Proposed Track
For
Open Hardware Management

Revision History

Date	Name	Description
12-07-2011	Grant Richard	Original draft based on the discussions before the NYC Open Compute Summit, at the Summit + individual conversions/email. Thanks to all who helped form this.
12-10-2011	Markus Fischer	General edits and clarifications. Further details on Firmware Lifecycle
12-11-2011	Joel Wineland	Various edits. Additional points added to strategic track examples.
12-22-2011	Grant Richard	Various edit from comments from 12/12 conference call.
12-12-2011	Matthew Liste	Various edits to smooth document and clarify points.

Contents

Summary

Open Compute's focus is on server hardware, with much of its effort concentrating on machine, cabinet and data centers design. Yet, a significant part of scale computing is managing the firmware that makes the machines run, alerting about their health, discovering their resources and accessing them remotely. For scale computing, tooling and standards around firmware behavior and maintenance is required to reduce the support burden and secure machines. These low/no touch, scalable, scriptable, secure and standard tools must maintain firmware versions/configurations, remote access solutions, retrieve machine configuration/information and fault/event alerts.

Presently, these tools are mostly platform/vendor specific and non-scalable. Because of this, existing scale compute users must create and maintain systems and processes to unify and augment existing tools -- with *every individual* scale users largely creating the same toolset. The OCP Hardware Management Track will address these issues by providing consistent, scalable and open source standards and tools for Open Compute servers.

License

As of April 7, 2011, the following persons or entities have made this Specification available under the Open Web Foundation Final Specification Agreement (OWFa 1.0), which is available at http://www.openwebfoundation.org/legal/the-owf-1-0 agreements/owfa-1-0:

Facebook, Inc.

You can review the signed copies of the Open Web Foundation Agreement Version 1.0 for this Specification at http://opencompute.org/licensing/, which may also include additional parties to those listed above.

Your use of this Specification may be subject to other third party rights. THIS SPECIFICATION IS PROVIDED "AS IS." The contributors expressly disclaim any warranties (express, implied, or otherwise), including implied warranties of merchantability, noninfringement, fitness for a particular purpose, or title, related to the Specification. The entire risk as to implementing or otherwise using the Specification is assumed by the Specification implementer and user. IN NO EVENT WILL ANY PARTY BE LIABLE TO ANY OTHER PARTY FOR LOST PROFITS OR ANY FORM OF INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION OR ITS GOVERNING AGREEMENT, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND WHETHER OR NOT THE OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Overview

Hardware Management can be thought of in four areas:

1. Firmware Lifecycle
2. Event, Alert and Logs
3. Remote Operations
4. Strategic Technologies

The problem space is:

- Inability to deploy firmware fixes and configuration changes quickly and at scale
- Instabilities due to many versions of firmware in different combinations
- Lack of agility caused by having to integrate new tools / new vendors into existing home-grown Hardware Management environments
- Lengthy and brittle bootstrap processes when attempting to automate firmware load and configuration process
- The bugs in unused portions of over-featured products cause unnecessary firmware updates
- Tooling is vendor specific and often does not scale to many tens of thousands of machines.

For these reasons, it is critical to maintain a machine's firmware and those processes must scale. Further examples of the larger problem set are:

1. There are no broadly adopted and consistently implemented capabilities around basic machine maintenance.
    - Vendor written maintenance tools *with identical functions* for the alerts/events, remote operations and firmware lifecycle are not interoperable. Vendor X tools can only be applied to their machines and can't be used on Vendor Y even though they perform the same functions.
    - For standard based processes like event/alert/log data and remote management, even though the delivery may be the same (SNMP, IPMI and syslog) the message numberings, payloads or commands may be different.

2. Software deployed across identical hardware with different firmware will appear to randomly fault due to lack of ability to test across all permutations and to discover toxic combinations of individual firmware payloads/configurations.

- In the first three years of a machine or add-in card's introduction, there are typically many (3-10) firmware releases for standard motherboard components that often need to be quickly rolled out.
- Lack of tools to rapidly and securely deploy firmware configuration and binaries.
- Vendor specific firmware lifecycle solutions generally deploy firmware payload in a piecemeal fashion.
- When vendors test their firmware as an integrated stack, PCI add-in components from different OEMs have a different revision release cadence and payload delivery system.
- Firmware configuration is often embedded in firmware payloads complicating configuration visibility and management.
- No broadly adopted standard for deployment of firmware bits or configurations exists.

