Introduction to IBM® Power® Reliability, Availability, and Serviceability for Power10 processor-based systems using IBM PowerVM®

IBM Infrastructure
Daniel Henderson, Irving Baysah
Updated December 2024

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® Power® Power6® Power7® Power7+™ Power8® Power® PowerHA® PowerLinux™

PowerVM® System x® System z® PowerVC Power Hypervisor™ Power Architecture™

IBM®

Additional Trademarks may be identified in the body of this document.
Google is a registered trademark of Google LCC. Intel is a registered trademark of the Intel corporation.
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 two principal authors/editors 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:

Kanisha Patel, Kevin Reilly, Marc Gollub, Julissa Villarreal, Michael Mueller, George Ahrens, Kanwal Bahri, Steven Gold, Jim O'Connor, K Paul Muller, Ravi A. Shankar, Kevin Reick, Peter Heyrman, Dave Stanton, Dan Hurlimann, Kaveh Naderi, Nicole Nett, John Folkerts and Hoa Nguyen.

Table of Contents

Introduction

Reliability generally refers to the infrequency of system and component failures experienced by a server.

Availability, broadly speaking, is how the hardware, firmware, operating systems and application designs handle failures to minimize application outages.

Serviceability generally refers to the ability to efficiently and effectively install and upgrade systems firmware and applications, as well as to diagnose problems and efficiently repair faulty components when required.

These interrelated concepts of reliability, availability and serviceability are often spoken of as" RAS".

Within a server environment all of RAS, but especially application availability, is really an end-to-end proposition. Attention to RAS needs to permeate all the aspects of application deployment. However, a good foundation for server reliability whether in a scale-out or scale-up environment is clearly beneficial.

Systems based on the Power processors are generally known for their design emphasizing Reliability, Availability and Serviceability capabilities. Previous versions of a RAS whitepaper have been published to discuss general aspects of the hardware reliability and the hardware and firmware aspects of availability and serviceability.

The focus of this whitepaper is to introduce the Power10 processor-based systems using the PowerVM hypervisor. Systems not using PowerVM will not be discussed specifically in this whitepaper.

This whitepaper is organized into five sections:

Section 1: RAS Overview of key Power10 processor-based systems
An overview of the RAS capabilities of the latest Power10 processor-based systems.

Section 2: General Design Philosophy
A general discussion of Power RAS design philosophy, priorities, and advantages.

Section 3: Power10 Subsystems RAS Details
A more detailed discussion of each sub-system within a Power server concentrating on the RAS features of processors, memory, and other components of each system.

Section 4: Reliability and Availability in the Data Center
Discussion of RAS measurements and expectations, including various ways in which RAS may be described for systems: Mean Time Between Failures, '9's of availability and so forth.

Section 5: Serviceability
Provides descriptions of the error log analysis, call-home capabilities, service environment and service interfaces.

Comparative Discussion

In September 2021, IBM introduced the first models using Power10 processors: The IBM Power E1080; a scalable server using multiple four socket Central Electronics Complex (CEC) drawers. The Power E1080 design was inspired by the Power E980 but has enhancements in key areas to complement the performance capabilities of the Power10 processor.
One of these key enhancements includes an all-new memory subsystem with Differential DIMMs (DDIMMs) using a memory buffer that connects to processors using an Open Memory Interface (OMI) which is a serial interface capable of higher speeds with fewer lanes compared to a traditional parallel approach.

Another enhancement is the use of both passive external and internal cables for the fabric busses used to connect processors between drawers eliminating the routing of signals through the CEC backplane in contrast to the Power9 approach where signals were routed through a backplane and the external cables were active. This design point significantly reduces the likelihood that the labor intensive and costly replacement of the main system backplane will be needed.
Another change of note from a reliability standpoint is that the processor clock design, while still redundant in the Power E1080 has been simplified since it is no longer required that each processor module within a CEC drawer be synchronized with the others.

Figure 1: Power9/Power10 Servers RAS Highlights Comparison

	Power9 1s and 2s Systems^	Power9 IBM Power E950	Power9 IBM Power E980	Power10 IBM Power E1080
Base Power9™ Processor RAS features including • First Failure Data Capture • Processor Instruction Retry • L2/L3 Cache ECC protection with cache line-delete • Power/cooling monitor function integrated into on chip controllers of processors	Yes*	Yes	Yes	Yes
Power9 Enterprise RAS Features • Core Checkstops	No	Yes	Yes	Yes
Multi-node SMP Fabric RAS • CRC checked processor fabric bus retry with spare data lane and/or bandwidth reduction	N/A	N/A	Yes	Yes – Power10 design removes active components on cable and introduces internal cables to reduce backplane replacements
PCIe hot-plug with processor integrated PCIe controller[$]	Yes	Yes	Yes	Yes
Memory DIMM ECC supporting x4 Chipkill*	Yes	Yes	Yes	Yes

Uses IBM memory buffer and has spare DRAM module capability with x4 DIMMs*	No	Yes	Yes	Yes – New Memory Buffer
x8 DIMM support with Chipkill correction for marked DRAM*	N/A	N/A	Yes	N/A
Custom DIMM support with additional spare DRAM capability*	No	No	Yes	Yes, New Custom DIMM
Active Memory Mirroring for the Hypervisor	No	Yes - Feature	Yes - Base	Yes- Base
Redundant/spare voltage phases on voltage converters for levels feeding processor and custom memory DIMMs or memory risers.	No	Redundant	Both redundant and spare	Yes, For Processors- DDIMMs use on board power management integrated Circuits (PMICs)
Redundant processor clocks	No	No	Yes	Yes – New Design
Redundant service processor and related boot facilities	No	No	Yes	Yes
Redundant TPM capability	No	No	Yes	Yes
Transparent Memory Encryption	No	No	No	Yes
Multi-node support	No	No	Yes	Yes

* In scale-out systems Chipkill capability is per rank of a single Industry Standard DIMM (ISDIMM); in IBM Power E950 Chipkill and spare capability is per rank spanning across an ISDIMM pair; and in the IBM Power E980, per rank spanning across two ports on a Custom DIMM.
The Power E950 also supports DRAM row repair
^ IBM Power® S914, IBM Power® S922, IBM Power® S924
IBM Power® H922 ,IBM Power® S924, IBM Power® H924

$Note: I/O adapter and device concurrent maintenance in this document refers to the hardware capability to allow the adapter or device to be removed and replaced concurrent with system operation. Support from the operating system is required and will depend on the adapter or device and configuration deployed.

In July 2022, the General Availability of the Power10 4-Socket enterprise mid-range server and the 1- and 2-sockets scaleout servers followed the GA of the E1080. The table below provide comparison highlights among the current Power10 based systems.

Figure 2: Power10 Servers RAS Highlights Comparison

	Power10 1s and 2s IBM Power S1014, S1022, S1024	Power10 IBM Power E1050	Power10 IBM Power E1080
Base Power10™ Processor RAS features including • First Failure Data Capture • Processor Instruction Retry • L2/L3 Cache ECC protection with cache line-delete • Power/cooling monitor function integrated into on chip controllers of processors	Yes	Yes	Yes
Power10 Enterprise RAS Features • Core Checkstops	Yes	Yes	Yes
Multi-node SMP Fabric RAS • CRC checked processor fabric bus retry with spare data lane and/or bandwidth reduction	N/A	N/A	Yes – Power10 design removes active components on cable and introduces internal cables to reduce backplane replacements
PCIe hot-plug with processor integrated PCIe controller	Yes No Cassette	Yes – with blindswap cassette	Yes – with blindswap cassette
Memory DIMM ECC supporting x4 Chipkill*	Yes	Yes	Yes
Dynamic Memory Row repair and spare DRAM capability	2U DDIMM – no spare DRAM 4U DDIMM (post GA) – 2 spare DRAM per rank Yes – Dynamic Row Repair	Yes - base 4U DDIMM – 2 spare DRAM per rank Yes – Dynamic Row Repair	Yes - base 4U DDIMM – 2 spare DRAM per rank Yes – Dynamic Row Repair
Active Memory Mirroring for the Hypervisor	Yes – Base New to scaleout	Yes - Base	Yes - Base
Redundant/spare voltage phases on voltage converters for levels feeding processor	No	Yes N+1	Yes N+2
Redundant On-board Power Management Integrated Circuits (PMIC) memory DDIMMs	No with 2U DDIMM Yes – optional 4U DDIMM (post GA)	Yes - Base 4U DDIMM	Yes - Base 4U DDIMM

Service Processor Type	Enterprise Base Board Management Controller (eBMC) – open standard with Redfish support	Enterprise Base Board Management Controller (eBMC) – open standard with Redfish support	Flexible Service Processor (FSP) – IBM proprietary
Processor Clocks Redundancy/Sparing	No	Integrated Spare (post GA)	Redundant
Redundant service processor and related boot facilities	No	No	Yes
Boot from more than 1 processor socket (redundant FSP advantage)	No	No	Yes
Redundant TPM capability (dual TPM module in each drawer)	No	No	Yes
Transparent Memory Encryption	Yes	Yes	Yes
Multi-node support	N/A	N/A	Yes
Concurrent Op-Panel Repair	No – Op Panel Base Yes - LCD	Yes – Op Panel Base Yes – LCD (post GA)	Yes – Op Panel Base Yes - LCD
TOD Battery Concurrent Maintenance	No	Yes	Yes
Internal Cables	Lots of internal cables	Very Few internal cables	Few internal cables

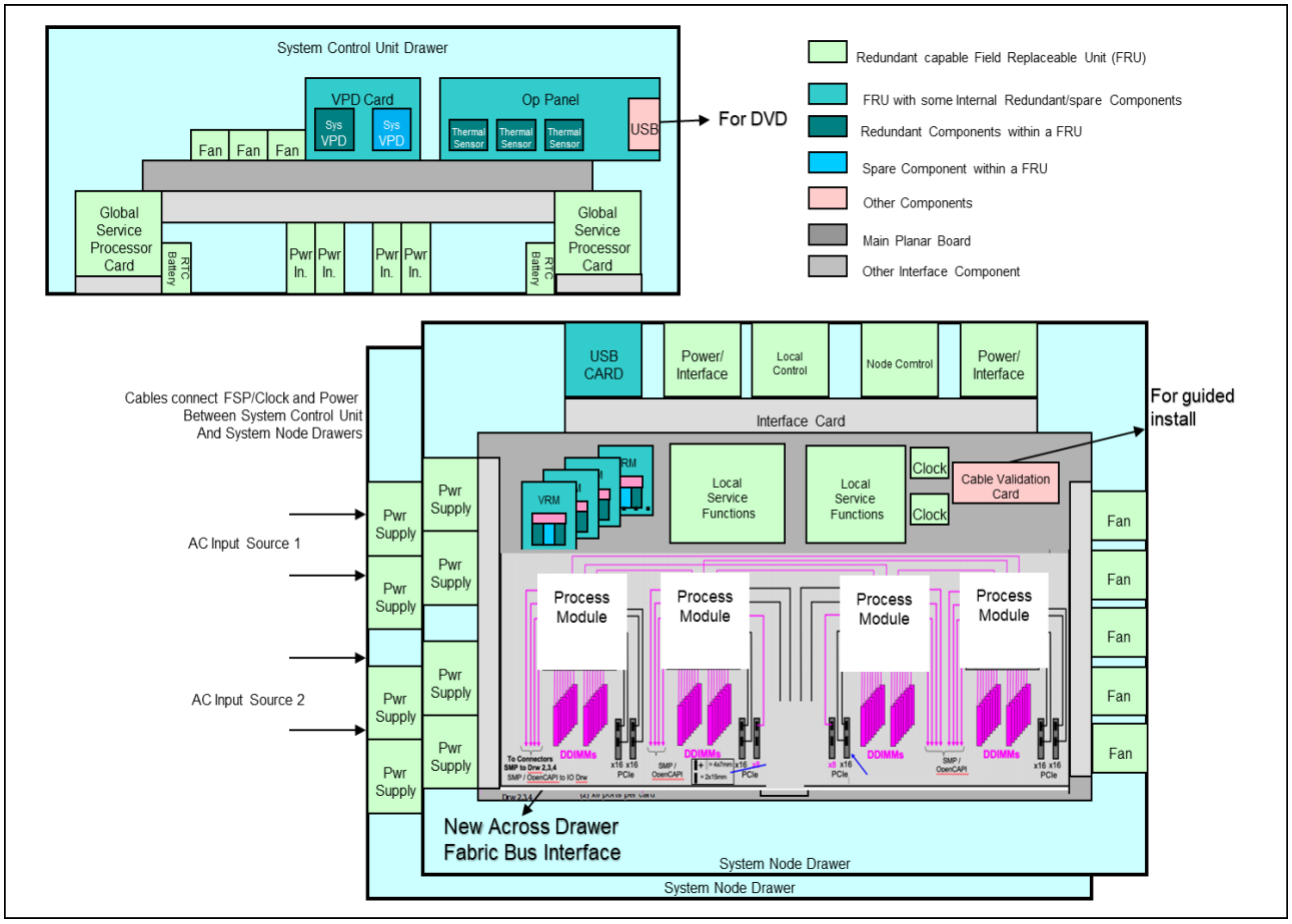
IBM Power E1080

As announced, the Power E1080 is designed to be capable of supporting multiple drawers each containing 4 processor sockets. The initial product offering will be limited to two drawers.
In addition to these CEC drawers, the design supports a system control drawer which contains the global service processor modules. Each CEC drawer supports 8 PCIe slots and I/O expansion drawers are also supported.

System Structure

A rough view of the Power E1080 design is represented in the figure below:
Compared to the Power E980 the most visible changes are in the memory sub-system and the fabric busses that connect the processors.

Figure 3: Power E1080 Structure Simplified View



Not illustrated in the figure above is the internal connections for the SMP fabric busses which will be discussed in detail in another section.
In comparing the Power E1080 design to the Power9-based Power E980, it is also interesting to note that the processor clock function has been separated from the local service functions and now resides in a dual fashion as separate clock cards.

The Power8 multi-drawer system design required that all processor modules be synchronized across all CEC drawers. Hence a redundant clock card was present in the system control drawer and used for all the processors in the system.

In Power9 only each CEC drawer was required to be synchronized using the same reference clock source. In the Power E1080, each processor can run asynchronous to all the other processors and use a separate/different reference clock source. While a clock and redundant clock is provided in each node (rather than for each processor) there is no need for logic to keep them coordinated. This allows for a significantly simpler clock design.

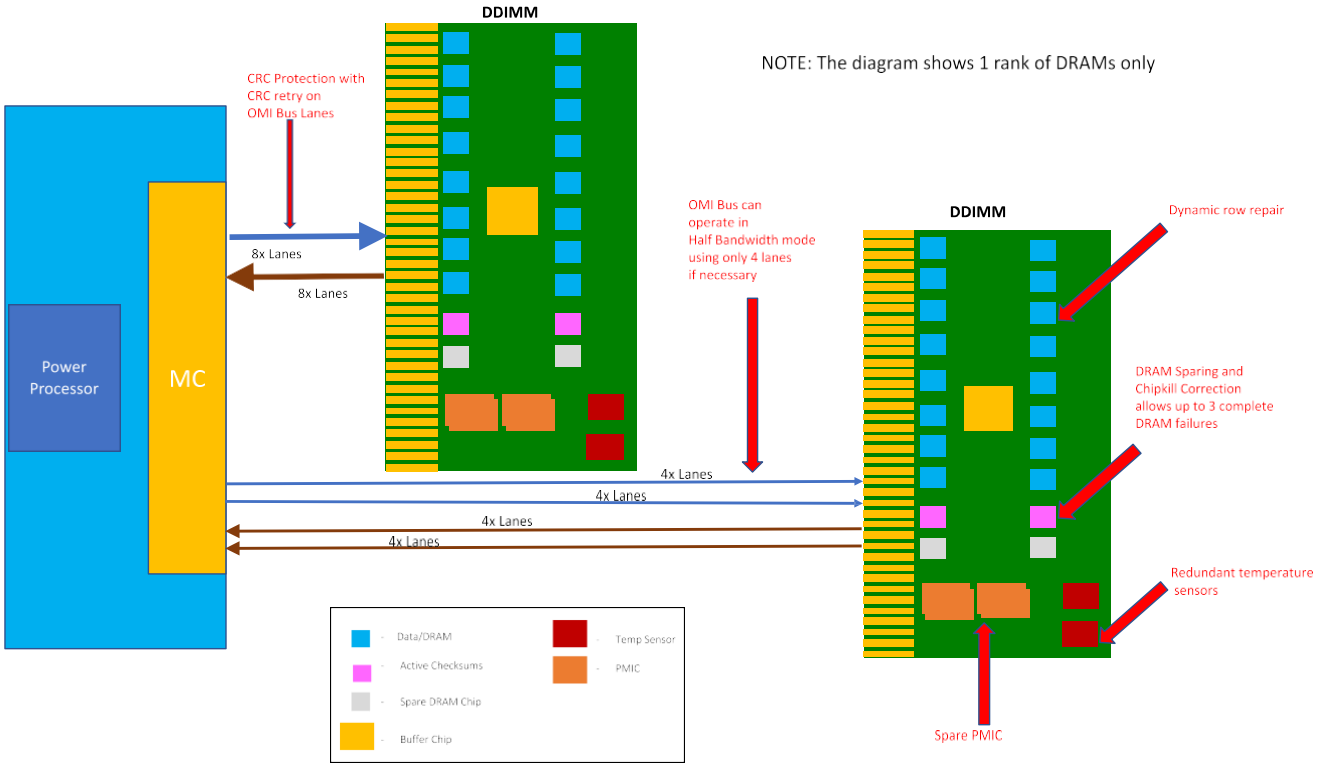
Power E1080 Processor RAS

While there are many differences internally in the Power10 processor compared to Power9 that relate to performance, number of cores and so forth, the general RAS philosophy for how errors are handled has remained largely the same.

Two items which have been changed are the memory controller function and the SMP connections between the processors. These will be discussed in the next two sub-sections. The processor clocking has also changed as previously discussed.

Power E1080 Memory Subsystem

Figure 4: DDIMM Memory Features



The memory subsystem of the Power E1080 has been completely redesigned to support Differential DIMMs (DDIMMs) with DDR4 memory that leverage a serial interface to communicate between processors and the memory.

A memory DIMM is generally considered to consist of 1 or more “ranks” of memory modules (DRAMs). A standard DDIMM module may consist of 1 or 2 ranks of memory and will be approximately 2 standard “rack units” high, called 2U DDIMM. The Power E1080 exclusively uses a larger DDIMM with up to 4 ranks per DDIMM (called a 4U DDIMM). This allows for not

only additional system capacity but also room for additional RAS features to better handle failures on a DDIMM without needing to take a repair action (additional self-healing features). The components of the memory interface include the memory controller on the processors, the open memory interface (OMI) that a memory controller uses to communicate with the memory buffer on each DIMM, the DRAM modules that support a robust error detection and correction capability (ECC), the on-DIMM power management integrated circuits (PMICs) and thermal monitoring capabilities.

These components are considered below looking first at what is standard for both 2U and 4U DDIMMs and then what is unique for the 4U DDIMMs the Power E1080s use.

Buffer
The DDIMM incorporates a new Microchip memory buffer designed to IBM's RAS specifications. Key RAS features of the buffer include protection of critical data/address flows using CRC, ECC and parity, a maintenance engine for background memory scrubbing and memory diagnostics, and a Fault Isolation Register (FIR) structure which enables firmware attention-based fault isolation and diagnostics.
Unlike Power9 systems, this memory buffer does not contain an L4 cache.

OMI
The OMI interface between the memory buffer and processor memory controller is protected by CRC retry/recovery facility to re-transmit lost frames to survive intermittent bit flips. A complete lane fail can also be survived by triggering a dynamic lane reduction from 8 to 4, independently for both up and downstream directions. A key advantage of the OMI interface is that it simplifies the number of critical signals that must cross connectors from processor to memory compared to a typical industry standard DIMM design.

Memory ECC
The DDIMM includes a robust 64-byte Memory ECC, with 8-bit symbols, capable of correcting up to five symbol errors (one x4 chip and one additional symbol), as well as retry for data and address uncorrectable errors.

Dynamic Row Repair
To further extend the life of the DDIMM, the dynamic row repair feature can restore full use of a DRAM for a fault contained to a DRAM row, while system continues to operate.

Spare Temperature Sensors
Each DDIMM provides spare temperature sensors, such that the failure of one does not require a DDIMM replacement.

Unique to 4U DDIMM: Spare DRAMs
4U DDIMMs used in the Power E1080 include two spare x4 memory modules (DRAMs) per rank. These can be substituted for failed DRAMs during runtime operation. Combined with ECC correction, the 2 spares allow the 4U DDIMM to continue to function with 3 bad DRAMs per rank, compared to 1 (single device data correct) or 2 (double device data correct) bad DRAMs in a typical industry standard DIMM design.
This extends self-healing capabilities beyond what is provided with dynamic row repair capability.

Power Management
In the Power E980, voltage regulator modules (VRMs) in the CEC drawers were separately used to provide different voltage levels to the CDIMMs where the levels would be used or further divided on the CDIMM.

The DDIMMs used in the Power E1080 use power management ICs (PMICs) to divide voltages and provide the power management for the DDIMMs. Separate memory VRMs are no longer used.

The 4U DDIMMs also include spare power management ICs (PMICs) such that the failure of one PMIC does not require a DDIMM replacement.

Power E1080 SMP Interconnections

Power E1080 Processor to Processor SMP Fabric Interconnect Within a CEC Drawer
To communicate between processors within a CEC drawer, the Power E1080 uses a fabric bus composed of eight bi-directional lanes of data. The physical busses are run from processor to processor through the CEC drawer main backplane.

