Intelligent platform management: Critical key to future products

By Curtis A. Schwaderer

CompactPCI Systems

The CompactPCI notion of hot swap introduced many years ago is continuing to extend its reach across the entire platform. It's no longer enough to be able to hot failover each card within a chassis. High availability is extending past the circuit boards and into power, temperature, and the enclosure itself. Power supplies, temperature regulation, and global status and health instrumentation are all extensions and important next steps of an evolution in a complete high-availability platform management solution.

In this month's Software Corner, we'll take a look at the Intelligent Platform Management Interface (IPMI) standard. We'll cover the problems it solves, who's who in IPMI, and the latest enhancements and features to the IPMI specification, version 1.5.

Platform management problem

Being able to perform a hot swap of a board is obviously an important characteristic of a high-availability system. However, this capability by itself points to a couple of important problems that need to be solved:

- Determination of why the board needed to be hot swapped. The ability to hot replace a board is important. The ability to figure out why is probably more important in the larger scheme of things. Board mean-time-before failure (MTBF) is typically very long under normal operating conditions. Usually board failure is a symptom of a larger problem that needs to be found and fixed before more failures occur.
- By the time the board needs to be hot swapped, usually no diagnostic communication with the board is possible. If the board is dead, communication with its central processor or any statistics or instrumentation usually isn't possible. Ideal platform management operation would allow for the ability to communicate with the platform management component independently, even if the board is not operational, to facilitate "post mortem" analysis and information gathering.

The IPMI standard solves these problems by extending the reach of the diagnostics and management beyond the circuit board and into the chassis and supporting elements like fan and temperature status, power supply voltage regulation, and chassis intrusion. IPMI also is specified to operate on a separate set of processor elements, thereby still being operational and allowing interaction, whether the server is running normally or out of operation for any reason.

The key benefits are the cost savings associated with the management of IPMI-enabled systems and the higher level of information available to diagnose and fix system problems. Lower operational expense is always welcome – and the ability to uniformly manage an entire platform is an important simplifying factor. There is also significant value in allowing IT managers to determine server and platform health for operational and non-operational components in order to minimize mean time to repair (MTTR) periods. Otherwise, IT managers are forced to poke and prod their way blindly to determine the cause of server failure. This blind searching may result in more failures and capital expense before the problem is found.

How IPMI works

The term platform, as defined in IPMI, refers to:

- A rack or racks containing multiple chassis (or card cages)
- Circuit boards
- Power supplies
- Fans
- Cabling

IPMI resides as multiple management elements on the circuit boards within a chassis. Therefore, platform management is the monitoring and control functions built into all these elements within the platform hardware. The primary purpose of IPMI is to monitor the health of the system hardware.

Independent operation implies that IPMI will not run on the central processor of the circuit board, but a dedicated management controller that implements IPMI. This dedicated controller is called the baseband management controller (BMC). The BMC implementation typically consists of a microcontroller of some kind that has the IPMI command software set running on it.

Figure 1 is the IPMI block diagram from the IPMI specification version 1.5 and details the overall IPMI system. The key component in the system is the BMC. Also notice that there are several

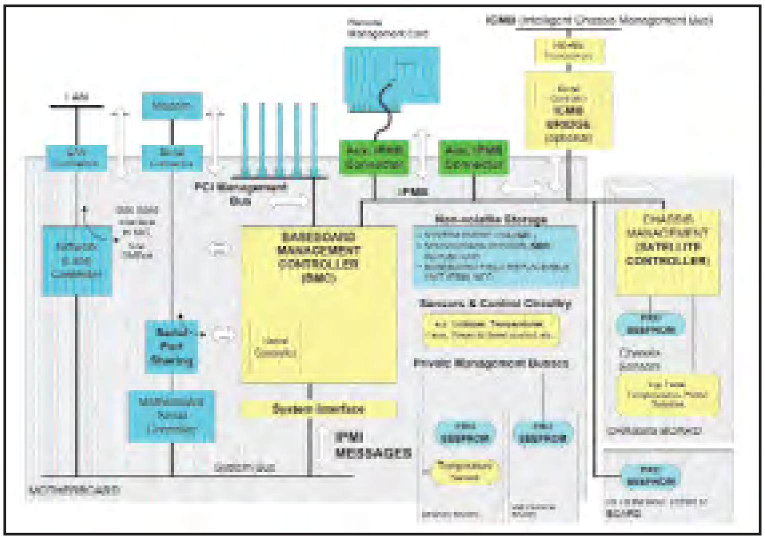

Figure 1

buses that interface with the BMC. Some of them are from the original 1.0 spec and some are new to enable the new remote management features of IPMI v1.5.

