

Remote Hardware Management Framework for Cluster Computing Toolkit

Yung-Chin Fang, Jenwei Hsieh;
Dell Inc.
{Yung-Chin_fang; Jenwei_hsieh; }@dell.com

Stephen L Scott[†], Thomas Naughton
Oak Ridge National Laboratory
{scottsl; naughtont}@ornl.gov

Abstract

Due to a variety of management specifications, hardware, firmware and software implementations, most of the cluster computing toolkits do not include remote hardware management functionalities. The remote, hardware level management is a necessary task for a computer center. The remote hardware management task requires cross-field knowledge such as specifications, management fabrics, hardware, firmware and software implementations. For a multi-generation hardware environment, there is little in common across hardware level management interface implementations. In a cluster computing toolkit, there is a need to have a common hardware management interface/solution across processor architecture, platform, firmware, software and management fabric implementations. This paper presents a framework to address these differences that can be used in a cluster computing toolkit.

1. Introduction to Specifications

A cluster life cycle [1] includes: design phase, deployment phase, operational phase and retirement phase. The management specifications described in this section are focused on the deployment and operational phases. The purpose of management specifications is to enhance uptime and reduce total cost of ownership. These specifications are designed

to facilitate the hardware, firmware and software implementations used on a platform to perform local/remote monitoring and management of the node level hardware health condition, without interfering with the node's computing performance.

The frequently used HPC cluster management features are primarily based on well-defined management specifications. For example: in order to remotely deploy the operating system and cluster computing software stack to a new cluster, system administrators will often utilize the hardware level remote power management specification implementation to remotely power up the nodes in serial to facilitate pre-boot execution environment (PXE) [2] or extensible firmware interface (EFI) [3] level network boot. These implementations are used to remote deploy the OS and cluster computing software stack to the nodes across a cluster. PXE is usually implemented in an IA32 platform's basic input output system (BIOS) or in a network interface card ROM. EFI level network boot is implemented in the network interface's EFI level driver and resides in NVRAM. Remote power management is defined in advanced configuration and power interface (ACPI) [4], and ACPI is included in the Wired for Management (WfM) [5] specification. Remote power management is also addressed in the intelligent platform management interface (IPMI) [6] specification. Along with LM sensor management [7], IPMI and WfM are the three most commonly implemented specifications.

[†] Research supported by the Mathematics, Information and Computational Sciences Office, Office of Advanced Scientific Computing Research, Office of Science, U. S. Department of Energy, under contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

LM sensor management is a de facto standard from the 1990's, designed to use an embedded management processor, such as the LM81 [8], which utilized many sensors; such as CPU temperature, voltage, and fan RPM sensors, etc. to monitor and manage the node-level hardware health condition. There are some dedicated management buses, which are independent of the host data/address/control bus. An administrator can use an operating system level agent to query and control a sensor's reading via the LM processor. The operating system level agent can pass sensor readings to a centralized management console via a management fabric, and this is usually referred to as in-band management. There are two common views on the definition of in-band management. The first school of thought is that a management task can only be performed with the presence of an operating system, and the other definition is that management traffic and the host operating system share the same communication bandwidth. While these two definitions are not mutually exclusive, they do correspond to two equivalent definitions for out-of-band management. The first is that a management task can be done without the presence of an operating system, and the second is that a management task will not share communication bandwidth with a host operating system.

