is relatively simple to implement such software assisted fault avoidance, and this is what IBM implemented as a primary means in older POWER processor-based systems, such as were sold in the early 2000s.

IBM Power Systems still retain software techniques to mitigate the impact of faults, as well as to predictively deallocate components. However, because “recovery” often simply means reducing system capacity or limiting what crashes rather than truly repairing something that is faulty or handling an error with no outage, more modern Power Systems consider these techniques as legacy functions. These are largely relegated to second or third lines of defense for faults that cannot be handled by the considerable fault avoidance and recovery mechanisms built in to the hardware.

This is still not the case in most alternative designs which also must contend with the limitations of a software approach to “recovery” as well as the multiplication of hypervisors, operating systems and applications that must provide such support.

Finally, alternative designs may offer use of an optional service processor to gain some ability to understand the source of a fault in the event of a server outage. Still without the extensive built-in error detection and fault isolation capabilities with the ability to access trace arrays, other debug structures and so forth, the ability to get to root cause of certain issues even with an optional service processor is likely not available. Special instrumentation and recreate of failures may be required to resolve such issues coordinating activities of several different vendors of the hardware and software components that may be involved.

IPMI/BMC Management Interfaces

Systems with the IBM dedicated flexible service processor will typically use the service processor as a system service management interface as well.

Alternative designs may choose to implement such functions using the Intelligent Platform Management Interface specification with a Base Management Controller (BMC) as the primary out-of-bands monitoring and service management interface.

Two systems designed to the IPMI specification can still vary considerably as to how errors are detected/isolated/self-healed and recovered even while reporting errors through the same interface.

IBM Power S812LC and S822LC systems are suitable for use of the IPMI, do not have an IBM designed service processor. They do make use of a BMC.

More information about those systems is given in the section dedicated to those models.

POWER8 Dedicated Service Processor and Initial Program Load

In previous POWER generations, the processors relied on the service processor not just for error handling, but also for initial program load (IPL) of code and for testing of processors during that initialization.

POWER8 has implemented a new feature that essentially does bulk of the initial program load of an SMP system through the processors themselves, in a fashion much more similar to what is done in alternative processor implementations.

This allows for a speedier system IPL and reduces the requirements on the service processor during system initialization.

It should also be noted that the POWER8 processor design and Power Systems infrastructure permits a system to be configured so that code running in a hypervisor can access the error information instead of a dedicated service processor.

However, there are still advantages in the use of a fully capable dedicated service processor including that:

Using the dedicated service processor means that routine recoverable errors are typically handled without interrupting processors running user workload, and without depending on any hypervisor or other code layer to handle the faults.

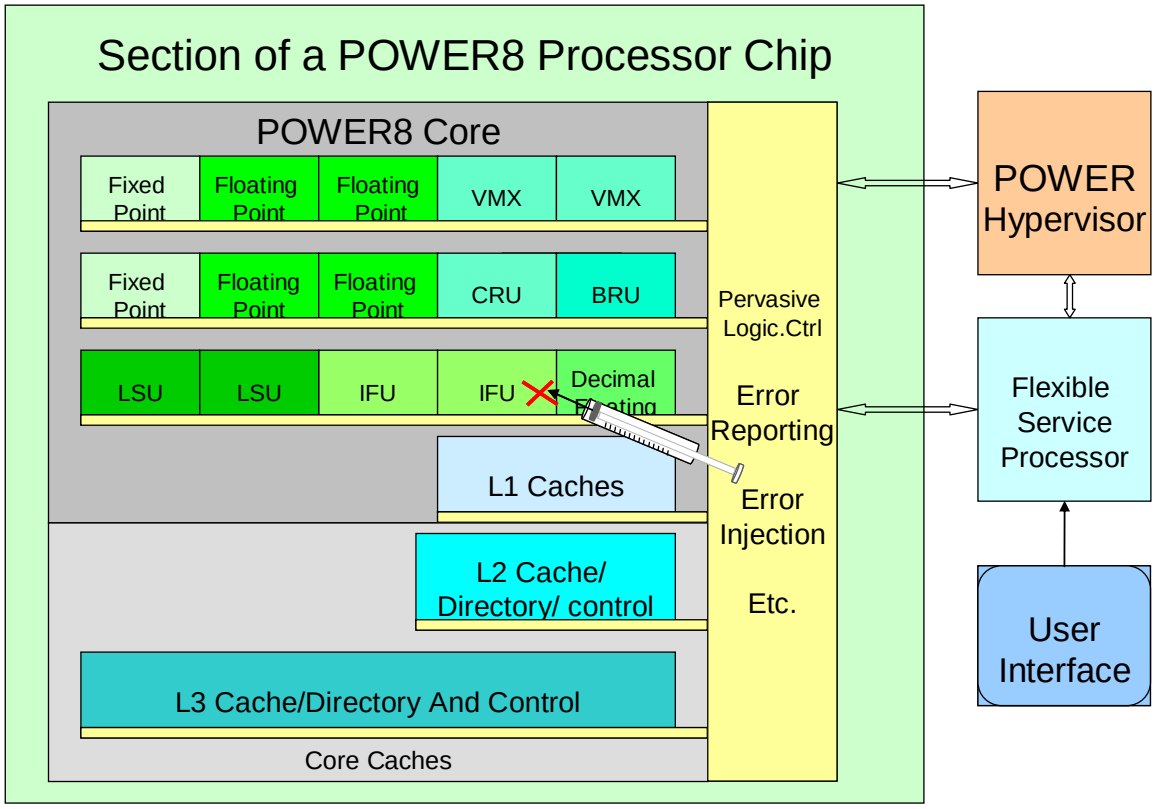
Even after a system outage, a dedicated service processor can access fault information and determine root cause of problems so that failing hardware can be deconfigured and a system can afterwards be trusted to restart afterwards – and in cases where restart is not possible, to communicate the nature of the fault and initiate necessary repairs.

Processor Module Design and Test

While Power Systems are equipped to deal with soft errors as well as random occasional hardware failures, manufacturing weaknesses and defects should be discovered and dealt with before systems are shipped. So before discussing error handling in deployed systems, how manufacturing defects are avoided in the first place, and how error detection and fault isolation is validated will be discussed.

Again, IBM places a considerable emphasis on developing structures within the processor design specifically for error detection and fault isolation.

Figure 19: Processor Error Injection



The design anticipates that not only should errors be checked, but that the detection and reporting methods associated with each error type also need to be verified. Typically when there is an error that can be checked and some sort of recovery or repair action initiated, there will be a method designed into the hardware for “injecting” an error to test directly the functioning of the hardware detection and firmware capabilities. Such error injecting can include different patterns of errors (solid faults, single events, and intermittent but repeatable faults.) Where direct injection is not provided, there will be a way to at least simulate the report that an error has been detected and test response to such error reports.

The ability in logic to inject errors allows simulation of the processor logic to be performed very precisely to make sure that the errors injected are detected and handled. This verifies the processor design for error handling before physical hardware is even built.

It is later used during system test to verify that the service processor code and so forth handle the faults properly.

During testing of a processor, all of this error infrastructure capability can have two uses:

1. Since the service processor can access the state of a processor module using the service processor error checking infrastructure, it also has the ability to alter the state of these registers as well. This allows the service processor to seed certain patterns into hardware, run the hardware in a special environment and monitor results. This "built-in-self" test capability is an effective way of looking for faults in the hardware when individual processor modules are manufactured, and before being built into systems, and to detect faults that might rarely be seen during functional testing.
2. Typically during manufacture of the individual processor modules, a certain amount of "burn-in" testing is performed where modules are exercised under elevated temperature and voltage conditions to discover components that may have otherwise failed during the early life of the product.

IBM's extensive availability to test these modules is used as part of the burn-in process.

Under IBM control, assembled systems are also tested and a certain amount of system "burn-in" may be also be performed, doing accelerated testing of the whole system to weed-out weak parts that otherwise might fail during early system life, and using the error reporting structure to identify and eliminate the faults.

Even during that system burn-in process, the service processor will be used to collect and report errors. However, in that mode the service processor will be more severe about the testing that is performed. A single fault anywhere during the testing, even if it's recoverable, will typically be enough to fail a part during this system manufacturing process.

In addition, processors have arrays and structures that do not impact system operation even if they fail. These include arrays for collecting performance data, and debug arrays perhaps only used during design validation. But even failures in these components can be identified during manufacturing test and used to fail a part if it is believed to be indicative of a weakness in a processor module.

Having ownership of the entire design and manufacture of both key components such as processor modules as well as entire systems allows Power System manufacturing to do all of this where it might not be possible in alternate system design approaches.

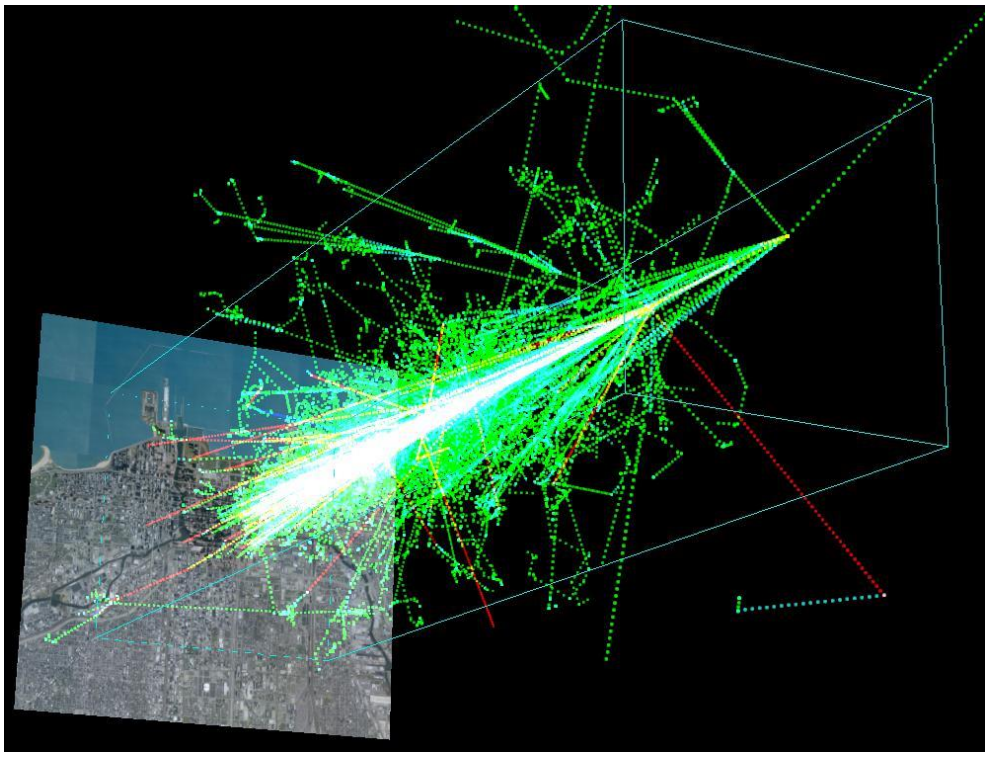
Soft Error Handling Introduction

It was earlier stated that providing protection against outages and repair actions for externally induced soft error events is a hallmark of Power Systems design.

Before discussing how this is done, an explanation of the origin of externally induced soft errors and their frequency will be provided.

Since the early 1900s, cosmic rays have been observed and analyzed. Cosmic rays are highly energetic particles originating from other galaxies. They consist mostly of protons and atomic nuclei. When these highly energetic particles go through earth's atmosphere, they produce a variety of daughter particles, e.g. neutrons, electrons, muons, pions, photons and additional protons. In 1936, Victor Hess and Carl David Anderson shared the Nobel Prize in Physics for their work on cosmic rays. Figure 20 shows a modern simulation of the daughter particle generation caused by a 1TeV proton hitting the atmosphere 20 km above Chicago.

Figure 20: Simulation of 1 TeV proton hitting atmosphere 20 km above Chicago -- University of Chicago, http://astro.uchicago.edu/cosmus/projects/aires/[2]



Cosmic ray events with these types of energies are not infrequent and the neutrons generated in this process are of concern for semiconductors. They can find their way into the active silicon layer and cause harm through an inadvertent change of the data stored e.g. in an SRAM cell, a latch, or a flip-flop, or a temporary change of the output of logic circuitry. These are considered to be soft error events since the problem is of a temporary nature and no device damage results.

To understand the nature and frequency of such events, monitoring facilities have been established at a number of locations around the world, e.g. Moscow (Russia), Newark (Delaware USA), as well as near the South Pole. Flux, a measurement of events over an area over time, is a function of altitude: Higher flux is observed at higher altitude due to less protection from the atmosphere. Flux also increases at the poles due to less protection from earth's magnetic field. The neutron flux can be mapped for all locations on planet earth.

In addition to cosmic neutrons, alpha particles emitted from semiconductor manufacturing and packaging materials play a role. Their energy is sufficiently high that they also can reach the active silicon layer, and cause soft errors.

The JEDEC Solid State Technology Association has published a standard related to the measuring and reporting of such soft error events in semiconductor devices. From the JEDEC standard 89A[3] it is frequently quoted that 12.9 cosmic ray-induced events of greater than 10MeV can be experienced per square-centimeter per hour of greater 10MeV in New York City (at sea level.)

As an example, a single processor module with die-size of 0.3 cm^2 might expect to see over ninety such events a day – for the cosmic ray-induced events only.

---

[2] Courtesy of Sergio Scuitto for AIRES and Maximo Ave, Dinoj Surendran, Tokonatsu Yamamoto, Randy Landsberg, and Mark SubbaRao for the image as released under Creative Commons License 2.5 (http://creativecommons.org/licenses/by/2.5/)

[3] Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices, *www.**jedec**.org/sites/default/files/docs/jesd**89a**.pdf*

Measuring Soft Error Impacts on POWER Processors

Clearly not every event in any semiconductor device will lead to application failure or loss of data. The soft error fail rates of circuitries are strongly dependent on the technology (bulk vs., SOI[4], elimination of ^{10}B from semiconductor manufacturing line[5]), the design style (hardened design[6,7] vs. cost optimized design), the storage element (SRAM vs., DRAM vs., flash vs. flip-flop) and other parameters.

The Silicon on insulator technology used in POWER8 processors provides approximately one order of magnitude advantage over bulk[2]. A low alpha emission base line is achieved through selecting semiconductor manufacturing and packaging materials carefully. Also, very significantly, SER hardened latches as described in [5] are used throughout.

Even so, Power Systems do not depend on technology alone to avoid the impact of soft error events. As will be explained in detail later, the POWER8 processor design preference is to build error correction or retry capabilities throughout the processor in caches, cores and other logic so that soft errors that still occur are handled without impact to applications or other software layers.

Therefore the Power Systems approach is to leverage as possible techniques in the design of latches and other components within a technology to minimize the occurrence of soft error events, then add to that design techniques to minimize or eliminate, if possible, their impact.

The JEDEC 89A standard discusses not only the potential for soft error events to occur, but also the need to develop methods to validate a processor's ability to handle them.

The means by which IBM validates the ability handle soft errors in the processor, including accelerated proton beam exposure and a radioactive-underfill testing, is discussed in detail in POWER7 RAS whitepaper.

Alternative Designs

Alternative designs that do not make the same technology choices as POWER8 may experience higher rates of soft errors. If the soft error handling in the hardware does not extend past single bit error correction in caches, the expectation would be that most events impacting logic, at best, would be reported to software layers for mitigation. Software mitigation typically means termination of applications, partitions or entire systems.

In addition to taking outages for soft errors, it is typically not possible after a crash to determine that the soft error was in fact an externally induced event instead of a hardware fault. Vendors implementing a service policy may be forced to either take away resources and cause downtime to remove and replace parts for the soft errors, or force customers to experience multiple outages for real hardware problems.

---

[4] P. Roche et al, "Comparisons of Soft Error Rate for SRAMs in Commercial SOI and Bulk Below the 130-nm Technology Node", IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 50, NO. 6, DECEMBER 2003

[5] "B10 FINDING AND CORRELATION TO THERMAL NEUTRON SOFT ERROR RATE SENSITIVITY FOR SRAMS IN THE SUB-MICRON TECHNOLOGY", Integrated Reliability Workshop Final Report (IRW), 2010 IEEE, Oct. 17-21, 2010, pp. 31-33, Shi-Jie Wen , S.Y. Pai, Richard Wong , Michael Romain , Nelson Tam

[6] "POWER7 Local Clocking and Clocked Storage Elements," ISSCC 2010 Digest of Technical Papers, pp. 178-179. James Warnock, Leon Sigal, Dieter Wendel, K Paul Muller, Joshua Friedrich, Victor Zyuban, Ethan Cannon, A.J. KleinOsowski

[7] US Patent 8,354,858 B2, 15-Jan-2013

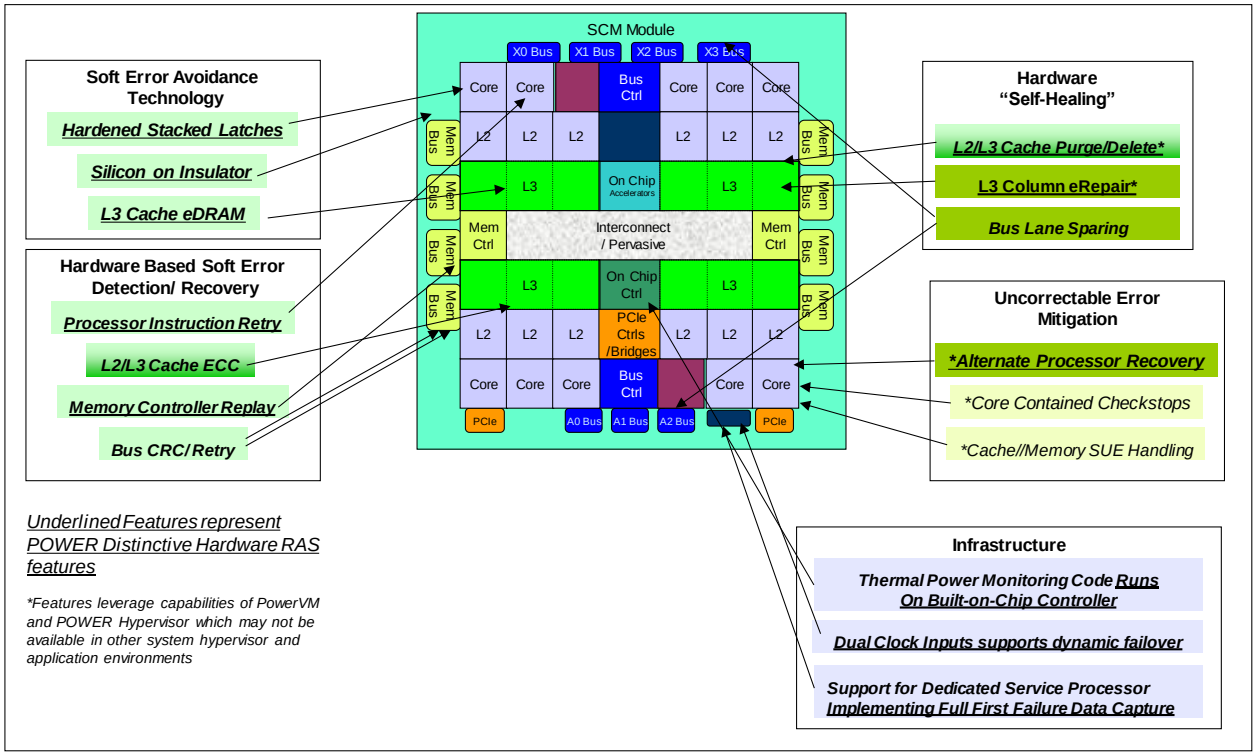
POWER8 Common: Processor RAS Details

The previous discussion in this section gives a general description of how a POWER processor is designed for error detection and fault isolation and how this design is used during processor manufacture and test. It also outlines how the POWER processor technology is hardened against soft errors.

The rest of this section now looks specifically at error handling described by type and area of the processor involved.

