

High Availability Systems: The Ever-Growing HA Stack

As High Availability systems have evolved, so has the equipment manufacturers' approach to product development. With the development of more highly reliable and fault tolerant architectures, equipment developers, whether in the communications, mil/aero or industrial sectors, now see they can offer their customers HA architectures that reduce operational expenditures and revenue loss due to service interruptions. By building always-on applications around these HA architectures, end customers are noticing significant reductions in total cost of ownership. Under the shadow of the current economy, minimizing total cost of ownership is a theme that resonates throughout the industry.

Traditionally, levels of HA were calculated based on the Mean Time Between Interruptions data collected from the hardware components in a system. This determined a number of "nines," or the percentage of time per year that they system remains functional. Now, HA can be defined by an integral suite of redundant hardware components and specialized layers of software components, commonly known as the "HA Stack."

1. THE HA STACK

There are several different layers that make up the HA Stack, comprised of both hardware and software. Each layer serves a purpose. Some are independently functional, and some are dependent on other layers in the stack. In the model discussed in this article, the ranking starts at the bottom with some of the most ubiquitous features for developing HA solutions. As shown in figure one, the HA Stack begins with layers that provide hardware features including Hot Swap and Component Redundancy. The third layer, Management, is comprised of both hardware and software (such as the capturing of data into event logs, sensor data records and interfaces such as SNMP). The final three layers are software-based and include Fault Tolerance, Redundant Host and Predictive Analysis/Policy-Based Management.

2. HOT SWAP AND COMPONENT REDUNDANCY

The common goal for High Availability is to increase Mean Time Between Interruptions (MTBI) and decrease Mean Time To Repair (MTTR). MTBI is an interruption of service to the system and is affected by the Mean Time Between Failure (MTBF) and redundancy of the system components. Availability = $(MTBF)/(MTBF + MTTR)$. This calculation generates a percentage that equates to the amount of uptime per year. For example, 99.999 percent, or "five nines," translates into a maximum allowable down time of 5.25 minutes per year. MTBI primarily speaks to the quality of components that are used in the design of all hardware components, while MTTR addresses minimizing system interruptions. MTTR can be broken into two

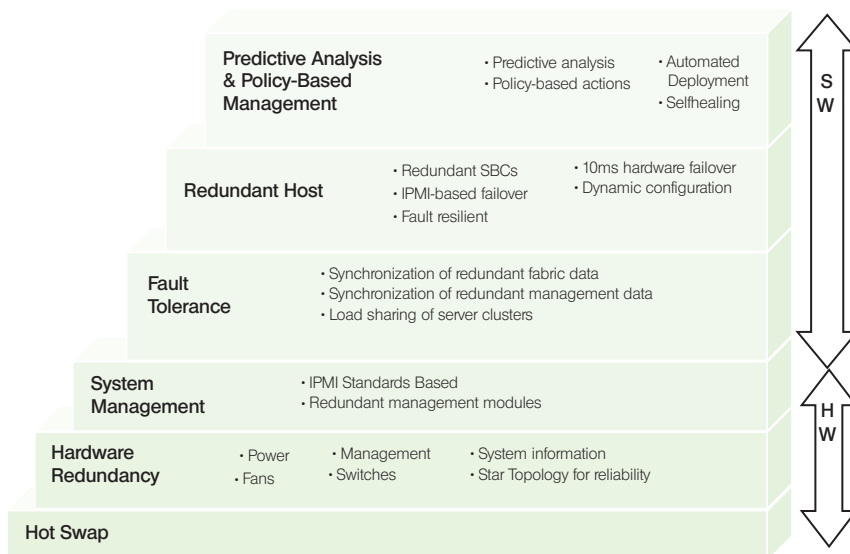


Figure 1. The HA Stack.

categories, serviceability and redundancy. For serviceability, the HA architecture requires that all components be Hot Swappable. This means the system must remain powered on during the removal and replacement of any failed components and that these components must be easily accessible for servicing. Redundancy ensures there is no single point of failure. Since failures occur randomly, it is typically adequate to assume that only one failure occurs at any given moment. In a platform, typical redundant components include power supplies, fan trays, Intelligent Platform Management Interface (IPMI)-based management modules, PICMG® 2.16 fabric switches and dual power inputs, but it can also include redundant Single Board Computers for application control or redundant capabilities on the backplane itself. We will see in the Fault Tolerant section how redundant components communicate and synchronize with each other to support a seamless failover.