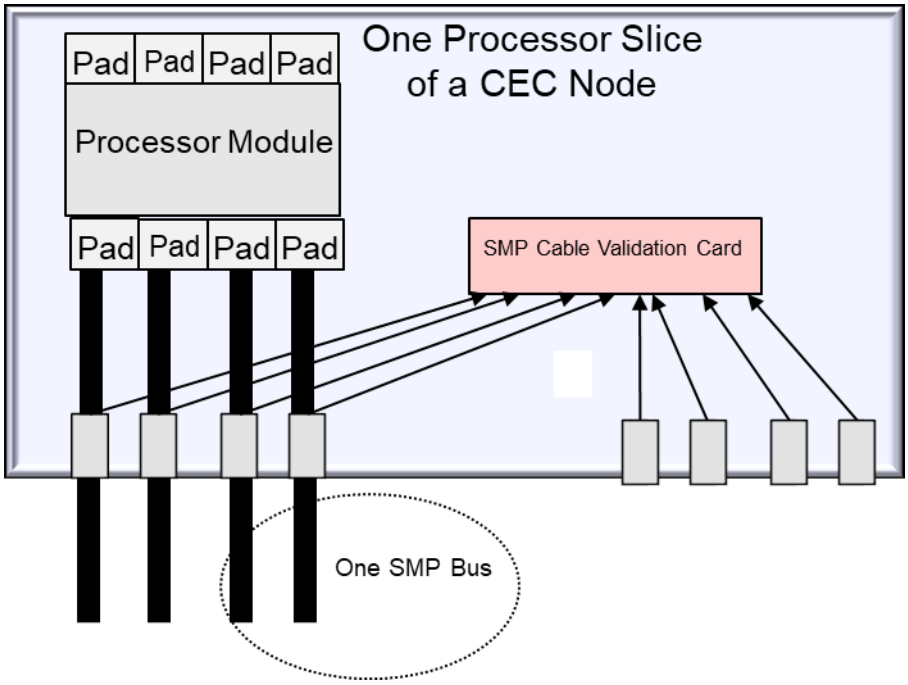
The data transferred is CRC checked. Intermittent errors can result in retry of operation. During run-time should a persistent error occur on a bus, the system can reduce from using eight lanes to four lanes. This capability is called "½ bandwidth mode".

Power E1080 CEC Drawer to CEC Drawer SMP Fabric Interconnect Design
The SMP Fabric busses used to connect processors across CEC nodes is similar in RAS function to the fabric bus used between processors within a CEC drawer. Each bus is functionally composed of eight bi-directional lanes of data. CRC checking with retry is also used. ½ bandwidth mode is supported.

Unlike the processor-to-processor within a node design, the lanes of data are carried from each processor module through internal cables to external cables and then back through internal cables to the other processor.

Physically each processor module has eight pads (four on each side of the module.) Each pad side has an internal SMP cable bundle which connects from the processor pads to a bulkhead in each CEC drawer which allows the external and internal SMP cables to be connected to each other.

Figure 5: SMP Fabric Bus Slice



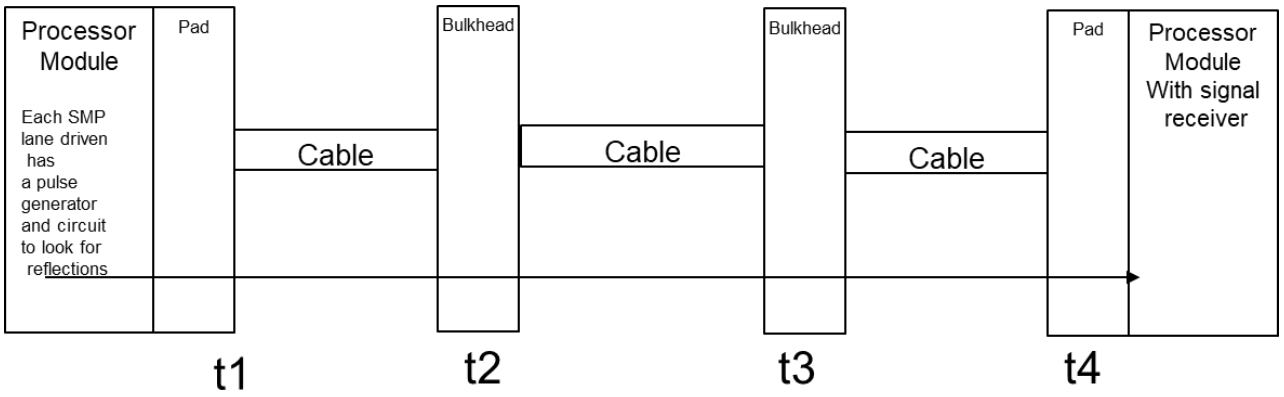
Where the illustration above shows just the connections on one side of the processor.
In addition to connecting with the bulkhead, each cable bundle also has connections to an SMP cable Validation Card which has logic used to verify the presence and type of a cable to help guide the installation of cables in a system.

Since physical cables are used, each bus also employs a spare data lane. This can be used to substitute for a failed lane without the need to enter ½ bandwidth mode or the need to schedule a repair.

The ability to concurrently repair an external cable that failed during run-time before a system is rebooted is also supported.

One key difference in this design compared to SMP9 is that the external SMP cable is no longer an active cable. This can be an advantage in reducing the number of components that can fail on the cable, but it does make it harder to isolate the root cause of a cable fault.
To maintain error isolation with the new cost-effective design, an advanced diagnostic capability called Time Domain Reflectometry (TDR) is built into the Power10 processor and a test performed whenever a bus goes into half-bandwidth mode.

Figure 6: Time Domain Reflectometry



Though it is beyond the scope of this whitepaper to delve into the exact details of how TDR works, as a very rough analogy it can be likened to a form of sonar where when desired, a processor module that drives a signal on a lane can generate an electrical pulse along the path to a receiving processor in another CEC drawer. If there is a fault along the path, the driving processor can detect a kind of echo or reflection of the pulse. The time it takes for the reflection to be received would be indicative of where the fault is within the cable path.
For faults that occur mid-cable, the timing is such that TDR should be able to determine exactly what field replaceable unit to replace to fix the problem. If the echo is very close to a connection, two FRUs might be called out, but in any case, the use of TDR allows for good fault isolation for such errors while allowing the Power10 server to take advantage of a fully passive path between processors.

Power E1080 SMP Fabric Interconnect Design within a CEC Drawer
Unlike the SMP fabric busses between processor drawers, the connections between processors are still routed processor-to-processor within the drawer “planar” or motherboard card instead of cables. The design maintains a CRC checking with retry and the ability to go into a ½ bandwidth mode if needed.

Internal I/O

The Power E1080 supports 8 PCIe Gen4/Gen5 PCIe slots in each CEC drawer. Up to 4 optional NVMe drives are supported. USB support for the System Control Unit, if desired can be provided by using a USB adapter in one of the 8 PCIe slots and a USB cable connected to the System Control Drawer.

To determine the exact processor to slot assignments, refer to the system documentation or "redbook" for the Power E1080.

Note, however that location codes have been changed for the CEC drawers.
Power9 used location codes that began with 1, e.g., P1-C1 would refer to the first planar in a unit and the first connector. In Power10, location codes used to identify elements within the system now begin with a zero., e.g., P0-C0 would be the first planar, first connector.

Power E1080 and E980 Telemetry Data

These models are designed with extensive thermal and power management capabilities. The E980 and E1080 Power Management unit consists of dedicated controllers that dynamically optimize processor frequency based on the workload. The firmware also has additional monitoring features for memory, IO adapters, system planars, etc. For instance, if a DIMM is running at high temperature, the energy monitoring firmware will detect the event and throttle the memory bandwidth to protect the DIMM.

With the release of HMC1040, the Hardware Management Console (HMC) provides API for customers to acquire power usage and thermal metric from Flexible Service Processor (FSP) based systems. The HMC API return raw, processed, or aggregated power and thermal data. Clients now have the ability to easily gather telemetry data and monitor clusters of E1080 and E980 servers in a datacenter. For more details on the FSP based system telemetry data collection, go to the IBM Power10 Energy Monitoring page.

IBM Power E1050

The Power E1050 is designed to support 2-socket, 3-socket and 4-socket processor configurations. The 2-socket system can be upgraded to 3-socket or 4-socket in the field or at the customer site. Each processor socket is a Dual Chip Module (DCM), which consists of 2 Power10 processor chips. A DCM connects to 16 memory channels, which equates to 64 DDIMMs per fully populated E1050.

As with the E1080, the E1050 customers will benefit from the highly reliable 4U DDIMM. With advanced memory RAS features like integrated spare PMIC, the ability to withstand multiple serial Chipkill events, dynamic DRAM row repair and OMI channel half bandwidth mode, the 4U DDIMM offers in-memory databases very high application availability.

The E1050 introduces the Enterprise Baseboard Controller (eBMC) as the service processor. This is a departure from the IBM proprietary FSP which is used in the E1080 and previous IBM Power enterprise servers. The E1050 eBMC design essentially brings the unmatched enterprise RAS features of the FSP architecture to the industry standard BMC. The eBMC supports the Redfish API which is an open standard designed for simple and secure management of hybrid cloud infrastructure. This allows the E1050 to be easily integrated into any hybrid cloud environment with simplified system serviceability.

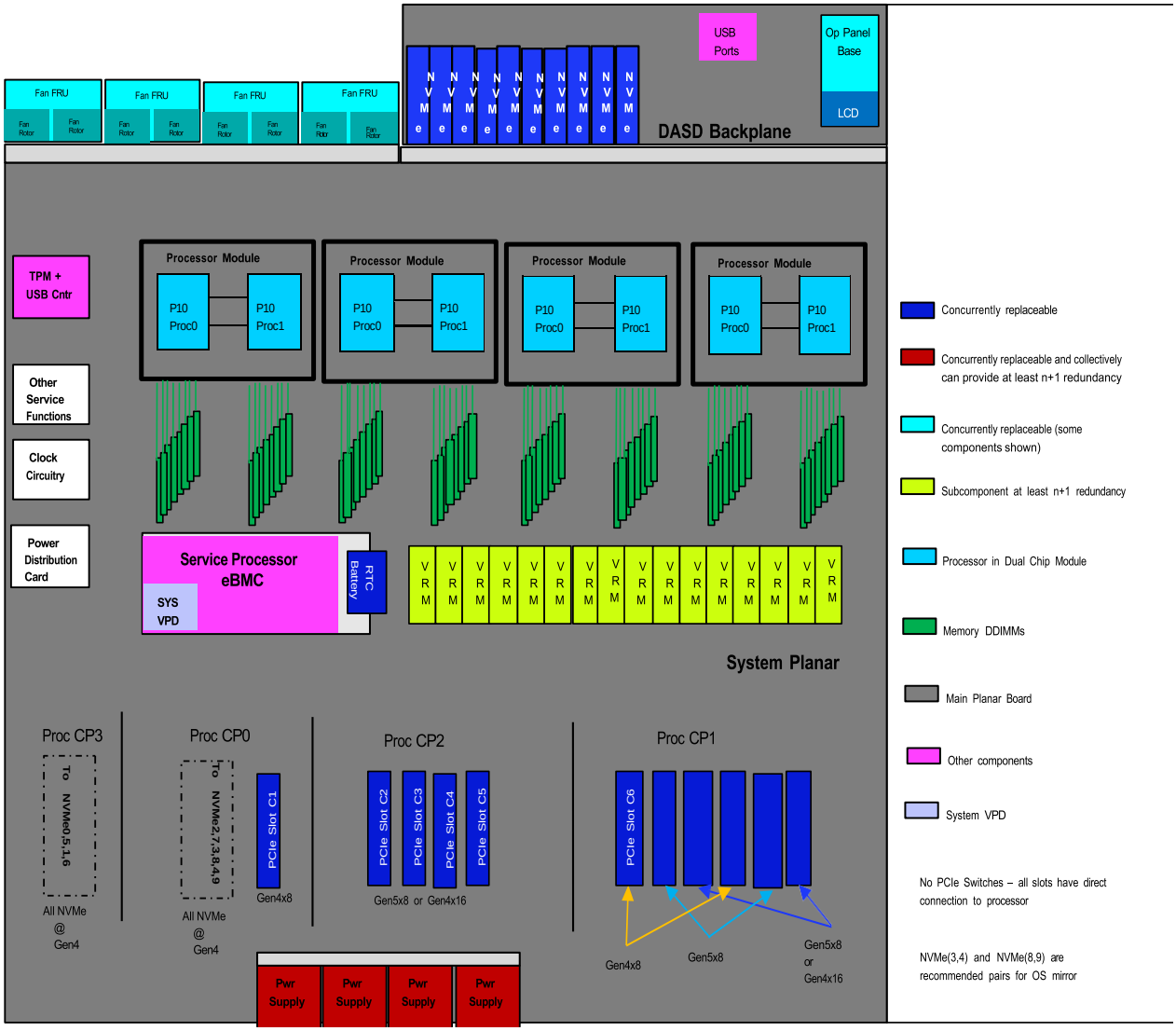
The E1050 brings a significantly higher performing machine to the same form factor as the E950 while maintaining the best-in-class RAS. There are more processor cores per die, faster processor interconnect, higher memory bandwidth, more PCIe Gen4/5 slots, to name but a few. These components are protected by CRC and/or ECC with retry capability where appropriate. The use of infrastructure redundancy and concurrent maintenance or hotplug is designed in critical components.

System Structure

A simplified view of the Power E1050 design is represented in the figure below:
The E1050 maintains the same system form factor and infrastructure redundancy as the E950. As depicted in the E1050 server diagram below, there are 4 Power Supplies and Fan Field Replaceable Units (FRU) to provide at least N+1 redundancy. These components can be concurrently maintained or hot add/removed. There is also N+1 Voltage Regulation Module (VRM) phase redundancy to the Processors and redundant Power Management Integrated Circuit (PMIC) supplying voltage to the 4U DDIMM that the E1050 offers.

The E1050 Op Panel base and LCD are connected to the same planer as the internal NVMe drives. The Op Panel base and LCD are separate FRUs and are concurrently maintainable. The NVMe backplane also has 2 USB 3.0 ports, accessible through the front of the system, for system OS. Not shown in the diagram, are 2 additional OS USB 3.0 ports at the rear of the system, connected through the eBMC card.

Figure 7: Power E1050 Structure Simplified View



Power E1050 Processor RAS

The E1050 processor module is a DCM which differs from that of the E950 which has Single Chip Module (SCM). Each DCM has 30 processor cores, which is 120 cores for a 4S E1050 system. In comparison, a 4S E950 server supports 48 cores. The internal processor buses are twice as fast with the E1050 running at 32Gbs. Despite the increased cores and the faster high speed processor bus interfaces, the RAS capabilities are essentially equivalent. With features like Processor Instruction Retry (PIR), L2/L3 Cache ECC protection with cacheline delete and CRC fabric bus retry that is a characteristic of P9 and P10 processors. As in the E950, when an internal or external fabric bus lane encounters a hard failure, the lane can be dynamically spared out with no impact to system availability.

Power E1050 Memory RAS

Unlike the processor RAS characteristics, the E1050 memory RAS varies significantly from that of the E950. The E1050 supports the same 4U DDIMM as the E1080.

The memory comparison DIMM table below highlights the differences between the E950 DIMM and the E1050 DDIMM. It also provides the RAS impacts of the DDIMMs, which are applicable to the E1080 servers. For more description on the DDIMM, refer to the E1080 System Memory section of this document.

Figure 8: E950 DIMM vs E1050 DDIMM RAS Comparison

	Power9 E950 Memory	Power10 E1050 Memory	RAS impact
DIMM Form Factor	Riser card plus ISDIMMs	4U DDIMM	• **P10 4U DDIMM**: Single FRU or fewer components to replace • **E950 DIMM**: Separate FRU used for memory buffer on riser card and the ISDIMMs
symbol Correction	Single Symbol correction	Dual Symbol correction	• **P10 4U DDIMM**: Data pin fail (1 symbol) lining up with single cell fail on another DRAM is still correctable • **E950 DIMM**: Data pin fail (1 symbol) lining up with single cell fail on another DRAM is uncorrectable
X4 Chip Kill	One spare DRAM per port or across a DIMM pair	Two spare DRAM per port	• **P10 4U DDIMM** ○ 1st chip kill fixed with spare ○ 2nd serial chip kill fixed with spare ○ 3rd serial chip kill fixed with ECC ○ 4th serial chip kill is uncorrectable • **E950 DIMM** ○ 1st chip kill fixed with spare ○ 2nd serial chip kill fixed with ECC ○ 3rd serial chip kill is uncorrectable
DRAM Row Repair	Static	Dynamic	• **P10 4U DDIMM**: Detect, fix, and restore at runtime without system outage • **E950 DIMM**: Detect at runtime, but fix and restore requires system reboot
L4 Cache	Yes	No	• **P10 4U DDIMM**: Avoids L4 cache failure modes • **E950 DIMM**: L4 cache fails contribute to DIMM replacements
Voltage Regulation Redundancy	No	Yes	• **P10 4U DDIMM:** can survive a voltage regulation component failure • **E950 DIMM:** voltage regulation and associated components are single point of failure

NOTE: A memory ECC code is defined by how many bits or symbols (group of bits) it can correct. The Power10 DDIMM memory buffer ECC code organizes the data into 8-bit symbols and each symbol contains the data from one DRAM DQ over 8 DDR beats.

Power E1050 I/O RAS

The E1050 provides 11 general purpose PCIe slots that allows for hot plugging of IO adapters. These PCIe slots operate at Gen4 and Gen5 speeds. As shown in the table below, some of the PCIe slots support I/O expansion drawer cable cards.

Unlike the E950, the E1050 location codes start from index 0, as with all Power10 servers. However, slot c0 is not a general purpose PCIe slot and it's reserved for the eBMC Service Processor card.

Another difference between the E950 and the E1050, is that all the E1050 slots are directly connected to a P10 processor. In the E950 some slots are connected to the P9 processor by way of I/O switches.

All 11 general purpose PCIe slots are available if 3S or 4S DCM are populated. In the 2S DCM configuration, only 7 PCIe slots are functional.

Figure 9: Power E1050 I/O Slot Assignments

Slot	Type	From	Supports
C1	x8 G4	CP0 = DCM0/C0	PCIe Adapters, Cable card for I/O expansion
C2	x8 G5/x16 G4	CP2 = DCM2/C1	PCIe Adapters, Cable card for I/O expansion
C3	x8 G5/x16 G4	CP2 = DCM2/C1	PCIe Adapters, Cable card for I/O expansion
C4	x8 G5/x16 G4	CP2 = DCM2/C0	PCIe Adapters, Cable card for I/O expansion
C5	x8 G5/x16 G4	CP2 = DCM2/C0	PCIe Adapters, Cable card for I/O expansion
C6	x8 G4	CP1 = DCM1/C1	PCIe Adapters
C7	x8 G5	CP1 = DCM1/C1	PCIe Adapters, Cable card for I/O expansion
C8	x8 G5/x16 G4	CP1 = DCM1/C1	PCIe Adapters, Cable card for I/O expansion
C9	x8 G4	CP1 = DCM1/C0	PCIe Adapters
C10	x8 G5	CP1 = DCM1/C0	PCIe Adapters, Cable card for I/O expansion
C11	x8 G5/x16 G4	CP1 = DCM1/C0	PCIe Adapters, Cable card for I/O expansion

DASD Options

The E1050 provides 10 internal NVMe drives at Gen4 speeds. The NVMe drives are connected to DCM0 and DCM3. In a 2S DCM configuration, only 6 of the drives are available. A 4S DCM configuration is required to have access to all 10 internal NVMe drives. Unlike the E950, the E1050 has no internal SAS drives. An external drawer can be used to provide SAS drives.

The internal NVMe drives support OS-controlled RAID0 and RAID1 array, but no hardware RAID. For best redundancy, the OS mirror and dual VIOS mirror can be employed. To ensure as much separation as possible in the hardware path between mirror pairs, the following NVMe configuration is recommended:

a.) Mirrored OS: NVMe3,4 or NVME8,9 pairs
b.) Mirrored Dual VIOS
    I. Dual VIOS: NVMe3 for VIOS1, NVMe4 for VIOS2
    II. Mirrored the Dual VIOS: NVMe9 mirrors NVMe3, NVMe8 mirrors NVMe4

Power E1050 Service Processor

The IBM Power10 E1050 comes with a redesigned service processor based on a Baseboard Management Controller (BMC) design with firmware that is accessible through open-source industry standard APIs, such as Redfish. An upgraded Advanced System Management Interface

(ASMI) web browser user interface preserves the required enterprise RAS functions while allowing the user to perform tasks in a more intuitive way.

Equipping the industry standard BMC with enterprise service processor functions that are characteristic of FSP based systems, like the E1080, has led to the name Enterprise BMC (eBMC). As with the FSP, the eBMC runs on its own power boundary and does not require resources from a system processor to be operational to perform its tasks.

The service processor supports surveillance of the connection to the Hardware Management Console (HMC) and to the system firmware (hypervisor). It also provides several remote power control options, environmental monitoring, reset, restart, remote maintenance, and diagnostic functions, including console mirroring. The BMC service processors menus (ASMI) can be accessed concurrently during system operation, allowing nondisruptive abilities to change system default parameters, view and download error logs, check system health.

Redfish, an Industry standard for server management, enables the Power Servers to be managed individually or in a large data center. Standard functions such as inventory, event logs, sensors, dumps, and certificate management are all supported with Redfish. In addition, new user management features support multiple users and privileges on the BMC via Redfish or ASMI. User management via Lightweight Directory Access Protocol (LDAP) is also supported. The Redfish events service provides a means for notification of specific critical events such that actions can be taken to correct issues. The Redfish telemetry service provides access to a wide variety of data (e.g. power consumption, ambient, core, DDIMM and I/O temperatures, etc.) that can be streamed on periodic intervals.