Different Lines of Defense

Figure 21: Key RAS Capabilities of POWER8 Processors by Error Type and Handling



Comprehensive Fault handling within a processor must deal with soft errors, hardware faults that do not cause uncorrectable errors in data or computation, and faults that do cause uncorrectable errors.

As much as possible, there should be error handling in every area of the processor sufficient to handle each of these errors so that soft errors cause no outages or parts replacement, correctable hard errors are repaired so that neither outages nor replacements result, and the outage impact of uncorrectable faults are minimized.

For POWER8 that generally means

1. Faults that have causes other than broken hardware – soft error events – are minimized through use of technology particularly resistant to these events. Where soft errors nonetheless result in impacts to logic or data, techniques in the hardware may be used to recover from the events without application impact.
2. When a solid fault is detected spare capacity and other techniques may be available to "self-heal" the processor without the need to replace any components for the faults.
3. The impact of uncorrectable errors is mitigated where possible using the minimum involvement of any hypervisor OS or applications. Leveraging a virtualized environment under PowerVM, this may include handling events without any OS or application involvement or knowledge. In certain virtualized environments, some methods of mitigation, such as

Alternate Processor Recovery may involve no application outages at all and the mitigation also includes a solid means of preventing subsequent outages.

Figure 21 illustrates key processor RAS features. It identifies those features distinctive to POWER compared to typical alternative designs where in the alterative designs:

1. Soft error protection is primarily for busses and data and for data primarily limited to use of error correction codes often without distinction between soft and hard errors. Soft errors in miscellaneous logic are not addressed by design.
2. Failing hardware that causes correctable error events are deconfigured by deallocating major system component such as entire processor cores, usually impacting performance. Depending on the software layers involved this may require involvement from the hypervisor, operating system and applications. Restoring system performance requires a repair action to replace the components.
3. The impact of any remaining uncorrectable errors, including perhaps those due to soft errors in logic, also involves software layers. Without cooperation from operating systems and application layers, mitigation at best means terminating something, typically a partition. When a fault impacts the hypervisor, mitigation may mean termination of the system. More importantly when a solid fault is “mitigated” the root cause is rarely addressed. Repeatedly encountering the problem while in the middle of recovery can be expected to cause a system outage even if the original incident could be mitigated.

Processor Core Details

Core Logic Soft Errors

As explained earlier, the first line of defense against soft-errors is the use of hardened latches and other technology advantages described earlier.

Certain soft error events within the core that cannot be handled in this way may still be transparently recovered. The POWER7 whitepaper goes into considerable detail into this RAS explaining how POWER7 core errors are handled.

For example, the paper explains the processor ability to detect many faults during the instruction in which they are executing and before execution is complete. When such a fault is detected in a core running partition code, it can be possible for the processor to retry the failing instruction. When the retry of the failing instruction succeeds, processing continues with no interruption of service and no need ever to take an outage.

Alternate Design for Core Logic

This is contrasted with what may be done by an alternate design that depends primarily on software. When such a fault is detected, typically whatever code level is handling “machine checks” from a processor (either OS or hypervisor) would be informed that there is an error.

The error report would indicate that there was a fault, and give details explaining what core took the error.

If the fault could be limited to an OS kernel in a virtualized system, rather than simply recording the event and allowing the processor to retry, recovery typically would consist of the OS taking a kernel panic and going down – taking with it whatever applications were running within the partition.

If the fault could not be contained to the partition, then typically an entire system outage will occur.

Core Logic Solid Faults

While recovery is useful for soft-error induced temporary faults. For solid faults, the error condition will persist and multiple retries will not fix the issue. In some cases in a POWER8 processor it is possible to still catch the solid fault before an instruction is complete and essentially migrate the state of the failed processor to another processor.

Presuming sufficient resources and virtualization under PowerVM, these faults result in the workload of the running processor simply being migrated to a spare, with a potential decrease in performance, but potentially also no application outage at all.

This method, called Alternate Processor Recovery (APR) does depend on the hypervisor in use, but with processor virtualization can work with any operating system.

The POWER7 whitepaper extensively discusses Alternate Processor Recovery along with techniques PowerVM uses to maintain application availability where possible according to a customer defined priority scheme.

As in POWER7, APR is not viable for all possible core uncorrectable errors. If the running state of one core cannot be migrated, or the fault occurred during certain window of time, then APR will not succeed. For a number of these cases, the equivalent of a machine-check could be generated and treated by the hypervisor. With PowerVM such “core contained checkstops” would allow termination of just the partition using the data when the uncorrectable error occurs in a core being used by a partition.

In the alternate design approach, soft errors are not retried, so any error that can be detected, hard or soft, will result in the same error behavior as previously described. And again, the POWER approach proves superior in the sense that seamless recovery may occur where in the “alternate design” recovery just means potentially limiting the crash to a partition or application instead of the entire system.

Still not every fault can be handled, even within a core, by any of these techniques. Where the result of such faults leads to system termination, having detailed fault analysis capabilities to allow the failed hardware to be detected and isolated allows a system to be restarted with confidence that it will continue to run after the failure.

Caches

The fact that advanced techniques for handling error events are architected is insufficient to understand whether such techniques cover a wide spectrum of faults, or maybe just one or two conditions.

To better illustrate the depth of POWER processors fault handling abilities, the example of the level 3 (L3) cache will be explored in depth for a system with a dedicated flexible service processor and POWER Hypervisor. For simplification this example speaks of the L3 cache but POWER8 processors are also capable of doing the same for the L2 cache.

A cache can be loosely conceptualized as an array of data, with both rows of data which are typically used to comprise a cache line and columns with bits contributing to multiple cache lines.

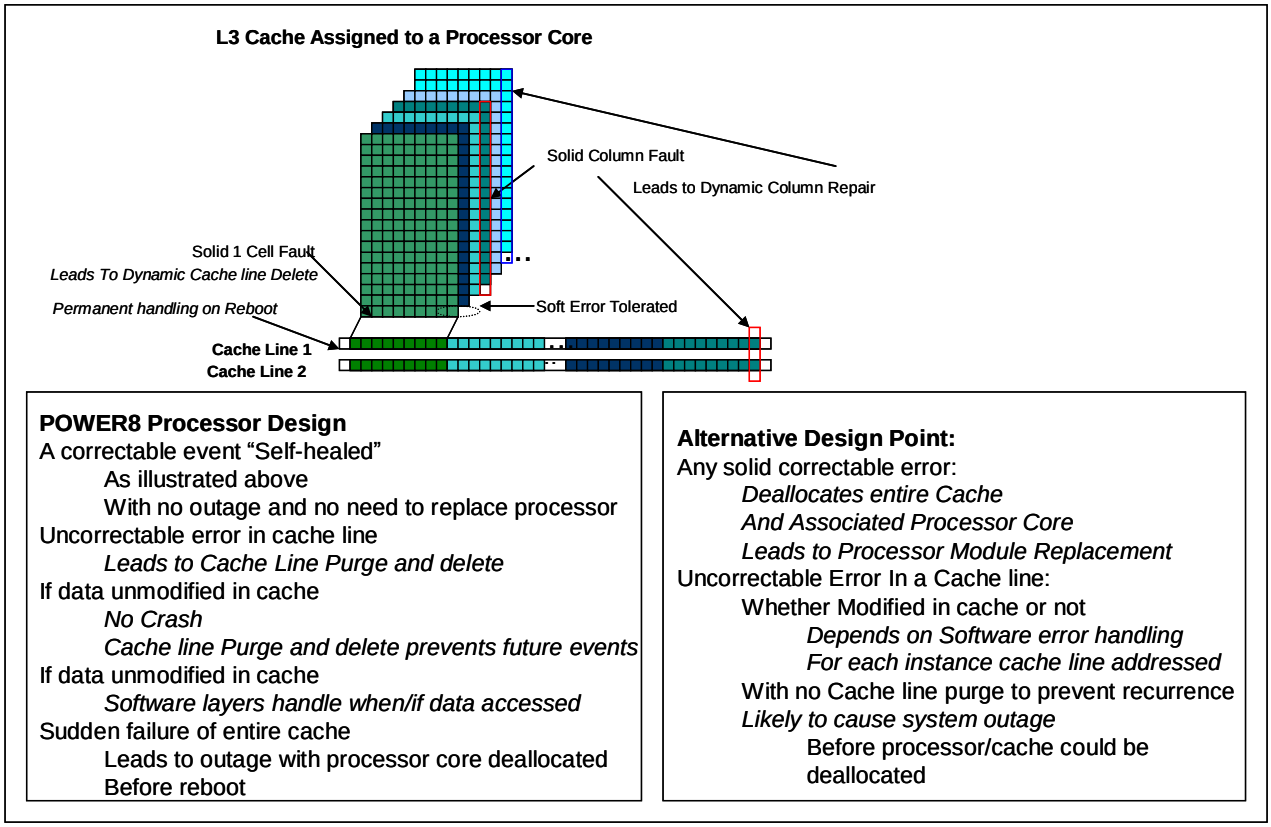
An L3 cache can contain data which is loaded into the cache from memory, and for a given cache line, remains unmodified (nothing written into the cache that has not also been written back in to memory.) An L3 cache line can also contain data that has been modified and is not the same as the copy currently contained in memory.

In an SMP system with multiple partitions, the data in an individual cache line will either be owned by a hypervisor or by some other partition in the system. But the cache line itself can contain memory from multiple partitions.

For the sake of this discussion it will be assume that an L3 cache is segmented into areas owned by different cores so that any given area of the cache is accessed only by a single processor core; though this may not always be the case.

Figure 22 below gives an illustrative representation of an L3 cache but is not meant in any way to accurately describe the L3 cache layout or error correction code implemented for each cache line.

Figure 22: L3 Cache Error Handling



Soft Errors

The data in the L3 cache is protected with an Error Correction Code (ECC) that corrects single bit errors as well as detects at least two bit errors. Such a code is sometimes abbreviated as SEC/DED ECC. With SEC/DED ECC a soft error event that upsets just a single bit by itself will never cause an outage.

Not all particle events, however, are of equal magnitude; some may impact not only one cell of data, but an adjacent cell as well.

The cache design accounts for that by the layout of cache lines so that adjacent cells being impacted by a soft error upset do not by themselves lead to uncorrectable errors. The eDRAM technology deployed for L3 and L4 caches also provides significant technology protection.

In addition logic used in controlling the L3 cache and elsewhere in the processor chip can also be subject to soft errors. The cache directory is therefore also protected with SEC/DED ECC. Certain other data structures for logic are implemented by hardened latches featuring series resistors in stack form (otherwise known as "stacked latches") making them less susceptible to soft error events. Detailed checking and even a level of recovery are implemented in various logic functions.

Intermittent and Sold Faults

Still a physical fault in a cell within the L3 cache can occur over time and cause a correctable error when that given cell is accessed – perhaps intermittently at first, and then more solidly. This would be termed a solid correctable error.

Other faults may impact logic within the L3 cache and cause uncorrectable errors on one or more cache lines. It is also possible to introduce a fault in the L3 cache that impacts an entire "column" of data with multiple cache lines impacted and requiring correction whenever the data is accessed.

Persistent correctable errors that impact a single cache line are prevented from causing an immediate outage because of the *SEC/DED ECC employed.* To prevent such errors from aligning with some other fault at some point in time (including a random soft error event) firmware can direct the hardware to write all of the contents of a cache line to memory (purge) and then stop using that cache line (delete). This purge and delete mechanism can permanently prevent a persistent fault in the cache line from causing an outage, without the needing to deallocate the L3 cache or processor core using the cache.

In addition, if a fault impacts a “column” of data, there is spare capacity within the L3 cache to allow the firmware to substitute dynamically a good column for a faulty one, essentially “self-healing” the cache without the need to take any sort of outage or replace any components in the system.

If it should happen, despite all of the above, that a cache line did experience an uncorrectable error, the system in purging the cache will mark the impacted data as stored in memory with a special code. When the data is subsequently referenced (if the data is subsequently referenced) by an application, a kernel or a hypervisor, a technique called “Special Uncorrectable Error Handling” is used to limit the impact of the outage to only what code owns the data. (See the POWER7 RAS whitepaper for additional details.)

It should be emphasized that special uncorrectable error handling is only needed if data in a cache is modified from what is stored elsewhere. If data in a cache was fetched from memory and never modified, then the cache is not purged. It is simply deleted and the next time needed, the will be data re-fetched from memory and stored in a different cache.

Cache “Alternate” Software driven approach

In an alternate design, without the ability to cache line delete or the extra capacity to repair a column, the discussion of L3 error handling would begin with the error correction capabilities. If parity protection is offered, a single bit error can be detected, but cannot be recovered from. Parity is not very useful in avoiding soft errors in a cache itself. If a SEC/DED ECC code is used, the single random soft error will be corrected.

Of course, as indicated above, soft errors can occasionally impact more than one bit and so simply saying SEC/DED ECC protection isn’t sufficient to state how well such an array is protected from soft errors.

Typical alternate designs do not handle “fixing” solid problems in the caches by either cache-line delete or bit-line repair, however. Instead it may be the responsibility of a software layer to be told or recognize when there is a pattern of correctable errors in the cache. Stopping use of the entire cache would involve migrating every application in every partition that is using the cache off of the cache by essentially deallocating the associated processor core.

If the system has a spare core available for some reason (not allocated to any partitions, etc.) then the spare core could also be assigned to the partitions using the old core to restore the system performance/capacity previously enjoyed.

All of this takes the co-operation of multiple software layers – a hypervisor doing partitioning (if partitioning is done) and the operating system most likely. In some cases even applications need to be involved because if a processor is dedicated to an application and the application never terminates a thread running on a core, then the core may never get the chance to be deallocated.

Typically also, to restore original capacity the processor module containing the cache would eventually have to be replaced.

While of this predictive error handling is in progress, a random soft error can occur and cause uncorrectable error in a cache line. Also possible is that an intermittent fault in the cache initially causes correctable errors but grows worse and eventually causes uncorrectable errors.

If the fault were in modified cache data, then the impact of these uncorrectable errors could be mitigated by some version of special uncorrectable error handling as described for POWER. This would depend on the software layers implementing this approach.

However, without any ability to correct the problem in the cache itself, if the problem were due to anything other than a single event, then re-fetching the data to the same cache would cause the same sort of problem. A hypervisor or other code able to handle multiple such cache events and their impact to code

will attempt to deallocate a processor core. This function may be neither easy to design nor effective at avoiding outages.

For a fault that impacts multiple cache lines, it is reasonable to expect that such faults will end up causing all partitions to terminate as well as the hypervisor.

Single Points of Failure

Given all of the details of the RAS design, a simplifying question is often asked: how many "single points of failure" does the system have? This question can be difficult to answer because there is no common definition of what that means.

Single Points of Failures and Computational Elements

One definition of a single point of failure (SPOF) is any element in the system which, if it fails causes at least one application to fail.

Using that definition, absent full mirroring of everything, arguably all the processors and memory in a system are single points of failure because there can be some failures that will cause at least one application to fail.

In practice, the degree to which these failures can be avoided is the degree to which such components are not single points of failure by this definition. Soft error protection, integrated sparing of key elements such as DRAM modules and bit-lines, and recovery techniques such as Alternate Processor Recovery are all meant to avoid such faults being single points of failure. Software driven "recovery" often is not.

Alternatively, one could say that a component is a SPOF only if it can fail in such a way as to take down *all* of the applications running in a system. Given "recovery" techniques the alternate design uses to minimize the impact of certain outages, it would be expected that such faults would be rarer than individual application outages.

Nevertheless if recovery means terminating whatever is running on a core at the time of a fault, and a core is running the hypervisor, then again, such faults will take down the hypervisor and all of the applications running on top of it – faults driving such events would therefore also be single points of failure by that definition.

Overall, it should be expected that any such design that uses a single hypervisor to manage the partitioning of a system can experience some processor related failures, at least, that take down the system. This can also be true of Power Systems design, though the key advantage in Power Systems is the considerable amount of hardware recovery built-in to handle faults and rely as little as possible on software based recovery.

It is worth noting that there are some alternate designs where a single "system" is physically partitioned into separate pieces and each piece runs its own hypervisor. The claim would then be that no fault in the computational elements, processors, memory, etc. can take down the entire system; though there certainly would be faults that can take down a hypervisor and everything running on top of it. Still by the prior definition it can be claimed that there is no system-wide single point of failure in the processors and memory. However, since the physical partitions don't share the same hypervisor image, they can't share any common resources to help with virtualization of resources, drive utilization and so forth. From an application outage viewpoint it is better, therefore, to run separate hypervisors in separate systems, not separate partitions in the same system. This is because any time a piece of hardware is shared between two components there is a potential of a common fault that causes an outage, even if what is shared is considered redundant. This will be explored further in an upcoming section.

Still another definition of single point of failure is that there should be no fault that can take a system down *and keep it down until repaired.* In other words, even if a rare fault in a processor should cause a hypervisor crash, the system should be able to re-boot and run after deconfiguring the failing hardware.

This is not something that should be taken for granted in any processor design. It depends on the ability to properly detect and isolate failures and to fence out the failing hardware. It should be noted that Power Systems are designed so that logic faults isolated to a processor core, cache, SMP communications path

or a memory subsystem (presuming sufficient memory in the system) do not keep a system down until repair even in the rare case that this may occur.

This is a key element of the over-all processor and processor-interconnect design structure.

POWER8 Common: Memory RAS Details

Just as processor cores and caches should be protected against soft errors, unplanned outages and the need to replace components, so too should the memory subsystem. Having such adequate protection is perhaps more critical given the amount of memory available in large scale server.

The POWER8 processor can support up to 8 DIMMS per processor module and as of this writing DIMMS supporting 64GB. Memory therefore can occupy a substantial portion of a 1s or 2s server footprint. The outages that can be caused (planned or unplanned) by inadequately protected memory can be significant.

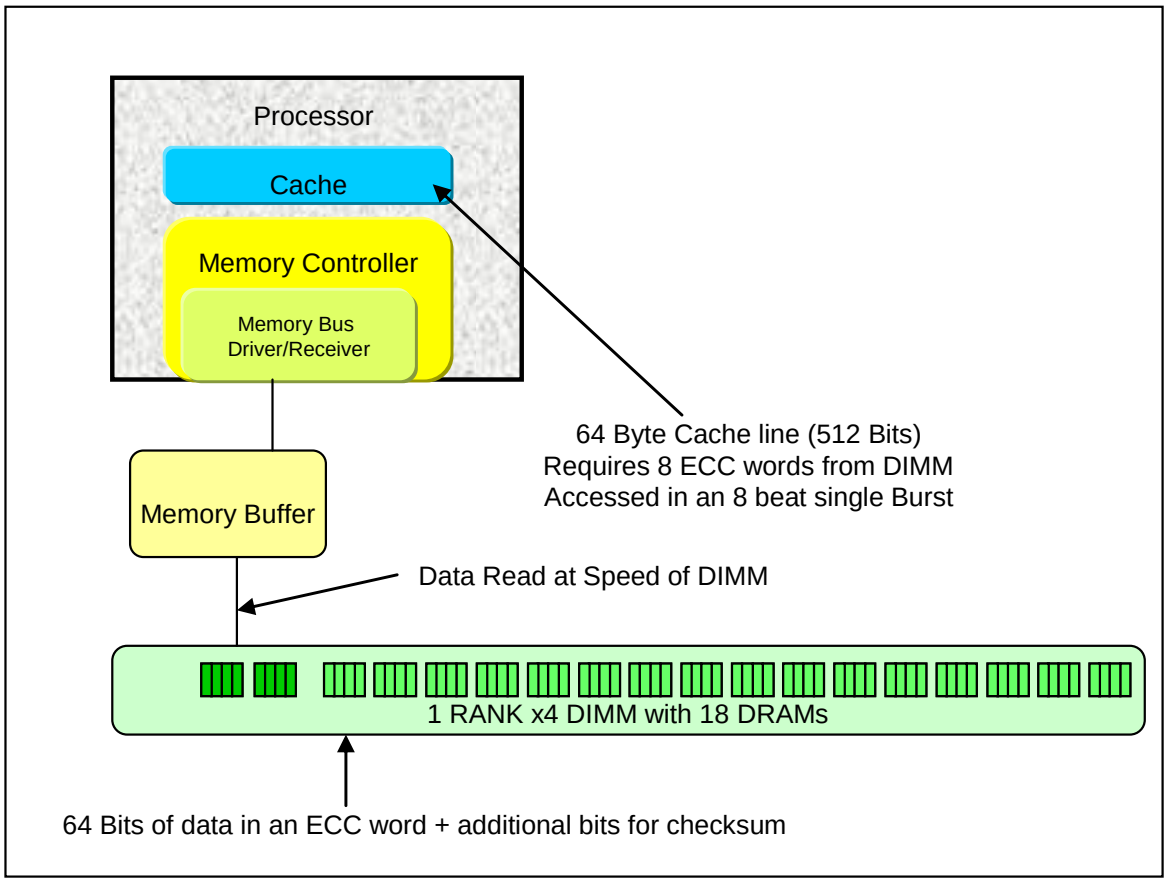
Memory Design Introduction

Before a detailed discussion of POWER8 processor-based systems memory design, a brief introduction into server memory in general will be useful.