Another hardware-based design element is the physical connection between components. There are typically two types of topologies; star and bus. In the PICMG 2.16 standard, a dual star topology connects each device via Ethernet over the backplane. The advantage to using a star topology versus a bus topology is if one failed device on the system is communicating erroneously on the bus, it does not adversely affect the rest of the system. This is because the point-to-point connections of each device to the fabric switches will isolate this from other traffic.

3. SYSTEM MANAGEMENT

The ratification of PICMG 2.9 defined a standardized management bus using the IPMI specification. Management occurs inherently at the core of most HA applications and all layers of the HA Stack, as it collects health status, environmental information such as temperature and voltage, and asset management information. It integrates to higher levels of software or to GUIs that allow intelligent actions to be made based on the system management information. Secondly, it manages slot control to power up/down or reset the components in the platform.

How it Works

Typically a system supports one or two redundant shelf management modules with a single IP address to interrogate and control some or all of the components in a system. In fault-tolerant applications, for example, the module could send commands over the system management bus (typically called the Intelligent Platform Management Bus or IPMB) to reset or cut power to failed cards. This isolates problems from other components in the system. Each IPMI-based managed component is designed with a micro controller running an interdependent small footprint operating system. On the shelf management module, it is called the Baseboard Management Controller (BMC). The BMC performs queries to the other managed objects and stores the information in an event log. If the component is a slave, such as server boards, line cards or even power supplies, the associated micro controller can be much simpler (called the

Satellite Management Controller or SMC), as it just reports information about itself.

Security and Independent Operation

The dedicated management framework provides security because management data isn't running on the same network as the application. Additionally, IPMI functions as a separate management plane that does not depend on the main application processors or operating system. Because the micro controller runs autonomously from its host, if the host's operating system crashes due to a kernel panic or "blue-screen," the SMC can report that a failure has occurred. The dedicated shelf management module in the platform receives this signal and can reset the board to get it back into a known state, or power it down until service personnel can diagnose the problem.

Event Reporting

One of the most critical elements to system management is the ability to capture and report out unusual events that can cause service interruptions. Management architectures that support user-defined thresholds allow the system manager to set early warning levels that allow the system manager to react to a problem before it becomes catastrophic.

Providing Framework for Predictive Analysis and Policy-Based Actions

At the simplest level, management can be used to report out failures or for slot control to power up and down specific components in the system. However, management can be used to perform intelligent and elegant failovers prior to catastrophic events, to set thresholds that trigger policy-based actions and to do automated recovery. With the management architecture closely and continuously monitoring the environmental information and stability of the application, higher levels of software, such as Predictive Analysis or Policy-Based Management, can determine what actions to take based on a single event or any combination of events. More of this will be discussed in a later section.

4. FAULT TOLERANCE

There are many levels of fault tolerance. At the basic level, it means that simplistic devices continue some level of operation after failure. Some redundant components in the chassis, such as power supplies, operate on a simple load sharing principal - if one power supply fails, the others will increase their share of the load until the failed supply is replaced. However, at a more intelligent level, fault-tolerant components mirror all operations - that is, every operation is performed on two or more duplicate systems, such as load sharing CPUs or redundant line cards operating in a cluster. Additionally, fault-tolerant components must synchronize important configuration data, such as MAC addresses for redundant fabric switches or management information collected by the shelf management modules. In order to achieve an elegant fault-tolerant failover, the latest, most reliable data must be synchronized between the two redundant components.

For instance, a system with redundant fabric boards can either communicate through one switch at a time in an active/standby mode or through both switches simultaneously for greater bandwidth. If a link, PHY or switching node fails in the active/standby mode, data can be re-routed to an alternate path, maintaining network connections. These boards will continuously check for health. If a problem is detected, the switch will de-assert all links, signaling the attached devices to communicate through the secondary port. The secondary switch learns the MAC addresses on the ports. During the change-out of the failed unit, the new unit "clones" its setup from the configuration stored on the secondary. Additionally, some switches support the virtual router redundancy protocol (VRRP), which allows several switch/routers on a multi-access link to utilize the same virtual IP address. One switch is elected as a master with the other switch acting as backup. The secondary switch "spoofs" the MAC and default gateway IP addresses of the primary. This supports a failover without any time outs on the network.