IBM Power S1014, S1022, S1024 Servers

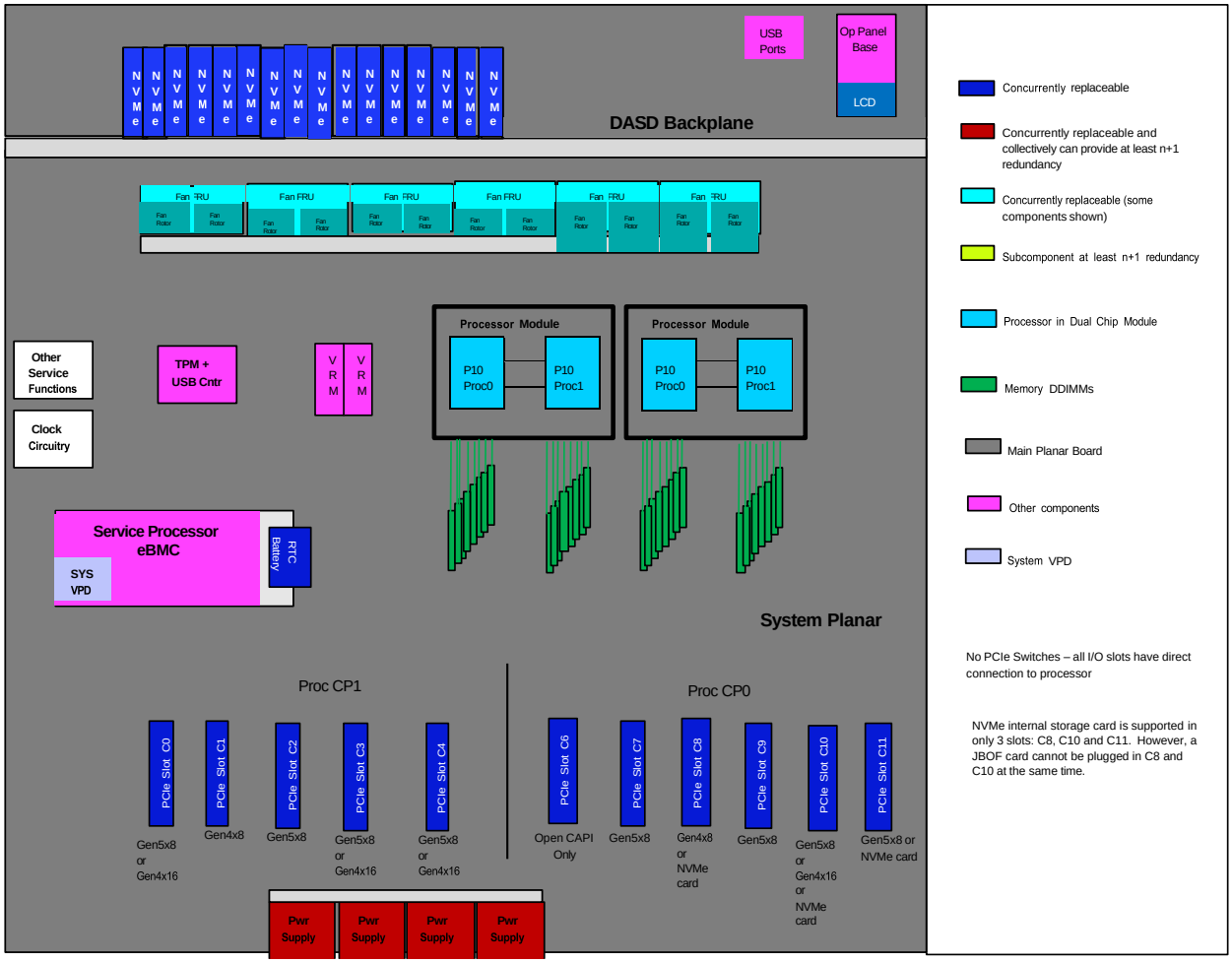
The Power S1014, S1022 and S1024 are the Power10 Scaled out servers. These systems are offered in the 1S and 2S DCM configuration with 2U and 4U CEC drawer options.

System Structure

There are multiple scale out system models (MTMs) supported. For brevity, this document focuses on the largest configuration of the scale out servers.

The simplified illustration, in Figure 10, depicts the 2S DCM with 4U CEC drawer. Similar to the S9xx, there is infrastructure redundancy in the power supplies and fans. In addition, these components can be concurrently maintained along with the Op Panel LCD, internal NVMe drives and IO adapters.

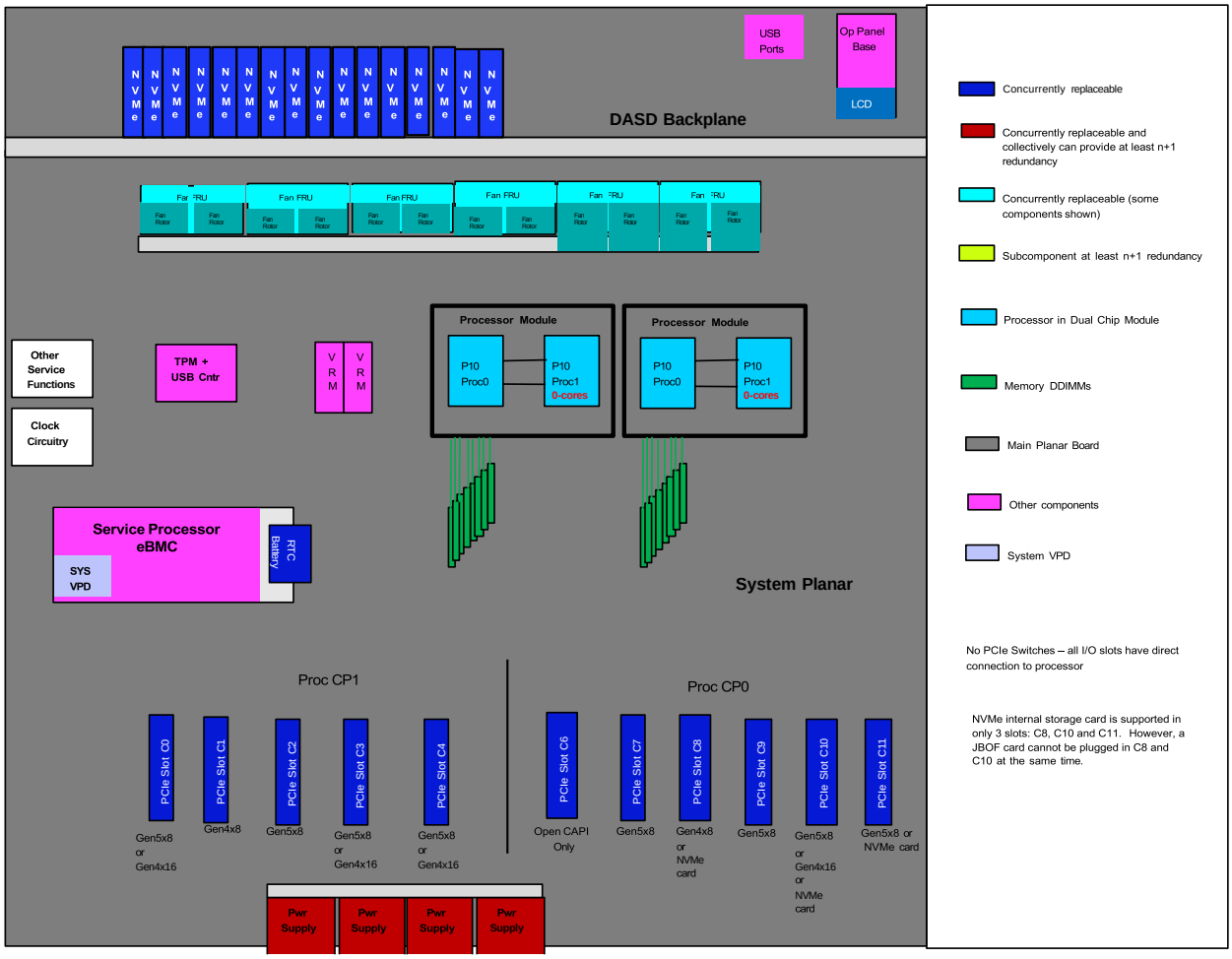
Figure 10: Power S1024 Structure Simplified View



As depicted in the simplified illustration in Figure 11, there is another variation of the Power10 processor module. This option contains 1 P10 chip with processor cores and OMI memory channels and another P10 chip whose primary purpose is to provide PCIe connections to I/O devices. This P10 entry Single Chip Module (eSCM) processor configuration gives customers the option to purchase cost reduced solution without losing any I/O adapter slots. A point of note is if the primary processor chip of the eSCM is nonfunctional and garded by the firmware, the

associated I/O only chip will be deconfigured as well and all the attached I/O slots will be unavailable.

Figure 11: Power S1024 Structure Simplified View With eSCM



Power S1014, S1022, S1024 Processor and Memory RAS

Other than the differences between the eSCM, DCM and SCM configurations outlined above, the processor RAS for these systems will be similar to enterprise systems. Refer to the E1050 processor RAS section and the general Power10 RAS comparison tables for more details.

The memory on these systems is very different from that of the S9xx systems. These scale out systems now support Active Memory Mirroring for the Hypervisor, which was not available in the S9xx servers. While an S9xx uses ISDIMMs, these systems support a 2U DDIMM which is a DDIMM form-factor that fits in 2U systems. The 2U DDIMM is a reduced cost and RAS

version of the 4U DDIMM. The 4U DDIMMs RAS characteristics were discussed in some details in the E1080 section of this document.

Figure 12: S9xx ISDIMM vs 2U DDIMM RAS Comparison

	Power9 S9xx Memory	Power10 S1014, S1022, S1024 2U Memory	RAS impact
DIMM Form Factor	Direct attached ISDIMMs	2U	• **P10 2U DDIMM**: Single FRU or fewer components to replace • **S9xx DIMM**: Separate FRU used for the ISDIMMs
Symbol Correction	Dual Symbol correction	Dual Symbol correction	• **P10 2U DDIMM**: Data pin fail (1 symbol) lining up with single cell fail on another DRAM is still correctable • **S9xx DIMM**: Data pin fail (1 symbol) lining up with single cell fail on another DRAM is still correctable

X4 Chip Kill	Single DRAM chipkill correction, but No spare DRAM	Single DRAM chipkill correction, but No spare DRAM	• **P10 2U DDIMM**
			  ○ 1st chip kill fixed with ECC
			  ○ 2nd serial chip kill is uncorrectable
			• **S9xx DIMM**
			  ○ 1st chip kill fixed with ECC
			  ○ 2nd serial chip kill is uncorrectable

DRAM Row Repair	No	Dynamic	• **P10 2U DDIMM**: Detect, fix, and restore at runtime without system outage • **S9xx ISDIMM**: Neither Static nor Dynamic row repair supported
Voltage Regulation Redundancy	No	No	• **P10 2U DDIMM:** voltage regulation and associated components are single point of failure • **S9xx DIMM:** voltage regulation and associated components are single point of failure

NOTE: A memory ECC code is defined by how many bits or symbols (group of bits) it can correct. The P10 DDIMM memory buffer ECC code organizes the data into 8-bit symbols and each symbol contains the data from one DRAM DQ over 8 DDR beats.

Power S1014, S1022, S1024 I/O RAS

The S1024 provides 10 general purpose PCIe slots that allows for hot plugging of IO adapters. Some of these PCIe slots operate at Gen5 speeds, while a few are limited to Gen4 speeds. As shown in Figure 13, some of the PCIe slots support NVMe JBOF (Just a Bunch Of Flash) cable card and I/O expansion drawer cable card.

Both DCMs must be installed to have connection to all 10 PCIe I/O slots. If only one DCM is installed, the 1S processor module drives 5 general purpose PCIe slots (C7 to C11).

Figure 13: Power S1024 I/O Slot Assignments

Slot	Type	From	Supports
C0	x8 G5/x16 G4	CP1 = DCM1/C1	PCIe Adapters, Cable card for I/O expansion
C1	x8 G4	CP1 = DCM1/C1	PCIe Adapters
C2	x8 G5	CP1 = DCM1/C1	PCIe Adapters, Cable card for I/O expansion
C3	x8 G5/x16 G4	CP1 = DCM1/C0	PCIe Adapters, Cable card for I/O expansion
C4	x8 G5/x16 G4	CP1 = DCM1/C0	PCIe Adapters, Cable card for I/O expansion
C7	x8 G5	CP0 = DCM0/C1	PCIe Adapters, Cable card for I/O expansion
C8	x8 G4	CP0 = DCM0/C1	PCIe Adapters, NVMe JBOF card
C9	x8 G5	CP0 = DCM0/C1	PCIe Adapters, Cable card for I/O expansion
C10	x8 G5/x16 G4	CP0 = DCM0/C0	PCIe Adapters, Cable card for I/O expansion, NVMe JBOF
C11	x8 G5	CP0 = DCM0/C0	PCIe Adapters, Cable card for I/O expansion, NVMe JBOF

DASD Options

These systems provide up to 16 internal NVMe drives at Gen4 speeds. The NVMe drives are connected to the processor via a plug-in PCIe NVMe JBOF (Just a Bunch Of Flash) card. Up to 2 JBOF cards can be populated in the S1024 and S1014 servers, with each JBOF card attached to an 8-pack NVMe backplane. The S1022 NVMe backplane only supports the 4-pack, which provides up to 8 NVMe drive per system. The JBOF cards are operational in PCIe slots C8, C10 and C11 only. However, the C8 and C10 slots cannot both be populated with JBOF cards simultaneously. As depicted in Figure 13 above, all 3 slots are connected to DCM0 which means a 1S system can have all the internal NVMe drives available. While the NVMe drives are concurrently maintainable, a JBOF card is not. Unlike the S9xx, these systems have no internal SAS drives. An external drawer can be used to provide SAS drives.

Power S1014, S1022, S1024 Service Processor

These systems use the same eBMC service processor as the E1050 servers. Refer to the Power E1050 Service Processor section of this document for more information.

IBM Power S1012

In May of 2024, IBM announced another Power10 processor-based system, The IBM Power S1012.

The S1012 is described as an edge-level server, designed for both edge computing and core business workloads. The edge-level server is a 1-socket half-wide system available in a 2U rack-mounted or tower chassis form factor.

While the system benefits from incorporating a Power10 processor, an eBMC and many firmware components shared in common with other systems, the system design, positioning and form-factor is different from others in the Power10 family. Consequently, there are a number of RAS characteristics that are different from the other Power10 servers described in this whitepaper.

A brief highlight of some of the key differences follows.

Rather than using 2U DDIMMs, the system supports DDR4 Industry Standard RDIMMs (ISDIMMS). The processor connects to the ISDIMMS through separate memory buffers incorporated on the system planar.

The S1012 does not support Active Memory Mirroring of the Hypervisor and dynamic memory deallocation for predictive memory failure. The PCIe I/O Adapters are not concurrently maintainable and external I/O drawers are not supported. Live Partition Mobility (LPM) is now supported for the Power S1012 with post General Availability (GA) firmware.

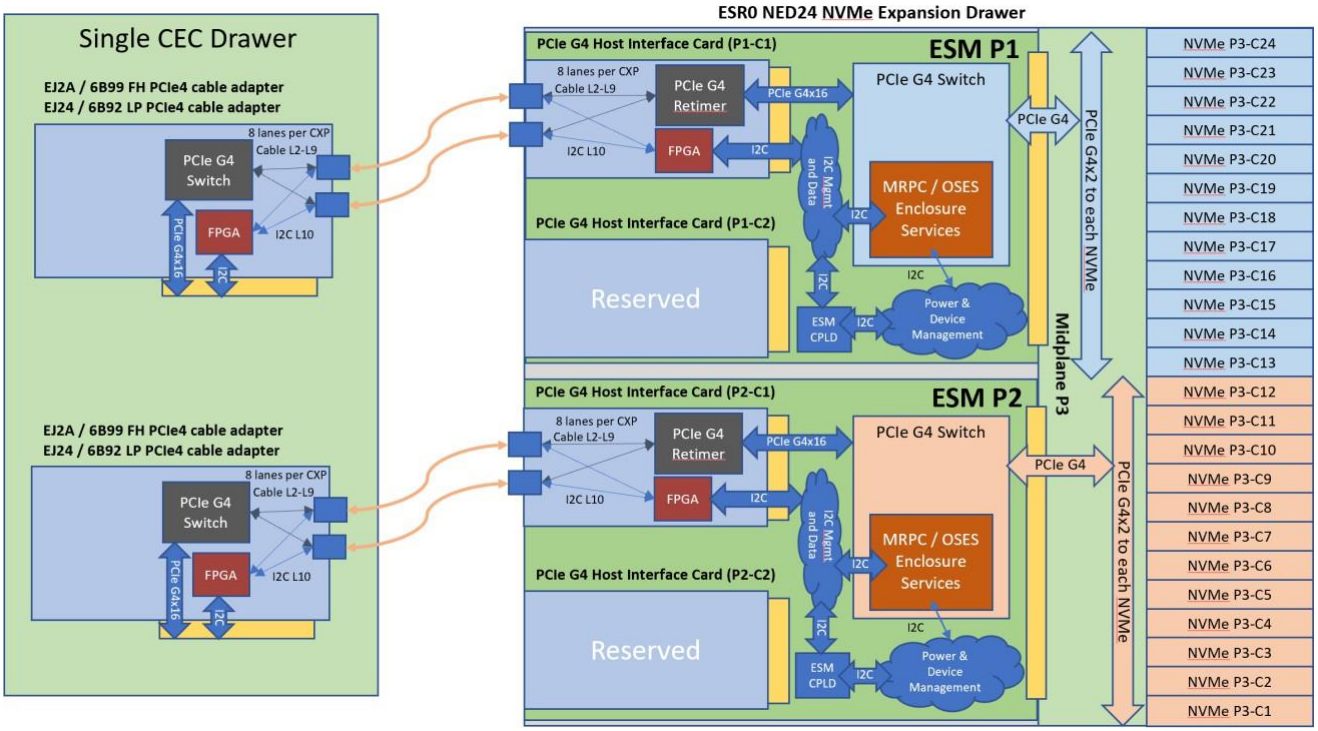
The system does support redundancy in system fans and power supplies as well as concurrent maintenance. The four NVMe drives are accessible for hot-plugging (when allowed by the operating system.) The 4 PCIe slots do not support concurrently maintainable or hot-plugging of adapters.

Due to limited number of components in the system, the use of fault indicators for failing part indication is more limiting compared to the scale-out systems.

NVMe Expansion Drawer (NED24)

The PCIe Gen4 NVMe Expansion Drawer can be used in systems to increase storage capacity. The NED24 NVMe expansion drawer provides extra NVMe U.2 drives. It is connected to system units by two expansion drawer cable pairs. The NED24 NVMe expansion drawer can hold up to 24 small form factor (SFF) NVMe U.2 drives. Depending on system model two or more can be attached for capacity or redundancy.

Figure 14: The NED24 I/O Subsystem Structure



The PCIe x16 uplink to the system cable cards is carried over a pair of x16 cables. Connectivity is fault tolerant of either cable of the pair failing and can be repaired independently. The management of the drawer is provided by two Enclosure Services Modules (ESM) with either being capable of managing the drawer.

For serviceability the drawer has front and back service indicators and IBM blue colored touchpoints. These operate according to the IBM Power indicator design for repair and maintenance operations. On the front of the Drawer is a display panel with a Green power LED, a Blue identify LED and an Amber fault LED.

Each of the NVMe U.2 drives can be concurrently repaired and can be configured using the same high availability options such as RAID or mirroring. Internally NED24 has redundant power supplies and over provisioned with 6 fans in a N+1 configuration.

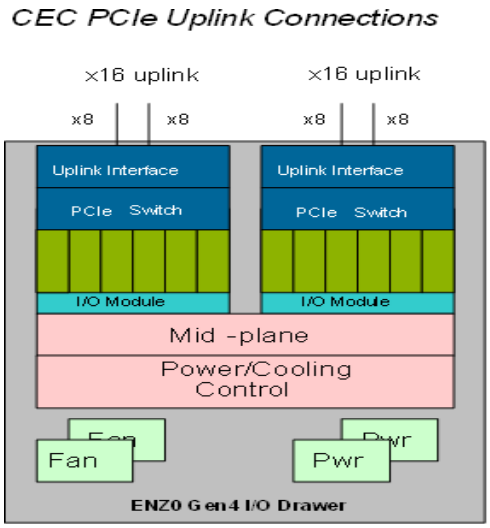
Figure 15: NVMe Expansion Drawer RAS Feature Matrix

Faulty Component	Impact of Failure	Impact of Repair	Prerequisites
NVMe Device in slot	Loss of function of the NVMe Device	NVMe device can be repaired while rest of the NED24 continues to operate	OS device mirroring is employed.
Enclosure Services Manager (ESM)	Loss of access to all the NVMe devices through the ESM unless in a multi path configuration	Associated ESM must be taken down for repair; rest of the NED24 can remain active.	Systems with an HMC
Other Failure of PCIe4 cable adapter card in System or ESM	Loss of access to all the NVMe devices connected to the ESM unless in a multi path configuration	Associated ESM must be taken down for repair; rest of the NED24 can remain active.	Systems with an HMC
A PCIe Lane fault or other failure of a cable	System continues to run but the number of active lanes available to ESM will be reduced	Associated ESM must be taken down for repair; rest of the NED24 can remain active.	Systems with an HMC
One Power Supply	NED24 continues to run with remaining PSU	Concurrently repairable	None
One Fan	NED24 continues to run with remaining fan	Concurrently repairable	None
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)	Systems with an HMC

ENZ0 PCIe Gen4 I/O Expansion Drawer

ENZ0 PCIe Gen4 I/O expansion drawer is used to increase PCIe adapter capacity. Each I/O expansion drawer provides up to 12 additional PCIe Gen4 slots, 8 x16 and 4 x8 widths. The maximum potential throughput for each I/O expansion drawer is 64GB/s in each direction concurrently. Actual realized throughput depends on application and overall system configuration.

Figure 16: ENZ0 PCIe Gen4 I/O Expansion Drawer



The I/O expansion drawer is attached using a converter card called a PCIe x16 to CXP Converter Card that plugs into certain capable PCIe slots of the CEC. The converter card connects a pair of CXP type copper or active optical cables that are offered in a variety of lengths to the I/O expansion drawer. CXP is a type of cable and connector technology used to carry wide width links such as PCIe. These cards have been designed to improve signal integrity and error isolation. These cards now contain a PCI-express switch that provides much better fault isolation by breaking the link to the I/O expansion drawer up into two links. Because each link is shorter with fewer components an error can be better isolated to a subset of failing component. This aids in the repair on a failure.