Memory Organization

Briefly, main memory on nearly any system consists of a number of dynamic random access memory modules (DRAM modules) packaged together on DIMMs. The DIMMs are typically plugged into a planar board of some kind and given access through some memory bus or channel to a processor module.

Figure 23: Simplified General Memory Subsystem Layout for 64 Byte Processor Cache Line



For better capacity and performance between the processor and DIMMS, a memory “buffer” module may also be used; the exact function of which depends on the processor and system design.

The memory buffer may be external to the processor or reside on the DIMM itself, as is done in POWER8 custom enterprise DIMMS.

Like caches, a DRAM module (or DRAM for short) can be thought of as an array with many rows of data, but typically either 4 or 8 “columns.” This is typically denoted as x4 or x8.

DRAM modules on a DIMM are arranged into what will be called here “ECC” words. ECC words are a collection of one row of data from multiple DRAMS grouped together. A typical design may use 16 x4 DRAMs grouped together in an ECC word to provide 64 bits of data. An additional 2 DRAMs might also be included to provide error checking bits, making for a total of 18 DRAMs participating in an ECC word.

When a processor accesses memory, it typically does so by filling a cache-line’s worth of memory. The cache line is typically larger than an ECC word (64 bytes or 128 bytes, meaning 512 or 1024 bits). Therefore several ECC words must be read to fill a single cache line. DDR3 memory is optimized to yield data in this manner, allowing memory to be read in a burst yielding 8 rows of data in 8 beats to fill a 64 byte cache line.

In Figure 23 above, it is presumed that the data of two additional x4 DRAMs worth of data are reserved for redundancy required for error checking.

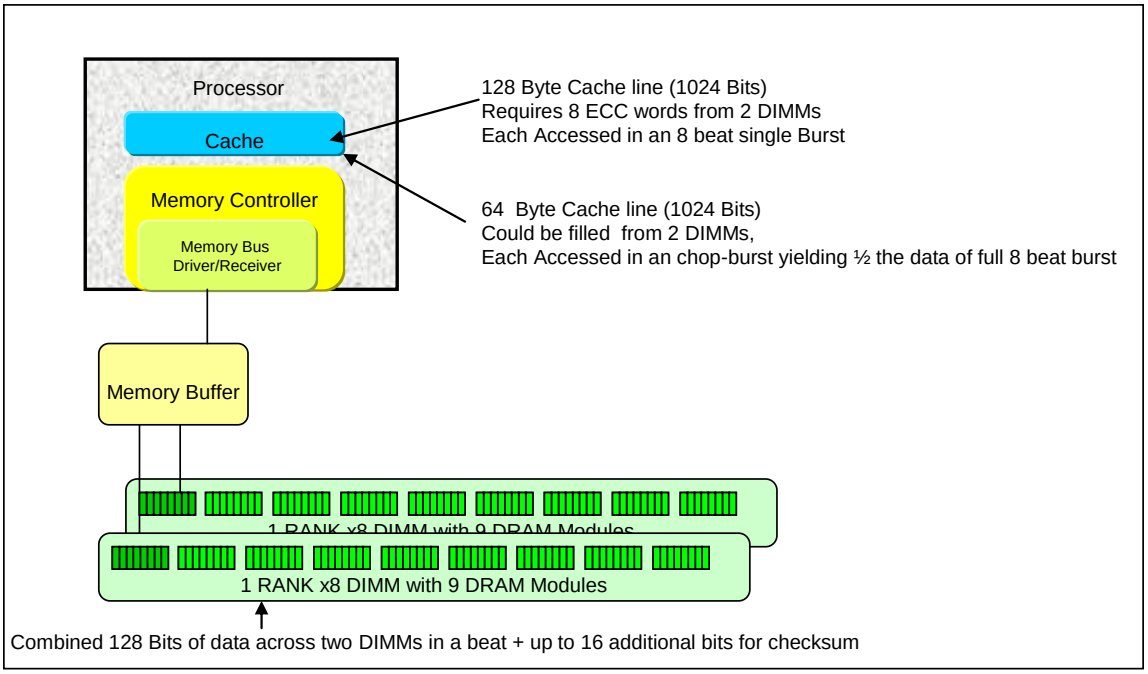
This amount is typically enough, given a proper layout of bits within the ECC word, to allow the system to correct failures in multiple DRAM modules and actually correct the failure of an entire DRAM module.

IBM calls the capability of correcting all of the data associated with a bad DRAM “Chipkill correction”. Other vendors use different names for the same concept (single device data correct, for example).

The same simplified picture could also be drawn using x8 DRAM modules instead of x4. x8 DRAM modules supply 8 bits to each ECC word, so only ½ as many are needed for the same sized ECC word. A typical DIMM also supplies another DRAM module for error correction checksum and other such information (giving a total of 9 DRAM modules.) However, even with that extra DRAM, there is typically insufficient checksum information available to do Chipkill correction.

POWER8 cache sizes are 128 bytes wide, rather than 64. To fill a cache line efficiently from such industry standard DIMMs, two DIMMs would need to be accessed.

Figure 24: Filling a Cache Line Using 2 x8 Industry Standard DIMMs



One of the advantages of accessing memory this way, is that each beat in total would access 128 bits of data and up to 16 bits of checksum and other such information. Advanced ECC algorithms would be capable of correcting a Chipkill event in such a configuration even when x8 DRAM chips are used (the

more challenging case since twice the number of consecutive bits would need to be corrected compared to the x4 case.)

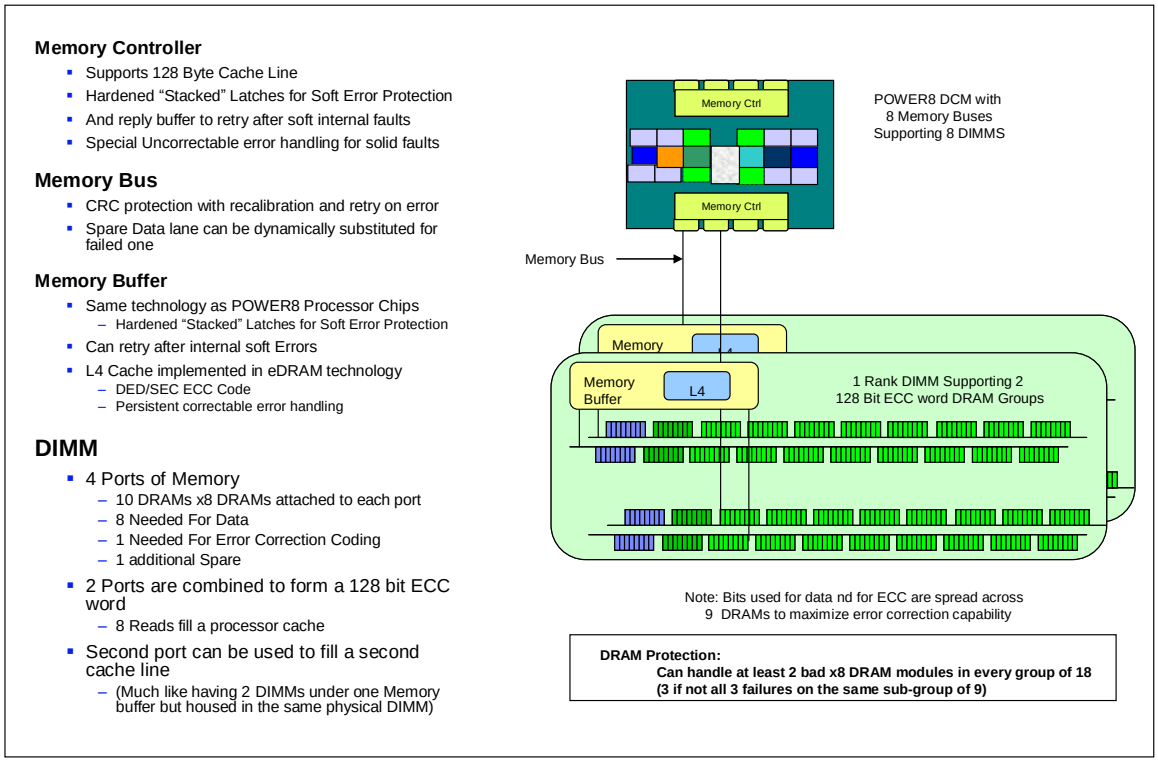
The example also shows that it would be possible to combine the output of two DIMMs when filling a 64 byte cache line. In such a case, a chop-burst mode of accessing data could be used. This mode, instead of supplying 8 bytes of data per x8 DRAM per access (as would be done in the normal 8 beat burst), supplies 4 bytes of data while using twice the number of DRAMs across each DIMM.

This mode also allows the full width of two DIMMS, 128 bits of data and up to 16 bits of checksum data, to be used for data and error correction and should be capable of correcting a Chipkill event using x8 DRAMs, or two Chipkill events if x4 DRAMs are used.

Typically such industry standard DIMMs are optimized for 8 beat bursts, the 4 beat chop-burst mode is somewhat akin to reading 8 beats but discarding or chopping the last 4. From a performance standpoint filling a 64 byte cache in this mode is not as efficient as reading from only one DIMM using 8 beat bursts.

POWER8 Memory

Figure 25: POWER8 Memory Subsystem



Rather than using the industry standard DIMMs illustrated, as of this writing all POWER8 processor-based systems offered by IBM use custom DIMM modules specified and manufactured for IBM and featuring an IBM designed and manufactured memory buffer module incorporated on the DIMM.

The memory buffer includes an L4 cache which extends the cache hierarchy one more level for added performance in the memory subsystem.

As for the main memory, on a single rank DIMM, the memory buffer accesses 4 ports of memory on a DIMM. If x8 DRAM modules are used, each port has 9 DRAM modules that are used for data and error checking. Significantly an additional DRAM module is included that can be used as a spare if one of the other 9 has a Chipkill event. Each port, or channel, therefore, is largely equivalent to a memory DIMM.

The DRAMs are still accessed using 8 beat bursts with each beat accessing 128 bits of data plus additional checksum information. The error checking algorithm used is sufficient to detect and correct an

entire DRAM module being bad in that group of 18. Afterwards the error correction scheme is at least capable of correcting another bad bit within each impacted ECC word.

Because of the sparing capability that is also added to these custom DIMMs, each such x8 DIMM can tolerate a single DRAM module being bad in each port – a total of 4 such events per DIMM. Because these are true spares, the DIMM does not have to be repaired or replaced after such events given that after sparing -- the Chipkill ECC protection is still available to repair an additional Chipkill event.

The buffer chip communicates with a processor memory controller using an IBM designed memory bus. Each memory controller has four such busses and a processor DCM has two memory controllers support a maximum of 8 custom DIMMs.

It should be noted that for additional capacity a DIMM may have multiple ranks where a rank of memory duplicates the DRAM structure of a single rank DIMM. This section discusses the RAS of DIMMS built with x8 DRAM modules since, as previously discussed, the error checking complexity is more difficult when using x8 DRAM modules. However, some x4 custom DIMMs are also supported in POWER8.

Comparing Approaches

Figure 26: Memory Needed to Fill a Cache Line, Chipkill Comparison

Characteristics	POWER8	Alternate Processor Design
Processor Cache Line Size	128 Bytes	64 Bytes
Number of Bits of Data Per Channel	64 Bits	64 Bits
Number of Channels Per Physical DIMM Per Rank	4	1

Characteristics	POWER8 Capability with x8 DIMMS	POWER8 Capabilities with x4 DIMMS	Alternate Design x8 single channel mode	Alternate Design x4 single channel mode	Alternate Design x4 dual channel mode+
Number of Channels Used	2	2	1	1	2
Number of Physical DIMMs Used	1	1	1	1	2
DRAMS used for data/ECC etc. before channel spares	18	37^	9	18	36
Additional Spare DRAMs Available	2*	2*	0	0	0
Chipkill Events Tolerated Without Using Sparing	1	2	0	1	2
Maximum Number of Chipkill Events That could be tolerated if all spares used* (As % of Total DRAM)	3 (15%)	4 (~10%)	0 (0%)	1 (5.56%)	2 (5.56%)
Total Number (Best case) Of Chipkill events tolerated without requiring service	2	3	0	?	?

* 1 spare per channel, can only be used in that channel

+ **May have performance implications compared to Single Channel Mode**

^Counting An Additional "ECC" DRAM available for use across the two channels

Figure 26 summarizes the capabilities of POWER8 single rank DIMM design capabilities compared to potential alternative designs that naturally fill a 64 byte cache.

Not shown in the diagram is use of dual channels (and chop-burst mode) for x8 DIMMs in the alternate design. This may be a potential configuration but at present does not seem to be commonly offered.

The table shows that even with x8 DIMMs, the capability offered by POWER8 processor-based systems with this custom DIMMs is better than the alternative in terms of total numbers of Chipkill events that can be survived, as well as the total number of DRAMs each spare DRAM module has to cover for.

In addition, because the alternate design does not make use of true spares, it may be unclear whether this "double Chipkill protection" actually provides much more in terms of RAS compared to the alternate

design single Chipkill. The question is whether the DIMM can really be allowed to operate indefinitely, without need to be replaced when a single Chipkill event is encountered in a pair of DIMMs.

This is not always as easy to determine as it might seem. The decision on when to replace a DIMM for failures could be the choice of the hypervisor or operating system used. The system vendor may not have much control over it. And even in cases where the vendor does exert some control over it, there may be good reasons to replace the DIMM after the first Chipkill if true spares are not used. Depending on the exact algorithm used, the overall error detection capabilities available can change when a DRAM is marked out -- there may be less overall ability to detect or properly correct multiple bit failures. Further, the error correction in the presence of a marked out DRAM may be slower than would occur without the bad DRAM and the presence of a Chipkill might even impact performance and capability of memory scrubbing.

Again, making use of true spares, as in done in POWER8, avoids any of these potential issues since, after a sparing is used, the state of the ECC word is the same as before the sparing event took place.

And again, with the Power System approach, the customer doesn't need to make a choice between buying DIMMs with Chipkill protection or not, or choose between performance and having spare DRAM modules.

Additional Memory Protection

It has been previously mentioned that the memory buffer and memory controller implement retry capabilities to avoid the impact of soft errors.

The data lines on the memory bus between processor and memory also have CRC detection plus the ability to retry and continuously recalibrate the bus to avoid both random soft errors and soft errors that might occur due to drifting of bus operating parameters. The bus also has at least one spare data lane that can be dynamically substituted for a failed data lane on the bus.

The memory design also takes advantage of a hardware based memory scrubbing that allows the service processor and other system firmware to make a clear distinction between random-soft errors and solid uncorrectable errors. Random soft errors are corrected with scrubbing, without using any spare capacity or having to make predictive parts callouts.

The POWER Hypervisor and AIX have also long supported the ability to deallocate a page of memory for cases where a single cell in a DRAM has a hard fault. This feature was first incorporated in previous generations of systems where the sparing and error correction capabilities were less than what is available in POWER7 and POWER8.

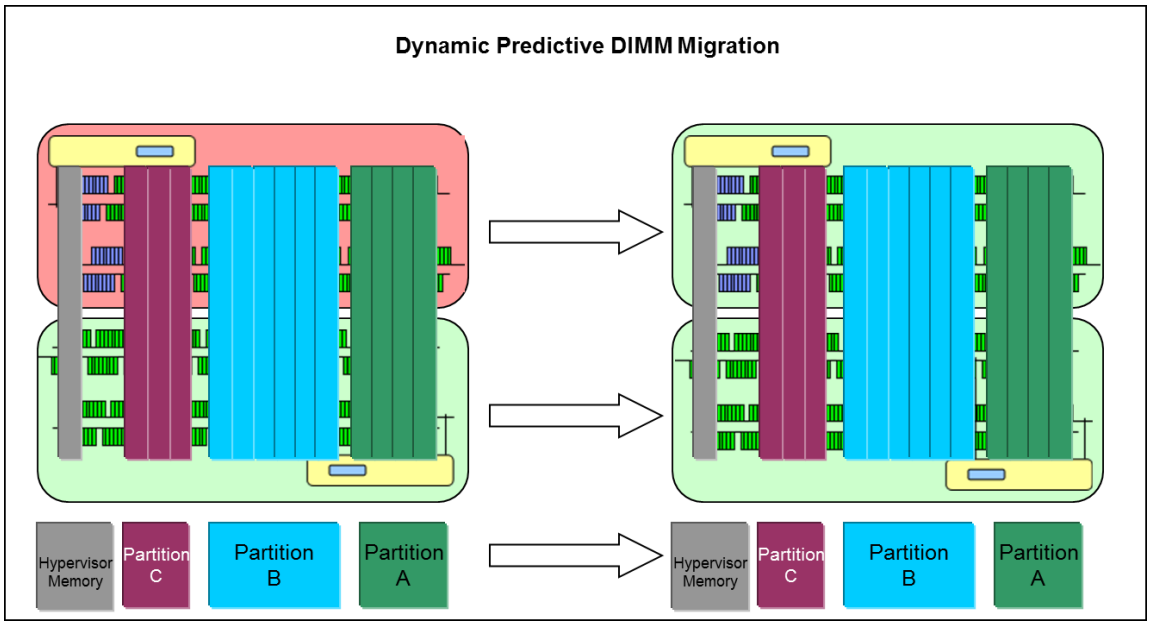
Dynamic Memory Migration and Active Memory Mirroring of the Hypervisor

To handle rare cases when memory does fail in a way that exhausts the extensive sparing or error correction capabilities in the DIMM, the POWER Hypervisor and the hardware can co-operate to mitigate the impact of the failures.

The mitigations are performed in the context of a highly virtualized system with the idea that memory anywhere within the system can be used to replace failed memory.

Several of these features are shown in Figure 27.