5. REDUNDANT HOST

One of the recently added layers to the HA Stack is Redundant Host. With a standards-based design founded on the PICMG 2.12 Redundant Host API, this architecture is open and not likely to change as a proprietary solution is prone to do. The Redundant Host set of APIs comprehends CompactPCI® and CompactTCA and can even be leveraged into AdvancedTCA in the future. Products built around the Redundant Host architecture can provide single-digit millisecond control failover, allowing system applications to recover almost immediately from any catastrophic control blade failure and circumvent losing valuable data. Switchovers can be triggered by predictive failure analysis, allowing the drivers and applications time to sync their databases and state information before handing over the control.

There are three types of device interaction in systems design: system data, management and control. They can occur over the same physical medium or over discrete mediums such as IPMB, CompactPCI, Ethernet or the H.110 telephony bus. Traffic such as voice, images and Internet traffic is exchanged in the data plane. This is the data that is being manipulated for the intended use of the system. The management plane is used to get status information and set thresholds for reporting or to take local actions regarding the functional health of the devices in the system. Some of the items that are monitored include CPU temperature, voltages and device signals such as BDSEL# and HEALTHY#. SNMP and IPMI are two examples of protocols used within the management plane. The control plane performs such functions as initialization, configuration (including hot swap) and control of the devices within the system. It is also through the control plane that an IO blade, such as an E1/T1 interface card, would interact for things like call routing, billing and various statistics gathering.

Redundant Host concentrates its efforts in the control plane. There is typically a 1:N relationship of control blade to IO blades. In CompactPCI, the System Master

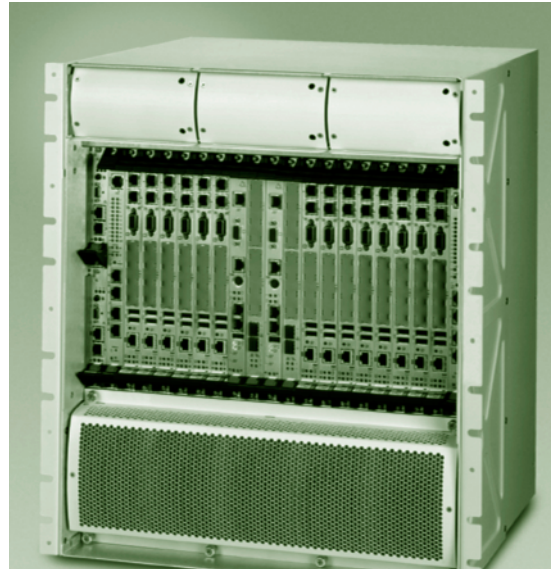


Figure 2. The Three Planes.

typically performs the control blade role while the peripherals perform the IO blade roles. This provides a single point of failure in that architecture, which is overcome by the Redundant Host design by providing management of a redundant control blade. This paradigm holds true for some PICMG 2.16 (and even pure Ethernet) architectures where IO devices manipulate the payload data and require a controlling device to either provide routing for further processing, store the data or simply collect statistics.

Imagine a database application configured with an Ethernet-based cluster. Typical failovers for a pure Ethernet cluster can range in multiple seconds. Processing data from a single Gigabit Ethernet interface for two seconds requires a buffer of 256 Megabytes of RAM to eliminate data loss. Compare this with a failover time of less than 10 milliseconds requiring a relative buffer size of less than one Megabyte. This doesn't even account for the processing time associated with playing "catch-up" with all of that data while the stream of new data continues to flow. Faster failover has an impact of reducing system requirements and therefore the cost of a control blade in any architecture.

In the case of a CompactPCI system requiring reliability and cost effectiveness, this architecture is ideal. Consider a chassis with 12 peripheral IO blades to perform E1/T1 trunking and routing. These blades can cost upward of \$10,000 each. The single point of failure in the CompactPCI system is the lone System Master, which typically costs \$2,500 or less. A clustering solution would require two identical systems to be built at a total cost of \$245,000. By making the System Master a redundant component, the high cost of the line cards is optimized, yielding a total blade cost of \$125,000 - almost half! Add to this the reliability of up time with the rapid failover and it is easy to see that choosing Redundant Host is a "no-brainer."