Each I/O expansion drawer contains up to two pluggable 6 slot fanout modules. Each can be installed and repaired independently. Each uses two x8 cables for its PCIe x16 width uplink. This implementation provides tolerance from a failure in either cable. The failure of one cable reduces the I/O bandwidth and lane width in half without a functional outage. Most client applications will see little or no performance degradation when the link recovers from errors in this fashion.

Additional RAS features allow for power and cooling redundancy. Dual power supplies support N+1 redundancy to allow for concurrent replacement or single power source loss. A voltage regulator module (VRM) associated with each PCIe4 6-slot fanout module includes built-in N+1

DC phase redundancy. The loss of a phase will not cause an outage to the fanout module. A single Air Moving Device (fan) can be replaced concurrently with operations.

Each PCIe4 6-slot fanout module is connected to the system using two copper or active optical cables. Each cable provides 8 PCIe lanes and a sideband channel. The 16 PCIe lanes span both cables making the two cables one uplink. The sideband channels of each cable are redundant and have built-in dynamic failover and fault recovery. Sideband channels provide functions such as service management power control, fan speed, thermal, voltage and current monitoring.

The table below summarizes the impact of failures of each component.

Figure 17: ENZ0 PCIe Gen4 I/O Expansion Drawer RAS Features

Faulty Component	Impact of Failure	Impact of Repair	Prerequisites
PCIe adapter in a PCIe slot	Loss of function of the PCIe adapter	PCIe adapter can be repaired while rest of the system continues to operate.	Multipathing PCIe adapter redundancy, where implemented, can be used to prevent application outages.
Fault on a PCIe lane or cable.	Link lane width and I/O bandwidth reduction using remaining PCIe lanes.	Repair can be deferred. Associated fanout module must be taken down for repair; rest of the system can remain active.	Concurrent repair requires attachment to HMC managed systems
Failure of PCIe x16 to CXP Converter Card	Loss of access to all the adapters of the connected fanout Module.	Associated fanout module must be taken down for repair; rest of the system can remain operational.	Concurrent repair requires attachment to HMC managed systems
Single Fan	System continues to run with remaining fans	Concurrently repairable	
Single Power Supply	System and drawer continue to operate with remaining power supplies.	Concurrently repairable	
Fanout Module Voltage Regulator Module (VRM)	System and drawer continue to run for a phase failure transition to n mode. Other faults will impact all the adapters in the module.	Associated fanout module must be taken down for repair; rest of the system can remain operational.	Concurrent repair requires attachment to HMC managed systems
Chassis Management Card (CMC)	No Impact to running system, but once powered off, the I/O expansion drawer cannot be re-integrated until CMC is repaired.	Drawer must be powered off to repair. Loss of use of all I/O in the drawer.	Concurrent repair requires attachment to HMC managed systems
Midplane	Depending on source of failure, may take down entire I/O expansion drawer.	I/O expansion drawer must be powered off to repair. Loss of use of all I/O in the I/O expansion drawer; rest of the system can remain active	Concurrent repair requires attachment to HMC managed systems

Section 2: General RAS Philosophy and Architecture

Philosophy

In the previous section three different classes of servers were described with different levels of RAS capabilities for Power10 processor-based systems. While each server had specific attributes, the section highlighted several common attributes.
This section will outline some of the design philosophies and characteristics that influenced the design. The following section will detail more specifically how the RAS design was carried out in each sub-system of the server.

Integrate System Design

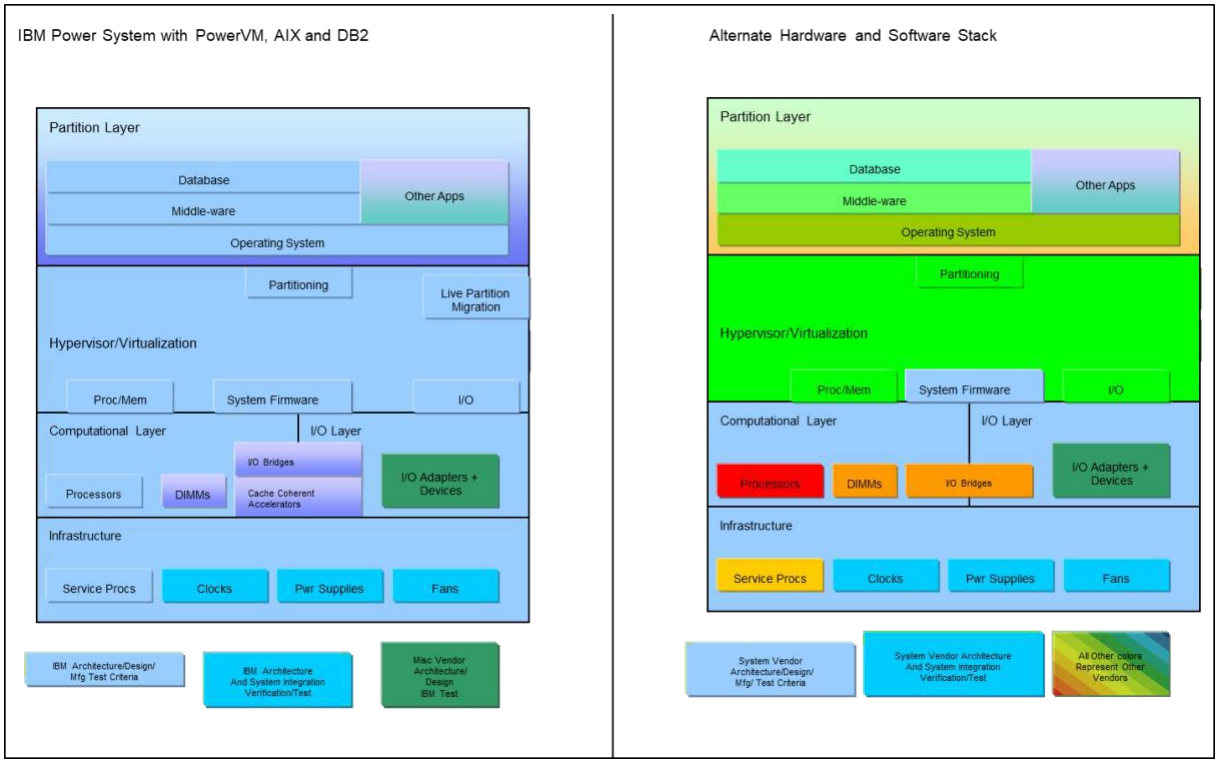
The systems discussed use a processor architected and designed by IBM. IBM systems contain other hardware content also designed by IBM including memory buffer components, service processors and so forth.

Additional components not designed or manufactured by IBM are chosen and specified by IBM to meet system requirements. These are procured for use by IBM using a rigorous procurement process intended to deliver 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. IBM's PowerVM hypervisor is also IBM designed and supported.

In addition, IBM offers two operating systems developed by IBM: AIX and IBMi. 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.

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 18: IBM Enterprise System Stacks (Power8 Design Point)



The figure above indicates how IBM design and influence may flow through the different layers of a representative enterprise system as compared to other designs that might not have the same level of control. Where IBM provides the primary design and manufacturing test criteria, IBM can be responsible for integrating all 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.

Incorporate Experience

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.

The detailed knowledge of previous system performance has a major influence on future systems design. This knowledge lets IBM invest in improving the discovered limitations of previous generations. Beyond that, it also shows the value of existing RAS features. This knowledge justifies investing in what is important and allows for adjustment to the design when certain techniques are shown to be no longer of much importance in later technologies or where other mechanisms can be used to achieve the same ends with less hardware overhead.

Architect for Error Reporting and Fault Isolation

It is not feasible to detect or isolate every possible fault or combination of faults that a server might experience, though it is important to invest in error detection and build a coherent architecture for how errors are reported and faults isolated. The sub-section on processor and memory error detection and fault isolation details the IBM Power approach for these system elements.

It should be pointed out error detection may seem like a well understood and universal hardware design goal. However, it is not always the goal of every computer sub-system design. Hypothetically, for instance, graphics processing units (GPU s) whose primary purpose is rendering graphics in non-critical applications may have options for turning off certain error checking (such as ECC in memory) to allow for better performance. The expectation in such case is that there are applications where a single dropped pixel on a screen is of no real importance, and a solid fault is only an issue if it is noticed.

In general, I/O adapters may also have less hardware error detection capability where they can rely on a software protocol to detect and recover from faults when such protocols are used.

Leverage Technology and Design for Soft Error Management

In a very real sense errors detected can take several forms. The most obvious is a functional fault in the hardware – a silicon defect, or a worn component that over time has failed.
Another kind of failure is what is broadly classified as a soft error. 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.

Data cells in caches and memory, for example, may have a bit-value temporarily upset by an external event such as caused by a cosmic ray generated particle. Logic in processors cores can also be subject to soft errors where a latch may also flip due to a particle strike or similar event. Busses transmitting data may experience soft errors due to clock drift or electronic noise.
The susceptibility to soft errors in a processor or memory subsystem is very much dependent on the design and technology used in these devices. This should be the first line of defense. Choosing latches that are less susceptible to upsets due to cosmic ray events was discussed extensively in previous whitepapers for example.

Methods for interleaving data so that two adjacent bits in array flipping won't cause undetected multi-bit flips in a data word is another design technique that might be used.
Ultimately when data is critical, detecting soft error events that occur needs to be done immediately, in-line to avoid relying on bad data because periodic diagnostics is insufficient to catch an intermittent problem before damage is done.

The simplest approach to detecting many soft error events may simply be having parity protection on data which can detect a single bit flip. When such simple single bit error detection is deployed, however, the impact of a single bit upset is bad data. Discovering bad data without being able to correct it will result in termination of an application, or even a system so long as data correctness is important.

To prevent such a soft error from having a system impact it is necessary not simply to detect a bit flip, but also to correct. This requires more hardware than simple parity. It has become common

now to deploy a bit correcting error correction code (ECC) in caches that can contain modified data. Because such flips can occur in more than just caches, however, such ECC codes are widely deployed in Power10 processors in critical areas on busses, caches and so forth. Protecting a processor from more than just data errors requires more than just ECC checking and correction. CRC checking with a retry capability is used on a number of busses, for example. When a fault is noticed maximum protection is achieved when not only is a fault noticed but noticed 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.

Deploy Strategic Spare Capacity to Self-Heal Hardware

Techniques that protect against soft errors are of limited protection against solid faults due to a real hardware failure. A single bit error in a cache, for example can be continually corrected by most ECC codes that allow double-bit detection and single bit correction.
However, if a solid fault is continually being corrected, the second fault that occurs will typically cause data that is not correctable. This would result in the need to terminate, at least, whatever is using the data.

In many system designs, when a solid fault occurs in something like a processor cache, the management software on the system (the hypervisor or OS) may be signaled to migrate the failing hardware off the system.

This is called predictive deallocation. Successful predictive deallocation may allow for the system to continue to operate without an outage. To restore full capacity to the system, however, the failed component still needs to be replaced, resulting in a service action.

Within Power, the general philosophy is to go beyond simple predictive deallocation by incorporating strategic sparing or micro-level deallocation of components so that when a hard failure that impacts only a portion of the sub-system occurs, full error detection capabilities can be restored without the need to replace the part that failed.

Examples include a spare data lane on a bus, a spare bit-line in a cache, having caches split up into multiple small sections that can be deallocated, or a spare DRAM module on a DIMM. With the general category of self-healing can be the use of spares. Redundancy can also be deployed to avoid outages. The concepts are related but there are differences between redundancy and use of spares in the Power approach.

Redundant Definition

Redundancy is generally a means of continuing operation in the presence of certain faults by providing more components/capacity than is needed to avoid outages but where a service action will be taken to replace the failed component after a fault.

Sometimes redundant components are not actively in use unless a failure occurs. For example, a processor may only actively use one clock source at a time even when redundant clock sources are provided.

In contrast, fans and power supplies are typically all active in a system. If a system is said to have “n+1” fan redundancy, for example, all “n+1” fans will normally be active in a system

absence a failure. If a fan fails occurs, the system will run with “n” fans. In cases where there are fans or power supply failures, power and thermal management code may compensate by increasing fan speed or making other adjustments according to operating conditions per power management mode and power/thermal management policy.

Spare Definition

A spare component is similar in nature though when a “spare is successfully used” the system can continue to operate without the need to replace the component.
As an example, for voltage regulator output modules, if five output phases are needed to maintain the power needed at the given voltage level, seven could be deployed initially. It would take the failure of three phases to cause an outage.

If on the first phase failure, the system continues to operate, and no call out is made for repair, the first failing phase would be considered spare. After the failure (spare is said to be used), the VRM could experience another phase failure with no outage. This maintains the required n+1 redundancy. Should a second phase fail, a “redundant” phase would then have been said to fail and a call-out for repair would be made.

Focus on OS Independence

Because IBM Power has long been designed to support multiple operating systems, the hardware RAS design is intended to allow the hardware to take care of the hardware largely independent of any operating system involvement in the error detection or fault isolation (excluding I/O adapters and devices for the moment.)

To a significant degree this error handling is contained within the processor hardware itself. However, service diagnostics firmware, depending on the error, may aid in the recovery. When fully virtualized, specific OS involvement in such tasks as migrating off a predictively failed component can also be performed transparent to the OS.

The PowerVM hypervisor is capable of creating logical partitions with virtualized processor and memory resources. When these resources are virtualized by the hypervisor, the hypervisor has the capability of deallocating fractional resources from each partition when necessary to remove a component such a processor core or logical memory block (LMB).
When an I/O device is directly under the control of the OS, the error handling of the device is the device driver responsibility. However, I/O can be virtualized through the VIOS offering meaning that I/O redundancy can be achieved independent of the OS.

Build System Level RAS Rather Than Just Processor and Memory RAS

IBM builds 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 achieve application availability. The failure of a fan, a power supply, a voltage regulator, or I/O adapter might be more likely than the failure of a processor module designed and manufactured for reliability.

Scale-out servers will maintain redundancy in the power and cooling subsystems to avoid system outages due to common failures in those areas. Concurrent repair of these components is also provided.

For the Enterprise system, a higher investment in redundancy is made. The Power E1080, for example is designed from the start with the expectation that the system must be largely shielded from the failure of these other components causing persistent system unavailability; incorporating substantial redundancy within the service infrastructure (such as redundant service processors, redundant processor boot images, and so forth.) An emphasis on the reliability of components themselves are highly reliable and meant to last.
This level of RAS investment extends beyond what is expected and often what is seen in other server designs. For example, at the system level such selective sparing may include such elements as a spare voltage phase within a voltage regulator module.

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 is not relegated to an operating system.

Further, the error detection and fault isolation capabilities are intended to enable retry and other mechanisms to avoid outages due to soft errors and to allow for use of self-healing features. This requires a very detailed approach to error detection.

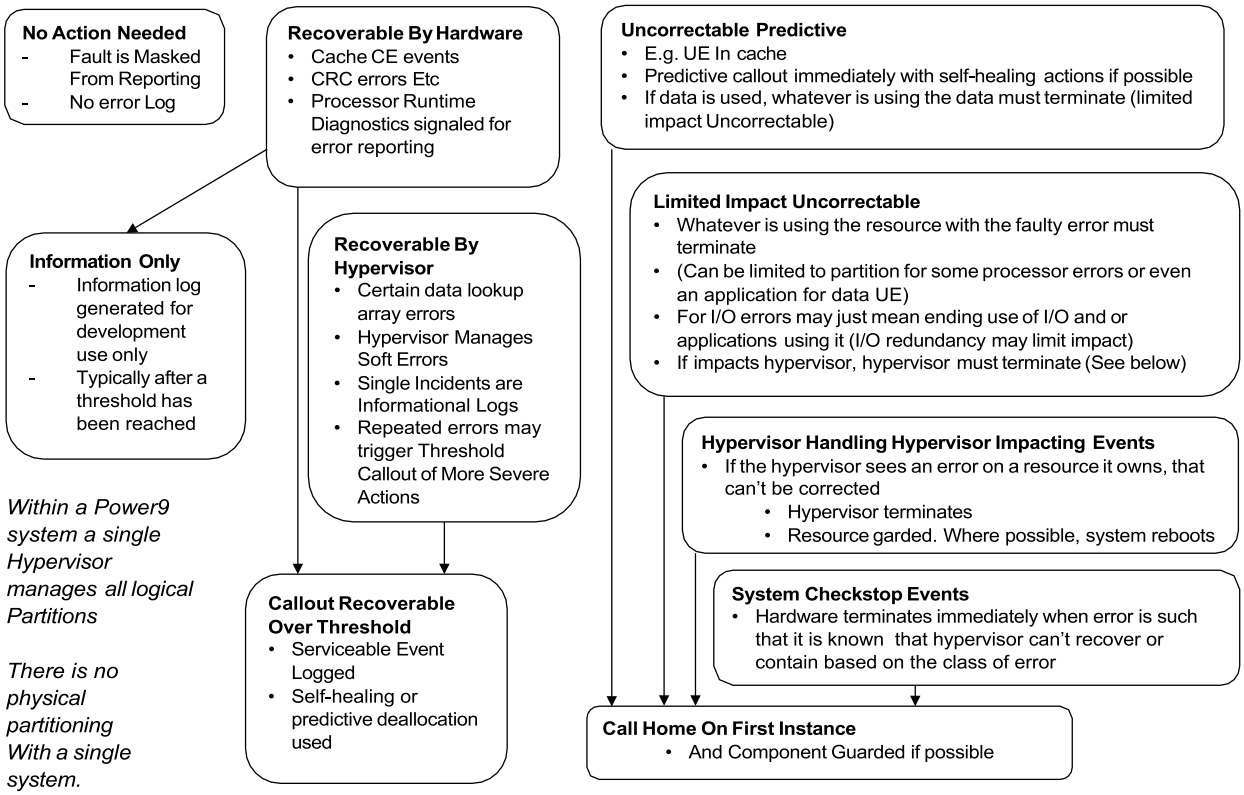
This approach is beneficial to systems as they are deployed by end-users, but also has benefits in the design, simulation, and manufacturing test of systems as well.

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 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 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.

The error detection and fault isolation capabilities maximize the ability to categorize errors by severity and handle faults with the minimum impact possible. Such a structure for error handling can be abstractly illustrated by the figure below and is discussed throughout the rest of this section.

Figure 19: Handled Errors Classified by Severity and Service Actions Required



First Failure Data Capture Analysis (Processor Runtime Diagnostics)

The first failure data capture design is meant for catching faults during run-time as they occur and provide isolation, mitigations and other actions at the time of detection. To provide full isolation, and take appropriate service actions, code is run which accesses the fault information available. This code is known as Processor Runtime Diagnostics (PRD). PRD is different from RPD which is described in the next section.

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

In a Power7 generation server, PRD and other service code was all run within the dedicated service processor used to manage these systems. The dedicated service processor was in charge of the IPL process used to initialize the hardware and bring the servers up to the state where the hypervisor could begin to run. The dedicated service processor was also in charge, as previously described, to run the PRD code during normal operation.

In the rare event that a system outage resulted from a problem, the service processor had 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 additional debug and trace arrays that could be used to further understand the root cause of faults.

Even if a severe fault caused system termination, this access provided the means for the service processor to determine the root cause of the problem, deallocate the failed components, and allow the system to restart with failed components removed from the configuration.
Power8 gained a Self-Boot-Engine which allowed processors to run code and boot using the Power8 processors themselves to speed up the process and provide for parallelization across multiple nodes in the high-end system. During the initial stages of the IPL process, the boot engine code itself handled certain errors and the PRD code ran as an application at later stages if necessary.

In Power9 the design has changed further so that during normal operation the PRD code itself runs in a special hypervisor-partition under the management of the hypervisor. This has the advantage of continuing to allow the PRD code to run even if the service processor is non-functional (important in non-redundant environments). Should the code running fail, the hypervisor can restart the partition (reloading and restarting the PRD.)

The system service processors are also still monitored at run-time by the hypervisor code and can report errors if the service processors are not communicating. The Power10 based servers maintain the Power9 design.

Periodic Processor Exerciser/Diagnostics Program (Runtime Processor Diagnostics)
As hyperscalers consolidate more and more computing power some have expressed a concern about the need to run periodic diagnostics for processors in addition to what is provided by the built-in error detection and fault isolation capabilities. Such diagnostics might better isolate faults that would otherwise be detected but poorly isolated. They may also detect faults that otherwise were not caught by hardware checkers. In such a case, since the diagnostics are only run periodically, incorrect operation or results could have occurred before the periodic diagnostics find an issue.

Google published an article talking about software and hardware defenses against the impact of faults (https://support.google.com/cloud/answer/10759085?hl=en). The document also describes a set of periodic diagnostics exercises for CPUs called, CPU Check, “focusing primarily on the x86_64 architecture.”

Intel has released a diagnostic tool (https://www.intel.com/content/www/us/en/support/articles/000005567/processors.html) that can be downloaded and run as an application in an operating system to, among other functions, “Verify the functionality of all the cores of Intel® Processor.”

Power customers may ask whether IBM has similar periodic diagnostic capabilities.

The possible exposure to failed hardware and the detectability can depend on the method of manufacture and test as well as the underlying micro-architecture of the processor including the capabilities of any First Failure Data Capture architecture.

While First Failure Data Capture (catching faults at the source where fault isolation and mitigation can best be performed) is considered vital to the Power approach to RAS, around the mid-2000s, a limited set of diagnostics tests, focused primarily on floating point calculations, were available to run periodically on Power servers. These tests were intended to complement the FFDC approach.

While the processor architecture, FFDC and recovery techniques were refined over several subsequent generations, these added tests were still available in each generation through Power9. In Power9 these diagnostics can be enabled by the customers choosing options to execute the tests on a periodic basis, staggering the execution over a period of time, or be set to run at a particular time.

During the development and functional test of Power servers, and during the manufacturing of the servers, IBM runs a diagnostics exerciser suite capable of exercising hardware cores.

To more directly align any periodic processor core diagnostics with what is used during test and manufacturing, starting with the FW 1050 release for Power10 servers, Power servers implemented a new run-time processor diagnostic (RPD) capability for periodic testing of the functionality of processor cores. RPD uses functions from the previously mentioned test code base with expanded coverage over the legacy periodic diagnostic capability aligned with the Power10 processor capabilities. These tests run within a special partition under PowerVM control and allow for testing processor cores in small time-slices during system operation.