Figure 27: Dynamic Memory Migration



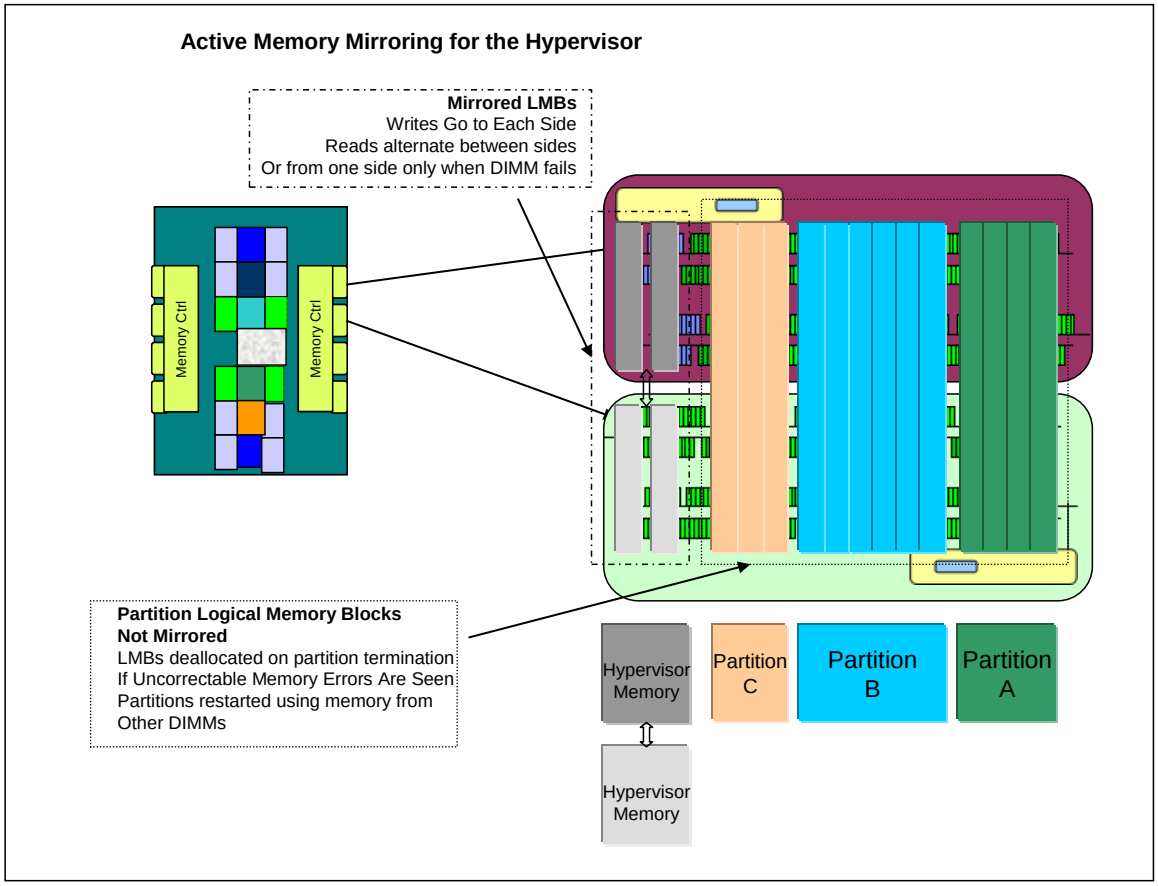
The figure illustrates that memory is assigned to the hypervisor and partitions in segments known as logical memory blocks. Each logical memory block (LMB) may contain memory from multiple DIMMS. Each DIMM may also contribute to multiple partitions.

Mechanisms are employed in systems to optimize memory allocation based on partition size and topology. When a DIMM experiences errors that cannot be permanently corrected using sparing capability, the DIMM is called out for replacement. If the ECC is capable of continuing to correct the errors, the call out is known as a predictive callout indicating the possibility of a future failure.

In such cases, if an E870 or E880 has unlicensed or unassigned DIMMS with sufficient capacity to handle it, logical memory blocks using memory from a predictively failing DIMM will be dynamically migrated to the spare/unused capacity. When this is successful this allows the system to continue to operate until the failing DIMM is replaced, without concern as to whether the failing DIMM might cause any future uncorrectable error.

This migration can occur regardless of where the unallocated memory resources are. There is no requirement that the resources be under the same processor or even the same node as the failing DIMM. In systems with not-yet-licensed Capacity Upgrade on Demand memory, the memory will be temporarily enabled for this purpose.

Figure 28: Active Memory Mirroring for the Hypervisor



In the previous example, as illustrated, if a DIMM were to fail in such a way that uncorrectable errors were generated, then the impact to the system would depend on how severe the impact was.

If a single uncorrectable error occurred in memory used by a partition, then special uncorrectable error handling would allow the OS to terminate whatever code was using the data is referenced for use. That might mean no termination if the data is never referenced, or terminating a single application. It might also mean termination of the partition if the data were critical to the OS kernel.

Most severely, without mirroring, if an uncorrectable error impacted a critical area of the hypervisor, then the hypervisor would terminate, causing a system-wide reboot.

However, the processor memory controller is also capable of mirroring segments of memory in one DIMM with segments of another. When memory is mirroring writes go to both copies of the memory, and reads alternate between DIMMs, unless an error occurs.

The memory segments used in mirroring are fine-grained enough that just the LMBs associated with hypervisor memory can be mirrored across all the DIMMs where they are allocated.

The example on Figure 28, illustrates mirroring hypervisor memory. This mirroring feature, standard in E870/E880 Systems, is used to prevent memory faults from causing hypervisor outages.

It should be noted that while mirroring the hypervisor can prevent even a catastrophic DIMM failure from causing a hypervisor outage, partitions using memory from such a DIMM can still terminate due to multiple DIMM errors. Whenever a partition terminates due to an uncorrectable fault in an LMB, the service processor and PowerVM hypervisor will ensure that the LMB is not reallocated to the failing partition or another partition on partition reboot. For catastrophic DIMM events they will proactively mark all LMBs on a DIMM as faulty and not be reassigned to partitions on partition reboots.

Alternate Approach

Alternate designs may take a less virtualized approach to handling some of the conditions previously described, though taking advantage of these features would depend on the software stack in use, and in particular on the hypervisor.

For example:

1. Rather than having spare DRAMs or system-wide Capacity Update on demand capabilities, they may designate spare rank or ranks of memory. The spare capacity so designated, however, might need to be used only to spare DIMMS underneath the same memory controller. This capability might only be used for adding memory capacity. Or depending on design, hypervisor and OS, it might be used to migrate from failing or predictively failing memory. In such a design, however, to be sure that a failed rank can be spared out anywhere within the system may require that a spare rank be designated for every memory controller in the system.
2. Systems may support mirroring of all the memory in the system. This option would prevent any uncorrectable error in memory from taking down any code. However, it would require doubling the amount of memory required for the system, and potentially reducing the performance or capacity of the system. If the memory subsystem is robust enough, customers may rarely take advantage of full system mirroring.
3. More fine-grained mirroring may also be supported, for example, mirroring all the memory under a single processor module, in systems with multiple modules. Such a design would necessitate limiting critical functions to just the portion of the system that is mirrored. Mirroring hypervisor memory might mean requiring that all hypervisor code be executed in just a portion of one processor.

   Taking advantage of such an approach would be OS/hypervisor dependent and may pose performance challenges compared to the POWER approach that mirrors the critical memory wherever it is allocated.

Final Note:

Many of these described memory options in alternate designs may only be available in certain configurations – say running128 bit ECC across DIMMs only and therefore not in “performance” mode. Some functions may also be mutually exclusive – for example full system mirroring might be incompatible with DIMM sparing or DIMM “hot-add”.

Section 4: Server RAS Design Details

Server Design: Scale-out and Enterprise Systems

IBM Power System S812L, S822, S822L, S814, S824 and S824L systems are 1 and 2 socket systems are described as systems intended for a scale-out application environment. Other, larger, Power Systems are referred to as Enterprise class.

To the extent that scale-out and enterprise systems are created with the same basic building blocks such as processors, memory and I/O adapters it can be difficult to understand the differences between such systems.

Clearly an enterprise system supporting 4, 8, or even more sockets is capable of greater performance within a single partition compared to a one or two socket server. Such systems may run at a greater processor frequency than the one or two socket, and support significantly more memory and I/O, all of which leads to greater performance and capacity.

From a hardware RAS standpoint; however, there are several elements that may separate a scale-out server from an enterprise server, a non-exhaustive list of such characteristics could include:

Node Structure to Avoid High Impact Outages

In a scale-out system, such as a 1 socket server, it is not difficult to understand how certain hardware faults such as certain catastrophic failures of a processor module can lead to a system that cannot be used to run work-load until the failed element is repaired. IBM typically refers to any such event as a High Impact Outage because of the extra downtime that may occur while replacement parts are procured for a system and the outage time then associated with the repair.

While the effectiveness of processor reliability and availability design can make these events rare, it is expected that practically no fault of a single component like a processor should take an enterprise system down and keep it down until repair.

Enterprise systems, especially those with more than 4 sockets use a combination of logical and physical hardware design to isolate these elements so that at least a portion of a system can run for even the most severe such failure instances.

POWER7 processor-based IBM POWER 795 systems, for example used 4 socket processor books as a basic system building-block, with the processor books integrated into a system by connecting to a common mid-plane. If necessary any given processor book within the system could be deconfigured after a failure to allow workload to run on the remaining resources.

Infrastructure redundancy

Many scale-out systems offer some redundancy for infrastructure elements such as power supplies and fans.

But this redundancy does not include other functions such as processor clocks, service interfaces, or module voltage regulation, temperature monitoring and so forth.

As a minimum, enterprise servers, to avoid high impact outages of the kind mentioned earlier, needs to have redundancy in global elements that are used across nodes.

Better enterprise design extends infrastructure redundancy within each node such as carrying out clock redundancy to each processor.

I/O redundancy

Enterprise system design requires that I/O redundancy be possible across the entire I/O path from each I/O adapter to the memory used to transfer data to or from the I/O device.

Most scale-out systems are capable of handling I/O redundancy for a limited number of adapters, but do not have the capacity to carry out I/O redundancy for a large number of adapters and

across node boundaries which is necessary to ensure that no I/O adapter fault will cause a High Impact Outage.

More Sparing Options

Enterprise systems may incorporate more spare capacity to avoid the need to take planned outages to replace parts. Spare capacity for future expansion, such as Capacity Update on Demand may be offered and used to provide additional options for continued operation in the face of processor or memory failures.

More extensive use of Highly Reliable Parts, Testing and Burn-in

Enterprise systems may incorporate even more reliable parts and perform component and system level extended burn-in and test to reduce early life failures.

This section focuses on these server design aspects, highlighting what is particular to Power Systems, again comparing Power System designs to alternative server design approaches.

Server Design: Infrastructure Redundancy and Outages

Introduction

In theory, providing redundant components in a system may be thought of as a highly effective way to avoid outages due to failures of a single component. The care with which redundancy is designed and implemented plays an important factor on how effective redundancy really is in eliminating failures in a subsystem.

There are a number of considerations.

Serial Failures, Load Capacity and Wear-out

Nearly any system running enterprise applications will supply redundant power supplies. This power supply redundancy is typically meant to assure that if one power supply fails, the other can take up the power load and continue to run.

Likewise most systems are designed to tolerate a failure in fans needed to cool the system. More sophisticated systems may increase fan speed to compensate for a failing fan, but a single fan failure itself should not cause a significant cooling issue.

In such a design, it would be expected that a system would never experience an outage due to a single power supply or fan fault.

However, if a power supply fails in a redundant design and the second power supply should happen to fail before it is repaired, then the system will obviously be down until one or the other of the supplies is fixed.

The expectation is that this would be a rare event.

If the system were incapable of determining that one of a pair of redundant parts had failed, then this can be more common, however. The ability to constantly monitor the health of the secondary component is therefore essential in a redundant design, but not always easy.

For example, two power supplies may share the load in a system by supplying power to components. When no power is supplied to a component, that condition is fairly easy to detect. If one power supply were able to supply some current, but an insufficient amount to carry the load by itself, that the supply is “weak” and depending on the design may not be detected until the good supply fails.

In a lightly loaded system, it may not even be possible to distinguish between a “weak” supply and one that is providing no current at all. In some redundant designs for light loads, only one supply may even be configured to carry the load.

Even when both supplies are operating optimally, if a power supply is not well tested designed and specified to run a system indefinitely on a single power source, then what may happen is that when the first power supply fails, the second carries a load that stresses it to the point where it soon also fails.

This kind of failure mode can be exasperated perhaps by environmental conditions: Say the cooling in the system is not very well designed so that a power supply runs hotter than it should. If this can cause a failure of the first supply over time, then the back-up supply might not be able to last much longer under the best of circumstances, and when taking over a load would soon be expected to fail.

As another example, fans can also be impacted if they are placed in systems well provided for cooling of the electronic components, but where the fans themselves receive excessively heated air that is a detriment to the fans long-term reliability.

These kinds of long term degradation impacts to power supplies or fans might not be apparent when systems are relatively new, but may occur over time.

IBM's experience with Power Systems is that they tend to be deployed in data centers for longer than may be typical of other servers because of their overall reliability and performance characteristics. IBM specifies that power supplies as well as fans be designed to run in systems over long periods of time without an increase in expected failure rate. This is true even though it is expected that because these components can be concurrently replaced, and that they will normally be replaced fairly soon after a failure has been reported.

What is specified for infrastructure component useful in Power Systems is typically longer than parts used in common alternate system designs and something that IBM continues to evaluate and improve as customer usage patterns dictate.

In addition, understanding that heat is one of the primary contributors to components "wearing out", IBM requires that even components providing cooling to other components should be protected from excessive heat by their placement in the system.

Common Mode Failures

Still even with well-designed redundancy and elimination of serial failures, some faults can occur within a sub-system where redundancy is insufficient to protect against outages.

An easily understood example is when two power supplies are given the same AC power source. If that source fails, then the system will go down despite the redundancy. But failures include events besides simple power loss. They can include issues with surges due to electrical storm activity, short dips in power due to brown-out conditions, or perhaps when converting to backup power generation.

These incidents will be seen by both supplies in a redundant configuration and all the power supplies need to be able to withstand transient faults.

As another example, suppose that two supplies end up providing voltage to a common component, such as a processor module. If any power input to the module were to be shorted to ground, it would be expected that both supplies would see this fault. Both would have to shut down to prevent an over-current condition.

In an installed and running system, one would rarely expect to see wires themselves suddenly shorting to ground. However, if a component such as a cable or card were allowed to be removed or replaced in a system, without a good design to protect against it, even an experienced person doing that service activity could cause a short condition.

Also, the failure of certain components may essentially manifest as a short from power to ground. Something as simple as a capacitor used for electrical noise decoupling could fail in that fashion.

Proper system design would mitigate the impact of these events by having soft-switches, or effectively circuit breakers isolating key components, especially those that can be hot-plugged.

But ultimately there is someplace electrically where redundant components do have to come together to provide a function, and failures there can cause outages.

Server Design: Power and Cooling Redundancy Details

Voltage Regulation

There are many different designs that can be used for supplying power to components in a system.

One of the simplest conceptually is a power supply that takes alternating current (AC) from a data center power source, and then converts that to a direct current voltage level (DC).

Modern systems are designed using multiple components, not all of which use the same voltage level. Possibly a power supply either can provide multiple different DC voltage levels to supply all the components in a system. Otherwise it may supply a voltage level (e.g. 12v) to voltage regulators which then convert to the proper voltage levels needed for each system component (e.g. 1.6 V, 3.3 V, etc.) Use of such voltage regulators can also ensure that voltage levels are maintained within tight specifications required for the modules they supply.

Typically a voltage regulator module (VRM) has some common logic plus a component or set of components (called converters, channels or phases). At a minimum a VRM provides one converter (or phase) that provides the main function of stepped-down voltage, along with some control logic. Depending on the output load required, however, multiple phases may be used in tandem to provide that voltage level.

If the number of phases provided is just enough to drive what the phase is driving, the failure of a single phase can lead to an outage. This can be true even when the 12V power supplies are redundant. Therefore additional phases may be supplied to prevent the failure due to a single phase fault. An addition additional phases may also be provided for sparing purposes.

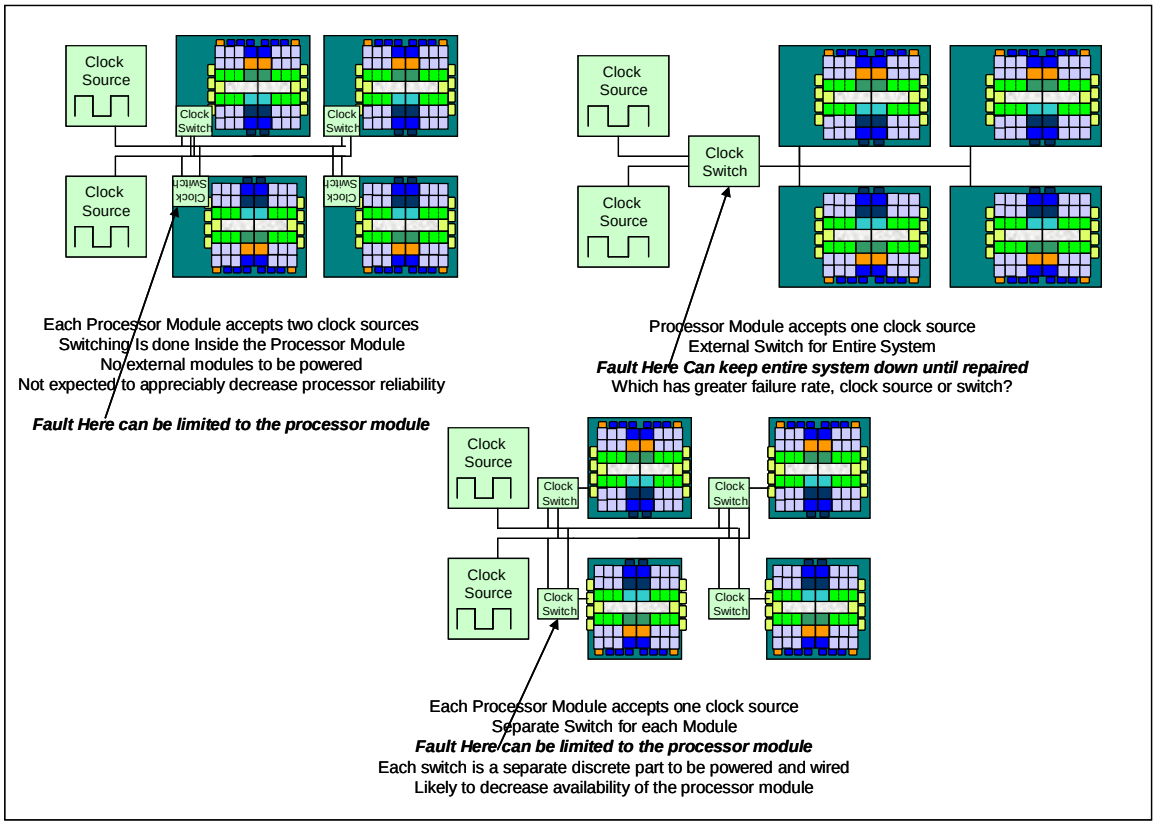
While such redudancy and sparing may be provided at at the CEC-level, for example by separate voltage regulator cards, some devices, such as memory DIMM, may take a voltage level and use some sort of voltage converter to further divide the voltage for puposes such as providing a reference voltage or signal termination. Such on-component voltage division/regulation is typically not as demanding an activitiy as previously discussed and is not included in the discussion of voltage regulation.

While it is important to provide redundancy to critical components to maximize availabiity, other components may also have individual voltage regulators when the component itself is redundant.

Redundant Clocks

Vendors looking to design enterprise class servers may recognize the desirability of maintaining redundant processor clocks so that the failure of a single clock oscillator doesn't cause a system to go down and stay down until repaired. However, a system vendor cannot really feed redundant clocks into a processor unless the processor itself is designed to accept redundant clock sources and fail over when a fault is detected.

Figure 29: Redundant Clock Options



Therefore what is often done is redundant clock oscillators are supplied, but then external components are used to feed a single clock to each processor module. If the external component feeding the processor module fails, then the processor will cause an outage – regardless of the fact that the "clock source" was redundant. This may or may not cause more outages than having a non-redundant clock source, depending on how many of these external switching modules are used, and if they can in fact switch dynamically (instead of after a system reboot). The biggest advantage in such a design is that when multiple modules are used, at least the number of processors taken off line (after a reboot) can be minimized.

In contrast the POWER8 SCM processor module accepts multiple clock inputs. Logic inside the processor module can switch from one clock source to another dynamically (without taking the system down).

It is still possible that the switching logic inside the processors could fail, the common mode failure point, but clock redundancy is carried as far as possible in Enterprise Power System designs by providing the switching logic within the processor module.

Interconnect Redundancy

A scalable SMP system 2s or greater will typically have multiple busses that provide communications between processors, from processors to memory, and typically from I/O controllers out to I/O hubs, I/O drawers or a collection of I/O slots.