Redundant Host saves development time by managing the internal mechanics of host failover. Developers

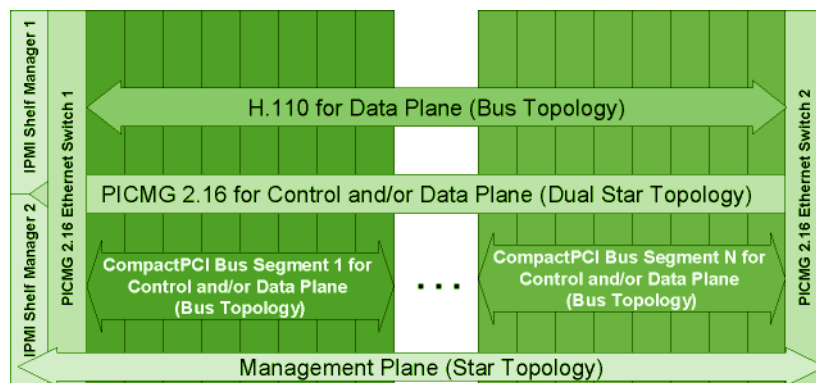


Figure 3. Performance Technologies' ZT 5085e Redundant Host Chassis.

need only design for synchronization of databases and state information along with notification of becoming active. By using a shared storage solution on the System Masters, each ensuing transaction from the IO blade could be committed to a database then acknowledged to the IO blade followed by synchronization with the backup host. In the case of a catastrophic failure to the active host, the backup would be up and running within 10 milliseconds. The newly active host would know of the last committed transaction, verify this with the database accessed through shared storage and request the IO blades to repeat the missing transaction. (Figure 3)

6. PREDICTIVE ANALYSIS & POLICY-BASED MANAGEMENT

Developers of HA solutions can take advantage of sophisticated management software to set user-defined thresholds to predict problems before they become catastrophic. This type of software controls the entire application in real-time and typically features a data-gathering engine that continuously monitors all of the system management information from the application infrastructure, the network and the business or service processes into a single repository. For instance, if shelves are overheating and have been monitored to climb over 20C in a 10 minute period, the software can trigger the air conditioning in the operations center. Or if the service is becoming unstable, (e.g. CPU utilization is too high, or available memory or hard drive capacity is low) the software can force a higher level of data backup to ensure no loss of information. A policy-based management system typically establishes a baseline and a set of rules to define an action to take when a specific event or combination of events occurs. These rules or policies can be fine tuned to establish a certain level of performance throughout the entire system. The system will monitor activities and perform trending to identify potential problems before they can cause a complete failure. When an event occurs, the system can automate lower level management tasks, open trouble tickets and send detailed automated notifications to the operations staff. This information is also useful for capacity planning. This enterprise-level of view management, which monitors all aspects of the

operations, ensures that policies are being enacted intelligently and globally, not based on one specific shelf or network module.

7. CONCLUSION

Vendors of standards-based embedded system architectures such as CompactPCI and PICMG 2.16 are offering products with these always-on design methodologies built in, which is good news, because designing these systems can be a daunting task. Since each level of the HA Stack is dependant on other levels, it is important to look for integrated solutions that offer as many of these features as possible; or at least the capabilities. Ultimately this will allow the developer to offer high availability, reparability, scalability and a lower total cost of ownership for the end user. ■

Sean O'Brien is a software engineering manager for Performance Technologies, managing the development of software and firmware for the Computing Products Group in San Luis Obispo, California. He has been with the company for five years. O'Brien is actively involved in the development of industry standards, and he recently chaired the PICMG® 2.14 Multi Computing Subcommittee. Prior to Performance Technologies, O'Brien worked for SPAR and ARK Telecom as a software engineer developing software and firmware for TDMA and FDMA satellite telecommunications equipment. He graduated from California Polytechnic Institute in 1991 with a BS in Computer Science.

Tony Romero is a senior product manager with Performance Technologies. For the past three years Tony has worked extensively in system architecture and product development of platforms with CompactPCI Packet Switched Backplanes, both pre-PICMG 2.16 and PICMG 2.16. His responsibilities include managing the CompactPCI computing platform products that comprise chassis, midplanes, system management, power supplies and cooling. Prior to working at Performance Technologies, Tony worked for Primus Knowledge Solutions and Dell Computer Corp.