# Unified Heterogeneous HPCC Hardware Management Framework

Yung-Chin Fang, Jeffrey Mayerson, Rizwan Ali, Monica Kashyap; Jenwei Hsieh; Tau Leng, Victor Mashayekhi
*Dell Inc.*
*{Yung-Chin_fang; Jeffrey_mayerson; Rizwan_ali; Monica_kashyap; Jenwei_hsieh; Tau_leng; Victor_Mashayekhi}@dell.com*

## Abstract

*The remote, hardware level management of heterogeneous clusters (such as the remote power cycling of a hung node) is a necessary task for a computer center. This task requires knowledge across multiple specifications, fabrics (hardware, firmware, software, management) and implementations. For a heterogeneous cluster environment, there is little in common across hardware level management interface implementations. In a heterogeneous HPCC, grid or cyber-infrastructure environment, there is a need to have a common hardware management interface across unique architecture, platform, firmware, software and management fabric implementations. This paper presents the framework of a unified interface across heterogeneous clusters to overcome these differences. This paper also addresses certain findings in the prototyping process.*

## 1. Management specifications

The management specifications described in this section are designed for the operational and deployment phases of the cluster life cycle model [1]. 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 network interface card ROM. EFI level network boot is implemented at? In? network interface's EFI level driver. 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.

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 to 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 is hung, 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 hung 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 has 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 hung 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), 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], DMI [13], etc. ACPI, WOL and PXE specification implementations are typically helpful to the cluster 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 advanced power management (APM) specification or their own proprietary remote power management implementation.

The WOL specification complements the ACPI power management feature. 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 specification, plus, the network interface card must support the WOL feature.

After a new node is powered up remotely, via ACPI or WOL implementation, 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 server 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 technician. Most of the popular cluster computing packages, such as Open Source Cluster Application Resources (OSCAR) [17] by the Open Cluster Group and the ROCKS by San Diego Supercomputer Center and others, facilitate the PXE implementation as the main remote system deployment mechanism building block. PXE implementation does have a hardware dependency, so a later extensible firmware interface (EFI) moved the boot-up portion of PXE to EFI level instead of 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 standardized industry-wide HPC cluster management specification. Current HPCC management approaches are designed to integrate and facilitate available implementations of these specifications to perform cluster management. To overcome this short coming, node level management features could be integrated and automated via a unified command line interface (CLI) implementation and provide cluster/grid/cyberinfrastructure level services and increase uptime.

## 2. Specifications Implementation

Standardized management specifications are defined by groups of professionals from many parties. When defining standards, there is usually some space for vendors to interpret, so each vendor can have its own implementation of a single standard. For example: IPMI 1.5 defines a set of baseboard management controller (BMC) level ASCII text command set for remote management. In order to communicate with the BMC level ASCII text command, a proxy based CLI is needed for a remote management console. Each vendor has interpreted the specification and implemented a unique command set for their own remote management console. So, even though the IPMI standard defines a command set, the remote proxy CLI for this command set will be different from vendor to vendor. The variety of implementations has caused heterogeneous cluster management inconvenience. When a system administrator needs to remote power cycle a hung node, the following items must be known:

1) What is the brand? Different vendors implement management of HW/FW/SW differently.
2) What is the architecture? Is it monolithic, blade, IA32, or IA64? Usually different architectures will lead to a different management HW/FW/SW stack implementation.
3) What is the platform model? A single vendor may have different implementations on different models. 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.
4) What are the available management fabrics? In-band, Serial-Over-LAN, Serial-Over-Telnet, EMP via dedicated Ethernet, shared Ethernet, or a direct serial connection?
5) What are the privilege requirements? Is root access required for this feature, or can a general user perform this command?
6) What is the available HW/FW/SW? A single platform may have different HW/FW/SW configurations. One configuration may support a remote management controller, one 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.
7) What are the version 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, etc.
8) What are the available management fabrics and corresponding command line interface? One platform can be equipped with more than one management fabric. For example:
    - The platform has OS level in-band manageability via in-band fabric
    - The platform is equipped with a proprietary IP addressable remote management card via a dedicated Ethernet port
    - The platform is equipped with a serial based proprietary remote management controller via a serial port connection
    - The platform is equipped with an IPMI Serial Over LAN (SOL) via a dedicated out-of-band Ethernet port
    - 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.