Some designs offer some sort of redundancy for these busses. Typically, however, there are not entirely redundant paths between elements, so taking advantage of redundancy usually means the loss of performance or capacity when such a bus is used. This is sometimes known as lane reduction.

If this level of redundancy can be used dynamically, then it has use in keeping applications running (perhaps with reduced performance). If the lane reduction only happens on a system reboot, it is questionable whether it would be preferable to simply not use the processor, memory or other resource rather than re-IPL in a reduced capacity.

For busses such as memory and processor-to-processor, Power Systems are typically designed to take full advantage of the busses put in to service. Therefore, POWER8 processor-based Systems utilize the concept of protecting such busses against soft errors, and then having spare capacity to permanently repair the situation when a single data bit on a bus breaks, maintaining full capacity.

I/O Redundancy

In a PCIe Gen3 I/O environment, not all I/O adapters require full use of the bandwidth of a bus; therefore lane reduction can be used to handle certain faults. For example, in an x16 environment, loss of a single lane, depending on location, could cause a bus to revert to a x8, x4, x2 or x1 lane configuration in some cases. This, however, can impact performance.

Not all faults that impact the PCIE I/O subsystem can be handled just by lane reduction. It is important when looking at I/O to not just consider all faults that cause loss of the I/O adapter function.

Power Systems are designed with the intention that traditional I/O adapters will be used in a redundant fashion. In the case, as discussed earlier where two systems are clustered together, the LAN adapters used to communicate between processors would all be redundant and the SAN adapters would also be redundant.

In a single 1s or 2s system, that redundancy would be achieved by having one of each type of adapter physically plugged into a socket controlled by one PCIe adapter, and the other in a slot controlled by another.

The software communicating with the SAN would take care of situation that one logical SAN device might be addressed by one of two different I/O adapters.

For LAN communicating heart-beat messages, both LAN adapters might be used, but messages coming from either one would be acceptable, and so forth.

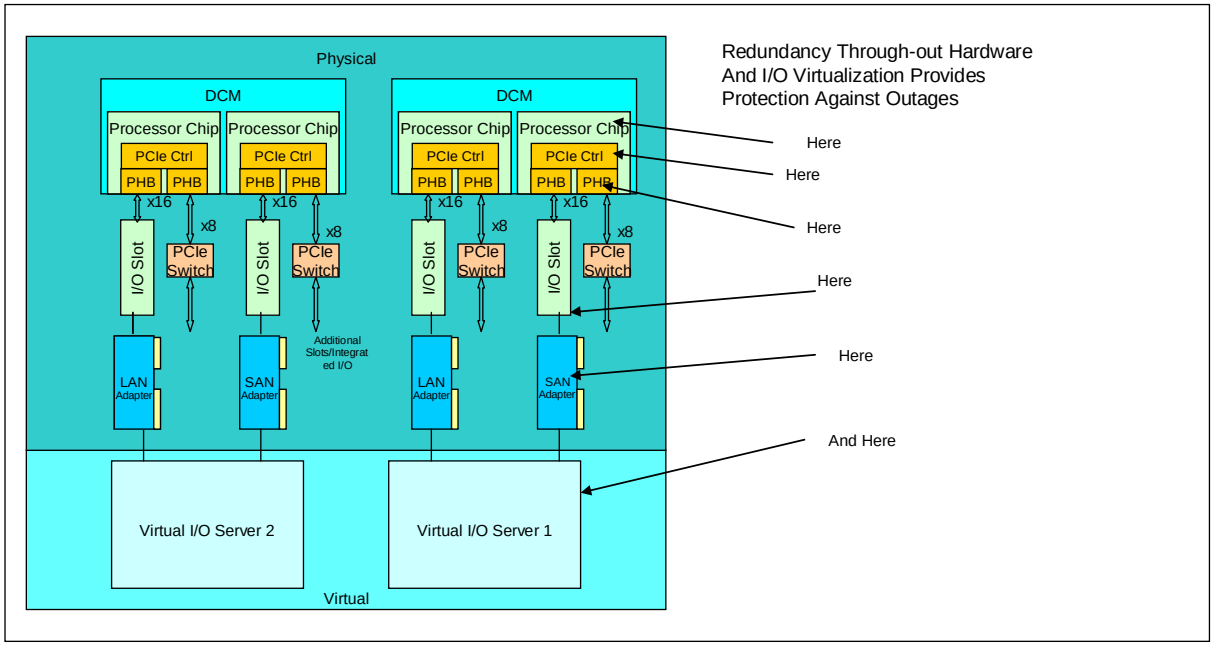
This configuration method would best ensure that when there is a fault impacting an adapter, the redundant adapter can take over. If there is a fault impacting the communication to a slot from a processor, the other processor would be used to communicate to the other I/O adapter.

The error handling throughout the I/O subsystem from processor PCIe controller to I/O adapter would ensure that when a fault occurs anywhere on the I/O path, the fault can be contained to the partition(s) using that I/O path.

Furthermore, PowerVM supports the concept of I/O virtualization with VIOS™ so that I/O adapters are owned by I/O serving partitions. A user partition can access redundant I/O servers so that if one fails because of an I/O subsystem issue, or even a software problem impacting the server partition, the user partition with redundancy capabilities as described should continue to operate.

This End-to-End approach to I/O redundancy is a key contributor to keeping applications operating in the face of practically any I/O adapter problem.

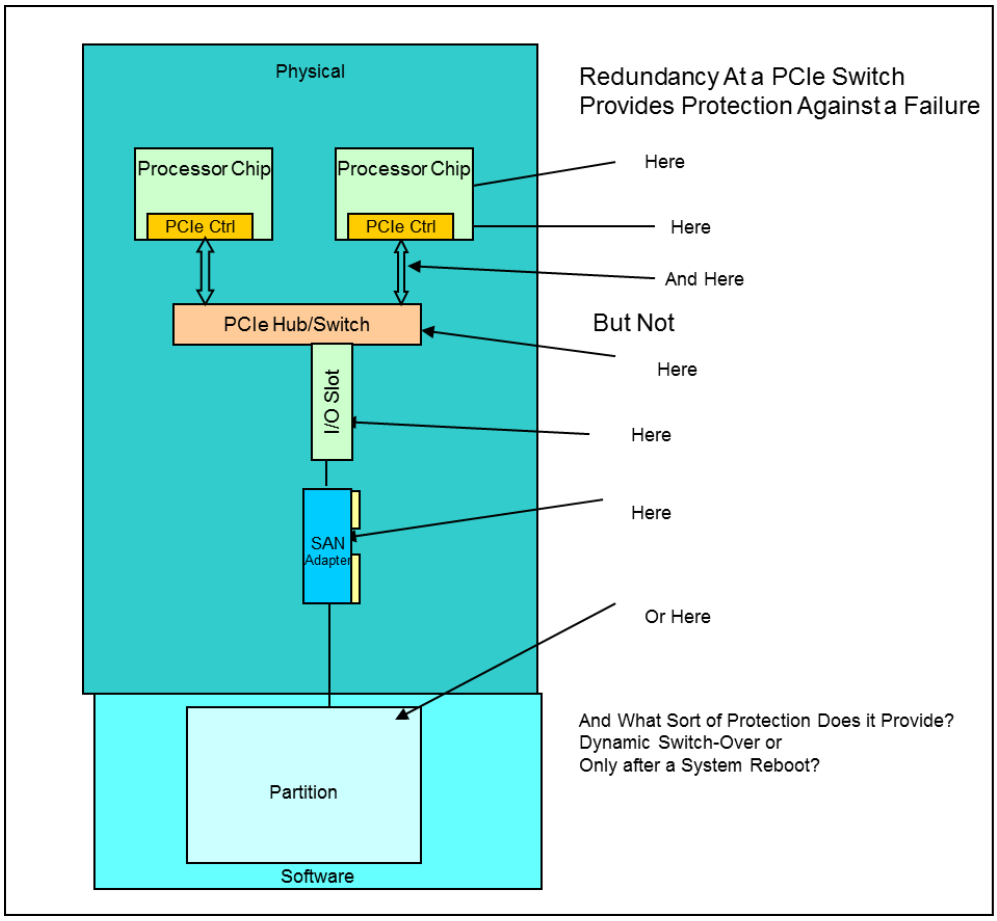
Figure 30: End-to-End I/O Redundancy



Alternate PCIe Hub/Switch Redundancy

In contrast an alternate design for a server might offer up an arrangement where several slots can be controlled by either of 2 processors using some I/O hub logic in-between so that if one processor module were to fail, after a system reboot, the same set of slots could be accessed by the other processor.

Figure 31: PCIe Switch Redundancy Only



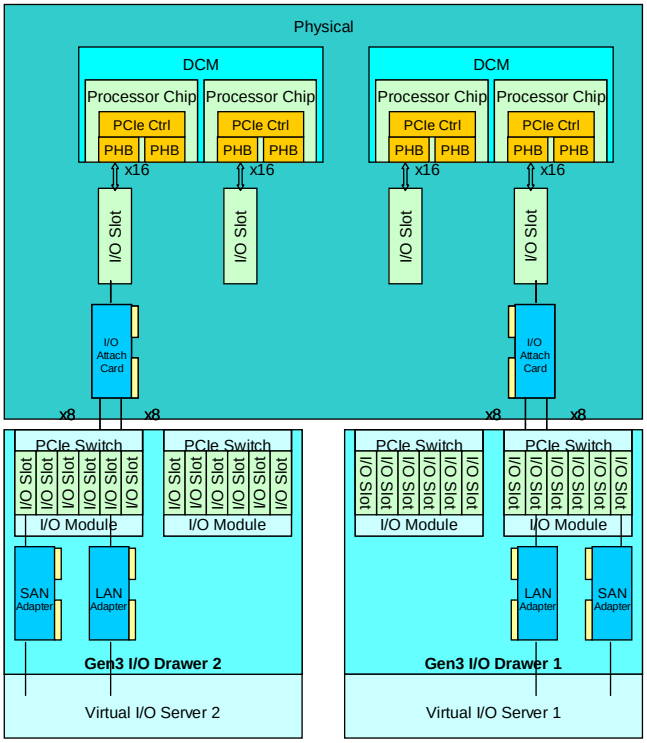
This design may be useful for processor faults, but does very little to protect against faults in the rest of I/O subsystem, and introduces an I/O hub control logic element in the middle which can also fail and cause outages.

PCIe Gen3 Expansion Drawer Redundancy

As elsewhere described the optically connected PCIe Gen3 I/O expansion drawer provides significant RAS features including redundant fans/power supplies and independently operating I/O modules. Certain components such as the mid-plane, will require that the drawer be powered off during the repair and could potentially impact operation of both I/O modules.

Therefore it is recommended for the greatest redundancy that redundant adapter pairs be connected to separate I/O drawers and these separate I/O drawers be connected to different processor modules where possible.

Figure 32: Maximum Availability with Attached I/O Drawers



I/O Drawer Built With Considerable Ability
To Isolate Failures Across I/O modules
And With Significant Infrastructure Redundancy

Availability still Maximized Using Redundant I/O Drawers

Server Design: Planned Outages

Unplanned outages of systems and applications are typically very disruptive to applications. This is certainly true of systems running applications standalone, but is also true, perhaps to a somewhat lesser extent, of systems deployed in a scaled-out environment where the availability of an application does not entirely depend on the availability of any one server. The impact of unplanned outages on applications in both such environments is discussed in detail in the next section.

Planned outages, where the end-user picks the time and place where applications must be taken off-line can also be disruptive. Planned outages can be of a software nature – for patching or upgrading of applications, operating systems or other software layers. They can also be for hardware, for reconfiguring systems, upgrading or adding capacity, and for repair of elements that have failed but have not caused an outage because of the failure.

If all hardware failures required planned downtime, then the downtime associated with planned outages in an otherwise well designed system would far-outpace outages due to unplanned causes.

While repair of some components cannot be accomplished with workload actively running in a system, design capabilities to avoid other planned outages are characteristic of systems with advanced RAS capabilities. These may include:

Updating Software Layers

Maintaining updated code levels up and down the software stack may avoid risks of unplanned outages due to code bugs. However, updating code can require planned outages of applications, partitions or entire systems.

POWER Systems are designed to generally allow a given level of “firmware” to be updated in the code used by service processors, the PowerVM hypervisor and other areas of system hardware, without needing an outage.

Depending on the nature of the fault, it might still be the case that some fixes would require outages. One of the improvements made in Power Systems beginning with the POWER7+ processor was the ability to reinitialize processors individually during system operation to allow for firmware fixes that required reinitializing processors.

Migrating from one firmware level to another, where a level provides new function, is not supported dynamically.

Dynamic updating of hypervisors other than the PowerVM hypervisor and of operating systems and applications depend on the capabilities of each such software layer.

Concurrent Repair

When redundancy is incorporated into a design, it is often possible to replace a component in a system without taking any sort of outage.

As examples, Enterprise Power Systems have the ability to concurrently remove and replace such redundant elements as power supplies and fans.

In addition Enterprise Power Systems as well as POWER8 processor-based 1s and 2s systems have the ability to concurrently remove and replace I/O adapters.

Integrated Sparing

To reduce replacements for components that cannot be removed and replaced without taking down a system, Power Systems strategy includes the use of integrated spare components that can be substituted for a failing ones. For example in POWER8, each memory DIMM has spare DRAM modules that are automatically substituted for failed DRAM modules; allowing a system to not only survive the failure of at least one DRAM in a DIMM, but also not ever have to take a system down to repair it.

Such integrated sparing is also used in more subtle ways such as by having extra “columns” of data in an L3 cache, or by providing more than redundant voltage regulator phases for CEC-level voltage regulators.

For items that cannot be concurrently repaired, or spared, focusing on the reliability of the part itself is also a key component in providing for single system availability.

These capabilities are often distinctive to IBM processor designs.

Server Design: Clustering Support

PowerHA SystemMirror

IBM Power Systems running under PowerVM and AIX™ and Linux support a spectrum of clustering solutions. These solutions are designed to meet requirements not only for application availability as regards to server outages, but also data center disaster management, reliable data backups and so forth. These offerings include distributed applications such as with db2 pureScale™, HA solutions using clustering technology with PowerHA™ SystemMirror™ and disaster management across geographies with PowerHA SystemMirror Enterprise Edition™.

It is beyond the scope of this paper to discuss the details of each of the IBM offerings or other clustering software, especially considering the availability of other material.

Live Partition Mobility

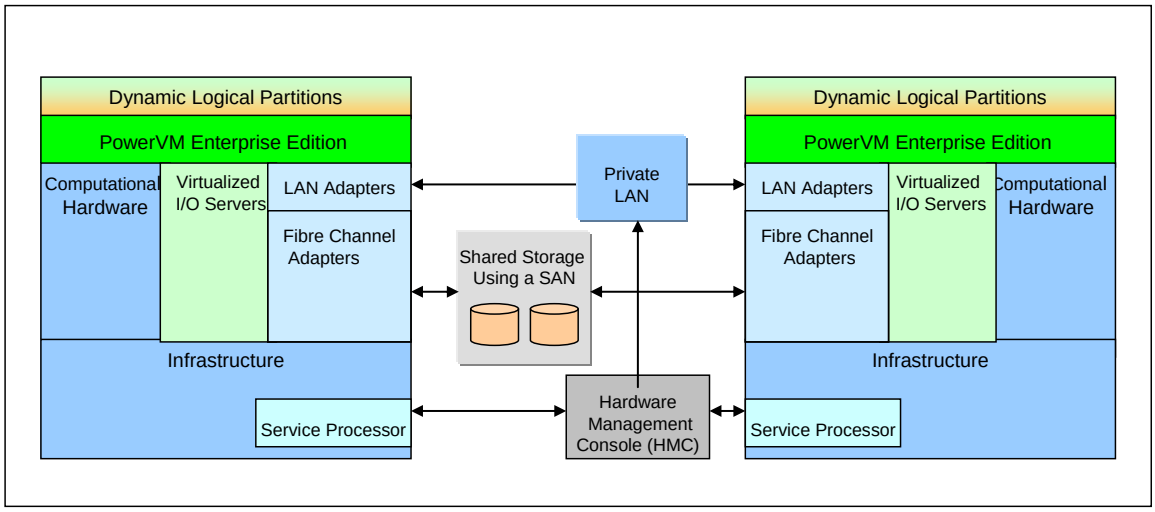
However, Live Partition Mobility (LPM), available for Power Systems running PowerVM Enterprise Edition, will be discussed here in particular with reference to its use in managing planned hardware outages.

Additional LPM details are available in an IBM Redbook titled: **IBM PowerVM Virtualization Introduction and Configuration**[8]

LPM Primer

LPM is a technique that allows a partition running on one server to be migrated dynamically to another server.

Figure 33: LPM Minimum Configuration



In simplified terms, LPM typically works in an environment where all of the I/O from one partition is virtualized through PowerVM and VIOS and all partition data is stored in a Storage Area Network (SAN) accessed by both servers.

To migrate a partition from one server to another, a partition is identified on the new server and configured to have the same virtual resources as the primary server including access to the same logical volumes as the primary using the SAN.

When an LPM migration is initiated on a server for a partition, PowerVM begins the process of dynamically copying the state of the partition on the first server to the server that is the destination of the migration.

[8] Mel Cordero, Lúcio Correia, Hai Lin, Vamshikrishna Thatikonda, Rodrigo Xavier, Sixth Edition published June 2013, *www.**redbooks**.ibm.com/**redbooks**/pdfs/sg247460.pdf*

Thinking in terms of using LPM for hardware repairs, if all of the workloads on a server are migrated by LPM to other servers, then after all have been migrated, the first server could be turned off to repair components.

LPM can also be used for doing firmware upgrades or adding additional hardware to a server when the hardware cannot be added concurrently in addition to software maintenance within individual partitions.

When LPM is used, while there may be a short time when applications are not processing new workload, the applications do not fail or crash and do not need to be restarted. Roughly speaking then, LPM, allows for planned outages to occur on a server without suffering downtime that would otherwise be required.

Minimum Configuration

For LPM to work, it is necessary that the system containing a partition to be migrated and the system being migrated to both have a local LAN connection using a virtualized LAN adapter. It is recommended that LAN adapter used for migration should be a high speed connection (10G.) The LAN used should be a local network and should be private and have only two uses.

LPM requires that all systems in the LPM cluster be attached to the same SAN (when using SAN for required common storage) which typically requires use of fibre channel adapters.

If a single HMC is used to manage both systems in the cluster, connectivity to the HMC also needs to be provided by an Ethernet connection to each service processor.

The LAN and SAN adapters used by the partition must be assigned to a Virtual I/O server and the partitions access to these would be by virtual lan (vlan) and virtual scsi (vscsi) connections within each partition to the VIOS.

Each partition to be migrated must only use virtualized I/O through a VIOS; there can be no non-virtualized adapters assigned to such partitions at time of migration.

I/O Redundancy Configurations

LPM connectivity in the minimum configuration discussion is vulnerable to a number of different hardware and firmware faults that would lead to the inability to migrate partitions. Multiple paths to networks and SANs is therefore recommended. To accomplish this, a VIOS server can be configured to use dual fibre channel and LAN adapters.

Externally to each system, redundant hardware management consoles (HMCs) can be utilized for greater availability. There can also be options to maintain redundancy in SANs and local network hardware.