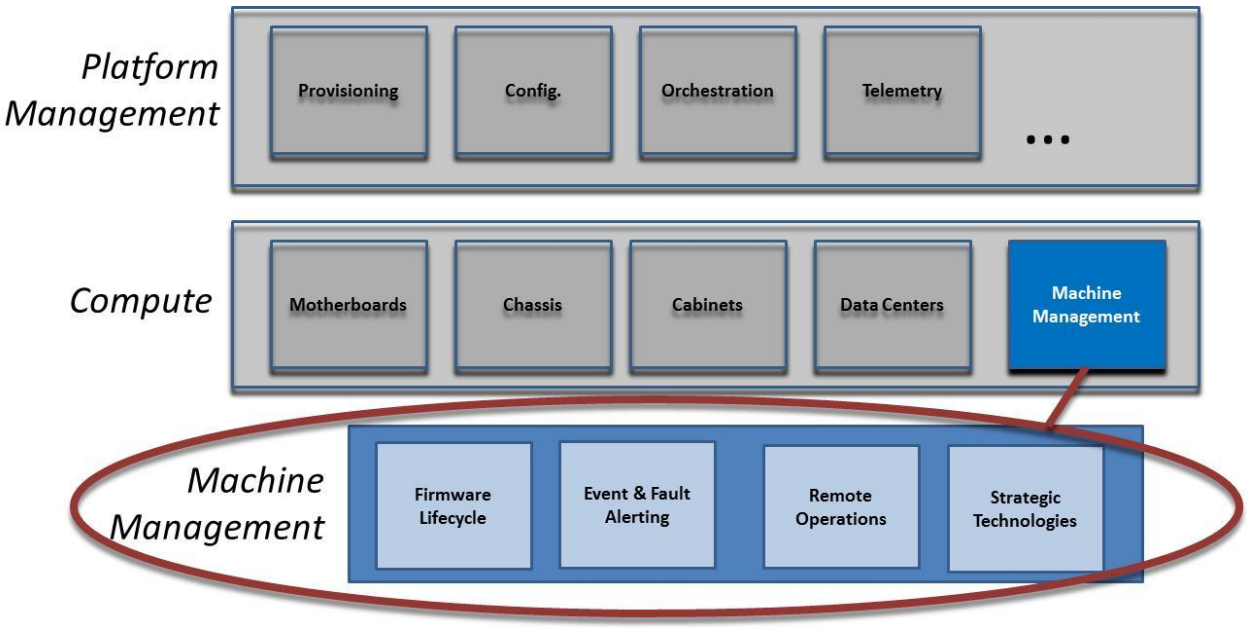
3. Expansive or inappropriately featured vendor remote management tools

    - Scale users' need very few functions.
    - Complex remote management products can introduce more firmware and machine instability.

4. Need to manage machines holistically as part of the overall data center management

    - No standard ways to integration of machine information into data center BMS/DCIS systems

These are some of the reasons it is important to include Open Hardware Management as part of the Open Compute effort and to establish it as a Track. Standardization for API/Interface and consistent tools will lower adoption barriers and simplify management of scale computing environments.



It is equally important to understand what Open Hardware Management is not. In this effort, we are making a distinction between Hardware Management and Platform Management. The Hardware Management effort is bounded by 1) the firmware on the machine, 2) event/alerting/logging about machine components and 3) basic remote management. Platform management encompasses all the other functions and is primarily about deploying and management operating systems and applications.

It is understood that these boundaries are somewhat arbitrary. Another, hopefully temporary, difference is that the items identified as Platform Management have multiple cross platform open and closed sourced solutions so there are good options whereas most Hardware Management tools provide few, if any, good cross vendor options.

It is important that Open Compute should not forget to enable Hardware Management technologies and Open Platform Management will contribute to this by providing an API/interface to access Open Hardware Management functions uniformly across all OCP platforms.

Mission Statement

During our breakout session in the past Open Compute Summit, the participants created the following mission statement that frames our effort.

"Provide uniform management of firmware, alerting of hardware events and remote hardware access. Our focus will be on process automation and scalability by leveraging existing open standards whenever possible. We will coordinate with other groups within the OCP foundation to drive efforts."

