Hardware Discovery and Maintenance Workflows in Shasta Systems

Steven Presser
Hardware Management Services
Cray Inc.
Minneapolis, MN, USA
spresser@cray.com

Brent Shields
Hardware Management Services
Cray Inc.
Minneapolis, MN, USA
bshields@cray.com

Abstract— **Cray's Shasta supercomputers support more varied hardware than any previous Cray system. This includes a significantly wider variety of processors, coprocessors, and accelerators than has previously been available on Cray systems. Further, Cray is supporting the use of certain commodity hardware in Shasta systems. The more complicated hardware ecosystem in Shasta makes hardware management more complicated than previous Cray systems.**

However, Cray is building solutions to minimize this complexity, with a goal of making hardware management easier under Shasta than previous Cray systems. This paper will cover Hardware State Manager (HSM), River Endpoint Discovery Service (REDS), Mountain Endpoint Discovery Service (MEDS), and Ideal Design of Equipment, Architecture, and Layout Service (IDEALS). These four services cooperate to geolocate, initialize, discover and track the hardware within a Shasta system. This paper begins with an introduction to these services and their responsibilities, continue by discussing the workflows used to manage hardware with these services, and conclude with a comparison to XC systems.

Keywords: Computer interfaces, Computer network management, Supercomputers, High performance computing

I. Introduction

Cray's next-generation Shasta systems are designed to break the exascale barrier and to be more flexible than any previous Cray system. A portion of this flexibility is a significant expansion of hardware options. By expanding the number of available hardware options, Cray will allow customers to pick hardware that more closely matches their needs than was previously possible.

One of the major changes Cray is making is the availability of two major categories of hardware packaging: Mountain and River.

Mountain hardware is highly specialized, engineered by Cray to provide as much computing power in as little space as possible. Mountain hardware is highly integrated and Cray controls all elements of the hardware. Mountain hardware offers the highly-integrated and well-supported supercomputer experience Cray is famous for.

In contrast, River hardware is commodity hardware that is not altered by Cray. River hardware is intended to allow access to a wider variety of hardware options than Mountain, including specialty hardware (such as certain accelerators). However, since Cray does not customize this hardware, it requires more effort to maintain and has fewer features.

This paper examines four major tasks performed on these types of hardware: endpoint discovery, initialization, geolocation, and inventory discovery of hardware in Cray Shasta systems. Endpoint discovery locates hardware on the hardware management network. Initialization is the process of loading basic configuration onto compute nodes. Geolocation assigns an identity to the node based on its physical location. Finally, inventory discovery identifies all the hardware in managed devices, including network addresses, component serial numbers, and more. This completes discovery and enables the use of the high speed network, if any. Together, these tasks are the discovery story.

Cray's Shasta supercomputers use microservices to ensure the individual components of the system are highly available and independently replaceable. The discovery story uses four microservices: IDEALS (Ideal Design of Equipment, Architecture, and Layout Service), REDS (River Endpoint Discovery Service), MEDS (Mountain Endpoint Discovery Service), and HSM (Hardware State Manager). All the services primarily interact with the out-of-band hardware management interface – e.g.: Baseboard Management Controller (BMC) on River nodes or node controller (nC) on Mountain nodes.

Portions of the technologies being discussed in this paper are in the process of implementation and are subject to change.

II. Technical Specifications

Several technical specifications are used by the services. This section discusses each of these specifications in sufficient detail for this paper. Interested readers may examine each specification in more depth, if desired.

A. Redfish

The most prominent of these technical specifications is Redfish. Redfish is a vendor-agnostic interface for interacting with hardware out of band [1], published by the DMTF (formerly the Distributed Management Task Force). It presents a standardized interface to any supported hardware. This interface is a RESTful HTTP interface, meaning that it can be interacted with using standard HTTP tools. Further, the standard is extensible, meaning that it can cover a myriad of different hardware configurations and be customized by vendors.

In Shasta systems, Redfish is used to communicate with the out-of-band management controllers in all devices which support it.

B. SNMP

SNMP (Simple Network Management Protocol) is a generic transport that allows communication with hardware management interfaces on a network [2]. Cray uses SNMPv3 for security reasons. SNMP is a popular standard in network switches and has been extensively standardized. SNMP is deployed on many types of hardware that do not yet have Redfish support, and is most frequently used to interact with hardware which does not yet have Redfish management interfaces.

C. IPMI

IPMI (Intelligent Platform Management Interface) is an out of band hardware management scheme used by server-grade hardware [3]. Typically IPMI is implemented on the BMC. IPMI allows users to control the power state of a server, connect to a serial console remotely, interrogate some properties and sensors in systems, or even inject raw commands onto a serial bus. In Shasta, most IPMI functions are replaced by Redfish and IPMI is only used to perform configuration of the Redfish interface.