Figure 34: I/O Infrastructure Redundancy

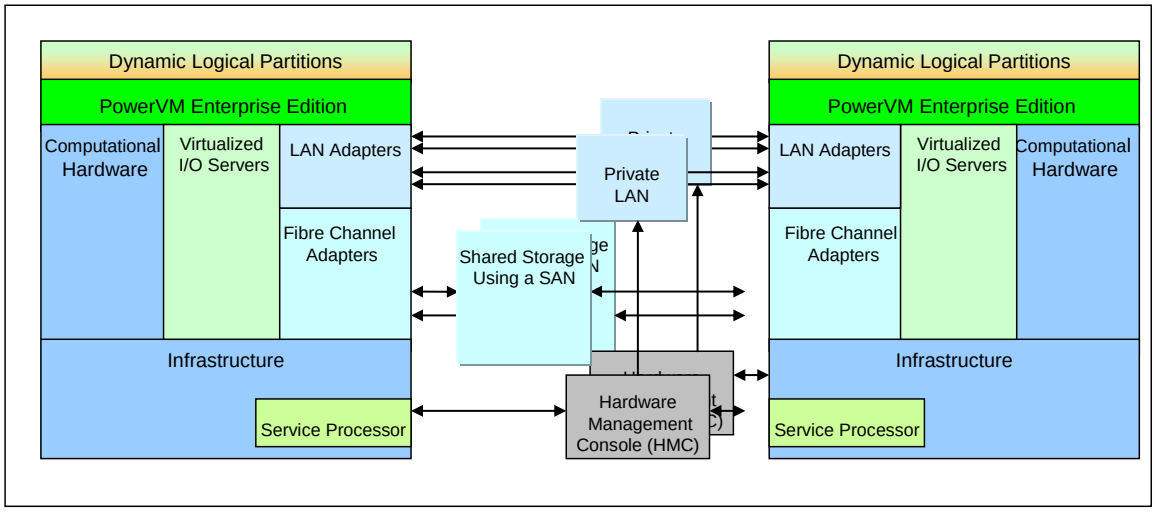
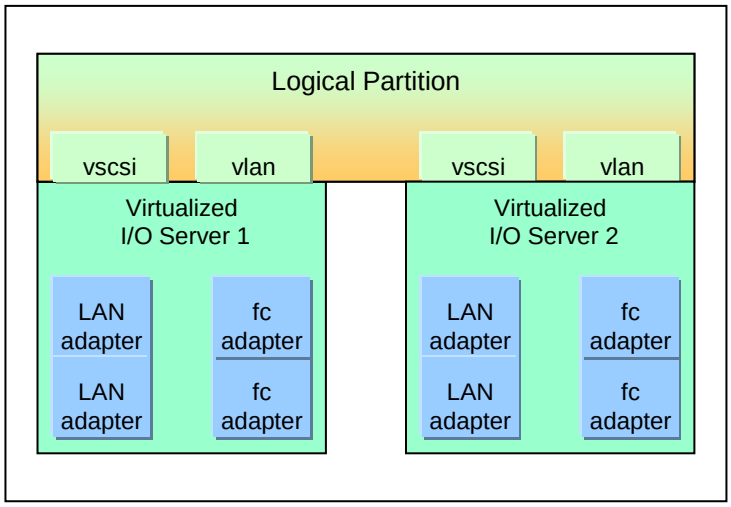


Figure 34 generally illustrates multi-path considerations within an environment optimized for LPM.

Within each server, this environment can be supported with a single VIOS. However, if a single VIOS is used and that VIOS terminates for any reason (hardware or software caused) then all the partitions using that VIOS will terminate.

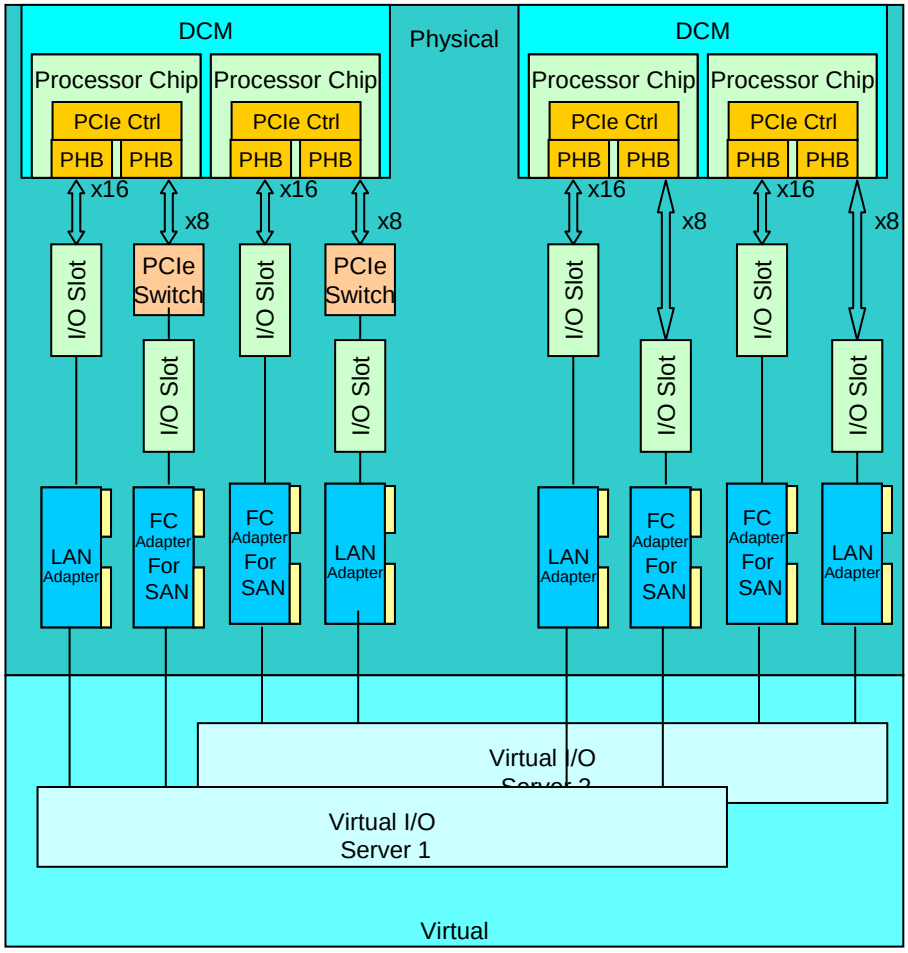
Using Redundant VIOS servers would mitigate that risk. There is a caution, however that LPM cannot migrate a partition from one system to another when a partition is defined to use a virtual adapter from a VIOS and that VIOS is not operating. Maintaining redundancy of adapters within each VIOS in addition to having redundant VIOS will avoid most faults that keep a VIOS from running. Where redundant VIOS are used, it should also be possible to remove the vscsi and vlan connections to a failed VIOS in a partition before migration to allow migration to proceed using the remaining active VIOS in a non-redundant configuration.

Figure 35: Use of Redundant VIOS



Since each VIOS can largely be considered as an AIX based partition, each VIOS also needs the ability to access a boot image, having paging space, and so forth under a root volume group or rootvg. The rootvg can be accessed through a SAN, the same as the data that partitions use. Alternatively, a VIOS can use storage locally attached to a server, either DASD devices or SSD drives. For best availability, however accessed, the rootvgs should use mirrored or RAID drives with redundant access to the devices.

Figure 36: I/O Subsystem of a POWER8 2-socket System



Currently IBM's POWER8 processor-based 1s and 2s servers have options for I/O adapters to be plugged into slots in the servers. Each processor module directly controls two 16 lane (x16) PCIe slots. Additional I/O capabilities are provided by x8 connections to a PCIe switch integrated in the processor board.

Figure 36 above illustrates how such a system could be configured to maximize redundancy in a VIOS environment, presuming the rootvgs for each VIOS are accessed from storage area networks.

When I/O expansion drawers are used, a similar concept for I/O redundancy can be used to maximize availability of I/O access using two I/O drawers, one connected to each processor socket in the system.

Section 5: Application Availability

Application Availability: Introduction

Trustworthiness

Users of a computer system intended to handle important information and transactions consider it essential that a computer system be trusted to generate accurate calculations, handle data consistently and universally without causing corruption, and not have faults that result in the permanent loss of stored data.

These essential elements are expected in any data system. They can be achieved through a combination of the use of reliable hardware, techniques in hardware and software layers to detect and correct corruption in data, and the use of redundancy or backup schemes, in hardware and software to ensure that the loss of a data storage device doesn't result in the loss of the data it was storing.

In the most important situations, disaster recovery is also employed to ensure that data can be recovered and processing can be continued in rare situations where multiple systems in a data center become unavailable or destroyed due to circumstances outside of the control of the system itself.

Another critical aspect related to the above is being able to trust that the computer system being used to handle important information has not been compromised – such as that no one denied physical access to the server is capable of acquiring unauthorized remote access.

Ensuring that a computer system can be trusted with the information and transactions given to it typically requires trusted applications, middle-ware, and operating systems, as well as hardware not vulnerable to remote attacks.

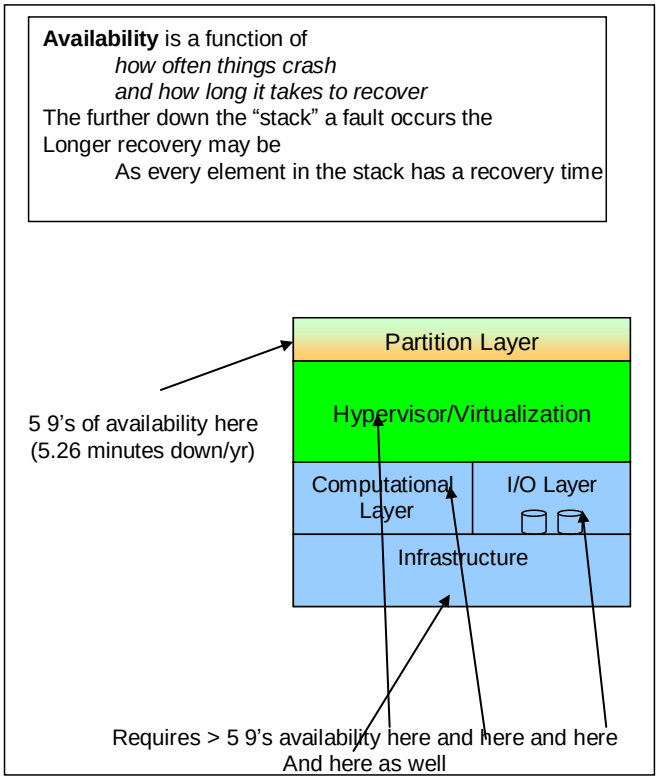
Application Availability Elements

Once a computer system delivers a reliable and trusted computing environment, the availability of that environment is often the next concern. Important applications need to be available at all times, regardless of hardware faults, code defects and so forth.

An often cited gold standard for availability is "5 9s" of availability – meaning that 99.999% of a given period of time, an application is up and running. To achieve 5 9s of availability over any given year requires that an application be un-available no more than 5.26 minutes in that year.

A true measurement of that availability needs to take in to account everything associated with running an application. If an application crashes, the time it is unavailable while restarting must be included in the down time. Likewise if there is a crash due to middle-ware, operating system or hypervisor, this must also be included in downtime.

Figure 37: 5 9s of Availability for the Entire System Stack



For a comprehensive measure of availability downtime associated with planned outages needed to upgrade applications and other code, reconfigure or to take down hardware for replacement or for upgrades would also need to be included.

The remainder of this section discusses in more detail what may be required to achieve application availability in different system environments, starting with the traditional single-server approach and then looking at various models of using multiple servers to achieve that availability or even go beyond 5 9s.

Application Availability: Defining Standards

What is "5 9s" of availability?

Availability is a function of how often things fail, and how long the thing that fails is unavailable after the failure.

How often things fail is usually expressed in terms of mean-time-between failure (MTBF) and is often given as a function of years, like 15 years or 20 years MTBF. It might sound, therefore, that on average a server with a 15 years MTBF might on average run 15 years before having a failure. That might be close to true if most hardware faults were due to components wearing out.

In reality most enterprise servers are designed so that components typically do not wear out during the normal expected life of a product.

MTBF numbers quoted therefore typically reflect the probability that a part defects or other issue causes an outage over a given period of time – during the warranty period of a product, as an example.

Hence 10 years MTBF really reflects that in a large population of servers one out of 10 servers could be expected to have a failure in any given year on average – or 10% of the servers. Likewise, a 50 years MTBF means a failure of 1 server in 50 in a year or 2% of the population.

To calculate application availability, in addition to knowing how often something failed, it's necessary to know how long the application is down because of the failure.

To illustrate, presume an application running on a server can recover from any sort of crash in 8 minutes. Presume that the MTBF for crashes due to any cause is 4 years. On average if 100 servers were running that application, one would expect 25 crashes on average in a year (100 servers run for 1 year /4 years between crashes/server).

An application would be unavailable after a crash for 8 minutes. In total then, for the population it could be said that 25*8 = 200 minutes of application availability would be expected to be lost each year.

The total minutes applications could be running, absent any crashes would be 100 servers * (365.25 days * 24 hours/day*60 minutes/hour) minutes in the average year. 200 minutes subtracted from that number, divided by that number and multiplied by 100% gives the percentage of time applications were available in that population over the average year. For the example, that would be 99.99962%. If the first 5 digits are all 9, then we would say that the population has at least "5 9s of availability."

For the example given, if the downtime for each incident were increased to slightly over 21 minutes so that the application unavailability for the population were 526, that would yield close to exactly 5 9s of availability.

The availability metric is dependent both how frequently outages occur and how long it takes to recover from them.

For the example given, the 99.99962% availability represented 25 outages in a population of 100 systems in a year, with each outage having 8 minutes duration. The same number would also have been derived if the projection were 100 outages of 2 minutes duration each, or 1 outage of 200 minutes.

The availability percentage metric doesn't distinguish between the two, but without such information it is very difficult to understand the causes of outages and areas where improvements could be made.

For example, Five 9s of availability for a population of 100 systems over a year could represent the impact of 25 outages each of about 21 minutes duration, or it could represent a single outage of 526 minutes duration.

Contributions of Each Element in the Application Stack

When looking at application availability it should be apparent that there are multiple elements that could fail and cause an application outage. Each element could have a different MTBF and the recovery time for different faults can also be different.

When an application crashes, the recovery time is typically just the amount of time it takes to detect that the application has crashed, recover any data if necessary to do so, and restart the application.

When an operating system crashes and takes an application down, the recovery time includes all of the above, plus the time it takes for the operating system to reboot and be ready to restart the application.

An OS vendor may be able to estimate a MTBF for OS panics and may have some basis for it -- previous experience, for example. The OS vendor, however, can't really express how many 9s of availability will result for an application unless the OS vendor really knows what application a customer is deploying, and how long its recovery time is.

Even more difficulty can arise with calculating application availability due to the hardware.

For example, suppose a processor has a fault. The fault might involve any of the following:

1. Recovery or recovery and repair that causes no application outage.
2. An application outage and restart but nothing else
3. A partition outage and restart.
4. A system outage where the system can reboot and recover and the failing hardware can subsequently be replaced without taking another outage
5. Some sort of an outage where reboot and recovery is possible, but a separate outage will eventually be needed to repair the faulty hardware.
6. That causes an outage but recovery is not possible until the failed hardware is replaced, meaning that the system and all applications running on it are down until the repair is completed.

The recovery times for each of these incidents is typically progressively longer with the final case very dependent on how quickly replacement parts can be procured and repairs completed.

Application Availability: Enterprise Class System

If enough information is available about failure rates and recovery times, it is possible to project expected application availability.

Figure 38 is an example with hypothetical failure rates and recovery times for the various situations mentioned above looking at a large population of standalone systems each running a single application.

Figure 38: Availability Potential of a Well-Designed Single System

Activities/Elements Involved in Recoveries	*Application restart and recovery*	*Reboot of OS*	*Reboot of Hypervisor and re-establishment of partitions*	*Hardware "Crash" and reboot to point of being able to load Hypervisor including fault diagnostic time*	*Time to Repair hardware in a system if part already identified and available and repair is not concurrent*	*Time to aquire failed parts and have at system ready to begin repair*
Average Recovery Time Per Element (Minutes)	*7*	*4*	*5*	*10*	*30*	*180*

Outage Reason	Mean time to Outage (in Years)	*Recovery Activities Needed*						Total Recovery minutes/Incident	Minutes Down Per Year	Associated Availabilty
Fault Limited To Application	3	*x*						7.00	2.33	*99.99956%*
Fault Causing OS crash	10	*x*	*x*					11.00	1.10	*99.99979%*
Fault causing hypervisor crash	80	*x*	*x*	*x*				16.00	0.20	*99.99996%*
Fault impacting system (crash) but system recovers on reboot with enough resources to restart application	80	*x*	*x*	*x*	*x*			26.00	0.33	*99.99994%*
Planned hardware repair for hw fault (where initial fault impact could be any of the above)	70	*x*	*x*	*x*	*x*	*x*		56.00	0.80	*99.99985%*
Fault impacting system where application is down until system is repaired	500	*x*	*x*	*x*	*x*	*x*	*x*	236.00	0.47	*99.99991%*
						Total For Application from all causes			5.23	*99.99901%*

The chart illustrates the recovery time for each activity that might be required depending on the type of failure. An application crash only requires that the crash be discovered and the application restarted. Hence there is only an x in the column for the 7 minutes application restart and recovery time.

If an application is running under an OS and the OS crashes, then the total recovery time must include the time it takes to reboot the OS plus the time it takes to detect the fault and recover the application after the OS reboots. In the example with an x in each of the first two columns the total recovery time is 11 minutes (4 minutes to recover the OS and 7 for the application.)

The worst case scenario as described in the previous section is a case where the fault cause a system to go down and stay down until repaired. In the example, with an x in all the recovery activities column, that would mean 236 minutes of recovery for each such incident.

This example shows 5 9s of availability.

To achieve that availability, the worst case outage scenarios is shown as being extremely rare compared to the application only-outages.

In addition the example presumed that:

1. All of the software layers can recover reasonably efficiently even from entire system crashes.
2. No more than a reasonable number of application driven and operating system driven outages.

3. A very robust hypervisor is used, expecting it to be considerably more robust that the application hosting OS.
4. Exceptionally reliable hardware is used. (The example presumes about 70 MTBF for hardware faults.)
5. Hardware that can be repaired efficiently, using concurrent repair techniques for the vast majority of the faults.
6. As previously mentioned, a system design that ensures few faults exist that could keep a system down until repaired. In the rare case that such a fault does occur it presumes an efficient support structure that can rapidly deliver the failed part to the failing system and efficiently make the repair.

It must also be stressed that the example only looks at the impact of hardware faults that caused some sort of application outage. It does not deal with outages for hardware or firmware upgrades, patches, or repairs for failing hardware that haven't caused outages.

Critical Application Simplification

Under a single operating system instance it is possible to run multiple applications of various kinds, though typically only one important application is deployed. Likewise in a system with multiple operating system partitions, it is possible to run multiple applications using multiple partitions.

To calculate the availability of each such application, throughout an entire system, calculations would have to change to account for the number of partitions and the outage time associated with each for each fault that could be experienced. The aggregate availability percentage would represent an aggregation of many different applications, not all of equal importance.

Therefore, in the examples in this section, a simplifying assumption is made that each server is running a single application of interest to availability calculations. In other words, the examples look at availability of a critical applications presuming one such application per system.

Enterprise Hardware Unplanned Outage Avoidance

In the example it is assumed that the MTBF for the hardware server, with regards to application impacting outages, was about 70 years (not including the hypervisor.)

This can be difficult to achieve by focusing on the reliability (including soft error resilience) of the server components alone, and the difficulty is compounded if there are pervasive issues in any critical components.

Avoiding these pervasive issues requires monitoring the performance of all the critical components in a system as they go through manufacturing and as they are deployed by customers. Otherwise low-level design defects and latent manufacturing flaws, some perhaps impacting only certain lots of a component, can erode the MTBF expected of a population of systems to something much less than what was theoretically expected. In addition to monitoring for such issues, being willing and able to take corrective actions – by design improvements, manufacturing techniques and testing, by field updates, and perhaps even performing proactive parts replacements on deployed systems if necessary – is also needed to maintain the desired levels of availability over time.