Approach

Using the Mission Statement, our approach to delivering and maintaining projects within this track are:

- Any solution must be scale to 100,000+ of physical/virtual machines.
- Use existing standards whenever possible. If standards require adjustment, partner with standards bodies to make changes. Examples of Platform Management relevant standards include:
    - DMTF – SMBIOS, CIM, WBEM, SMASH, ASF, WS-Man, SMI-S
    - Intel – (Broad adoption) DCMI, IPMI, IPMB, ICMB
    - AMD – OPMA
    - IETF – SNMP, NETCONF
    - SAForum – HPI
    - OASIS – DPWS, WSDM
    - SMIF – SMBUS – (Orig Intel), PMBUS
    - PICMG – ATCA (IPMI / IPMB / ICMP – mgt)
- Define requisite features, attributes and interface standards for Open Compute hardware designs.
- Encourage and work with Platform Management tools vendors to use OCP Hardware Management interfaces/APIs.
- Work with OCP Hardware designers to implement required functionality
- For each function, there will be “base” set that defines the minimal functionally to support scale computing. And there may be an “extended” set that may be developed that would serve scale compute provider with increased needs.
- As part of the every deliverable, a methodology to validate the functionality and maintain validity thereof must be included.
- As it is common to have some non-scale platforms in a scale environment, encourage closed source and specialty hardware manufacturers to comply with Open Hardware Management and to submit those platforms for validation.
- For every tool or function, make sure that security is considered and that tool use and firmware deployment will not be compromised.

Focus Areas

Logically, the Hardware Management Track divides into four areas of focus, that can be thought of as sub-projects that integrate into common solutions for Hardware Management.

It is imperative for each sub-project not only to define the standard, but also to define a way of testing and validating compliance with the defined standard.

Sub-Project	***Description***
Firmware Lifecycle	To provide a uniform interface to independently deploy and update firmware's binaries and configurations
Events, Alerts and Logs	Standard way for OCP machines to produce and format machine event, and logged messages.
Remote Management	Consistent way to remotely explore a machines configuration and perform systems operations such as reboot and open a remote console.
Strategic Technologies	Follow and encourage exploration of products and standards of potential benefit for future Open Compute specifications. Such activities may include survey of alternative system management wireline protocols , integration with data center building management systems, management coprocessor alternatives and etc.

Firmware Lifecycle

Most physical compute components have associated software most often referred to as firmware. Common firmware examples include BIOS, NIC and BMC firmware. Frequently, and especially during the early part of the component's life, firmware is revised to fix bugs and introduce capabilities.

In scale environments, it is important be able to rapidly deploy, securely update and have known combinations of firmware. Over the years, there have been toxic combinations between different versions of firmware on motherboards and components (e.g. conflicts between BIOS and NIC firmware) and firmware and OSs [like BIOS version X and Linux version Y]). To further complicate the task:

- Motherboard and component manufactures often ship the most recent version of the firmware when delivering components, so even for the same motherboard or component, there will be multiple versions without coordinated firmware management.
- The cadence between firmware revisions is different for each manufacturer.
- Each manufacturer creates their environment for delivering the firmware.
- Manufactures firmware update tools do not always run in the same OS environment – causing any solution to have multiple boots to apply updates.
- The delivering of the firmware software updates includes configuration as well.

For these reasons, it is important that deploying and updating both the firmware payload and its configuration be available for each OCP compute platform.

Approach

- All components with firmware are in scope. Examples include motherboards, NICs, PCI SSD, HBAs and RAID Controllers.
- Develop architecture and requirements for the firmware lifecycle configuration, deployment, updating, security and auditing firmware.
- Any solution will have the capabilities of deploying the firmware's configuration separately.
- Specify management framework/API/interfaces for providing this service to permit platform tool provider's access to integrate this with their products.
- The solution needs to include a "push" and a "pull" model. A standalone method may also be needed as there may be no way to remotely recover from a failed firmware update
- Firmware and configuration needs to be deliverable through a centralized server via the network.
- Firmware and configuration needs to be changeable with or without an OS running.

The general strategy is:

- Firmware is a “flat file” that needs to exist in the right format and placed in the right location.
- Base firmware configuration is a flat file that needs to exist in the right format and placed in the right location.
- This combination of firmware and firmware configuration needs additional properties , for example, checksums, version numbers and dependencies
- The total process needs a fall-back in case of catastrophic failure (either firmware or firmware configuration is corrupt / does not match checksum)
- Tools are required for the user to generate the firmware configuration (this could be as simple as a text editor creating an XML file)
- The firmware and firmware configuration combo would be repeated for every component in the system that requires firmware

Sub-projects for Firmware Lifecycle

OCP Firmware Lifecycle

Task	Through member code donation and development, create a framework for independently distributing firmware binaries and configuration.
Effort	- Create architecture and design document - Survey member tools to establish base version - Indentify contributors to project - Co-ordinate the releases and test cycles
Time	Nine to twelve months to initial draft vote

Sub-Project: Events, Alerts and Logs

Automation of knowledge of a standard machine condition is at the heart of a scale environment. For automation, ***Alerts***, ***Events*** and ***Logs*** need to be reliable and standard.