The primary bus is the Intelligent Platform Management Bus (IPMB). The IPMB is an I^{2}C-based serial bus used to communicate between management controllers. This provides the external card management connection that can be used to connect directly to a remote management card or to other devices with an IPMB interface. This bus is also used to communicate between chassis via the Intelligent Chassis Management Bus (ICMB). To communicate with the ICMB, an ICMB bridge device may be needed.

The architecture of the IPMB is very important. The support for multiple controllers enables scaling of architectures for as simple or complex a management system as required for the application. Simple configurations can be managed by a single BMC. More complex configurations that may involve managing a large number of platform elements can be split among any number of BMCs. The IPMB architecture even allows communication between chassis. A standards-based management interface like IPMB enables companies to create products that will be interoperable at the hardware level. For example, PICMG 3.1 takes advantage of IPMI and the IPMB, enabling an architecture that allows for multi-vendor interoperability at the chassis-management level. IPMI also has provisions to communicate with devices over raw I^{2}C without any of the IPMB framing.

IPMI messages are request/response based. In other words, the originator will issue a request (or command) and the receiver will issue a response in return. This protocol is independent of the framing or physical layer topology, so it can be used over IPMB, LAN, PCI, or serial. These remote extensions are introduced by the IPMI specification version 1.5 we'll introduce later.

Messages are grouped into functional command sets called Network Function Codes. Each network function code has a set of fields associated with it representing the data being exchanged in the message. The IPMI request messages have a function code and optional data. Responses have function code, data, and completion code that indicate the result of the processing initiated by the request message.

The information contained in the optional data section request or contain:

- Monitoring information – temperature, voltage, bus errors, physical system security
- Recovery – capabilities, i.e. reset or power off/on operations
- Logging – abnormal or out-of-range conditions
- Alerting – unique thing here is the platform can issue the alert without using run-time software on any of the servers
- Inventory – inventory capability to help find any failed hardware unit in the system

There are 12 logical management device definitions within the IPMI v1.5 specification as shown in Table 1 (on page 17). These logical devices may be in the form of the system controller, special ASICs within the platform elements, or other implementation. These devices are expected to respond to queries of their class.

IPMI and overall network management

IPMI is most effective when used within the context of an overall network management scheme. IPMI handles the hardware-related platform management chores. Even though it can operate independently, it is specified to fit within the context of a larger network management plan. Figure 2 also comes from the IPMI specification and outlines the IPMI context within an overall network management architecture.

The hardware layer is abstracted by an IPMI hardware interface. The first layer of software is the IPMI interface code that will form the abstraction that higher layer management software can build on for managing the hardware layer.

Above the IPMI interface layer is the standard network management components that enable local and remote network management into the larger managed system. Remember that IPMI can

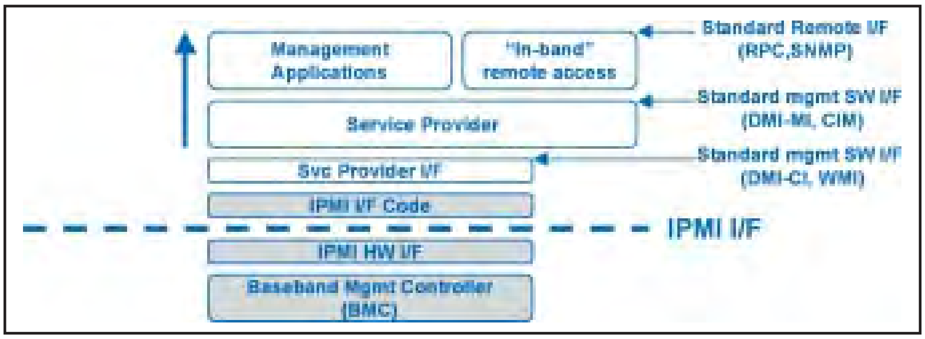
Management Applications
"In-band" remote access
Standard Remote I/F (RPC,SNMP)
Service Provider
Standard mgmt SW I/F (DMI-MI, CIM)
Svc Provider I/F
Standard mgmt SW I/F (DMI-CI, WMI)
IPMI I/F Code
IPMI I/F
IPMI HW I/F
Baseband Mgmt Controller (BMC)

Figure 2

be accessed and communicated with outside of the network management structure by incorporating any kind of local or remote connection between the hardware interface layer and the peer. You could probably envision a scenario where the higher layer network management software runs on the server CPU and hooks into the IPMI network management through the system bus. If the CPU happens to be dead, communication with the IPMI component can still occur to provide diagnostic information for the dead server, an important and highly desirable characteristic a new, intelligent network management scheme.

Who's who in IPMI

The IPMI specification is largely driven by is authors – Intel, Hewlett Packard, NEC, and Dell. The IPMI specification and other useful information can be found on the Intel developer Web site, developer.intel.com/design/servers/ipmi/spec.htm.