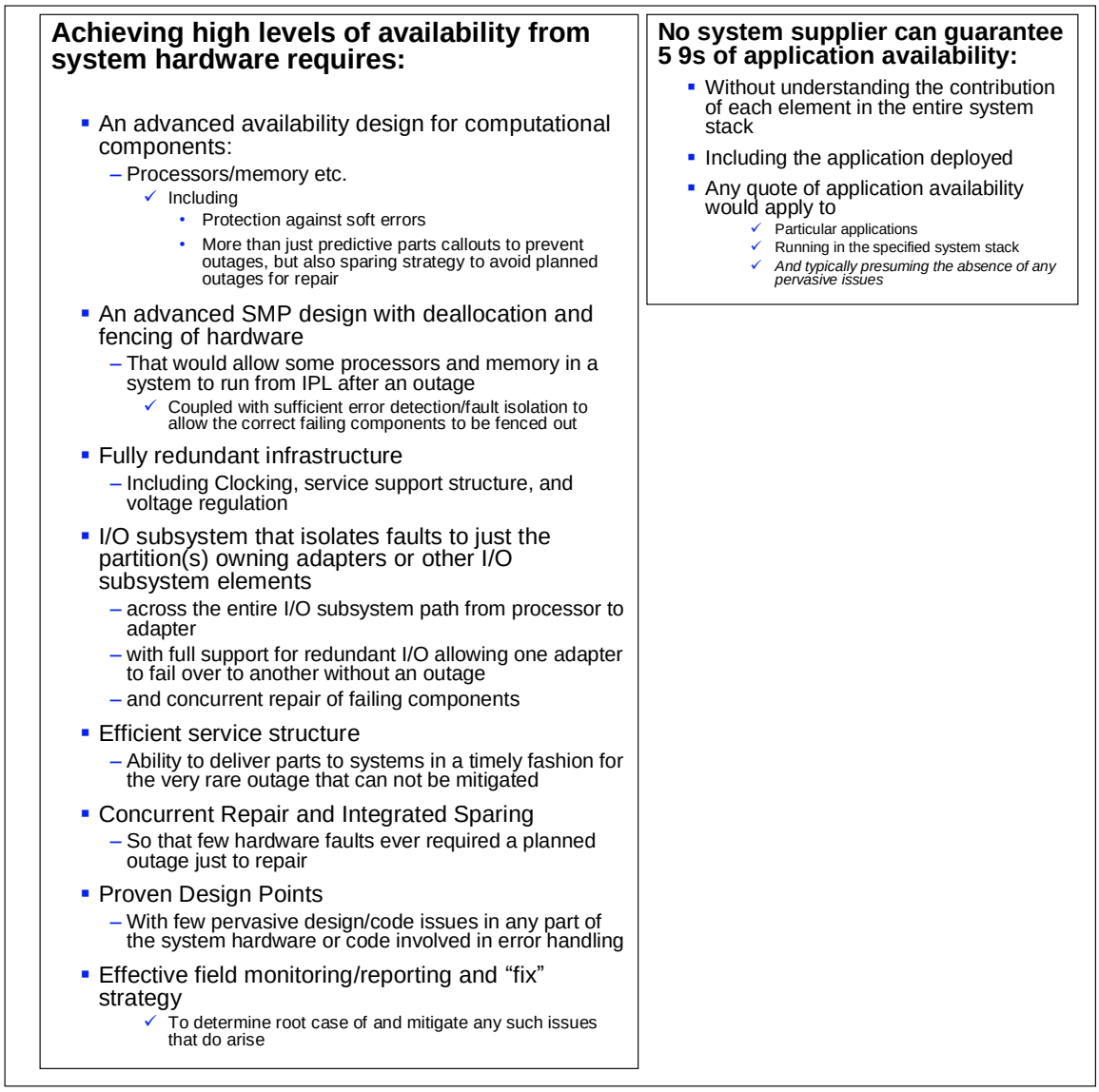
It is more of a challenge to have few if any faults that can keep a system down until repair, but this is also an element in achieving the 5 9s in this example. Systems achieving this need a design with a focus on the system infrastructure (components besides processors and memory).

Again in all of this, the size of the system must be considered. The larger the system, absent advanced techniques for maintaining availability; the more difficult it is to achieve the same MTBF level. Yet smaller systems cannot take advantage of the throughput and utilization capabilities of larger systems.

Enterprise System Design

Overall, IBM's experience with availability suggests that a vendor wishing to provide a server that can run enterprise-sized workloads with this level of availability needs to build one with at least most of the following kinds of characteristics as shown Figure 39 below.

Figure 39: Hardware Design for 5 9s of Availability



What a Hardware Vendor Can Control

A processor vendor wanting to ensure of availability of an application due to hardware has to be in control of many elements beyond the basic hardware reliability. This includes the entire system hardware design, fault diagnosis, a parts stocking strategy for repair of parts, performance of the repairs, and the software stack associated with all of the above including an effective way to track the performance of all of these in the field, and a way to identify and correct pervasive issues, design problems, bugs and etc.

If a hardware vendor depends on a hypervisor, operating system or application to prevent outages from hardware faults, then this must also be under the control of the vendor.

Absent all of that, the hardware vendor can estimate what all of these other elements ought to be able to perform – perhaps in an ideal environment – but success in achieving such availability may be out of the hands of the vendor.

If such a vendor quotes an availability number without explaining the expectations on downtime for all the different fault types up and down the stack, it becomes very difficult to make a comparative analysis or to determine what a real-world implementation would actually experience.

IBM is a processor vendor that also designs and manufactures entire systems and software stacks. IBM supports and repairs systems as deployed by customers, and tracks system performance. This gives IBM a real-world perspective on what it takes to maintain system availability, the means to track performance, and perhaps most importantly, the means to take steps to ensure that availability goals are met, regardless of where in the stack a problem may arise.

Application Availability: Less Capable Systems

The expectations for the enterprise server environment in the previous example are rather stringent.

It has been asked whether a less capable design can nonetheless be considered “good enough” in either a standalone or clustered environment.

Figure 40 gives an example of what might be expected for availability of a single application in a standalone environment for a system with less robust RAS characteristics. Comparing to the enterprise example:

1. The application and operating system are the same.
2. The hypervisor is presumed to be one-third as likely to suffer a crash as the underlying operating system, but is not as robust as the enterprise case. This may be representative of a design approach where the hypervisor is built on an OS base.
3. The hardware MTBF for system outages is presumed to be 20 years based on expectations concerning use of alternate processor/memory recovery designs and lack of infrastructure resiliency.
4. It presumes a less robust error detection fault isolation and service support so it takes a little longer to identify the root cause of the problem and source parts when needed for replacing a failed one.
5. In addition, since the system infrastructure is less robust, it presumes that ½ the faults that take the system down keep the system down until repaired.

Figure 40: Availability in an Ideal System Lacking Enterprise RAS Capabilities

Activities/Elements Involved in Recoveries	*Application restart and recovery*	*Reboot of OS*	*Reboot of Hypervisor and re-establishment of partitions*	*Hardware "Crash" and reboot to point of being able to load Hypervisor including fault diagnostic time*	*Time to Repair hardware in a system if part already identified and available and repair is not concurrent*	*Time to aquire failed parts and have at system ready to begin repair*
Average Recovery Time Per Element (Minutes)	*7*	*4*	*5*	*20*	*30*	*240*

Outage Reason	Mean time to Outage (in Years)	*Recovery Activities Needed*						Total Recovery minutes/Incident	Minutes Down Per Year	Associated Availabilty
Fault Limited To Application	3	*x*						7.00	2.33	*99.99956%*
Fault Causing OS crash	10	*x*	*x*					11.00	1.10	*99.99979%*
Fault causing hypervisor crash	30	*x*	*x*	*x*				16.00	0.53	*99.99990%*
Fault impacting system (crash) but system recovers on reboot with enough resources to restart application	40	*x*	*x*	*x*	*x*			36.00	0.90	*99.99983%*
Planned hardware repair for hw fault (where initial fault impact could be any of the above)	40	*x*	*x*	*x*	*x*	*x*		66.00	1.65	*99.99969%*
Fault impacting system where application is down until system is repaired	40	*x*	*x*	*x*	*x*	*x*	*x*	306.00	7.65	*99.99855%*
						Total For Application from all causes			14.17	*99.99731%*

Since the “expected” amount of outage for any given system is still measured in minutes, one might be tempted to say that the system has “almost 5 9s” of availability. As discussed earlier, stating availability in minutes like this is giving an average of events over multiple systems. The difference in minutes of availability on the average means more outages with significant duration. In this case, it represents more than 2.5 times more system-wide outages and over all 2.5 times more downtime.

This can directly impact service level agreements and cause significant customer impacts.

The analysis also presumes that the enterprise system and less robust system are running an equivalent amount of workload. Large scale enterprise systems are also typically capable of running larger workloads that might not be containable in a single less-robust system .Roughly speaking, needing two systems to run the same workload as one means doubling the number of outages.

Application Availability: Planned Outage Avoidance

Many discussions of hardware availability revolve around the concept of avoiding unplanned outages due to faults in hardware and software that unexpectedly take an application down.

A true measure of application availability considers not just the impact of these unplanned events, but also of any planned outage of an application, partition or any other element on the hardware stack.

Systems designed with concurrent repair capabilities for power supplies, fans, I/O adapters and drawers may typically avoid planned outages for the most common hardware failures.

The ability to generally apply fixes to code levels such as hypervisors, and other hardware related code, or firmware without an outage is also important.

These capabilities are largely the responsibility of the hardware system design. POWER8 based-servers are designed with these capabilities, including 1s and 2s systems.

Beyond the hardware, however, planned outages should be expected at the very least when applications or code levels are updated to new releases with new functional capabilities.

In looking at application availability, planning for these outages is essential and clustered environments are often leveraged to manage the impact of these planned events.

Application Availability: Clustered Environments

Avoiding Outages Due to Hardware

Depending on Software for RAS

Given the challenges involved in architecting, designing, manufacturing and supporting server systems that fit the “enterprise” category in its entirety, it has been suggested that it is sufficient instead to put the onus for application availability primarily on the shoulders of the applications themselves. In such an environment hardware is “good enough” if it can report faults so that the software and applications can deal with them. Little in the way of advanced RAS capability is presumed besides handling soft errors in data.

The preceding discussion shows the difficulty in controlling such an environment and truly achieving 5 9s of availability in a standalone environment.

Distributed Applications

Sometimes availability can be achieved by creating applications that are distributed across multiple systems in such a way that the failure of a single server, in the absence of anything else, will at most result in some loss of performance or capacity, but no application outage.

Sometimes a distributed application may be a very well defined entity such as a distributed file system where all of the “files” in the file-system are distributed across multiple servers. The file system management software is capable of running on multiple servers and responding to requests for data regardless of where the data is contained.

If a distributed file system can maintain and update multiple copies of files on different servers, then the failure of any one server has little impact on the availability of the file system.

Database middleware can be built in a similar fashion.

In addition in a web-commerce environment it is typical for a number of different servers to be used primarily as the front-end between the customer and a database. The front-end work can be farmed out to multiple different servers and when one server fails, workload balancing techniques can be used to cover for the lack of a server.

Even in such environments availability can still depend on the ability to maintain a complex multi-system application relatively “bug-free.”

Fail-over Clustering for High Availability (HA)

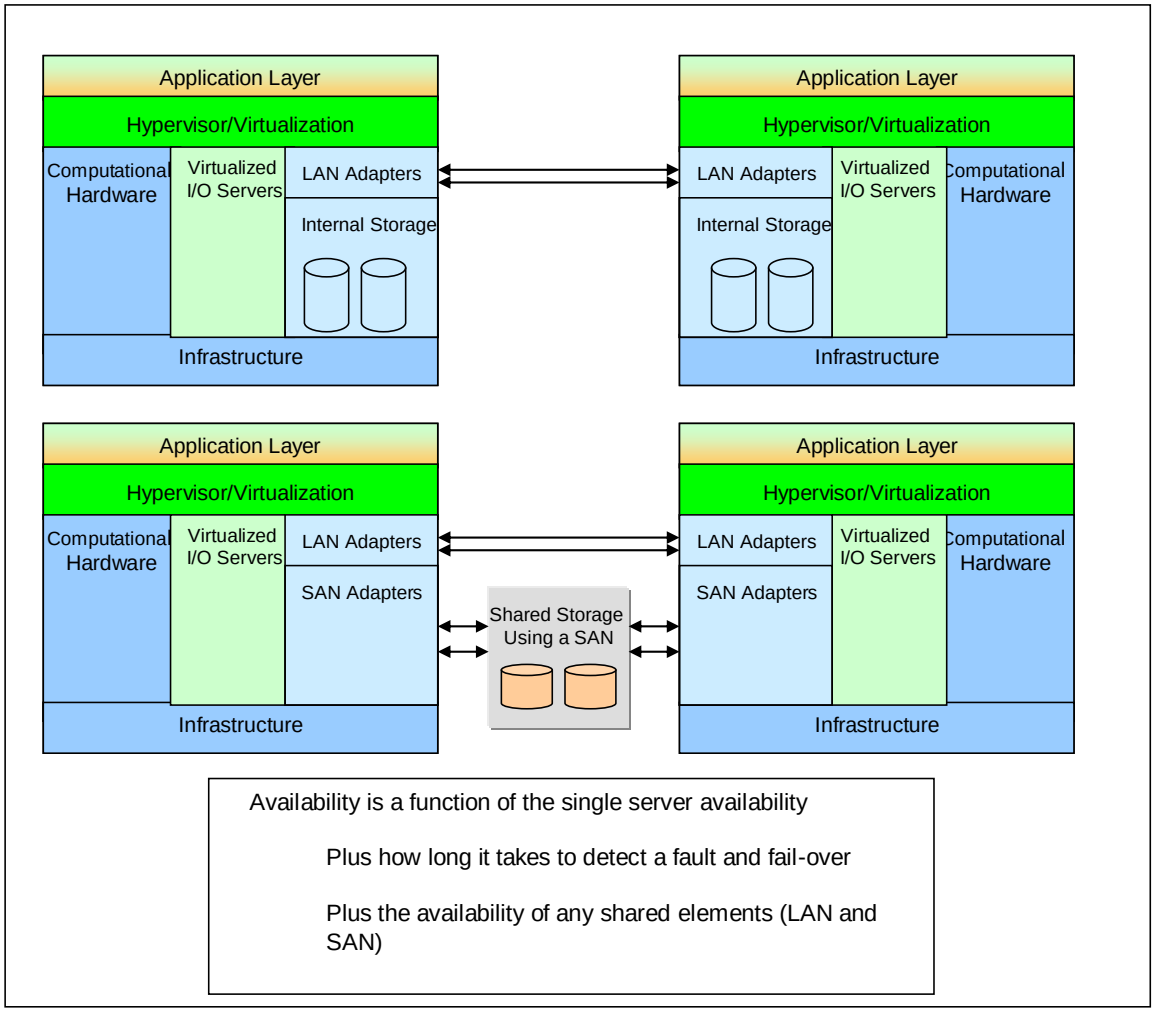
Even when some applications, can be successfully distributed across multiple systems, it is not always effective to distribute every application in an enterprise in the same way.

In these cases what can be described as “fail-over” clustering is often used to achieve better availability for a less distributed workload. Configuring and managing systems in this fashion is often called implementing a High Availability or HA solution.

In the most basic of such solutions, an application runs in a partition on one system. It communicates with a partition on another system. That communication includes some form of “heart-beat” that tells the secondary system that the primary is still up and running and functioning properly.

The two systems may share a common storage subsystem separate from each server, or the primary partition may also communicate to the secondary partition information about whatever changes are made to data that the primary system uses or processes.

Figure 41: Some Options for Server Clustering



When a failure is detected in the primary partition, the secondary partition will attempt to take over for the failed partition. The time it may take to do so would depend on a number of different things, including whether the secondary partition was already running the application, what it might need to take over IP addresses and other resources, and what it might need to do to restore application data if shared data were not implemented.

The totality of the time it takes to detect that an application has failed in one partition, and have the application continuing where it left off, is the “fail-over” recovery time.

This recovery time is typically measured in minutes. Depending on the application in question, the recovery time may be a number of minutes, or even a fraction of a minute, but would not be zero in a fail-over clustering environment. As a result, the availability of the underlying hardware is still of interest in maintaining responsiveness to end users.

Clustered Databases

Some vendors provide cluster-aware databases with HA software to implement a clustered database solution. Because many systems may join in a cluster and access the database these solutions are often considered to be distributed databases. However, implementations that have multiple servers sharing common storage, perhaps in a storage area network, are more properly considered to be clustered databases.

It is often touted that a clustered database means close to 100% database availability. Even in some of the most comprehensive offerings, however, distributed database vendors typically still quote unplanned outage restoration time as “seconds to minutes.”

In cases where applications using the database are clustered rather than distributed, application outage time consists not just of database recovery, but also application recovery even in a clustered database environment.

Measuring Application Availability in a Clustered Environment

It should be evident that clustering can have a big impact on application availability since if recovery time after an outage is required for the application, the time for nearly every outage can be fairly reliability predicted and limited to just that “fail-over” time.

Figure 42 shows what might be achieved with enterprise hardware in such a clustered environment looking at a single application.

Figure 42: Ideal Clustering with Enterprise-Class Hardware

	Time to detect outage in minutes	*1*		
	Time Required To Restart/Recover Application in minutes	*5*		
Outage Reason	Mean time to Outage (in Years)	Total Recovery minutes/Incident	Minutes Down Per Year	Associated Availability
Fault Limited To Application	3	6.00	2.00	*99.99962%*
Fault Causing OS crash	10	6.00	0.60	*99.99989%*
Fault causing hypervisor crash	80	6.00	0.08	*99.99999%*
Fault impacting system (crash) but system recovers on reboot with enough resources to restart application	80	6.00	0.08	*99.99999%*
Planned hardware repair for hw fault (where initial fault impact could be any of the above)	70	6.00	0.09	*99.99998%*
Fault impacting system where application is down until system is repaired	500	6.00	0.01	*100.00000%*
	Total for for all Outage Reasons		2.85	*99.99946%*

Similar to the single-system example, it shows the unavailability associated with various failure types. However, it presumes that application recovery occurs by failing over from one system to another. Hence the recovery time for any of the outages is limited to the time it takes to detect the fault and fail-over and recover on another system. This minimizes the impact of faults that in the standalone case, while rare, would lead to extended application outages.

The example suggests that fail-over clustering can extend availability beyond what would be achieved in the standalone example.

Figure 43 below is an illustration of the same with using systems without Enterprise RAS characteristics.

Figure 43: Ideal Clustering with Reliable, Non-Enterprise-Class Hardware

	Time to detect outage in minutes	*1*		
	Time Required To Restart/Recover Application in minutes	*5*		
Outage Reason	Mean time to Outage (in Years)	Total Recovery minutes/Incident	Minutes Down Per Year	Associated Availability
Fault Limited To Application	3	6.00	2.00	*99.99962%*
Fault Causing OS crash	10	6.00	0.60	*99.99989%*
Fault causing hypervisor crash	30	6.00	0.20	*99.99996%*
Fault impacting system (crash) but system recovers on reboot with enough resources to restart application	40	6.00	0.15	*99.99997%*
Planned hardware repair for hw fault (where initial fault impact could be any of the above)	40	6.00	0.15	*99.99997%*
Fault impacting system where application is down until system is repaired	40	6.00	0.15	*99.99997%*
	Total for for all Outage Reasons		3.25	*99.99938%*

It shows that clustering in this fashion does greatly reduce the impact of the system-down-until-repaired High Impact Outage (HIO) failures and does improve the expected average application availability. Clustering does not reduce the number of outages experienced, however, again seeing more than 2.5 times the number of system-wide outages compared to the enterprise hardware approach.

Recovery Time Caution

The examples given presume that it takes about six minutes to recover from any system outage. This may be realistic for some applications, but not for others. As previously shown, doubling the recovery time has a corresponding large impact on the availability numbers.

The expectation for In addition, whatever the recovery time associated with such a fail-over event is for a given application needs to be very well understood. Such a HA solution is really no HA solution at all if a service level agreement requires that no outage exceed, for example, 10 minutes, and the HA fail-over recovery time is 15.

Clustering Infrastructure Impact on Availability

A clustering environment might be deployed where a primary server runs with everything it needs under the covers, including maintaining all of the storage, data and otherwise within the server itself. This is sometimes referred to as a “nothing shared” clustering model.