The disadvantage for in-band management is: when a node locks up, a system administrator can not judge if it is a software issue, a platform/hardware issue or a network related issue. An administrator loses management access to the locked up machine. Under this implementation, there is no way to remote power cycle a hung node. Under this limited framework, Advanced Power Management (APM) [9] and Emergency Management Protocol/Port (EMP) [10] specification were introduced and implemented to overcome these disadvantages. An OS level APM daemon is designed for power management and it is BIOS based. If a system BIOS does not support APM, then the APM utility can not function as expected. Since APM is BIOS based, an operating system has no knowledge about what APM does. A wide variety of implementations and functionality have created inconsistent hardware environments for a system administrator. EMP is designed to facilitate the use of a system's serial port as the second management fabric, so the administrator can use this fabric to remote power cycle a locked up node if the platform's BIOS implements both the APM and EMP specifications. An EMP implementation usually supports BIOS level console redirection, and the administrator can configure the operating system for

OS level console redirection. This provides remote power cycle and console redirection features for an operational phase system. APM/EMP provide a limited feature set for the operational phase of a cluster's lifecycle, but features for the deployment phase are not addressed, and features for the operational phase can be further enhanced. With these enhancements in mind, the Wired for Management (WfM) specification was designed and has been widely implemented.

WfM incorporates the following specifications: Preboot eXecution Environment (PXE), Wake on LAN (WOL, also known as remote wakeup), and Advanced Power Configuration Interface (ACPI), along with interfaces to many management protocols such as: Simple Network Management Protocol (SNMP) [11], Common Interface Mode (CIM) [12], and DMI [13]. ACPI, WOL and PXE specification implementations are typically helpful for the deployment phase. ACPI can be operated with and without the presence of an operating system. An ACPI implementation can be used to remote power up a new cluster with no OS and also remote power cycle a hung node. ACPI can also be used in conjunction with an OS to provide Operating System directed Power Management (OSPM) functions. With OSPM, the OS determines when to provide power management and the BIOS determines how to do it. Some manufacturers have implemented the ACPI specification on their platform, while other manufacturers have chosen to implement the advanced power management (APM) specification or their own proprietary remote power management mechanism.

The WOL specification complements the ACPI power management functionality. WOL enables cluster management tools to send a WOL packet to a compute node to wake it up. This requires that the cluster nodes be implemented with either ACPI or APM, plus, the network interface card must support WOL.

After a new node is powered up remotely, via ACPI or WOL, the next step is to deploy an operating system to that node. PXE is designed for remote boot-up and is usually implemented in conjunction with a network chip and the system BIOS. When a node boots up, it can execute the PXE, which resides on system BIOS or network interface card option ROM, and send out a Dynamic Host Configuration Protocol (DHCP) [14] request to a remote boot server asking for an IP address. Once the IP address is received, the PXE routine can interact with a remote boot server to dynamically retrieve the requested boot image over the

network. These items make it possible to remotely install the operating system and applications, and to remotely configure a new cluster without the presence of a console. Most of the popular cluster computing packages, such as Open Source Cluster Application Resources (OSCAR) [17] by the Open Cluster Group facilitate the PXE implementation as the main remote system deployment mechanism building block. PXE does have a hardware dependency, so a later extensible firmware interface (EFI) specification moved the boot-up portion of PXE to the EFI level instead of a firmware level. EFI network boot is usually implemented on IA64 platforms (which have a dependency on the Extensible firmware interface).

WfM also supports a wide range of management specifications such as: DMI, CIM, SNMP, Boot integrity service (BIS) [15], Network PC Guidelines, Solution Exchange Standard (SES)/Service Incident Exchange Standard (SIS) [16], System Management BIOS (SMBIOS), Web-Based Enterprise Management (WBEM), Windows Management Instrumentation (WMI), and others. This provides operating system level interoperability.

There is no commonly adopted industry-wide high performance computing cluster management specification standard. Current computing-cluster management approaches are designed to integrate and facilitate available implementations of management specifications to perform local/remote management tasks. To overcome this short coming, node level hardware management features could be integrated and automated via a unified command line interface implementation and provide cluster/ grid/ cyber-infrastructure level services and increase uptime.