Unlike the approach other vendors may recommend where such diagnostics would have to be downloaded and executed in each partition, the Power10 processor diagnostics running under the PowerVM can test cores without customer intervention beyond selecting the desired method of operation, and regardless of how the system is partitioned. There is no need to have a copy running in each VM of the system.

Using run-time processor diagnostics to exercise cores can result in faults handled by the First Failure Data Capture capabilities of the processor design which will be treated according to the previously outlined FFDC processor error handling approach.

Should the run-time diagnostics itself detect and isolate a core fault, an SRC will be posted as a service actionable event. (B7005400 for Power9 and B7005194 for Power10). This will be handled as an unrecoverable processor core error.

Further explanation for the SRCs: *Background diagnostics have detected an uncorrectable error on a processor core. The failing processor core has been marked defective. Prior to detecting this failure, the failing processor may have been assigned to a running partition and it is recommended to run the appropriate application data consistency checks. Please contact your next level of support with any questions.*

The server operator can control if and how the runtime diagnostics are used. To take advantage of these periodic processor diagnostics in Power9 and Power10 with the FW 1050 release, or later, customers should verify that the function is enabled with the option desired.

For Power10, RPD runs in a special service partition created by the PowerVM hypervisor. In the background, using limited cycles, PowerVM can assign processor cores one at a time to the service partition to assess for faults while customer workload is running.
From the service processor menu, customers can select one of the following RPD operation modes:

1. Run Now – Begin testing processors and end once all processors have been tested.
2. Staggered – Periodically test all the processors and begin again when finished.
3. Scheduled – Periodically test processors only during a scheduled time window each day.
4. Disabled – Do not run

Generally, Power10 servers shipping with RPD will have RPD enabled by default in the **Staggered** mode. However, Because RPD requires system resources, servers with less than 128G of installed memory will default to **Disabled** mode. Depending on the system configuration and the RPD mode selected, the testing may take multiple days to exercise all processor cores.

PowerVM Partitioning and Outages

The PowerVM hypervisor provides logical partitioning allowing multiple instances of an operating system to run in a server. At a high level, a server with PowerVM runs with a single copy of the PowerVM hypervisor regardless of the number of CEC nodes or partitions.
The PowerVM hypervisor uses a distributed model across the server's processor and memory resources. In this approach some individual hypervisor code threads may be started and terminated as needed when a hypervisor resource is required. Ideally when a partition needs to access a hypervisor resource, a core that was running the partition will then run a hypervisor code thread.

Certain faults that might impact a PowerVM thread will result a system outage if they should occur. This can be by PowerVM termination or by the hardware determining that for, PowerVM integrity, the system will need to checkstop.

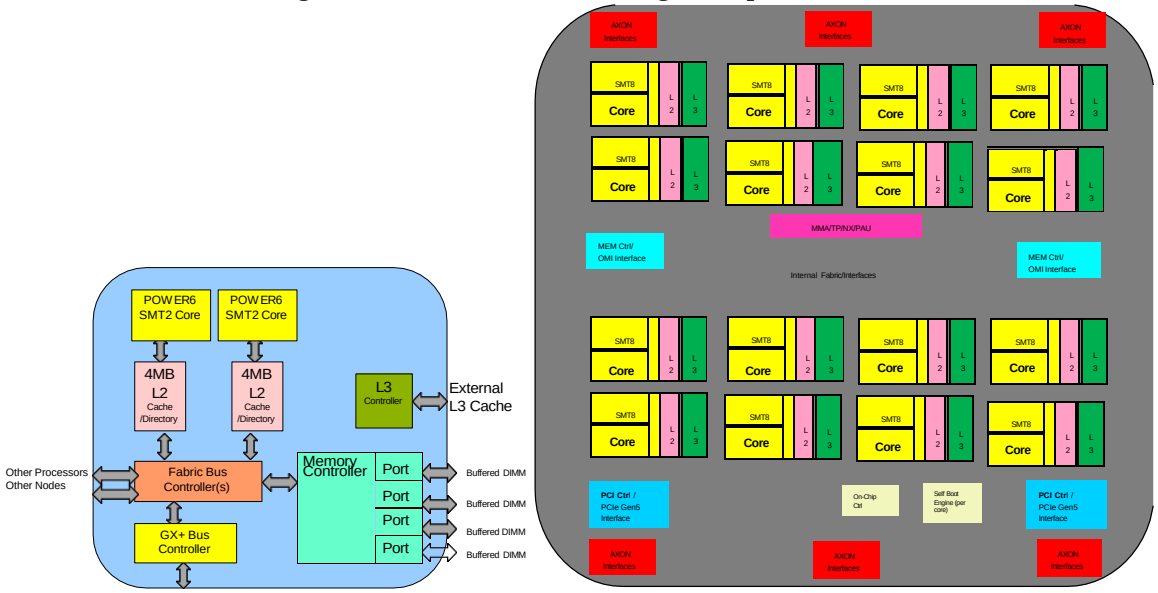
The design cannot be viewed as a physical partitioning approach. There are not multiple independent PowerVM hypervisors running in a system. If for fault isolation purposes, it is desired to have multiple instances of PowerVM and hence multiple physical partitions, separate systems can be used.

Not designing a single system to have multiple physical partitions reflects the belief that the best availability can be achieved if each physical partition runs in completely separate hardware. Otherwise, there is a concern that when resources for separate physical partitions come together in a system, even with redundancy, there can be some common access point and the possibility of a "common mode" fault that impacts the entire system.

Processor RAS Details.

It is worth noting that the functions integrated within a processor has changed much over time. One point of comparison could be a Power6 processor compared to the current Power10.

Figure 20: Power6 Processor Design Compared to Power10



The illustration above shows a rough view of the Power10 scale-up processor design leveraging SMT8 cores (a maximum of 16 cores shown.)
The Power10 design is certainly more capable. There is a maximum of 16 SMT8 cores compared to 2 SMT2 cores on Power6. The core designs architecturally have advanced in function as well. The number of memory controllers has doubled, and the memory controller design is also different.

The addition of system-wide functions such as the Matrix Multiply Accelerator (MMA), NX accelerators and the CAPI interfaces provide functions just not present in the hardware of the Power6 system.

The Power10 design is also much more integrated. The L3 cache is internal, and the I/O processor host bridge is integrated into the processor. The thermal management is now conducted internally using the on-chip controllers.

There are reliability advantages to the integration. It should be noted that when a failure does occur its more likely to be a processor module at fault compared to previous generations with less integration. Some benefits of this integration approach is that Power10 based systems can leverage the advanced error isolation and recovery mechanisms of the processor to improve system uptime.

Hierarchy of Error Avoidance and Handling

In general, there is a hierarchy of techniques used in POWER™ processors to avoid or mitigate the impact of hardware errors. At the lowest level in the hierarchy is the design for error detection and fault isolation, the technology employed, specifically as relates to reducing the

instances of soft error not only through error correction, but in the selection of devices within the processor IC.

Because of the extensive amount of functionality beyond just processor cores and caches, listing all the RAS capabilities of the various system elements would require considerable detail. In somewhat broader strokes, the tables below discuss the major capabilities in each area.

Figure 21: Error Handling Methods Highlights

Cache Error Handling	The L2 and L3 caches in the processor use an ECC code that allows for single bit correction. During operation, when a persistent correctable error occurs in these caches, the system has the capability of purging the data in the cache (writing to another level of the hierarchy) and deleting it (meaning subsequent cache operations won't use the cache line during operation.) This is an example of "self-healing" that avoids taking a planned outage for a correctable error. The L1 cache (usually thought of as part of the processor Core element) checks for single bit errors. But instead of using an error correcting code, intermittent L1 cache errors can be corrected using data from elsewhere in the cache hierarchy. A portion of an L1 cache can be disabled (set delete) to avoid outages due to persistent hard errors. If too many errors are observed across multiple sets the core using the L1 can be predictively deallocated. In addition to the system caches as described above, there are the cache directories which provide indexing to the caches. These also have single bit error correction. Uncorrectable directory errors will typically result in system checkstops as discussed below.
Other ECC Checking	Beyond these elements, single bit correcting ECC is used in multiple areas of the processor as the standard means of protecting data against single bit upsets (beyond the reliability design features previously mentioned.) This includes a number of the internal busses where data is passed between units.
Special Uncorrectable Error Handling	Where there is ECC in the path for data and an uncorrectable error is encountered, the desire to prevent reliance on bad data means that whatever is using the data will need to be terminated. In simpler system designs, that would mean termination of something, at least the owner of the data, as soon as the uncorrectable error is encountered. Power has long had the concept of marking the data with a special ECC code and watching for when and if the data is going to be "consumed" by the owner (if the data is ever used.) At that point whatever the data owner is can be terminated. This can be a single application if a processor running under AIX is consuming user data. For kernel data, the OS kernel may be terminated. If PowerVM attempts to use the data in a critical error, then PowerVM will terminate. One additional advantage of the special error correction code is that the hardware is able to distinguish between a fresh ECC error when data is transferred and one that has been passed along. This allows the correct component, the one originating the fault, to be called out as the component to be replaced.
Bus CRC	As previously mentioned, ECC is used internally in various data-paths as data is transmitted between units. Externally to the processor, however, high speed data busses can be susceptible to the occasional multiple bit errors due to the nature of the bus design. A cyclic redundancy check code can be used to determine if there are errors within an entire packet of data. A transient fault can be corrected with a retry. CRC checking is now done for the processor fabric bus interfaces sending data between processors.

Lane Repair	By itself CRC checking has the advantage of handling multiple bit errors. For persistent problems it is in certain ways inferior to bit correcting ECC if a single persistent error cannot be corrected by retry. The Power10 busses between processors also have a spare data lane that can be substituted for a failing one to "self-heal" the single bit errors.
Split Data Bus (½ bandwidth mode)	In busses going between processor nodes (between CEC drawers in a Power E1080 server) if there is a persistent error that is confined to the data on a single cable (and the bus is split between cables) **Power10** can reduce the bandwidth of the bus and send data across just the remaining cable. This allows for correction of a more systematic fault across the bus. In addition, the bus between processors within a node or single CEC drawer system is also capable of a split-bus or ½ bandwidth mode. Full support of SMP bus features may depend on firmware level.
Processor Instruction Retry	Within the logic and storage elements commonly known as the "core" there can be faults other than errors in the L1 mentioned above. Some of these faults may also be transient in nature. The error detection logic within the core elements is designed extensively enough to determine the root cause of a number of such errors and catch the fault before an instruction using the facility is completed. In such cases processor instruction retry capabilities within the processor core may simply retry the failed operation and continue. Since this feature was introduced in the Power6, processor field data has shown many instances where this feature alone had prevented serious outages associated with intermittent faults. In Power10 faults that are solid in nature and where retry does not solve the problem will be handled as described further down in this table.
Predictive Deallocation	Because of the amount of self-healing incorporated in these systems as well as the extensive error recovery it is rare that an entire processor core needs to be predictively deallocated due to a persistent recoverable error. If such cases do occur, PowerVM can invoke a process for deallocating the failing processor dynamically at run-time (save in the rare case that the OS application doesn't allow for it.)
Core Checkstops	On scale-up systems where the use of many partitions may be common, if a fault cannot be contained by any of the previous features defined in the hierarchy, it may be still beneficial to contain an outage to just the partitions running the threads when the uncorrectable fault occurred. The Core Checkstop feature (sometimes called a **core-contained checkstop**) may do this in these systems supporting the feature for faults that can be contained that way. This allows the outage associated with the fault to be contained to just the partition(s) using the core at the time of the failure if the core is only used at the time by partition(s) other than in hypervisor mode. It should be noted that in such an environment system performance may be impacted for the remaining cores after the core contained checkstop. Note: A core checkstop signaled for a fault occurring on a core with any thread running a hypervisor instruction will typically result in hypervisor termination as described below. The possibility of encountering hypervisor running on a thread may have a correlation to the number of threads in use.
PowerVM Handled errors	There are other faults that are handled by the hypervisor that relate to errors in architected features – for example handling a SLB multi-hit error.
PCIe Hub Behavior and Enhanced Error Handling	Each Power10 processor has PCIe Host Bridges (PHB) called PCIe hubs which generate the various PCIe Gen4/5 busses used in the system. The hub is capable of "freezing" operations when certain faults occur, and in certain cases can retry and recover from the fault condition. This hub freeze behavior prevents faulty data from being written out through the I/O hub system and prevents reliance on faulty data within the processor complex when certain errors are detected.

	Along with this hub freeze behavior is what has long been termed as Enhanced Error Handling for I/O. This capability signals device drivers when various PCIe bus related faults occur. Device drivers may also attempt to restart the adapter after such faults (EEH recovery) depending on the adapter, device driver and application. A clock error in the PCIe clocking can be signaled and managed using EEH in any system that incorporates redundant PCIe clocks with dynamic clock failover enabled.
Hypervisor Termination and System Checkstops	If a fault is contained to a core, but the core is running PowerVM code, PowerVM may terminate to maintain the integrity of the computation of the partitions running under it. In addition, each processor fault checked is thoroughly reviewed during design. If it is known in advance that a particular fault can impact the hypervisor or if there is a fault in a processor facility whose impact cannot be localized by other techniques, the hardware will generate a platform checkstop to cause system termination. This by design allows for the most efficient recovery from such errors and invokes the full error determination capabilities of the service processor.
Processor Safe-mode	Failures of certain processor wide facilities such as power and thermal management code running on the on-chip controller (OCC) and the self-boot-engine (SBE) used during run-time for out of band service processor functions can occur. To protect system function, such faults can result in the system running in a “safe mode” which allows processing to continue with reduced performance in the face of errors where the ability to access power and thermal performance or error data is not available.
Persistent Guarding of Failed Elements	Should a fault in a processor core become serious enough that the component needs to be repaired (persistent correctable error with all self-healing capabilities exhausted or an unrecoverable error), the system will remember that a repair has been called for and not re-use the processor on subsequent system reboots, until repair. This functionality may be extended to other processor elements and to entire processor modules when relying on that element in the future means risking another outage. In a multi-node system, deconfiguration of a processor node for fabric bus consistency reasons results in deconfiguration of a node.

As systems design continues to mature the RAS features may be adjusted based on the needs of the newer designs as well as field experience; therefore, this list differs from the Power8 design.

For example, in Power7+, an L3 cache column repair mechanism was implemented to be used in addition to the ability to delete rows of a cache line.

This feature was carried forward into the Power8 design, but the field experience in the Power8 technology did not show the benefit based on how faults that might implement multiple rows surfaced. For Power9 and Power10 enterprise systems, given the size of the caches, the number of lines that were deleted were extended instead of continuing to carry column repair as a separate procedure.
Going forward, the number of rows that need to be deleted is adjusted for each generation based on technology.

Likewise, in Power6 a feature known as alternate processor recovery was introduced. The feature had the purpose of handling certain faults in the Power6 core. The faults handled were limited to certain faults where the fault was discovered before an instruction was committed and

the state of the failing core could be extracted. The feature, in those cases, allowed the failing workload to be dynamically retried on an alternate processor core. In cases where no alternate processor core was available, some number of partitions would need to be terminated, but the user was allowed to prioritize which partitions would be terminated to keep the most important partition running.

The mixture of which parts could be handled by this kind of design, and the method of delivering the design itself, changed from generation to generation. In Power8, field observations showed that essentially, and in part due to advances in other error handling mechanisms, the faults where alternate processor recovery was required and viable had become exceedingly rare. Therefore, the function is not being carried over in Power9 or Power10.

Processor Module Design and Test

While IBM Power servers are equipped to deal with certain 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.

The design anticipates that not only should errors be checked, but also that the detection and reporting methods associated with each error type also need to be verified. When there is an error class that can be checked, and some sort of recovery or repair action initiated, there may be a hardware method to “inject” an error that will test 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 should be a way to at least simulate the report that an error has been detected and test response to such error reports.

Under IBM control, assembled systems are also tested and a certain amount of system “burn-in” may 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.

In that mode the criteria may be more severe as to what constitutes a failure. A single fault, even if it’s recoverable, might be enough to fail a part during this system manufacturing process.

Memory RAS Details

Memory Hierarchy of Error Avoidance and Handling

In general, there is a hierarchy of techniques used in POWER™ processors to avoid or mitigate the impact of hardware errors. At the lowest level in the hierarchy is the design for error detection and fault isolation, the technology employed, specifically as relates to reducing the instances of soft error not only through error correction, but in the selection of devices within the processor IC.

Because of the extensive amount of functionality beyond just processor cores and caches, listing all the RAS capabilities of the various system elements would require considerable detail. In somewhat broader strokes, the tables below discuss the major capabilities in each area.

Figure 22: Error Handling Methods Highlights

DRAM Error Handling	RAS coverage for various DRAM faults: • Single cell fail: Correctable by ECC. • Fault contained to one DQ: Correctable by ECC, diagnosed and repaired dynamically by FW using symbol mark, or one of 2 spare DRAMs (spares available on 4U DDIMM only). • Fault contained to two DQs: Correctable by ECC, diagnosed and repaired dynamically by FW using symbol mark, or 2 spare DRAMs (spares available on 4U DDIMM only). • Fault contained to single DRAM row: Correctable by ECC, diagnosed and repaired dynamically by FW using row repair. • Fault contained to single DRAM, multiple rows: Correctable by ECC, diagnosed and repaired dynamically by FW using chip mark, or one of 2 spare DRAMs (spares available on 4U DDIMM only)
Memory Scrubbing	The memory controller periodically scrub the DRAMs for soft errors. This HW accelerated scrubbing involves reading the memory, checking the data and correcting the data if an ECC fault is detected. If a UE is detected, corrective action can safely be taken in maintenance mode.
Bus CRC	Like all high speed data busses, the OMI can be susceptible to the occasional multiple bit errors. A cyclic redundancy check code is used to determine if there are errors within an entire packet of data. If a transient CRC fault is detected, the packet is retried and the operation continues.
Split Memory Channel (½ bandwidth mode)	If there is a persistent CRC error that is confined to the data on half the OMI channel (1 to 4 lanes), the bandwidth of the bus is reduced and all the memory channel traffic is sent across just 4 good lanes.
Predictive Memory Deallocation	Refer to the **Dynamic Deallocation/Memory Substitution** section below for details.
Memory Channel Checkstops and Memory Mirroring	The memory controller communicating between the processor and the memory buffer has its own set of methods for containing errors or retrying operations. Some severe faults require that memory under a portion of the controller become inaccessible to prevent reliance on incorrect data. There are cases where the fault can be limited to just one memory channel. In these cases, the memory controller asserts what is known as a channel checkstop. In systems without hypervisor memory mirroring, a channel checkstop will usually result in a system outage. However, with hypervisor memory mirroring the hypervisor may continue to operate in the face of a memory channel checkstop. Refer to the **Hypervisor Memory Mirroring** section below for additional details.

Memory Safe-mode	Memory is throttled based on memory temperature readings in order to thermally protect the DDIMMs. If memory temperature readings are not available, then the memory will be throttled to safe-mode throttle values in order to thermally protect the DDIMMs.
Persistent Guarding of Failed Memory Elements	As with the processor, should a fault in a memory become serious enough that the component needs to be repaired (persistent correctable error with all self-healing capabilities exhausted or an unrecoverable error), the system will remember that a repair has been called for and not re-use the DDIMM on subsequent system reboots, until repair.

Memory RAS Beyond ECC

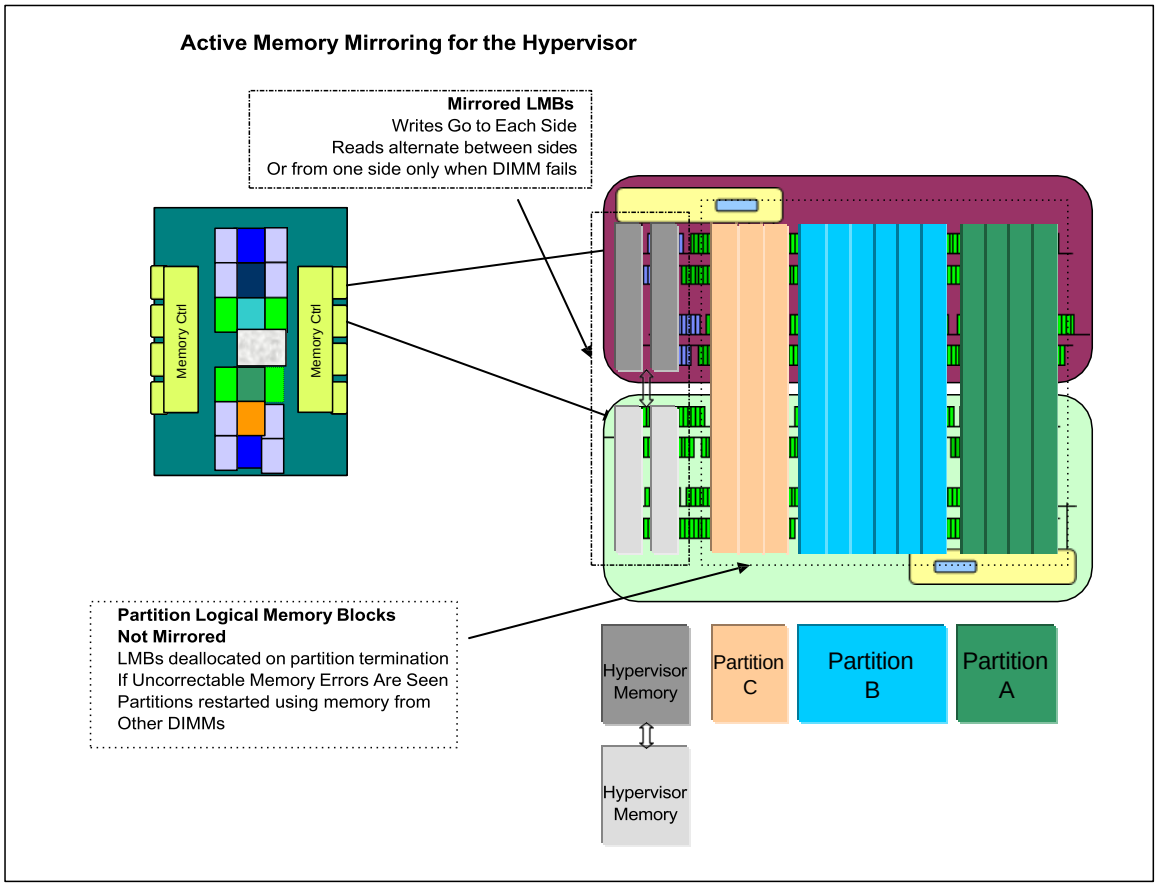
A truly robust memory design incorporates more RAS features than just the ECC protection of course. The ability to use hardware accelerated scrubbing to refresh memory that may have experienced soft errors is a given. The memory bus interface is also important.