In such a case, clustering involves copying data from one server to the backup server, with the backup server maintaining its own copy of the data. The two systems communicate by a LAN and redundancy is achieved by sending data across the redundant LAN environment.

It might be expected in such a case that long outages would only happen in the relatively unlikely scenarios where both servers are down simultaneously, both LAN adapters are down simultaneously, or there is a bug in the failover itself.

Hardware, software and maintenance practices must ensure that high availability for this infrastructure is achieved if high application availability is to be expected.

The “nothing shared” scenario above does not automatically support the easy migration of data from one server to another for planned events that don’t involve a failover.

An alternative approach makes use of a shared common storage such as a storage area network (SAN), where each server has accessed to and only makes use of the shared storage.

Real World Fail-over Effectiveness Calculations

The previous examples do also presume that such high availability solutions are simple enough to implement that they can be used for all applications and that fail-over, when initiated, always works.

In the real world that may not always be the case. As previously mentioned, if the secondary server is down for any reason when a failover is required, then failover is not going to happen and the application in question is going to be down until either the primary or secondary server is restored. The frequency of such an event happening is directly related to the underlying availability characteristics of each individual server – the more likely the primary server is to fail, the more likely it needs to have the backup available and the more likely the backup server is to be down, the less likely it will be available when needed.

Any necessary connections between the two must also be available, and that includes any shared storage.

It should be clear that if the availability of an application is to meet 5 9s of availability, then shared storage must have better than 5 9s of availability.

Different kinds of SAN solutions may have sufficient redundancy of hard drives to assure greater than 5 9s availability of data on the drives, but may fall down in providing availability of the I/O and control that provide access to the data.

The most highly available SAN solutions may require redundant servers under the cover using their own form of fail-over clustering to achieve the availability of the SAN.

Advanced HA solutions may also take advantage of dual SANs to help mitigate against SAN outages, and also to support solutions across multiple data centers. This mechanism effectively removes the task of copying data from the servers and puts it on the SAN.

Such mechanism can be very complicated both in the demands of the hardware and also in the controller software.

The role of software layers in cluster availability is crucial, and it can be quite difficult to ensure that all of the software deployed is bug-free. Every time an application fails, it may fail at a different point. Ensuring detection of every hardware fault and data-lossless recovery of an application for every possible failure is complicated by the possibility of exercising different code paths on each failure. This difficulty increases the possibility of latent defects in the code. There are also difficulties in making sure that the fail-over environment is properly kept up to date and in good working order with software releases that are consistently patched and known to be compatible.

To ensure proper working order of a fail-over solution as regards to all of the above, testing may be worthwhile, but live testing of the fail-over solution itself can add to application unavailability.

When the fail-over solution does not work as intended, and especially when something within the clustering infrastructure goes wrong, the recovery time can be long. Likewise if a primary server fails when the secondary is not available, or if the state of the transactions and data from the one server cannot be properly shadowed, recovery can include a number of long-downtime events such as waiting on a part to repair a server, or the time required to rebuild or recover data.

A good measurement of availability in a clustered environment should therefore include a factor for the efficiency of the fail-over solution implemented; having some measure for how frequently a fail-over fails and how long it takes to recover from that scenario.

In the figures below, the same Enterprise and Non-Enterprise clustering examples are is evaluated with an added factor that one time in twenty a fail-over event doesn't go as planned and recovery from such events takes a number of hours.

Figure 44: More Realistic Model of Clustering with Enterprise-Class Hardware

	Time to detect outage in minutes	*1*		*Ratio of Problem Failover Events to Succcessful Events*	*How Long For Application Recovery after a Problem Failover*		
	Time Required To Restart/Recover Application in minutes	*5*		*1:20*	*60*		
Outage Reason	Mean time to Outage (in Years)	Total Recovery minutes/Incident	Minutes Down Per Year	Mean Time to Fail-over Issue (years)	Additional Time To Account for Fail-over issues (minutes)	Total Minutes Down Per Year	Availablity Associated with Fault Type
Fault Limited To Application	3	6.00	2.00	*60*	*1*	3.00	*99.99943%*
Fault Causing OS crash	10	6.00	0.60	*200*	*0.3*	0.90	*99.99983%*
Fault causing hypervisor crash	80	6.00	0.08	*1600*	*0.0375*	0.11	*99.99998%*
Fault impacting system (crash) but system recovers on reboot with enough resources to restart application	80	6.00	0.08	*1600*	*0.0375*	0.11	*99.99998%*
Planned hardware repair for hw fault (where initial fault impact could be any of the above)	70	6.00	0.09	*1400*	*0.042857143*	0.13	*99.99998%*
Fault impacting system where application is down until system is repaired	500	6.00	0.01	*10000*	*0.006*	0.02	*100.00000%*
			Total Minutes of downtime per year			4.27	
						Availability	*99.99919%*

Figure 45: More Realistic Clustering with Non-Enterprise-Class Hardware

	Time to detect outage in minutes	*1*		*Ratio of Problem Failover Events to Succcessful Events*	*How Long For Application Recovery after a Problem Failover*		
	Time Required To Restart/Recover Application in minutes	*5*		*1:20*	*150*		
Outage Reason	Mean time to Outage (in Years)	Total Recovery minutes/Incident	Minutes Down Per Year	Mean Time to Fail-over Issue (years)	Additional Time To Account for Fail-over issues (minutes)	Total Minutes Down Per Year	Availablity Associated with Fault Type
Fault Limited To Application	3	6.00	2.00	*60*	*2.5*	4.50	*99.99914%*
Fault Causing OS crash	10	6.00	0.60	*200*	*0.75*	1.35	*99.99974%*
Fault causing hypervisor crash	30	6.00	0.20	*600*	*0.25*	0.45	*99.99991%*
Fault impacting system (crash) but system recovers on reboot with enough resources to restart application	40	6.00	0.15	*800*	*0.1875*	0.34	*99.99994%*
Planned hardware repair for hw fault (where initial fault impact could be any of the above)	40	6.00	0.15	*800*	*0.1875*	0.34	*99.99994%*
Fault impacting system where application is down until system is repaired	40	6.00	0.15	*800*	*0.1875*	0.34	*99.99994%*
			Total Minutes of downtime per year			7.31	
						Availability	*99.99861%*

The example presumes somewhat longer recovery for the non-enterprise hardware due to the other kinds of real-world conditions described in terms of parts acquisition, error detection/fault isolation (ED/FI) and so forth.

Though these examples presume too much to be specifically applicable to any given customer environment, they are intended to illustrate two things:

The less frequently the hardware fails the better the ideal availability and also less perfect clustering has to be to achieve desired availability.

If the clustering and failover support elements themselves have bugs/pervasive issues, or single points of failure besides the server hardware, less than 5 9s of availability (with reference to hardware faults) may still occur in a clustered environment. It is possible that availability might be worse in those cases than in comparable stand-alone environment.

Reducing the Impact of Planned Downtime in a Clustered Environment

The previous examples did not look at the impact of planned outages except for deferred repair of a part that caused some sort of outage.

Planned outages of systems may in many cases occur more frequently than unplanned outages. This is especially true of less-than-enterprise class hardware that:

1. Require outages to patch code (OS, hypervisor, etc.)
2. Have no ability to repair hardware using integrated sparing and similar techniques and instead must predictively take off-line components that may otherwise subsequently fail and require an outage to repair.
3. Do not provide redundancy and concurrent repair of components like I/O adapters.

In a clustered environment, it seems reasonable that when it is known in advance that a system needs to be taken down during a planned maintenance window, that recovery and fail-over times could be minimized with some advanced planning. Still, so long as fail-over techniques are used for the planned outages, one should still expect recovery time to be in the minutes range.

However, it is also possible to take advantage of a highly virtualized environment to migrate applications from one system to another in advanced of a planned outage, without having to recover/restart the applications.

PowerVM Enterprise Edition™ offers one such solution called Live Partition Mobility (LPM). In addition to use in handling planned hardware outages, LPM can mitigate downtime associated with hardware and software upgrades and system reconfiguration and other such activities which could also impact availability and are otherwise not considered in even the “real world” 5 9s availability discussion.

HA Solutions Cost and Hardware Suitability

Clustering Resources

One of the obvious disadvantages of running in a clustered environment, as opposed to a standalone system environment, is the need for additional hardware to accomplish the task.

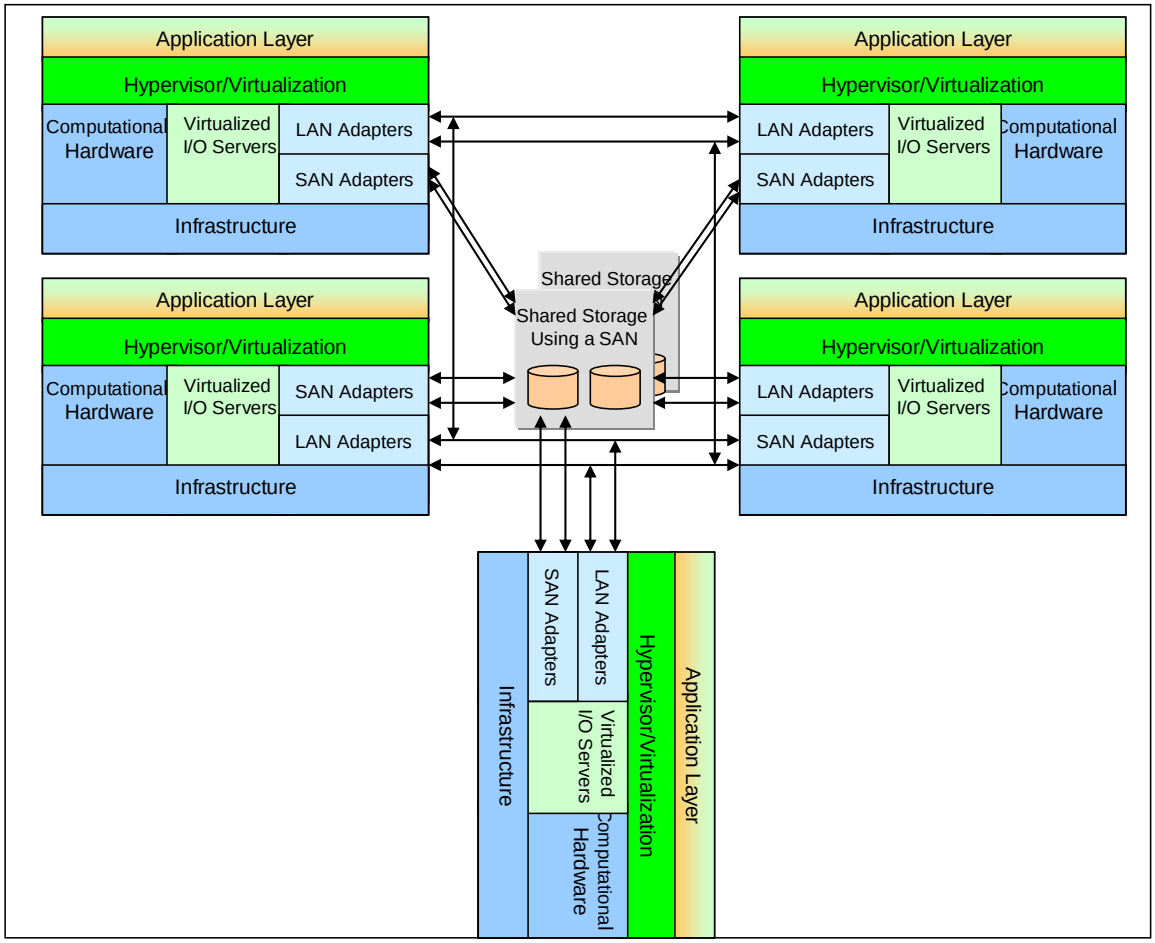
An application running full-throttle on one system, prepared to failover on another, needs to have a comparably capability (available processor cores, memory and so forth) on that other system.

There does not need to be exactly one back-up server for every server in production, however. If multiple servers are used to run work-loads, then only a single backup system with enough capacity to handle the workload of any one server might be deployed.

Alternatively, if multiple partitions are consolidated on multiple servers, then presuming that no server is fully utilized, fail-over might be planned so that one failing server will restart on different partitions on multiple different servers.

When an enterprise has sufficient workload to justify multiple servers, either of these options reduces the overhead for clustering.

Figure 46: Multi-system Clustering Option



There are a number of additional variations that could be considered.

In practice there is typically a limit as to how many systems are so clustered together. These include concerns about increased risk of simultaneous server outages, relying too much on the availability of shared storage where shared storage is used, the overhead of keeping data in sync when shared storage is not used, and practical considerations ensuring that where necessary all systems aligned in terms of code-updates, hardware configuration and so forth.

It many cases it should also be noted that applications may have licensing terms that make clustering solutions more expensive. For example, applications, databases etc. may be licensed on a per core basis. If the license is not based on how many cores are used at any given time, but on how many specific cores in specific systems the application might run on, then clustering increases licensing cost.

Using High Performance Systems

In these environments, it becomes not only useful to have a system that is reliable, but also capable of being highly utilized. The more the processors and memory resources of a system are used, the fewer total system resources are required and that impacts the cost of backup server resources and licenses.

Power Systems are designed with per core performance in mind and highly utilized. Deploying PowerVM allows for a great depth in virtualization allowing applications to take the most advantage of the processing, I/O and storage capabilities. Power Systems thus have a natural affinity towards use in clustered environments where maximizing resources is important in reducing ownership costs.

Summary

Heritage and Experience

Since at least the time of the IBM System/360™ “mainframe” computers IBM has been promoting and developing the concept of enterprise computing dependability through RAS: incorporating Reliability (making component fails infrequent), Availability (maintaining operation in the face of failures) and Serviceability (precisely identifying and reporting faults and efficiently and non-disruptively repairing failures.) These concepts have been honed over the course of various generations resulting in what is now known as the IBM System z™.

Power Systems have benefited from the knowledge and experience of System z designers and at least for the last fifteen years have worked specifically to adopt the developed methods and techniques in multiple generations of POWER processor-based systems.

The concept of virtualization and partitioning, for instance, has nearly as long a history in these mainframes and POWER based systems.

And in 1990 IBM introduced Sysplex™ for the System z Heritage mainframes as a means of clustering systems to achieve high availability.

Clustering techniques for POWER processor-based systems have also had a long history, perhaps even preceding the adoption of many of the availability techniques now incorporated into these systems.

System design must respond to changes in applications and the ways in which they are deployed. The availability techniques sufficient for servers designed in 2003 or 2005, when software based approaches for recovery were the norm, are not really sufficient in 2014 or 2015 to maintain the kinds of availability in server systems that are expected, given the capacity and performance and technology used in these servers.

IBM is perhaps the most keenly aware of what is required because IBM servers are deployed in many mission critical enterprise environments. IBM is responsible for the performance and quality of much of the hardware deployed in these environments, and often much of the software stack as well.

To support such an environment, IBM has an infrastructure that is geared towards quickly solving problems that arise in any customer situation and also gathering information on root causes to improve products going forward.

The error checking capacity built in to Power Systems is designed not merely to handle faults in some fashion when they occur but more importantly to allow the root cause to be identified, and on the first (and sometimes only occurrence) of an issue. This allows IBM to develop a fix, workaround or circumvention for issues where it is possible to do so. Owning processor design and manufacture, the design and manufacture of systems as well as the design of many software components perhaps uniquely positions IBM to make such fixes possible.

Consequently, POWER8 processor-based systems benefit from IBM’s knowledge and experience in processor design and the challenges involved in moving to the next generation of processor fabrication and technology. They also benefit from IBM’s experience on previous generations of products throughout the entire system, including power supplies, fans, and cabling technology and so forth. IBM offerings for clustered systems also benefit from IBM’s experience in supplying clustering software and clustering capable hardware on both Power and System z systems as well as the same dedication to enterprise server design and support in these products.

Application Availability

IBM’s unique approach to system reliability and enterprise availability can be clearly contrasted to alternate approaches that depend much more on software elements to attempt to recover or mitigate the impact of outages.

Power Systems intend to deliver substantial single system availability advantageous to any application or operating system environment.

Clustering is supported to achieve the highest levels of availability. With the understanding that having the most sound single system server availability gives the best foundation for achieving the highest levels of availability in real-world settings when clustering is employed.

Appendix A. Selected RAS Capabilities by Operating System

Legend:

- ● = supported, - = not supported, NA = Not used by the Operating System
- ●* supported in POWER Hypervisor, not supported in PowerKVM environment,
- ●^ supported in POWER Hypervisor, limited support in PowerKVM environment

Note: HMC is an optional feature on these systems.

RAS Feature	AIX V7.1 TL3 SP3 V6.1 TL9 SP3	IBMi V7R1M0 TR8 V7R2M0	Linux RHEL6.5 RHEL7 SLES11SP3 Ubuntu 14.04
Processor			
FFDC for fault detection/error isolation	●	●	●
Dynamic Processor Deallocation	●	●	●*
Dynamic Processor Sparing Using capacity from spare pool	●	●	●*
Core Error Recovery			
◆ Processor Instruction Retry	●	●	●
◆ Alternate Processor Recovery	●	●	●*
◆ Partition Core Contained Checkstop	●	●	●*
I/O Subsystem			
PCI Express bus enhanced error detection	●	●	●
PCI Express bus enhanced error recovery	●	●	●^
PCI Express card hot-swap	●	●	●*
Memory Availability			
Memory Page Deallocation	●	●	●
Special Uncorrectable Error Handling	●	●	●
Fault Detection and Isolation			
Storage Protection Keys	●	NA	NA
Error log analysis	●	●	●^

RAS Feature	AIX V7.1 TL3 SP3 V6.1 TL9 SP3	IBMi V7R1M0 TR8 V7R2M0	Linux RHEL6.5 RHEL7 SLES11SP3 Ubuntu 14.04
Serviceability			
Boot-time progress indicators	●	●	●
Firmware error codes	●	●	●
Operating system error codes	●	●	●^
Inventory collection	●	●	●
Environmental and power warnings	●	●	●
Hot-swap DASD / Media	●	●	●
Dual Disk Controllers / Split backplane	●	●	●
Extended error data collection	●	●	●
SP “call home” on non-HMC configurations	●	●	●*
IO adapter/device standalone diagnostics with PowerVM	●	●	●
SP mutual surveillance w/ POWER Hypervisor	●	●	●
Dynamic firmware update using HMC	●	●	●
Service Indicator LED support	●	●	●
System dump for memory, POWER Hypervisor, SP	●	●	●
Infocenter / Systems Support Site service publications	●	●	●
System Support Site education	●	●	●
Operating system error reporting to HMC SFP app.	●	●	●
RMC secure error transmission subsystem	●	●	●
Health check scheduled operations with HMC	●	●	●
Operator panel (real or virtual)	●	●	●
Concurrent Op Panel Maintenance	●	●	●
Redundant HMCs supported	●	●	●
Automated server recovery/restart	●	●	●
High availability clustering support	●	●	●
Repair and Verify Guided Maintenance with HMC	●	●	●
PowerVM Live Partition / Live Application Mobility With PowerVM Enterprise Edition	●	-	●
Power and Cooling and EPOW			
N+1 Redundant, hot swap CEC fans	●	●	●
N+1 Redundant, hot swap CEC power supplies	●	●	●
EPOW error handling	●	●	●*

Legend:

● = supported, - = not supported, NA = Not used by the Operating System

●* supported in POWER Hypervisor, not supported in PowerKVM environment,

●^ supported in POWER Hypervisor, limited support in PowerKVM environment

Note: Hardware Management Console (HMCs), required for some functions, are optional features for these systems.

About the principal/editor:

Daniel Henderson is an IBM Senior Technical Staff Member. He has been involved with POWER and predecessor RISC based products development and support since the earliest RISC systems. He is currently the lead system hardware availability designer for IBM Power Systems platforms.

Notices:

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

IBM Corporation
Systems and Technology Group
Route 100
Somers, New York 10589

Produced in the United States of America
January 2016

The Power Systems page can be found at: http://www-03.ibm.com/systems/power/

The IBM Systems Software home page on the Internet can be found at: http://www-03.ibm.com/systems/software/