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. The need

To learn the management specifications and various implementations from vendors for different platforms is a very time consuming process and the implementation will change along with age. Usually a heterogeneous cluster will have more than one CLI. Different platforms from the same vendor can be equipped with different remote management CLIs. Each CLI has its own command set. A large number of management commands across generations of clusters is also a barrier for efficient remote hardware management. A unified interface to cross all hardware, firmware, software, CLIs, specifications and implementations is needed for heterogeneous cluster management.

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 single interface 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. Dependency analysis

Dependency analysis falls into two categories: (1) management console and (2) managed node. For the management console category: any hardware level remote command line interface implementation will have a dependency list, such as: platform architecture, operating system type and kernel version, development tool/environment, and runtime environment dependencies. Platform architecture dependencies can be IA32, IA64, blade and/or monolithic architecture dependent. OS type and kernel version dependencies usually are associated with development tools. An example of development tool/environment dependencies are database version, compiler version, java version, etc. Runtime environment dependencies usually indicate the settings needed for the runtime environment. Different CLIs from different vendors will have different requirements for these dependencies.

For the managed node category: architecture dependencies such as IA32, IA64, monolithic and blade management architecture is the root cause for the remote hardware management inconvenience, while embedded management feature implementation is another major dependency. Embedded management feature implementations can include: management processor type, performance and capability; number of management buses implemented; dedicated management bus (such as SM bus or $I^2C$ bus) clock rate; usable NVRAM size; management processor topology to each component; the management processor firmware implementation; OS level management agent implementation, etc.

## 5. Unified CLI framework

After understanding the specifications, the need, and investigating the complexity of the layers of dependencies, the design of a unified CLI becomes feasible.

The first consideration for heterogeneous cluster management is scalability. A backend database engine can be used for efficient 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 are all 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 heterogeneous nodes by using a single CLI command.

In order to overcome the complicated hardware, firmware and software configuration and dependency, an automated installation mechanism is implemented. This installation will extract known information from the database and install all the needed components to the centralized management console and remotely update the management components of managed nodes. This resolved the complicated dependencies. After the unified interface is installed, the 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 unified CLI, all existing CLI commands and sub-commands are entered into the database. An intelligent parser is 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, then submit the command to the managed node. A plug-in mechanism is also 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 unified interface and tell the unified interface which nodes to execute the command on. At this point, the unified interface will map the command to the existing implemented CLI from vendors 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 both have strict authentication requirements. The unified CLI also implements 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 is created and exact command examples are included in the on-line help.

## 6. Intelligence

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

1) Cluster level power management: when remote powering up a large cluster, power failures can occur.

2) Command failover: when a command failed, the administrator needs to submit another sub-command to perform the same operation.

Cluster level power management. The remote power up of a cluster 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 power up, the unified CLI also implemented a segmented parallel power up algorithm. When an administrator submits a command to remotely power up a cluster, the unified 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 can be associated in the database 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 definable in a dedicated database to provide compatibility for future platforms.

Command failover. 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 unified CLI will auto failover to another command. For example: when an OS level in-band command for power cycling fails, the unified 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 unified 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 mission.

The unified CLI can be used in scripts, in order to make it more compatible with other software. The unified CLI is designed with an XML output capability

so it can be integrated with Ganglia or other OS level monitoring/management utilities.

## 7. Conclusion

This framework provides a unified CLI interface to remove the heterogeneous cluster hardware management inconvenience. It also provides: database class scalability, grouping features, OS and hardware/firmware level user account and password management, a transparent authentication mechanism, command line level parser, a one to many command submission mechanism, a command mapping mechanism, a runtime environment capability and compatibility verification mechanism, output format conversion, a command acknowledgement mechanism, a parallel remote power up mechanism, and a command failover mechanism.

The prototype has been developed on a heterogeneous cluster environment, and the prototype successfully proved the framework can enhance an administrator's performance and reduce the cost of ownership.

To make this framework to be more thorough, SNMP services is planned for integration into the framework to provide even higher level management coverage, such as the remote management of Ethernet or other interconnect switches, the setting of the management controller IP, and the remote management of KVM switches, etc.

## 10. 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 el., "The Penguin in the Pail -- OSCAR Cluster Installation Tool," *The 6th World MultiConference on Systemic, Cybernetics and Informatics (SCI 2002)*, Invited Session of SCI'02, Commodity, High Performance Cluster Computing Technologies and Applications, Orlando, FL, USA, 2002.