For purposes of this document, they are defined as:

- ***Event*** is a recorded machine state change that has significance
- ***Alert*** is an urgent notification of event.
- ***Log*** contains a collection of events and also refers to the placing of events into a collection.

Events and Alerts can be recorded a number of ways:

1. SNMP[1] (Simple Network Management Protocol)
2. WS-MAN[2] (WS-Management)
3. Syslog[3]
4. Publish/subscribe eventing services: allowing other devices to subscribe to asynchronous event messages produced by a given service. WS-Eventing / WS-Notification

Approach

- Define consistent event numbers and associated text payload information. For example, event 501 would always be associated with a disk fault and the payload would contain the message "Disk fault – Drive %" where % is the drive identifier.
- Leverage SNMP/syslog for "base" functionality and SNMP/syslog/WS-Man for "extended" functionality.
- Whenever possible, define both a push and pull method for collecting event and alert information.
- Concentrate on the standardization of the events rather than the actual implementations.
- Message number and message payloads will be consistent when delivered by different ways (like SNMP, WS-MAN & syslog).
- The approach should accommodate both in-band and out of band agents.
- Define mechanisms to validate and secure Event / Alert notification transports.

[1] http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol
[2] http://en.wikipedia.org/wiki/WS-MAN
[3] http://en.wikipedia.org/wiki/Syslog

Sub-projects for Events, Alerts and Logs

Base Event Message Standardization

Task	Define a minimum (base) set of events, their event # and their accompanying messages
Effort	- Research and identify requisite events/alerts - Determine payload format - Assign each event a unique number and create payload information - Draft proposal for member approval
Time	Two to four months

Hardware Management OCP SNMP MIB

Task	Create OCP SNMP MIB
Effort	- Review open source MIBs (like IF.MIB) that can be incorporated - Solicit MIB donations from existing hardware vendors - Integrate OMM base events messages in MIB - Draft proposal for member approval
Time	Three to five months

Syslog Standardization

Task	Standardize local and remote syslog and log using OCP Hardware Management standard messages
Effort	- Research and document standard strategies for local and remote syslog - Integrate OMM base events into syslog - Draft proposal for member approval
Time	Three to six months

Wake-on-LAN / Wake-on-Reboot

Task	Document using existing standard an implementation of WOL and WOR.
Effort	- Research and document existing solution for WOL & WOR and performance gap analysis - Draft recommendation and proposal for member approval - Work with OCP Compute Track to implement recommendations
Time	Two to four months

Reference Design for Event/Alert/Log Repository

Task	Create and implement reference design for an event repository to permit historical and filtered queries.
Effort	- Research and document existing systems (OpenTSDB?) - Survey contributors - Draft recommendation and proposal for member approval on architecture - Source effort to complete project
Time	Three to Nine months

Remote Hardware Management

Scale environments require a way to perform operations that in non-scale environments are manually performed at the console. Basic remote operations must be scriptable on every OCP machine as well as an extended set for those machines that either have increased complexity and/or greater service levels.

The examples of remote machines management operation that this Sub-Project is concerned with are:

- Remote power on / off
- Remote console
- Discover a machine's hardware/firmware configuration
- Soft reboot / shutdown
- Graphical console / VGA redirect
- Wake-on-LAN (WOL) and Wake-on-Reboot (WOR)
- Basic authentication / LDAP authentication

Approach

- Delineate the remote management capabilities that fall with the Sub-Project.
- Categorize the remote management capabilities into two sets: 1) Basic that must be present in all machines and 2) Extended that may vary from platform to platform but will always have a uniform interface and performance.
- Provide both individually managed and directory based authorization.
- Survey existing remote management technologies and implementations to determine the best technology to leverage. Identify gaps between Remote Hardware Management and existing standards. This will provide command line and API/interfaces.

Sub-projects for Remote Management

Below is a high level description of each sub-project.