2. About Implementation

Earlier computing hardware has little to no hardware level remote management capability and most of the hardware health management/monitoring capability is via an OS level agent. Later, certain platforms were equipped with proprietary hardware level remote management capability and are not compatible from vendor to vendor. Until IPMI was designed, remote hardware health monitoring/management did not seem to have a standard to follow. The standardized management specifications are defined by groups of professionals from many organizations. When defining standards, flexibility is kept for designers to select components and interpret technical details, so

each vendor can choose its own implementation of a feature. For example: IPMI 1.5 defines a set of baseboard management controller (BMC) level ASCII text commands for remote management of hardware health conditions. In order to communicate with the BMC level ASCII text commands, a proxy based CLI is needed for the remote management console software. Each vendor has interpreted the specification differently and implemented their unique proxy level command set for their remote management console. So, even though the IPMI standard defines a standard ASCII text SYS command set, the remote proxy CLI for this command set will be different from vendor to vendor. The varieties of implementations lead to compatibility issue and cluster management inconvenience. When a system administrator needs to remote power cycle a locked up node, the following items must be known:

- Maker. Different vendors implement management of HW/FW/SW differently.
- Architecture. Is it monolithic, blade, IA32, EM64T, or IA64 architecture? Usually different architectures will lead to a different management HW/FW/SW stack implementations.
- Platform model and generation. A single vendor may have different implementations on different models and generations. For example: Model A may have in-band manageability only, Model B may have a proprietary out-of-band management implementation, and Model C may have a standardized out-of-band implementation.
- Available management fabrics. In-band, Serial-Over-LAN, Serial-Over-Telnet, EMP via dedicated Ethernet, shared Ethernet, direct serial connection, or other proprietary management fabric?
- Access privilege requirements. Is root access required for this feature, or can a general user perform this command?
- Available HW, FW and SW. A single platform may have different HW/FW/SW configurations. One configuration may support a remote management controller; another configuration may support an IPMI implementation via X command line interface. The same platform with a different version of IPMI firmware may behave differently, and may not support the same command set. An administrator needs to figure out what the current HW/FW/SW configuration is for the platform.

- Version compatibility requirements. In order to remote power cycle a node, the centralized management console may need to be equipped with certain software modules. In many instances, these modules may have runtime environment dependency such as operating system kernel version dependency, development tool/runtime dependency, SNMP version dependency, etc.
- Implemented management fabrics and corresponding command line interface. One platform can be equipped with more than one management fabric. For example:
 - o The platform has OS level in-band manageability via in-band fabric
 - o The platform is equipped with a proprietary IP addressable remote management card via a dedicated Ethernet port
 - o The platform is equipped with a serial based proprietary remote management controller via a serial port connection
 - o The platform is equipped with an IPMI Serial Over LAN (SOL) via a dedicated out-of-band Ethernet port
 - o The platform is equipped with EMP via a direct serial connection

In many cases, one platform can have more than one dedicated management fabric. Also, an OS level hardware management CLI usually has a kernel version dependency due to hardware instrumentation module implementation.

Different vendors interpret and implement management specifications differently, and usually one platform will be implemented with more than one management specification. For example, one platform can have full PXE, ACPI, EMP plus part of the IPMI and WfM implementation. The chance of two models having the same management specification implementation is rare.

3. One management CLI

To learn the management specifications and various implementations from vendors for different platforms is a very time consuming process and the implementations will change as the architectures evolve. Usually a multi-generation cluster will have more than one CLI. Different platforms from the same vendor can be equipped with different remote management CLIs. For example: One platform can have three different CLIs, (1) CLI is OS level agent

implementation, (2) CLI is firmware level implementation and requires OS support, and (3) the CLI uses its dedicated management processor, ROM, memory and bus and does not require OS support. Each CLI has its own command set. Another barrier for efficient remote hardware management is the large number of management commands across cluster generations. A single remote cluster hardware monitoring/management interface to cross all hardware, firmware, software, CLIs, specifications and implementations is needed for the cluster computing toolkit.