Hypervisor Memory Mirroring

As an additional capability to memory sub-systems RAS, the ability to mirror memory can provide an additional protection against memory related outages.
In general, mirroring memory has a down-side. There is additional cost of the mirrored memory, mirroring memory can reduce memory capacity and may also have an impact on performance such as due to the need to write to two different memory locations whenever data is stored.

All Power10 processors provide a means to mirror the memory used in critical areas of the PowerVM hypervisor. This provides protection to the hypervisor memory so that it does not need to terminate due to just a fault on a DIMM that cause uncorrectable errors in the hypervisor. There are rare conditions when AMM of the hypervisor may not prevent system termination. For instance, there are times when mirroring will be disabled in response to a DIMM error which could lead to system termination if the second DIMM in mirrored pair were to experience an uncorrectable event.

Figure 23: Active Memory Mirroring for the Hypervisor



By selectively mirroring only the segments used by the hypervisor, this protection is provided without the need to mirror large amounts of memory.
It should be noted that the PowerVM design is a distributed one. PowerVM code can execute at times on each processor in a system and can reside in small amounts in memory anywhere in the system. Accordingly, the selective mirroring approach is fine grained enough not to require the hypervisor to sit in any particular memory DIMMs. This provides the function while not compromising the hypervisor performance as might be the case if the code had to reside remotely from the processors using a hypervisor service.

Dynamic Deallocation/Memory Substitution
For Enterprise Systems with custom DIMMs, the primary means of deallocating predictively bad memory in the system is leveraging the extensive set of spare DRAMs provided.
For errors impacting only a small amount of memory, PowerVM can deallocate a single page of memory through the OS for a single cell fault in memory.
For more severe errors, after spare memory is used, PowerVM can deallocate memory at logical memory block level. That may mean terminating partitions using memory after uncorrectable errors are encountered. For predictive errors, deallocation can be accompanied by substituting unallocated logical memory blocks for the deallocated blocks where there is enough unallocated memory. Where not enough unallocated memory, PowerVM may deallocate additional blocks once workload using the predictively failing memory frees up the memory.

It should be understood that typically, memory is interleaved across multiple DIMMs for maximum performance and when deallocating memory, the entire set of memory in an interleave

group has to be deallocated and, where available, substituted for the equivalent amount of spare memory.

Interleaving amounts may vary but it is possible that all of the DIMMs controlled by a processor module may be interleaved together. Providing fully spare memory would necessitate having another processor's worth of memory unallocated in such a case.

Where there is insufficient spare memory available, as memory is released within partitions, up to an entire processor's worth of memory may eventually become unavailable to the system pending repair of the failed DIMM.

On a reboot, the memory interleaving will be changed to just deconfigure the minimum number of DIMMs required to account for the failed DIMM while allowing the rest of the memory controlled by the processor to be used.

For instance, if 8 - 128GB DIMMs are attached to a processor socket they can be grouped as a single interleave unit. An unrecoverable error (UE), on one of the 8 DIMMs, will result in the other 7 – 128GB DIMMs being garded at system runtime. All LPARs that are allocated memory in the 1024GB memory group that contains the UE, will experience an outage. Upon a system reboot, the remaining 7 good memory DIMMs can be recovered and reallocated to LPARs, after being garded. A system reboot is required, because the size of the interleave group cannot be adjusted dynamically at runtime.

RAS beyond Processors and Memory

Introduction

Processor and memory differences aside, there are many design considerations and features that distinguish between scale-out and enterprise systems and between different enterprise systems as well. One of these is the amount of infrastructure redundancy provided.

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 as described below.

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 allow 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 not 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, depending on the design, the fact that the supply is “weak” 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; it may happen 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 that provide well 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. Therefore, understanding that excessive heat is one of the primary contributors to component “wear-out”, IBM requires that even components providing cooling to other components should be protected from excessive heat.

Common Mode Failures

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

One such category includes faults that are not detected in the primary source, code issues, and intrinsic limitations on fail-over capabilities.

Another kind of limitation may be called a common mode failure. For example, 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. Ultimately there will be someplace electrically where redundant components come together to provide a function, and failures there can cause outages.

Fault Detection/Isolation and Firmware and Other Limitations
The effectiveness of redundancy, especially when a failover is required, can depend also on when and how faults are detected and whether any firmware is required to properly implement a failover. It may not be possible to cover all events. Further, even when a fault is detected and failover is initiated, defects in the hardware or firmware design, including subtle timing windows, could impact whether the failover was successful.
The ability of an organization to inject errors and test failing modes, therefore, is essential in validating a redundant design strategy.

Power and Cooling Redundancy Details

Power Supply Redundancy
Power supplies in a system broadly refer to components that take utility power from the data center (typically alternating current, or AC power) and convert to DC power that is distributed throughout the system. Power supplies generally supply one DC voltage level (e.g., 12 volts) and voltage regulators may be used to supply different voltage levels required by various system voltages.

Power supply redundancy has two main goals. The first is to make sure that a system can continue to operate when a power supply fails. This is usually known as power supply redundancy. Conceptually if a system had four power supplies and could continue to run with one supply failed, that would be considered n+1 redundancy with 3 supplies needed for operation and 1 redundant supply.

The second goal is to allow for the data center to supply two sources of power into the system (typically using two power distribution units or PDUs). Should one of these sources fail the system should continue to operate.

In theory this could be achieved by feeding each power supply with two separate line cords, one from each PDU. Depending on the design and how one power source failed, however, there could be scenarios of power supply failure where operation cannot be maintained.
Alternative, multiple power supplies might be supplied. For example, a system may be designed with four power supplies, E1, E2, E3 and E4 with E1 and E2 designed to connect to one power distribution unit (PDU) and E3 and E4 to another PDU.

If the system can run one of the PDUs failing, even if it causes some performance degradation for some loads, it may be said to also have “line cord redundancy.”
If 4 power supplies are used in this fashion then when a PDU fails, the system will be running with two power supplies. It may be expected, therefore that this would provide 2+2 or n+2 redundancy. However, this need not always be the case. The system may be designed to operate when certain power supplies fail that are supplied to a single PDU, but not when one power supply under each PDU fails. Or it may be designed to run with two supplies but have performance degradation for certain configurations/loads.

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

As described above, power supplies may take alternating current (AC) from a data center power source, and then convert 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 can provide multiple different DC voltage levels to supply all the components in a system. Failing that, 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 work to maintain voltage levels within the 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 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 for the load it 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. Additional phases may also be provided for sparing purposes. The distinction between spare and redundant is that when a spare phase fails, the system continues to operate without the need to repair. After any spare phases fail, the failure of a redundant phase will require a service action. While it is important to provide redundancy to critical components within a device such as an adapter or DIMM may not be necessary if the devices themselves are redundant within the system.

There are other applications for voltage conversion or division that are not as power-demanding as the applications above: A regulator only used during IPL for initialization of a component for example, or a memory DIMM or riser which takes supplied voltage and further divides on card for purposes such as reference voltage or signal termination. Such uses are not included in the discussion of voltage regulation or voltage regulator modules discussed in the rest of this paper. Power10 1 and 2 socket systems as scale-out models do not provide redundant voltage phases. This is very typical of systems in the scale-out space and often in systems that are considered Enterprise.

The Power E1050 and E1080 provide an n+1 redundant phase for VRMs feeding the processor and VRMs for certain other components.

The Power E1080 server goes a step further and provides an additional spare phase for these elements.

Redundant Clocks

Vendors looking to design enterprise class servers may recognize the desirability of maintaining redundant processor clocks so that any hard failure of a single clock oscillator doesn't require a system to go down and stay down until repaired.

The Power9 scale-up processor is designed with redundant clock inputs. In Power8 servers a global clock source for all the processor components was required. Hence two global processor clock cards were provided in the system control drawer of the systems and a dynamic failover was provided.

In the Power E980 the design has been changed so that the main (run-time) processor core clock needs to stay synchronized within each CEC drawer (node) rather than the entire system. Hence the processor clock logic is now duplicated for each drawer in the system.
In this design the clock logic for the processor is now separate from a multi-function clock used for such components as PCIe bus. A fault on this multi-function clock where redundancy is provided can be handled through PCIe recovery.

As mentioned in the **IBM Power E1080 section**, in additional to the clocks being asynchronous across nodes, each processor reference clocks are asynchronous within a node as well. The E1080 reference clock sources are redundant and implemented significantly simpler logic design compared to the predecessor systems.

Service Processor and Boot Infrastructure

There is more function than just the service processor itself that can be considered part of the service processor infrastructure in a system.

Systems with a single service processor still have some service processor infrastructure that may be redundant (i.e., having two system VPD modules on a VPD card.) Still, the Power E1080 system provides the highest degree of service processor infrastructure redundancy among the systems being discussed.

In particular, it should be noted that for a system to boot or IPL, a system needs to have a healthy service processor, a functioning processor to boot from (using the internal self-boot engine) as well as functioning firmware.

In systems with a single service processor there is a single processor module and self-boot-engine that can be used for booting, and a single module used to store the associated firmware images.

In the Power E1080, each system has two service processors. In each node each service processor is connected to a different processor module. This redundancy allows the system to IPL for a single fault isolated to the service processor, the path from the service processor, or to the processor module. Other systems use a single eBMC service processor connected to a single processor module. Therefore, IPL may not be possible for certain faults in eBMC, it's path to the processor module, or the processor module itself.

Trusted Platform Module (TPM)

Each of the systems discussed in this paper also incorporates a Trusted Platform Module (TPM) for support of Secure Boot and related functions.[1] For redundancy purposes, the Power E1080 system ships with two TPMs per node. Other systems have a single TPM.
Within the active service processor ASMI interface, a "TPM required" policy can be enabled. A system or node will not boot if the system is in secure mode, TPM is required, and no functional TPM is found.

Concerning dual TPM, when a TPM is found to be faulty during initial IPL test, the second TPM module can be used. Once a TPM module is chosen for IPL, the other TPM module in the pair is disabled and cannot be further used. During run-time systems with multiple drawers may be able to take advantage of the multiple TPMs should one TPM fail. To understand the latest function, IBM documentation concerning the TPM use should be referenced.

I/O Subsystem and VIOS™

The descriptions for each system describe how internal I/O slots and external I/O drawers can be used for PCIe adapters.

In a PCIe 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.

In enterprise RAS designs, there is an expectation 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 and the SAN adapters would all 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 PCIe socket controlled by one PCIe processor, and the other in a slot controlled by another.

The software communicating with the SAN would take care of the 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.

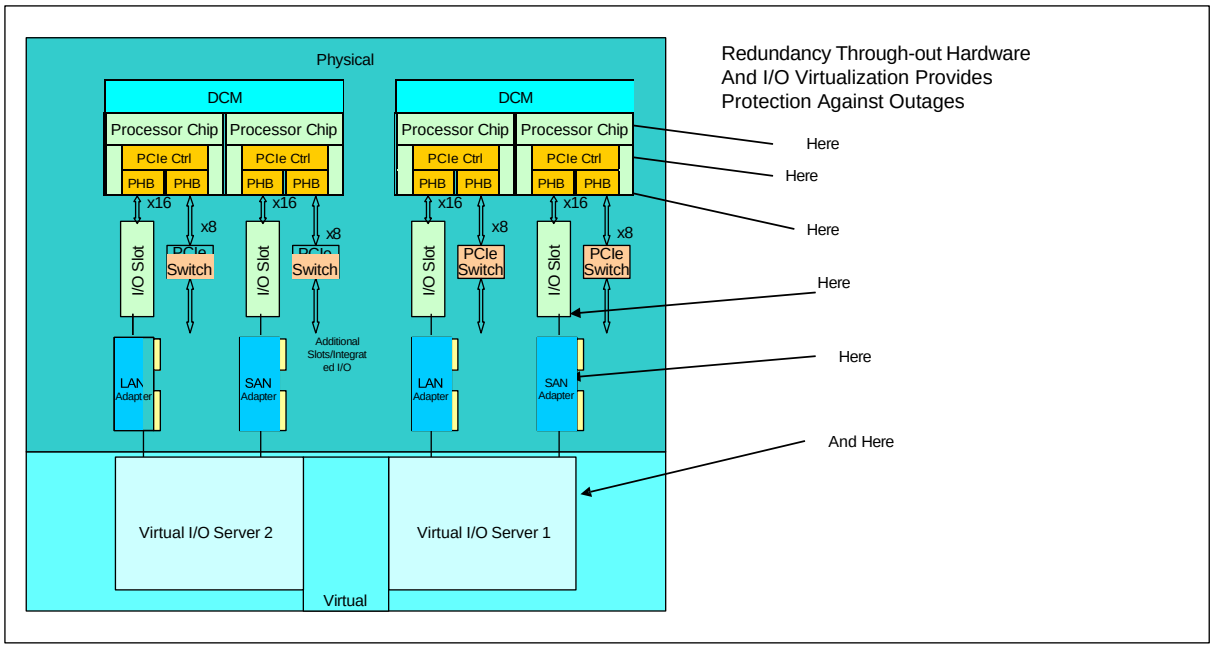
With the configuration when there is a fault impacting an adapter, the redundant adapter can take over. If there is a fault impacting the communication to a slot from a processor, the other processor would be used to communicate to the other I/O adapter.
The error handling throughout the I/O subsystem from processor PCIe controller to I/O adapter is intended so 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.

[1] https://www.ibm.com/docs/en/power10/9105-22A?topic=powervm-signatures-keys-in-secure-boot

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. This concept is illustrated below using a figure first published in the Power8 RAS whitepaper.

Figure 24: End-to-End I/O Redundancy

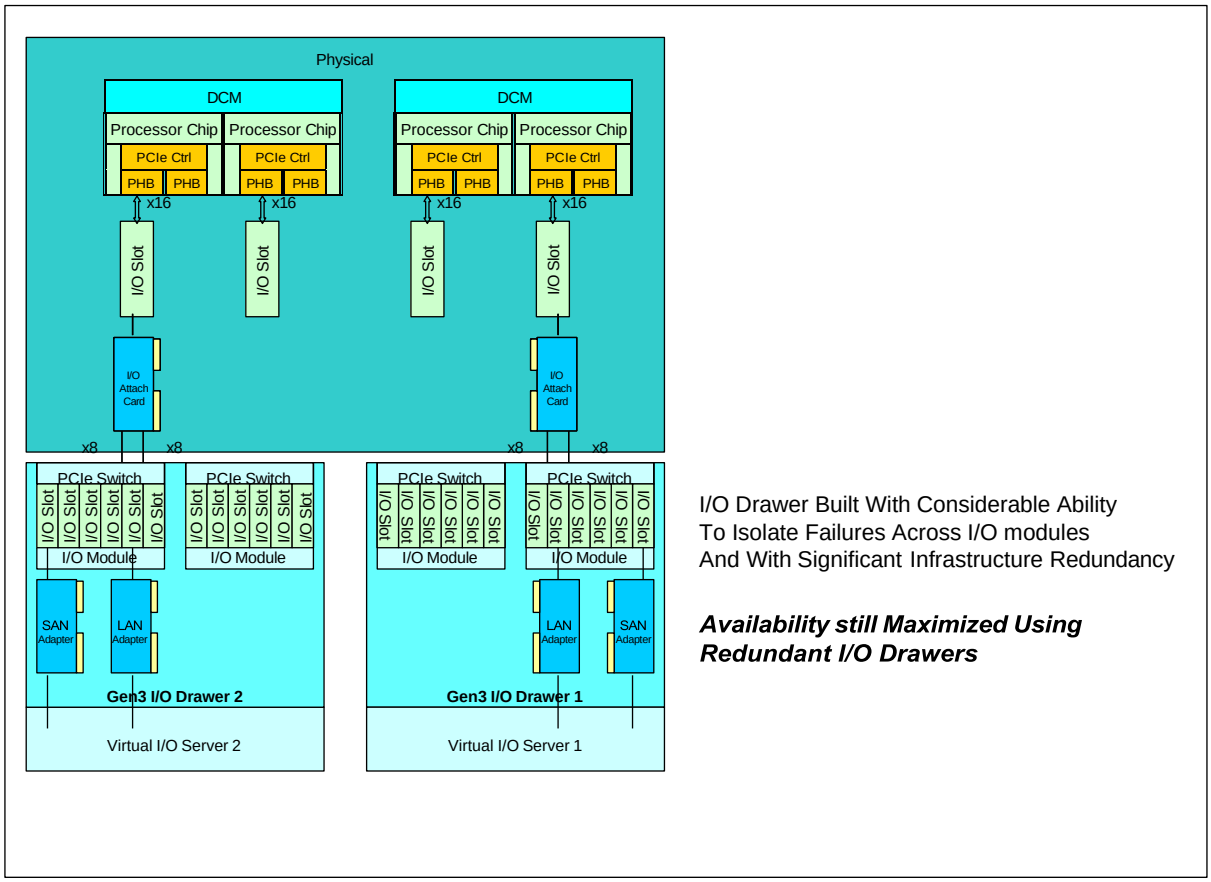


PCIe Gen4 Expansion Drawer Redundancy

As elsewhere described the optically connected PCIe Gen4 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.

For the highest level of redundancy, it is recommended 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 25: Maximum Availability with Attached I/O Drawers



Planned Outages

Unplanned outages of systems and applications are typically very disruptive to applications. This is certainly true of systems running standalone applications, 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.

Generally, systems are designed to 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, though exceptions can occur.

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 the entire system down.

As examples, Enterprise models support concurrently removeable and replaceable elements such as power supplies and fans. Most models also support concurrently removing and replacing I/O adapters according to the capabilities of the OS and applications.

Integrated Sparing
As previously mentioned, to reduce replacements for components that cannot be removed and replaced without taking down a system, design strategy includes the use of integrated spare components that can be substituted for failing ones.

Clustering and Cloud Support

PowerHA SystemMirror

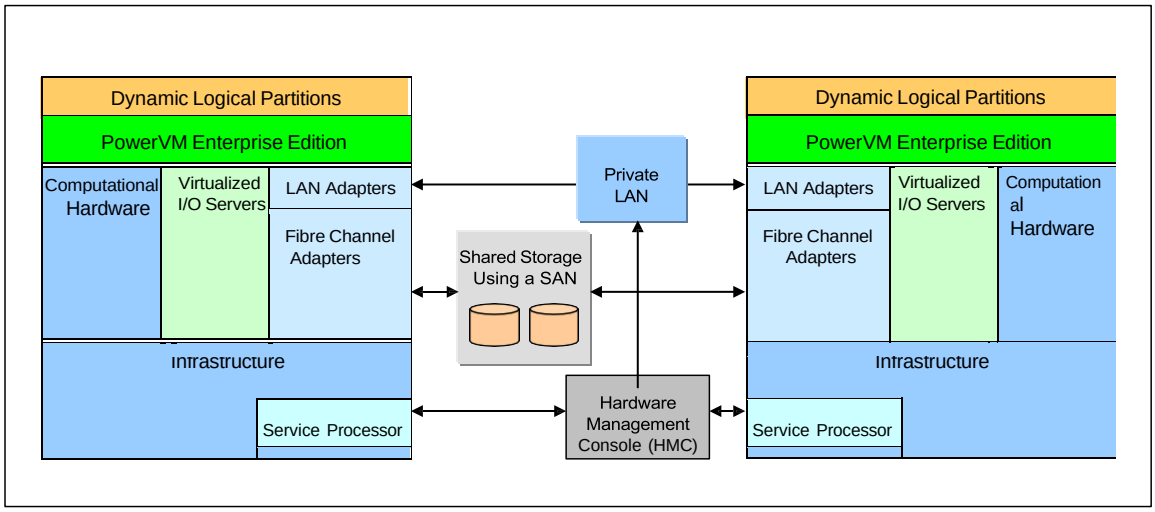
IBM Power running under PowerVM and AIX® and Linux support a spectrum of clustering solutions. These solutions are designed to meet requirements not only for application availability as regards to server outages, but also data center disaster management, reliable data backups and so forth. These offerings 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), a component of PowerVM Enterprise Edition, will be discussed here in particular with reference to its use in managing planned hardware outages.

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

Figure 26: LPM Minimum Configuration



In simplified terms, LPM typically works in an environment where all 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.

Thinking in terms of using LPM for hardware repairs, if all 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 detailed information on how LPM can be configured the following references may be useful: An IBM Redbook titled: **IBM PowerVM Virtualization Introduction and Configuration**[2] as well as the document **Live Partition Mobility**[3]

In general terms LPM requires that both the system containing a partition to be migrated and the system being migrated have a local LAN connection using a virtualized LAN adapter. In addition, LPM requires that all systems in the LPM cluster be attached to the same SAN. 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.

I/O Redundancy Configurations and VIOS
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 are therefore recommended. To accomplish this, Virtual I/O servers (VIOS) can be used.

VIOS as an offering for PowerVM virtualizes I/O adapters so that multiple partitions will be able to utilize the same physical adapter. VIOS can be configured with redundant I/O adapters so that the loss of an adapter does not result in a permanent loss of I/O to the partitions using the VIOS. 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.