Base Set – Remote Management through IPMI

Task	Standardize IPMI calls for Remote Power on/off and Remote Serial Console implementations
Effort	- Review materials for IPMI's open source API/Interface and tools. - Understand gap between current OCP BMCs and proposed standard. - Draft proposal for member approval that includes recommendation, specification and certification. - Member discussion and vote - Work with OCP Compute Hardware track to include in spec
Time	Two to three months

Extended Set – Remote Management through IPMI

Task	Extend the Base Remote Management feature set to include features that are not required by all scale compute users. These features may include VGA redirect/remote graphical console, advanced power management and monitoring, etc.
Effort	- Understand all the possible features - Need policy for compliance - Draft proposal for member approval - Work with OCP Compute track and closed source hardware vendors to implement
Time	Three to six months after Track Approval

Strategic Enabling Technology

Given the legacy of personal computers that extended to servers, the design of machines management was "bolted" on rather than designed into the platform. There are a number of efforts to provide a richer management network and stronger integration with external systems that have the promise of increasing efficiency and easing management.

Examples of these are:

- Separate, Subordinate or multi-system Management Networks
  - I2c-Bus
  - RS485
  - VDM over PCIe
- Consolidated hardware management practices between components that comprise a cabinet (servers, storage, network, MOAs,etc.) through a variety of DCIM vendors.

Sub-projects for Strategic Enabling Technologies

Dedicated System Management Buses

Task	Investigate alternative uses of I2c-Bus, SMBus, and VDM over PCIe, RS485 etc. with various Open Compute design track.
Effort	- Work with other Open Compute tracks to understand technical and cost implications for including these. - Develop overall strategy for using dedicated system management buses
Time	Six to twelve months

Integration with data center control systems

Task	Coordinate or co-develop with OCP Data Center track to integrate into their Data Center Information Management systems (DCIM).
Effort	- Work with Data Center Track to understand integration opportunities - Research open and closed source DCIM solutions - Develop bi-direction information and control standards - Work with OCP/open source DCIM providers to implement standards.
Time	Nine to twenty-four months.

Organization & Meeting Cadence

Organization

Although there are various roles within the organization (like committer, etc.), there will be three main types of participates in the project:

1. Project Chairs – who will facilitate the flow of information, determine consensus and commit documents.
2. Working Group – are track member who are committed moving the project forward between meetings. Examples of contributions can be advice, specification and code.
3. Advisory – who are the people who are engaged in monthly meetings and discussions.
4. General Assembly - who are people are following the topic and want to be part of the decision process.

Meeting Cadence

The formal meetings will have the following meeting schedules:

Working Group	Will meet as needed between other project formal meetings. A notice of any meeting /conference call will be sent to the general list for anyone interested.
General Assemblies	Will be co-terminus with the Open Compute Summits. These meeting will be for a wider audience with update on the past efforts and anticipated progress.
Advisory	Will be take place approximately every month and will discuss the progress made, open issues and anticipated progress. These calls are intended provide direction/focus of efforts and approve any new projects.

It is anticipated the Sub-Project meeting cadences will follow this pattern although Sub-Projects may decide on different cadences based on their requirements.

Sub-Projects and Groups

The information below is to provide a high level view on the initial organization and its priorities. This will be further developed once the charter is approved.

Sub-Projects in Priority Order

Based on a preliminary survey, the following sub-projects are listed in priority rank order:

Sub-Project	Rank
Systems Management: Coordinate with Data Center Track to integrate into their management systems	1
Remote Management: Standardize IPMI calls for Remote Power on/off and Remote Serial Console	2
SNMP Alerting: Define a minimum (base) set of events, their event ID and their accompanying messages	3
Systems Management: Investigate I2x-Bus and SMBus with OCP Compute Track	4
SNMP Alerting: Create OCP SNMP MIB for minimum set of events	5
Firmware Management: Through member code donation and development, create a framework for independently distributing firmware binaries and configuration.	5
Syslog: Standardize local and remote syslog and log using OCP Machine Management standard messages	6
Remote Management: Extend the base IPMI features to include functionality that are not required by all scale compute users. These are features such as VGA redirect/remote graphical console, etc.	7
Remote Management: Document using existing standard an implementation of Wake-on-LAN and Wake-on-Reboot.	8

Groups

Co-Chairs: Matthew Liste & Grant Richard

Working Group: Matthew Liste, Grant Richard, Markus Fischer, Joel Wineland and John Keveney

Advisory: Dlist of Open Compute / Hardware Management (hardwaremngt@opencompute.org)