Since the inception of IPMI, over 70 other companies have adopted and/or begun to provide various IPMI solutions. How hard could implementing nuts and bolts hardware health be? Well, as it turns out, implementing IPMI isn't simple – the specification itself is over 400 pages. Therefore, a number of companies that can provide plug-in silicon and base software solutions for IPMI will be important in its mainstream adoption. References to these companies can also be found at the Intel developer Web site.

IPMI is not only a system motherboard and embedded PC server standard. Multiple OEMs, silicon vendors, and operating system vendors are also on-board with IPMI. For example, the Advanced Telecom Compute Architecture (ATCA) defined by PICMG 3.x also incorporates IPMI as part of its network management solution. A variety of enterprise and telecom blade server initiatives are also looking at incorporating IPMI into their blade server products.

IPM device	Implements the baseline IPMI command set.
Sensor device	Set of commands to discover, access, or configure sensors.
SDR repository device	Set of commands to retrieve sensor data records (SDRs).
System event log device	Set of commands for managing the system event log (for post-mortem analysis, etc., separate from the operation of the main CPUs in the system).
FRU inventory device	Contains Field Replaceable Unit (FRU) serial, part number, revision tags that provide information on information for unit replacement if required.
Event receiver device	Accepts & acknowledges system event messages, passes information for logging in system event log, and records event requests as required.
Event generator device	The event generator device sets up the format and delivers event messages to the event receiver device on specific event triggers.
Application device	This is the application-specific set of IPMI management that a product developer can add value for management specific to the unique capabilities of their device.
PEF device	Compares a message against a set of event filters to generate a selectable action on a match.
Alert processing device	Queuing up and processing alerts and alert policies.
Chassis device	Focuses on function control and recovery at the chassis level, i.e. power cycle, on/off, reset, chassis ID.
Message handler	Deals with message authentication and routing within the BMC and between the BMC elements.

Table 1

What IPMI v1.5 adds to the party

You might have wondered why during the discussion of the IPMI block diagram in Figure 1, we didn't discuss the SMBus, serial, or PCI interfaces. Well, these are new for IPMI version 1.5. IPMI version 1.5 specifies remote management over LAN, PCI management bus, or serial topologies. Since the IPMI messaging is physical-layer independent, the messages can be sent and received over any topology. The 1.5 version of the spec introduces standards that take advantage of a number of remote messaging capabilities specified by the initial IPMI standard.

Associated with the additions of LAN, PCI, and serial interfaces, also come extensions to the IPMI message set to provide remote alerting. LAN alerts come in the form of SNMP traps that will be retried if not acknowledged. Serial/modem alerting can be configured to perform dial paging using a modem, alphanumeric paging, or through PPP/serial (similar to LAN alerts).

Platform Event Filtering (PEF) is another important addition that takes advantage of the new remote communication plumbing introduced in the previous paragraphs. PEF is a mechanism that provides the ability for configuring a BMC to take specific action based on event messages received. This is done through the implementation of an event filter table. This is a table that can be configured using IPM device messages that can configure these events.

In addition to the event filter table, there is an alert policy table. This table works in conjunction with the event filter table to determine alerting based on the match of a particular event or set of events.

How the event filter and alert policy table work together is probably best illustrated in an example. Let's say there are three important events we need to watch for in our system – over temperature, board failure, and low voltage. We could set up three events in the event table with actions on each of these events. Maybe on over temperature, we want to increase fan speed. On low voltage we want to power cycle, and on board failure log. Now if we're over temperature and boards fail, we want a page. So, we could make a fourth entry in the event filter table on the logical AND of both events indicating an ALERT action. At this point, we'd need to add an alert to the alert policy table. This is the first, so alert policy 1 is to send me an e-mail and page me on my pager. The event table has the ability to perform multiple methods of alerting along with strings that will be sent. In this case, if we are over temperature and boards fail, the IPMI controller would see the action ALERT and pointer to policy 1. The controller would look at the alert policy 1 and then send an e-mail and page with the text string configured in the message, "Get to the office – the system is hot, fans not helping, and boards are dying." Pretty interesting and pretty useful – with IPMI, now your system can call you and tell you when it's not well! (Scary to think we're touching on a space where the movie *2001: A Space Odyssey* took us – "Dave, what are you doing Dave?")

Event actions may also be to simply log the event with no alert policy. Events and logging are two independent facilities and either can be enabled or disabled separately.

Conclusion

No single vendor produces all components that go into a platform. Therefore it's critical to have a pervasive system management standard that all vendors can incorporate into their products to assure interoperability. At this point, with no historical standards, integrators and system administrators have struggled with a network management strategy dealing with a large number of disparate piece-parts. The IPMI standard offers multi-vendor interoperability with a unified and comprehensive platform management structure that will enable high-availability systems to be more easily managed and diagnosed.

For more information, Curt Schwaderer can now be reached at software@compactpci-systems.com.