[2] Mel Cordero, Lúcio Correia, Hai Lin, Vamshikrishna Thatikonda, Rodrigo Xavier, Sixth Edition published June 2013,
[3] IBM, 2018, ftp://ftp.software.ibm.com/systems/power/docs/hw/p9/p9hc3.pdf

Figure 27: I/O Infrastructure Redundancy

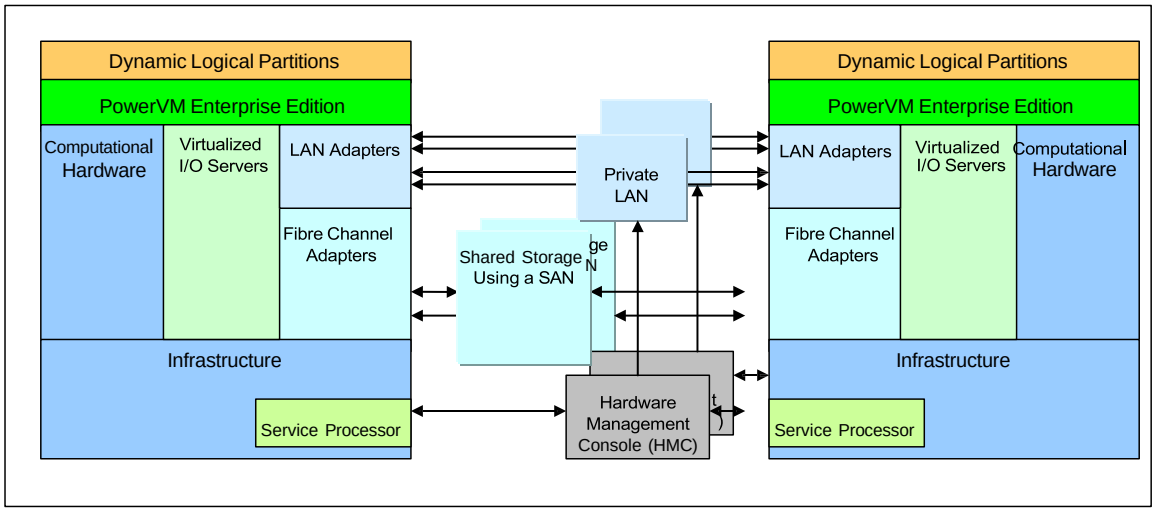
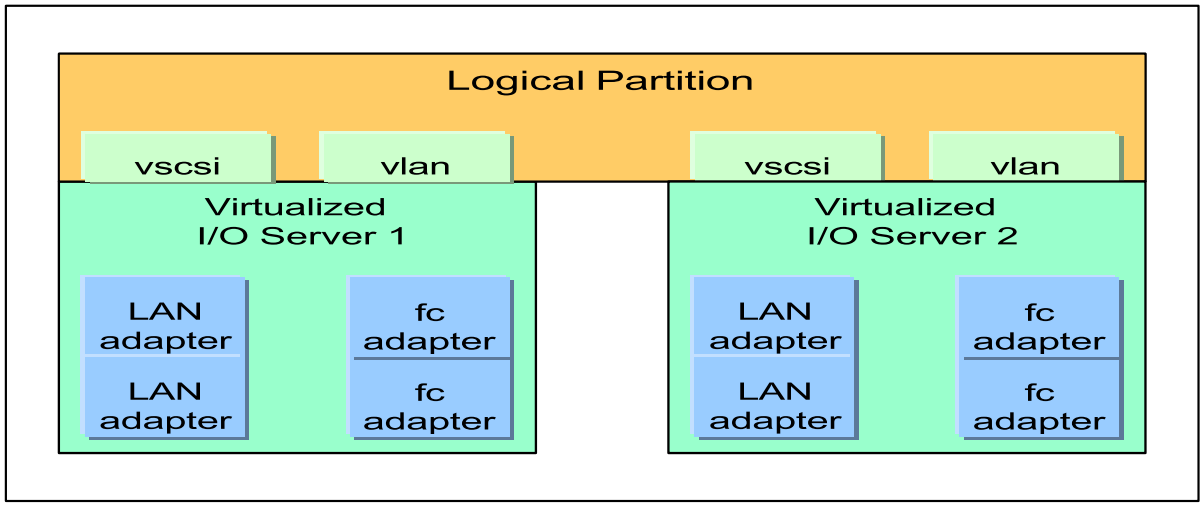


Figure generally illustrates multi-path considerations within an environment optimized for LPM. Within each server, this environment can be supported with a single VIOS. However, if a single VIOS is used and that VIOS terminates for any reason (hardware or software caused) then all the partitions using 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 28: 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 such as the internal NVMe drives provided for the Power E1080 and Power E1050

systems. For best availability, the rootvgs should use mirrored or other appropriate RAID drives with redundant access to the devices.

PowerVC and Simplified Remote Restart

PowerVC is an enterprise virtualization and cloud management offering from IBM that streamlines virtual machine deployment and operational management across servers. The IBM Cloud PowerVC Manager edition expands on this to provide self-service capabilities in a private cloud environment; IBM offers a Redbook that provides a detailed description of these capabilities. As of the time of this writing**: IBM PowerVC Version 1.3.2 Introduction and Configuration**[4] describes this offering in considerable detail.

Deploying virtual machines on systems with the RAS characteristics previously described will best leverage the RAS capabilities of the hardware in a PowerVC environment. Of interest in this availability discussion is that PowerVC provides a virtual machine remote restart capability, which provides a means of automatically restarting a VM on another server in certain scenarios (described below).

Systems with a Hardware Management Console (HMC) may also choose to leverage a simplified remote restart capability (SRR) using the HMC.

Error Detection in a Failover Environment

The conditions under which a failover is attempted It is important when talking about any sort of failover scenario. Some remote restart capabilities, for example, operate only after an error management system, e.g., an HMC reports that a partition is in an Error or Down State.
This alone might miss hardware faults that just terminate a single application or impact the resources that an application uses, without causing a partition outage. Operating system “hang” conditions would also not be detected.

In contrast, PowerHA leverages a heartbeat within a partition to determine when a partition has become unavailable. This allows for fail-over in cases where there is a software or other cause while a partition is not able to make forward progress even if an error is not recorded.
It is a consideration that for the highest level of application availability an application itself might want to leverage some sort of heartbeat mechanism to determine when an application is hung or unable to make forward progress.

[4] January 2017, International Technical Support Organization, Javier Bazan Lazcano and Martin Parrella

Section 4: Reliability and Availability in the Data Center

The R, A and S of RAS

Introduction
All of the previous sections in this document discussed server specific RAS features and options. This section looks at the more general concept of RAS as it applies to any system in the data center. The goal is to briefly define what RAS is and look at how reliability and availability are measured. It will then discuss how these measurements may be applied to different applications of scale-up and scale-out servers.

RAS Defined
Mathematically, reliability is defined in terms of infrequently something fails.
At a system level, availability is about how infrequently failures cause workload interruptions. The longer the interval between interruptions, the more available a system is.
Serviceability is all about how efficiently failures are identified and dealt with, and how application outages are minimized during repair.

Broadly speaking systems can be categorized as ”scale-up” or ”scale-out” depending on the impact to applications or workload of a system being unavailable.

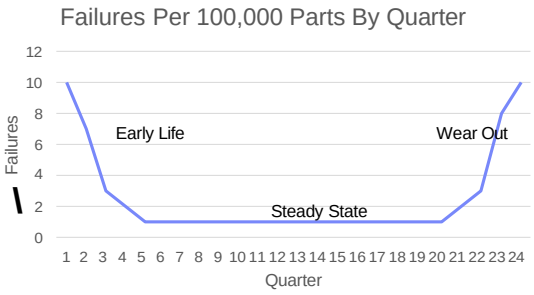
True scale-out environments typically spread workload among multiple systems so that the impact of a single system failing, even for a short period of time is minimal.
In scale-up systems, the impact of a server taking a fault, or even a portion of a server (e.g., an individual partition) is significant. Applications may be deployed in a clustered environment so that extended outages can in a certain sense be tolerated (e.g., using some sort of fail-over to another system) but even the amount of time it takes to detect the issue and fail-over to another device is deemed significant in a scale-up system.

Reliability Modeling
The prediction of system level reliability starts with establishing the failure rates of the individual components making up the system. Then using the appropriate prediction models, the component level failure rates are combined to give us the system level reliability prediction in terms of a failure rate.

In literature, however, system level reliability is often discussed in terms of Mean Time Between Failures (MTBF) for repairable systems rather than a failure rate. For example, 50 years Mean Time Between Failures. A 50 years MTBF may suggest that a system will run 50 years between failures, but means more like that given 50 identical systems, one in a year will fail on average over a large population of systems.

The following illustration explains roughly how to bridge from individual component reliability to system reliability terms with some rounding and assumptions about secondary effects:

Figure 29: Rough Reliability Cheat Sheet*



Ideally Systems are composed of multiple parts each with very small failure rates that may vary over time. For Example

- 10 parts in 100,000 fail in first quarter (early life failure) then tailing off
- 1 part in 1 100,000 for a some time (steady state failure rate)
- Then increasing rate of failures until system component in use is retired (wear-out rate)
- Typically this is described as a bathtub curve

If 1 year is rounded up to 10000 hrs then Roughly:

FIT = Failure In Time
1 FIT = 1 Failure per 1 Billion hours

***Roughly* 100 FITs means**
A Failure Rate of 1 Failure in 1000 systems per year

Mean Time Between Failures is the inverse of Failure Rate

1 fail/1000 Systems per year
= 1000 Year MTBF

When FITs are small

FIT of Component 1 +
FIT of Component 2 +
FIT of Component 3 +
…
FIT of Component n +

System Failure Rate

$$\frac{1}{\text{System Failure Rate}} = \text{Mean Time Between Failure (MTBF)}$$

Example

100	FITs +
50	FITs +
70	FITs +
30	FITs +
200	FITs +
50	FITs +
30	FITs +
170	FITs +
50	FITs +
250	FITs =
1,000	FITs For System

If 100 Fits is 1 Fail per 1000 systems per year

1000 Fits =
10 Failures Per 1000 Systems per Year

1000 Systems/10 Failures in a year equates to 100 Years MTBF

Different Levels of Reliability
When a component fails, the impact of that failure can vary depending on the component.
A power supply failing, in a system with a redundant power supply will need to be replaced. By itself, however, a failure of a single power supply should not cause a system outage and should lead to a concurrent repair with no down-time to replace.

There are other components in a system might fail causing a system-wide outage where concurrent repair is not possible.

Therefore, it is typical to talk about different MTBF numbers.
For example:
MTBF – Resulting Repair Actions
MTBF – That require concurrent repair

MTBF – That require a non-concurrent repair

MTBF – Resulting in an unplanned application outage
MTBF – Resulting in an unplanned system outage
Scale-out systems may invest in having a long MTBF for unplanned outages even if it means more recurrent repair actions.

Costs and Reliability

Service Costs
It is common for software costs in an enterprise to include the cost to acquire or license code and a recurring cost for software maintenance. There is also a cost associated with acquiring the hardware and a recurring cost associated with hardware maintenance – primarily fixing systems when components break.

Individual component failure rates, the cost of the components, and the labor costs associated with repair can be aggregated at the system level to estimate the direct maintenance costs expected for a large population of such systems.

The more reliable the components the less the maintenance costs should be.
Design for reliability can not only help with maintenance cost to a certain degree but also with the initial cost of a system as well – in some cases. For example, designing a processor with extra cache capacity, data lanes on a bus or so forth can make it easier to yield good processors at the end of a manufacturing process as an entire module need not be discarded due to a small flaw. At the other extreme, designing a system with an entire spare processor socket could significantly decrease maintenance cost by not having to replace anything in a system should a single processor module fail. However, each system will incur the costs of a spare processor for the return of avoiding a repair in the small proportion of those that need repair. This is usually not justified from a system cost perspective. Rather it is better to invest in greater processor reliability.

On scale-out systems redundancy is generally only implemented on items where the cost is relatively low, and the failure rates expected are relatively high – and in some cases where the redundancy is not complete. For example, power supplies and fans may be considered redundant in some scale-out systems because when one fails, operation will continue. However, depending on the design, when a component fails, fans may have to be run faster, and performance-throttled until repair.

On scale-up systems redundancy that might even add significantly to maintenance costs is considered worthwhile to avoid indirect costs associated with downtime, as discussed below.

End User Costs
Generally, of greater concern are the indirect costs that end users will incur whenever a service action needs to be taken. The costs are the least when a component fails and can be concurrently replaced, and the highest when a component fails such that the system goes down and must stay down until repaired.

The indirect cost typically depends on the importance of the workloads running in the system and what sort of mechanisms exist to cope with the fault. If an application is distributed across multiple systems and there is little to no impact to the application when a single system goes down, then the cost is relatively low and there is less incentive to invest in RAS. If there is some application downtime in such an environment, or at least a reduction in workload throughput,

then the cost associated with the downtime can be quantified and the need for investing in greater RAS can be weighed against those costs.

The previous section discussed how the highest levels of availability are typically achieved even in scale-up environments by having multiple systems and some means of failing over to another in the event of a problem. Such failovers, even when relatively short duration can have costs. In such cases, investing in RAS at the server level can pay particular dividends.

When a failover solution is not employed, the impact to workload is obvious. Even in cases where failover is enabled, not every application may be considered critical enough to have an automated failover means. The impact of a prolonged service outage on these applications can be significant.

It is also significant to note that even when plans are put into place for automated failover when a fault occurs, there are times when failover does not happen as smoothly as expected. These are typically due to multiple factors, unavailability of the backup system, unintended code level mismatches, insufficient testing, and so forth.

Perhaps the less reliable the system, the more often the failover mechanisms might be tested, but also the more likely that some reason for an incident to occur in failover leading to an extended outage. The latter suggests also that investing in both hardware reliability and failover testing can be beneficial.

Measuring Availability

Measuring Availability
Mathematically speaking, availability is often expressed as a percentage of the time something is available or in use over a given period of time. An availability number for a system can be mathematically calculated from the expected reliability of the system so long as both the mean time between failures and the duration of each outage is known.

For example: Consider a system that always runs exactly one week between failures and each time it fails, it is down for 10 minutes. For the 168 hours in a week, the system is down (10/60) hours. It is up 168hrs – (10/60) hrs. As a percentage of the hours in the week, it can be said that the system is (168-(1/6))*100% = 99.9% available.

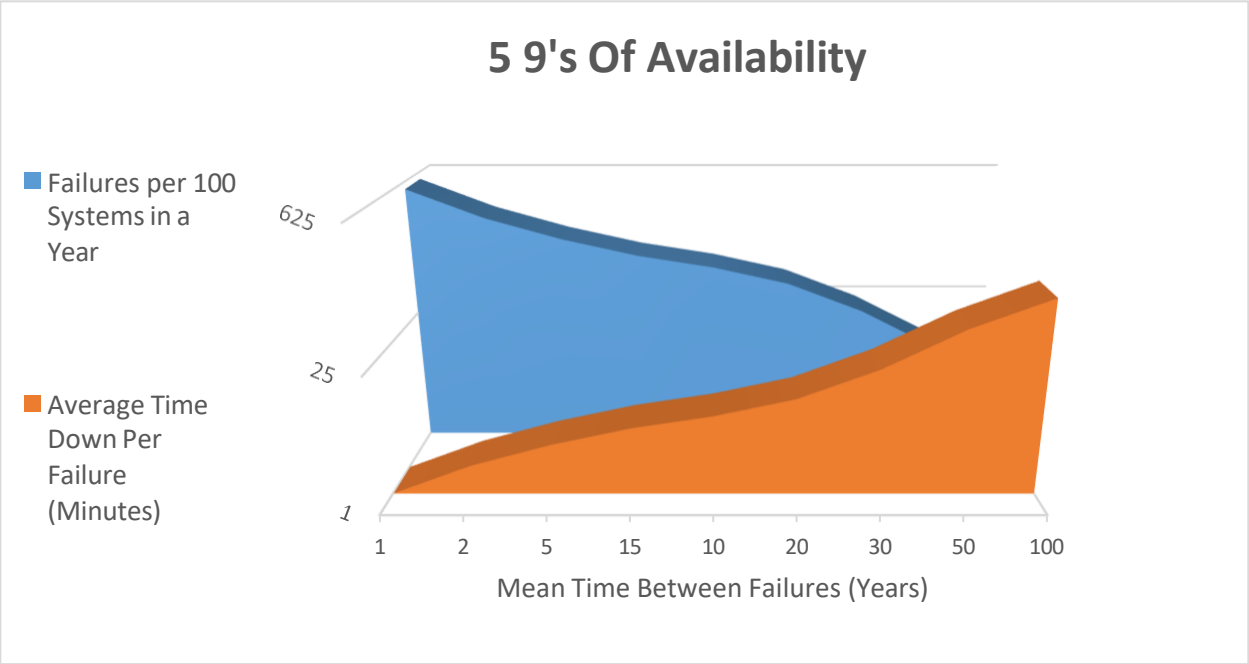
99.999% available means approximately 5.3 minutes down in a year. On average, a system that failed once a year and was down for 5.3 minutes would be 99.999% available. This is often called 5 9’s of availability.

When talking about modern server hardware availability, short weekly failures like in the example above is not the norm. Rather the failure rates are much lower and the mean time between failures (MTBF) is often measured in terms of years – perhaps more years than a system will be kept in service.

Therefore, when a MTBF of 10 years, for example, is quoted, it is not expected that on average each system will run 10 years between failures. Rather it is more reasonable to expect that on average, in a given year, one server out of ten will fail. If a population of ten servers always had exactly one failure a year, a statement of 99.999% availability across that population of servers would mean that the one server that failed would be down about 53 minutes when it failed.

In theory 5 9's of availability can be achieved by having a system design which fails frequently, multiple times a year, but whose failures are limited to very small periods of time. Conversely 5 9's of available might mean a server design with a very large MTBF, but where a given server takes a fairly long time to recover from the very rare outage.

Figure 30: Different MTBFs but same 5 9's of availability



The figure above shows that 5 9's of availability can be achieved with systems that fail frequently for miniscule amounts of time, or very infrequently with much larger downtime per failure.

The figure is misleading in the sense that servers with low reliability are likely to have many components that, when they fail, take the system down and keep the system down until repair. Conversely servers designed for great reliability often are also designed so that the systems, or at least portions of the system can be recovered without having to keep a system down until repaired.

Hence on the surface systems with low MTBF would have longer repair-times and a system with 5 9's of availability would therefore be synonymous with a high level of reliability.
However, in quoting an availability number, there needs to be a good description of what is being quoted. Is it only concerning unplanned outages that take down an entire system? Is it concerning just hardware faults, or are firmware, OS and application faults considered?
Are applications even considered? If they are, if multiple applications are running on the server, is each application outage counted individually? Or does one event causing multiple application outages count as a single failure?

If there are planned outages to repair components either delayed after an unplanned outage, or predictively, is that repair time included in the unavailability time? Or are only unplanned outages considered?

Perhaps most importantly when reading that a certain company achieved 5 9's of availability for an application - is knowing if that number counted application availability running in a standalone environment? Or was that a measure of application availability in systems that might have a failover capability.

Contributions of Each Element in the Application Stack
When looking at application availability it is 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 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 based on previous experience. 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. A condition 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. Figure 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 31: Hypothetical Standalone System Availability Considerations

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 acquire 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 Availability
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%*

This is not intended to represent any given system. Rather, it is intended to illustrate how different outages have different impacts. 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 causes 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.
In the hypothetical example numbers were chosen to illustrate 5 9s of availability across a population of systems.

This required that the worst-case outage scenarios to be extremely rare compared to the application only-outages.
In addition, the example presumed that:

1. All the software layers can recover reasonably efficiently even from entire system crashes.
2. There were no more than a reasonable number of applications driven and operating system driven outages.
3. A very robust hypervisor is used, expecting it to be considerably more robust than 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 most of the faults.
6. As previously mentioned, the system design is intended that 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.
7. A method of ensuring quick restart of a system after hardware outages, which might impact the ability to do extensive fault analysis.

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 have not caused outages.
The ability to recover from outages may also assume that some failed hardware may be deconfigured to allow the application to be restarted in cases where restart is possible. The system would need to be configured in such a way that there are sufficient resources after a deconfiguration for the application to restart and perform useful work.

The hypothetical example also presumes no significant time to diagnose the fault and planned hardware repair times. It may be more typical that some significant time is taken to collect extended error data concerning a failure. These numbers assume that action is taken not to collect such data and that the built-in error/detection fault isolation capabilities are generally sufficient to isolate the fault to a fault domain. It is also common when talking about availability to not consider any planned outages for repair.

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.

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 reliably predicted and limited to just that “fail-over” time.

Figure shows what might be achieved with the earlier proposed hypothetical enterprise hardware example in such a clustered environment.

Figure 32: Ideal Clustering with Enterprise-Class Hardware Example

	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 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.

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.

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 minutes.

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 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 work together to achieve the desired high availability level.

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 enough redundancy of hard drives to for greater than 5 9s availability of data on the drives but may fail 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 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 in the controller software.

The role of software layers in cluster availability is crucial, and it can be quite difficult to verify that all 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 verify 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 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.

	Time to detect outage in minutes	*1*		*Ratio of Problem Failover Events to Successful 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	Availability 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 34: More Realistic Clustering with Non-Enterprise-Class Hardware

	Time to detect outage in minutes	*1*		*Ratio of Problem Failover Events to successful 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	Availability 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 the less perfect clustering must 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 advance 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.