The design considerations for such a unified interface should include: efficient scalability; expandability for future CLIs and management components; a grouping feature for group users, platforms and groups; a one-to-many CLI mechanism; a mechanism to auto resolve runtime environment dependencies; and also a single interface to cross all the available platform hardware, management fabrics, firmware, software and CLIs is desired.

4. Dependencies

Remote hardware monitoring/management module dependencies fall into two categories: management console and managed node. For the management console category: any hardware/OS level remote command line interface implementation has a dependency list. For example: platform architecture dependency can be IA32, EM64T, IA64, blade and/or monolithic and others. Operating system level dependencies are OS type and kernel version, this dependency usually associates with development tools. Development tool/environment dependency examples are compiler, library, database web library version, and others. Runtime environment dependencies usually indicate the settings needed for the runtime environment and specific service version such as SNMP version. Different modules/CLIs from different vendors usually have different dependencies. These dependencies must be satisfied so a monitoring/management module can work properly. The resolution of these dependencies can be challenging for many cluster users/administrators.

The fact that the specifications are open to interpretation in places and the associated influence on implementations are the major dependencies for the managed node category: different architecture leads to different dependencies. Embedded management implementations can include: management processor

type, performance and capability; number of management buses implemented; dedicated management bus (such as SM bus or I²C bus) clock rate; usable NVRAM size; management processor topology to each component; the management processor firmware implementation; OS level management agent implementation, and supported management fabric, etc.

5. Single Management Framework

After understanding the specifications, establishing the need for a single CLI, and having investigated the complexity of the layers of dependencies, the design of a single CLI becomes feasible.

An HPC cluster architecture is meant to scale out to achieve cost effective computing power, so the first consideration for a multi-generation cluster management framework is scalability. A backend database engine is needed to efficiently manage the various data and provide good scalability. Information such as user account, password and privilege; required and existing node hardware, firmware and software configuration; user groups; individual cluster, and/or cluster of clusters can be recorded in the database. The second consideration is to have a one-to-many execution engine, so one command can be sent to one or more nodes by using a single CLI command.

In order to overcome the complicated hardware, firmware and software configuration and dependencies, an automated installation mechanism can be implemented. This installation can extract known information from the database, detect runtime environment and install all the needed components to the centralized management console and remotely update the management components for managed nodes. This installation can resolve the complicated dependencies. After the single interface is installed, this interface will scan all the possible management fabrics to auto discover existing management hardware, firmware and software, then update the corresponding node information in the database.

In order to make a single CLI work properly, all existing CLI commands and sub-commands are entered into the database. An intelligent parser should be implemented. Thus when a user enters a command, the parser will convert the high level command to the exact corresponding command based on the hardware, firmware and software information stored in the database, and then submit the command to the

managed node. A plug-in mechanism should be implemented for the managed node level command set expansion, to allow for future platform and management specification compatibility.

A self-contained installer mechanism is also implemented to resolve firmware and software dependencies along with providing hardware level capability information. The installer will check the management console and managed node runtime environments from the firmware version up to the OS level management component versions and then migrate the runtime environment to the latest known “best fit” configuration.

When an administrator needs to remote power cycle a hung node or pull hardware information such as CPU temperature, memory error count, or hard drive health status, the administrator only needs to submit a single command to the interface and tell the interface which nodes to execute the command on. At this point, the interface will map the command to the existing CLI implementation provided by the vendor and execute the appropriate command across the nodes as directed by the administrator.

Most of the existing hardware management components, such as IPMI BMC logon and OS level hardware management agent logon have strict authentication requirements. The CLI also should implement a transparent auto authentication mechanism and group account management, so the system administrator can use a group account to remote power cycle a cluster without knowing the corresponding hardware and command set.

In order to help enhance the usability, a scenario based on-line help should be created and exact command examples are included in this on-line help.

6. More Improvements

In the prototyping phase, authors found two areas, which could be further improved to enhance the framework:

- 1) Power management: when remote powering up a large cluster, due to surge spike, power failures can occur.