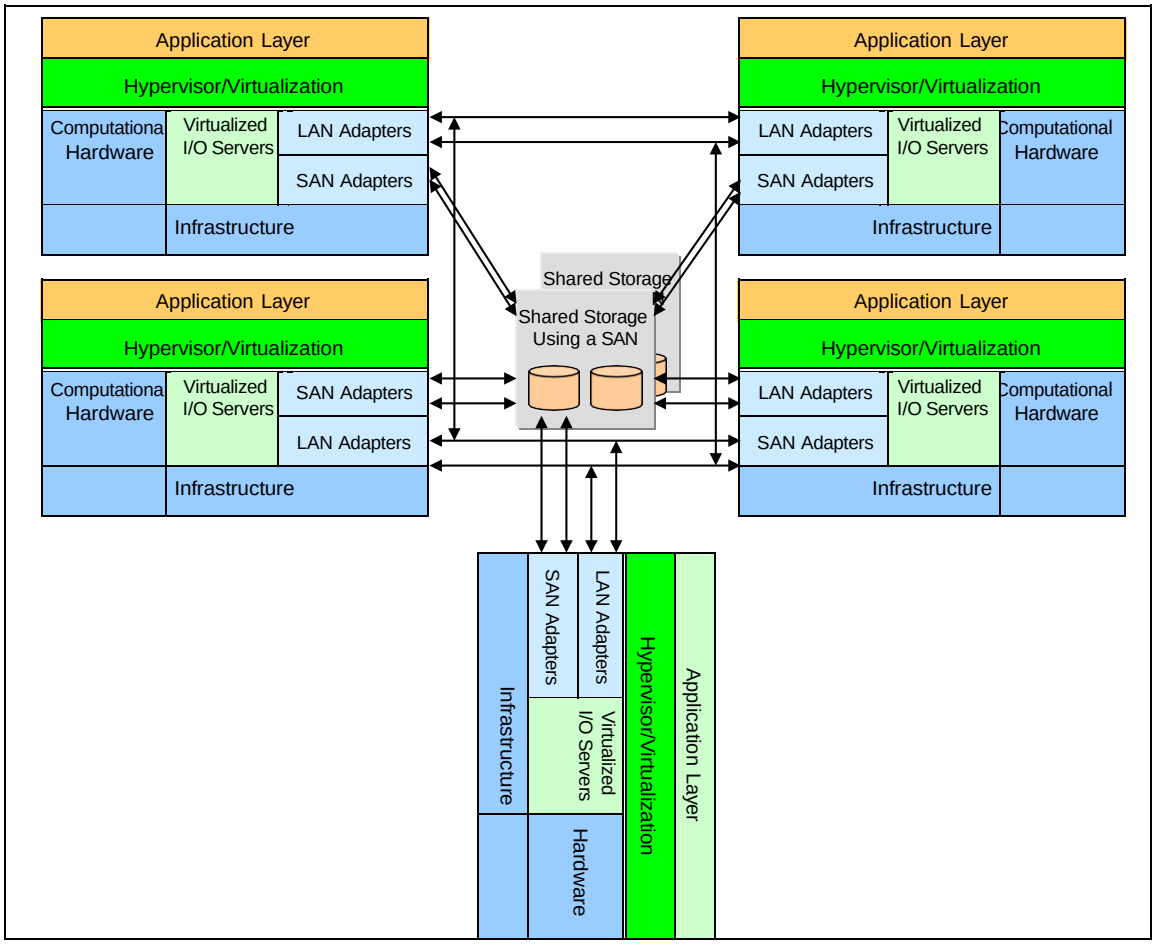
HA Solutions Cost and Hardware Suitability

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 comparable 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 35: Multi-system Clustering Option



There are several additional variations that could be considered.
In practice there is typically a limit as to how many systems are 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.

IBM Power is designed with per core performance in mind and the expectation of high utilization. Deploying PowerVM allows for a great depth in virtualization allowing applications to take the most advantage of the processing, I/O and storage capabilities. Thus, there is a natural

affinity towards use in clustered environments where maximizing resources is important in reducing ownership costs.

Cloud Service Level Agreements and Availability

The preceding analysis establishes some means of understanding application availability regardless of where applications are hosted – on premises, by a cloud provider, or using a hybrid approach.

It requires a thorough understanding of the underlying hardware capabilities, failover or remote restart capabilities, and HA applications. This may be difficult to achieve in all circumstances. For example, providers of cloud services may offer service level agreements (SLA) for the cloud services they provide. An SLA may have availability guarantees. Such SLAs typically pertain to availability within a given month with a failure to meet an SLA in a month providing a credit for the next month's services.

A typical SLA provides several tiers of availability, say 10% credit if a 99.95 % availability isn't maintained or 30% credit if 99% is not maintained and 100% credit if a level of 96% isn't achieved.

Figure 36: Hypothetical Service Level Agreement

Hypothetical Monthly Service Level Agreement Tiers	Credit Given on next month's charges
Availability less than 96%	100%
Availability greater than 96% but less than 99%	30%
Availability greater than 99% but less than 99.99	10%

*Where "down" and availability refer to the service period provided, not the customer application.

In understanding the service level agreement, what the "availability of the service" means is critical to understanding the SLA.

Presuming the service is a virtual machine consisting of certain resources (processor cores/memory) these resources would typically be hosted on a server. Should a failure occur which terminates applications running on the virtual machine, depending on the SLA, the resources could be switched to a different server.

If switching the resources to a different server takes no more than 4.38 minutes and there is no more than a single failure in a month, then the SLA of 99.99% would be met for the month. However, such an SLA might take no account of how disruptive the failure to the application might be. While the service may be down for a few minutes it could take the better part of an hour or longer to restore the application being hosted. While the SLA may say that the service achieved 99.99% availability in such a case, application availability could be far less.

Consider the case where an application hosted on a virtual machine (VM) with a 99.99% availability for the VM. To achieve the SLA, the VM would need to be restored in no more than about 4.38 minutes. This typically means being restored to a backup system.
If the application takes 100 minutes to recover after a new VM is made available (for example), the application availability would be more like 99.76% for that month.

Figure 37: Hypothetical Application Downtime meeting a 99.99% SLA

Outage Reasons	Downtime in minutes	Actual Availability for the Month	Downtime if one outage in a year (minutes)	Actual availability for that year	Downtime if 12 outages in a year	Actual Availability for the Year
Service Outage	4.38		4.38		52.56	
Downtime for application to restore on new virtual machine	100.00		100.00		1200	
Total Downtime (Sum of Service outage + application recovery)	104.38	99.76%	104.38	99.98%	1252.56	99.76%

If that were the only outage in the year, the availability across the year would be around 99.99%. The SLA, however, could permit a single outage every month.

In such a case with the application downtime typically orders of magnitude higher than the server outage time, even an SLA of 99.99% Availability or 4.38 minutes per month will prove disruptive to critical applications even in a cloud environment.

The less available the server is, the more frequent the client application restarts are possible. These more frequent restarts mean the client do not have access to their applications and its effect is compounded over time.

In such situations the importance of using enterprise class servers for application availability can't be understood just by looking at a monthly service level agreement.
To summarize what was stated previously, it is difficult to compare estimates or claims of availability without understanding specifically:

1. What kind of failures are included (unplanned hardware only, or entire stack)?
2. What is the expected meantime between failures and how is it computed (monthly SLA, an average across multiple systems over time, etc.)?
3. What is done to restore compute facilities in the face of a failure and how recovery time is computed?
4. What is expected of both hardware and software configuration to achieve the availability targets?
5. And for actual application availability, what the recovery time of the application is given each of the failure scenarios?

Section 5: Serviceability

The purpose of serviceability is to efficiently repair the system while attempting to minimize or eliminate impact to system operation. Serviceability includes new system installation, Miscellaneous Equipment Specification (MES) which involves system upgrades/downgrades, and system maintenance/repair. Based on the system warranty and maintenance contract, service may be performed by the client, an IBM representative, or an authorized warranty service provider.

Service Environment

In the PowerVM environment, the HMC is a dedicated server that provides functions for configuring and managing servers for either logical partitioned or full-system partition using a GUI or command-line interface (CLI) or REST API. An HMC attached to the system enables support personnel (with client authorization) to remotely or locally login to review error logs and perform remote maintenance if required.

There are multiple service environment options:

- **HMC Attached** - one or more HMCs or vHMCs are supported by the system with PowerVM. This is the default configuration for servers supporting logical partitions with dedicated or virtual I/O. In this case, all servers have at least one logical partition.
- **HMC less** - There are two service strategies for non-HMC managed systems.
    1. **Full-system partition with PowerVM**: A single partition owns all the server resources and only one operating system may be installed. The primary service interface is through the operating system and the service processor.
    2. **Partitioned system with NovaLink**: In this configuration, the system can have more than one partition and can be running more than one operating system. The primary service interface is through the service processor.

Service Interface

Support personnel can use the service interface to communicate with the service support applications in a server using an operator console, a graphical user interface on the management console or service processor, or an operating system terminal. The service interface helps to deliver a clear, concise view of available service applications, helping the support team to manage system resources and service information in an efficient and effective way. Applications available through the service interface are carefully configured and placed to give service providers access to important service functions.

Different service interfaces are used, depending on the state of the system, hypervisor, and operating environment. The primary service interfaces are:

- LEDs
- Operator Panel
- BMC Service Processor menu
- Operating system service menu
- Service Focal Point on the HMC or vHMC with PowerVM

In the light path LED implementation, the system can clearly identify components for replacement by using specific component-level LEDs and can also guide the servicer directly to the component by signaling (turning on solid) the enclosure fault LED, and component FRU fault LED. The servicer can also use the identify function to blink the FRU-level LED. When this function is activated, a roll-up to the blue enclosure identify LED will occur to identify an enclosure in the rack. These enclosure LEDs will turn on solid and can be used to follow the light path from the enclosure and down to the specific FRU in the PowerVM environment.

First Failure Data Capture and Error Data Analysis

First Failure Data Capture (FFDC) is a technique that helps ensure that when a fault is detected in a system, the root cause of the fault will be captured without the need to re-create the problem or run any sort of extended tracing or diagnostics program. For the vast majority of faults, a good FFDC design means that the root cause can also be determined automatically without servicer or human intervention.

FFDC information, error data analysis, and fault isolation are necessary to implement the advanced serviceability techniques that enable efficient service of the systems and to help determine the failing items.

In the rare absence of FFDC and Error Data Analysis, diagnostics are required to re-create the failure and determine the failing items.

Diagnostics

General diagnostic objectives are to detect and identify problems so they can be resolved quickly. Elements of IBM's diagnostics strategy is to:

- Provide a common error code format equivalent to a system reference code with PowerVM, system reference number, checkpoint, or firmware error code.
- Provide fault detection and problem isolation procedures. Support remote connection capability that can be used by the IBM Remote Support Center or IBM Designated Service.
- Provide interactive intelligence within the diagnostics, with detailed online failure information, while connected to IBM's back-end system.

Automated Diagnostics

The processor and memory FFDC technologies are designed to perform without the need for problem re-creation nor the need for user intervention. The firmware runtime diagnostics code leverages these hardware fault isolation facilities to accurately determine system problems and to take the appropriate actions. Most solid and intermittent errors can be correctly detected and isolated, at the time the failure occurs that is whether during runtime or boot-time. In the few situations that automated system diagnostics cannot decipher the root cause of an issue, service support intervention is required.

Stand-alone Diagnostics

As the name implies, stand-alone or user-initiated diagnostics requires user intervention. The user must perform manual steps, which may include:

- Booting from the diagnostics CD, DVD, USB, or network
- Interactively selecting steps from a list of choices

Concurrent Maintenance

The determination of whether a firmware release can be updated concurrently is identified in the readme information file that is released with the firmware. An HMC is required for the concurrent firmware update with PowerVM. In addition, as discussed in more details in other sections of this document, concurrent maintenance of PCIe adapters and NVMe drives are supported with PowerVM. Power supplies, fans and op panel LCD are hot pluggable as well.

Service Labels

Service providers use these labels to assist them in performing maintenance actions. Service labels are found in various formats and positions and are intended to transmit readily available information to the servicer during the repair process. Following are some of these service labels and their purpose:

- **Location diagrams**: Location diagrams are located on the system hardware, relating information regarding the placement of hardware components. Location diagrams may include location codes, drawings of physical locations, concurrent maintenance status, or other data pertinent to a repair. Location diagrams are especially useful when multiple components such as DIMMs, processors, fans, adapter cards, and power supplies are installed.
- **Remove/replace procedures**: Service labels that contain remove/replace procedures are often found on a cover of the system or in other spots accessible to the servicer. These labels provide systematic procedures, including diagrams detailing how to remove or replace certain serviceable hardware components.
- **Arrows**: Numbered arrows are used to indicate the order of operation and the serviceability direction of components. Some serviceable parts such as latches, levers, and touch points need to be pulled or pushed in a certain direction and in a certain order for the mechanical mechanisms to engage or disengage. Arrows generally improve the ease of serviceability.

QR Labels

QR labels are placed on the system to provide access to key service functions through a mobile device. When the QR label is scanned, it will go to a landing page for Power10 processor-based systems. The landing page contains links to each MTM service functions and its useful to a servicer or operator physically located at the machine. The service functions include things such as installation and repair instructions, reference code look up, and so on.

Packaging for Service

The following service features are included in the physical packaging of the systems to facilitate service:

- **Color coding (touch points)**: Blue-colored touch points delineate touchpoints on service components where the component can be safely handled for service actions such as removal or installation.

- **Tool-less design**: Selected IBM systems support tool-less or simple tool designs. These designs require no tools or simple tools such as flathead screw drivers to service the hardware components.
- **Positive retention**: Positive retention mechanisms help to assure proper connections between hardware components such as cables to connectors, and between two cards that attach to each other. Without positive retention, hardware components run the risk of becoming loose during shipping or installation, preventing a good electrical connection. Positive retention mechanisms like latches, levers, thumbscrews, pop Nylatches (U-clips), and cables are included to help prevent loose connections and aid in installing (seating) parts correctly. These positive retention items do not require tools.

Error Handling and Reporting

In the event of system hardware or environmentally induced failure, the system runtime error capture capability systematically analyzes the hardware error signature to determine the cause of failure. The analysis result will be stored in system NVRAM. When the system can be successfully restarted either manually or automatically, or if the system continues to operate, the error will be reported to the operating system. Hardware and software failures are recorded in the system error log filesystem.

When an HMC is attached in the PowerVM environment, an ELA routine analyzes the error, forwards the event to the Service Focal Point (SFP) application running on the HMC, and notifies the system administrator that it has isolated a likely cause of the system problem. The service processor event log also records unrecoverable checkstop conditions, forwards them to the SFP application, and notifies the system administrator.

The system has the ability to call home through the operating system to report platform-recoverable errors and errors associated with PCIe adapters/devices.

In the HMC-managed environment, a call home service request will be initiated from the HMC and the pertinent failure data with service parts information and part locations will be sent to an IBM service organization. Customer contact information and specific system related data such as the machine type, model, and serial number, along with error log data related to the failure, are sent to IBM Service.

Service Action Alert

Power10 eBMC systems firmware release 1050 provides a periodic reminder to service team and customer admin of the unresolved or deferred hardware repair action in the HMC managed system. This reminder also occurs at the end of each HMC repair and verify hardware service action, during power on, if unresolved hardware repair is present. This proactive alert can prevent a system outage. This service feature is enabled by default.

Call Home

Call home refers to an automatic or manual call from a client location to the IBM support structure with error log data, server status, or other service-related information. Call home invokes the service organization in order for the appropriate service action to begin. Call home can be done through the Electronic Service Agent (ESA) imbedded in the HMC or through a version of ESA imbedded in the operating systems for non-HMC managed or a version of ESA that runs as a standalone call home application. While configuring call home is optional, clients are encouraged to implement this feature in order to obtain service enhancements such as

reduced problem determination and faster and potentially more accurate transmittal of error information. In general, using the call home feature can result in increased system availability. See the next section for specific details on this application.

IBM Electronic Services

Electronic Service Agent (ESA) and Client Support Portal (CSP) comprise IBM Electronic Services solution, which is architected for providing fast, exceptional support to IBM clients. IBM ESA is a no-charge tool that proactively monitors and reports hardware events such as system errors and collects hardware and software inventory. ESA can help customers focus on their core company business initiatives, save time, and spend less effort in managing their day-to-day IT maintenance issues. In addition, Call Home Cloud Connect Web and Mobile capability extends the common solution and offers IBM Systems related support information applicable to Servers and Storage.

Details are available here - IBM Client Vantage

System configuration and inventory information collected by ESA also can be used to improve problem determination and resolution between the client and the IBM support team. As part of an increased focus to provide even better service to IBM clients, ESA tool configuration and activation comes standard with the system. In support of this effort, an HMC External Connectivity security whitepaper has been published, which describes data exchanges between the HMC and the IBM Service Delivery Center (SDC) and the methods and protocols for this exchange. To read the whitepaper and prepare for ESA installation, see the "Security" section at IBM Electronic Service Agent

Benefits of ESA

- **Increased Uptime**: ESA is designed to enhance the warranty and maintenance service by potentially providing faster hardware error reporting and uploading system information to IBM Support. This can optimize the time monitoring the symptoms, diagnosing the error, and manually calling IBM Support to open a problem record. And 24x7 monitoring and reporting means no more dependency on human intervention or off-hours client personnel when errors are encountered in the middle of the night.
- **Security**: The ESA tool is designed to help secure the monitoring, reporting, and storing of the data at IBM. The ESA tool is designed to help securely transmit through the internet (HTTPS) to provide clients a single point of exit from their site. Initiation of communication is one way. Activating ESA does not enable IBM to call into a client's system. For additional information, see the IBM Electronic Service Agent website.
- **More Accurate Reporting**: Because system information and error logs are automatically uploaded to the IBM Support Center in conjunction with the service request, clients are not required to find and send system information, decreasing the risk of misreported or misdiagnosed errors. Once inside IBM, problem error data is run through a data knowledge management system, and knowledge articles are appended to the problem record.

Remote Code Load (RCL)
The HMC 1030 release supports remote code load for firmware. It's a feature to upgrade or update code by a remote support engineer. RCL is supported with the Expert Care Premium package. For more details, please refer to the IBM Remote Code Load website.

Client Support Portal
Client Support Portal is a single internet entry point that replaces the multiple entry points traditionally used to access IBM Internet services and support. This web portal enables you to gain easier access to IBM resources for assistance in resolving technical problems.

This web portal provides valuable reports of installed hardware and software using information collected from the systems by IBM Electronic Service Agent. Reports are available for any system associated with the customer's IBM ID.

For more information on how to utilize client support portal, visit the following website: Client Support Portal or contact an IBM Systems Services Representative (SSR).

Summary

Investing in RAS
Systems designed for RAS may be more costly at the “bill of materials” level than systems with little investment in RAS.

Some examples as to why this could be so:
In terms of error detection and fault isolation: Simplified, at the low level, having an 8-bit bus takes a certain amount of circuits. Adding an extra bit to detect a single fault, adds hardware to the bus. In a class Hamming code, 5 bits of error checking data might be required for 15 bits of data to allow for double-bit error detection, and single bit correction. Then there is the logic involved in generating the error detection bits and checking/correcting for errors.
In some cases, better availability is achieved by having fully redundant components which more than doubles the cost of the components, or by having some amount of n+1 redundancy or sparing which still adds costs at a somewhat lesser rate.

In terms of reliability, highly reliable components will cost more. This may be true of the intrinsic design, the materials used including the design of connectors, fans and power supplies. Increased reliability in the way components are manufactured can also increase costs. Extensive time in manufacture to test, a process to ”burn-in” parts and screen out weak modules increases costs. The highest levels of reliability of parts may be achieved by rejecting entire lots –even good components - when the failure rates overall for a lot are excessive. All of these increase the costs of the components.

Design for serviceability, especially for concurrent maintained typically is more involved than a design where serviceability is not a concern. This is especially true when designing, for example, for concurrent maintenance of components like I/O adapters.
Beyond the hardware costs, it takes development effort to code software to take advantage of the hardware RAS features and time again to test for the many various ”bad-path” scenarios that can be envisioned.

On the other hand, in all systems, scale-up and scale-out, investing in system RAS has a purpose. Just as there is recurring costs for software licenses in most enterprise applications, there is a recurring cost associated with maintaining systems. These include the direct costs, such as the cost for replacement components and the cost associated with the labor required to diagnose and repair a system.

The somewhat more indirect costs of poor RAS are often the main reasons for investing in systems with superior RAS characteristics and overtime these have become even more important to customers. The importance is often directly related to:

- The importance associated with discovery errors before relying on faulty data or computation including the ability to know when to switch over to redundant or alternate resources.
- The costs associated with downtime to do problem determination or error re-creation, if insufficient fault isolation is provided in the system.
- The cost of downtime when a system fails unexpectedly or needs to fail over when an application is disrupted during the failover process.

The costs associated with planning an outage to or repair of hardware or firmware, especially when the repair is not concurrent.
In a cloud environment, the operations cost of server evacuation.

In a well-designed system investing in RAS minimizes the need to repair components that are failing. Systems that recover rather than crash and need repair when certain soft errors occur will minimize indirect costs associated with such events. Use of selective self-healing so that, for example, a processor does not have to be replaced simply because a single line of data on an I/O bus has a fault reduces planned outage costs.

In scale-out environments the reliability of components and their serviceability can be measured and weighed against the cost associated with maintaining the highest levels of reliability in a system.

In a scale-up environment, the indirect costs of outages and failovers usually outweigh the direct costs of the repair. An emphasis is therefore put on designs that increase availability in the face of frequent costs – such as having redundancy – even when the result is higher system and maintenance costs.

Final Word
The Power9 and Power10 processor-based models discussed leverage the long heritage of servers designed for RAS. The different servers aimed at different scale-up and scale-out environments provide significant choice in selecting servers geared towards the application environments end-users will deploy. The RAS features in each segment differ but in each provide substantial advantages compared to designs with less of an up-front RAS focus.

About the principal authors/editors:

Daniel Henderson is an IBM Senior Technical Staff Member. He has been involved with IBM 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 PowerVM based platforms.

Irving Baysah is a Senior Hardware development engineer with over 26 years of experience working on IBM Power. He designed Memory Controller and GX I/O logic on multiple generations of IBM Power processors. As a system integration and post silicon validation engineer, he successfully led the verification of complex RAS functions and system features. He has managed a system cluster automation team that developed tools to rapidly deploy in a Private Cloud, the PCIe Gen3/4/5 I/O bringup team and the Power RAS architecture team. He is currently the lead RAS Architect for IBM Power.

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 is intended to give a general understanding of concepts only. This information could include technical inaccuracies or typographical errors. Changes may be made periodically made to the information herein in new editions of this publication. IBM may make improvements and/or changes in the product(s) and/or 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

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 Group
Route 100
Somers, New York 10589
Produced in the United States of America
December 2024