- 2) Command failover: when a command failed, the administrator needs to submit a different command via a different management fabric to perform the same operation.

The simultaneous remote power up of all cluster nodes can cause unexpected power failures, such as the triggering of a circuit breaker, a blown fuse, etc. due to the large power spike. In order to prevent unexpected downtime caused by powering up a large number of nodes in a short time, the unified CLI should be implemented with a segmented parallel power up algorithm. When an administrator submits a remote power up command to a cluster, the CLI will power up the first node on every rack at the same time. After waiting for a period of time, the unified CLI will power up next row of nodes. The rack and node relationship, or power breaker and power bus can be associated using the database's grouping feature. Based on empirical measurements, certain high end servers can generate a 200 ampere spike for approximately 20ms when powering up, so the default parallel power up delay time can be set to 25ms to prevent unexpected downtime due to power spike. The node ampere and spike duration can be defined in a dedicated database to provide compatibility for future platforms. Different platforms can have different power up delay values. These delay times can be stored in the backend database.

Certain platforms have more than one management fabric, such as in-band, out-of-band, direct serial port connection, etc. The implementation expands the command mapping mechanism and creates a chain of corresponding commands. When one native command fails, the CLI will auto failover to another command. For example: when an OS level in-band command for power cycling fails, the CLI will try to find the corresponding out-of-band command, then submit the out-of-band command to the node. If the out-of-band command fails, the unified interface will try to use the direct serial connection and use the EMP feature to reset the node. If all efforts fail, then the interface will return an error message to indicate which node failed a specific command execution.

This design brings command line level automation; an administrator only needs to issue one command and the unified interface should have the knowledge and intelligence to complete the request.

7. Conclusion

This framework provides a unified command line interface (CLI) to remove the inconveniences of multi-generation cluster hardware management tools. It also provides: database class scalability, grouping features, OS and hardware/ firmware level user account and

password management, transparent authentication, one to many command submission, command mapping mechanism, runtime environment capability and compatibility verification, output format conversion, a command acknowledgement mechanism, parallel remote power up, and a command failover mechanism.

The prototype has been developed in a multi-generation cluster environment, and the prototype successfully proved the framework could enhance an administrator's performance and reduce the cost of ownership.

References

- [1] Fang, Yung-Chin; Mayerson, Jeffrey; Hsieh, Jenwei; Mashayekhi, Victor; Scott, Stephen; Naughton, Thomas, "The impact of industry standard to cluster management," High available and performance computing workshop, Santa Fe, NM Oct/2003.
- [2] Preboot Execution Environment (PXE) Version 2.1
- [3] EFI Specification 1.10 update
- [4] Advanced Configuration and Power Interface specification Revision 2.0c, August 25, 2003
- [5] Wired for Management Baseline Version 2.0
- [6] IPMI V1.5 Rev 1.1 Specification
- [7] Management Hardware Design, Interface and Layout Guidelines
- [8] National Semiconductor, "LM81 Serial Interface ACPI-Compatible Microprocessor System Hardware Monitor"
- [9] Advanced Power Management Specification V. 1.2
- [10] Emergency Management Technical Committee Requirements Working Draft: 25 March 2003
- [11] SNMP Version 3
- [12] CIM Schema v2.8
- [13] DMI v2.0s Specification
- [14] RFC 2131 Dynamic Host Configuration Protocol March 1997

- [15] Intel® Boot Integrity Services Application Programming Interface Version 1.0
- [16] DMTF DSP0132 Solution Exchange and Service Incident Specification
- [17] Thomas Naughton, Stephen L. Scott, et al., "The Penguin in the Pail -- OSCAR Cluster Installation Tool," *The 6th World MultiConference on Systemic, Cybernetics and Informatics (SCI 2002)*, Invited Session of SCI02, Commodity, High Performance Cluster Computing Technologies and Applications, Orlando, FL, USA, 2002.