D. SSDP

SSDP (Simple Service Discovery Protocol) is a multicast, HTTP-like protocol for services to announce their presence on a network [4]. SSDP is used by Cray's Mountain hardware to announce when it is available on the hardware management network and is the backbone of Mountain hardware discovery.

III. Tasks

The discovery story consists of a number of separate tasks: endpoint discovery, initialization, geolocation, and inventory discovery. We will discuss each of these, with a specific focus on the responsibilities required of the implementation of each task.

A. Endpoint Discovery

The endpoint discovery task consists of determining when a new endpoint is present on the hardware management network and configuring it. For example, when a compute node first boots, this task requires determining that a new out-of-band system management interface is available on the network. An implementation of endpoint discovery will communicate with network infrastructure (e.g.: switches) to determine when the network topology changes and how. It will further communicate with the infrastructure to determine how new elements of the network topology may be communicated with.

B. Initialization

Initialization is the process of configuring hardware in the system to work correctly with the rest of the system. This means configuring the IP address, credentials/authentication, and other information on the out-of-band system management interface (e.g.: the BMC).

C. Geolocation

Cray's Shasta systems assign every node an ID based on their physical location in a system. In Shasta, this is known as an xname and encodes both physical location as well as hardware type and responsibility. They bear significant similarities to cnames [5] in previous Cray systems. This makes the use of a system significantly easier in a number of ways. For example, if a node fails, it makes it very easy to locate and replace it. The xname thus provides a constant identifier for a system location, regardless of the physical component that resides there. It can also be used to indicate a location is valid but currently unpopulated.

D. Inventory Discovery

Finally, the Inventory Discovery process uses the Redfish interface to determine what hardware is available in each compute node as well as its initial state. This information is used to enable the use of the system on the high-speed and node management networks, for administrative tasks, such as collecting processor and memory properties for job scheduling, and system maintenance tasks, such as recording the serial and part numbers of Field Replaceable Units (FRUs) This level of discovery detail allows tracking of hardware as it moves through the system and allows administrators to examine snapshots of the historical state of the system. While endpoint discovery, initialization, and geolocation all generally take place only when hardware is first introduced to a system, inventory discovery re-occurs periodically to ensure hardware changes are detected and recorded.

IV. Services

There are four major services involved in hardware discovery and maintenance. This section will discuss each in depth.

These services are designed to work well in large system builds. Cray's large systems are often too large to be built and tested entirely in Cray's factory space and are never completely assembled until they arrive at the customer facility. Further, in larger systems, it is a statistical certainty that hardware will fail. Often it's a statistical certainty that hardware will fail during each system boot. From this, we extracted several primary design requirements:

1. The compute nodes must be discoverable and bootable even if all hardware is not present
2. It must be possible to add or remove compute nodes without affecting other compute nodes
3. Discovery should be an ongoing process so that a minimum of commands are required to replace failed hardware. Ideally failed hardware can be replaced without interacting with the management software at any point.

Figure 1 shows the information flow between the various services. This figure includes all services discussed, plus the Boot Script Service (BSS) and Artifact Repository Service (ARS). BSS manages which image is booted on a node, while

ARS stores the images and transfers them to the nodes on boot.

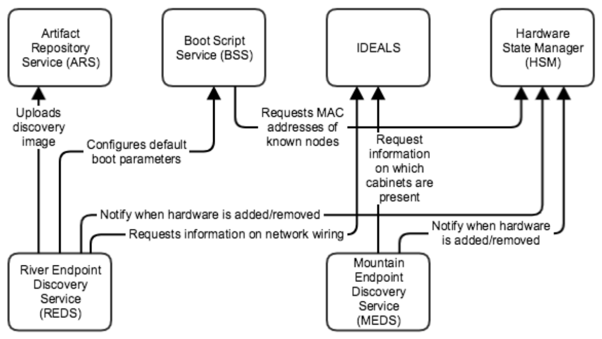
Artifact Repository Service (ARS)
Boot Script Service (BSS)
IDEALS
Hardware State Manager (HSM)
Uploads discovery image
Configures default boot parameters
Requests MAC addresses of known nodes
Request information on which cabinets are present
Notify when hardware is added/removed
Requests information on network wiring
Notify when hardware is added/removed
River Endpoint Discovery Service (REDS)
Mountain Endpoint Discovery Service (MEDS)

Figure 1: Information flow between hardware microservices

A. HSM

Hardware State Manager (HSM) is the "hub" of the hardware management story. HSM serves a number of functions, but primarily serves as both the main collection point for hardware state and inventory data, as well as the main distribution point of that data to the rest of the system and users. HSM responds to queries as well as transmitting events when important changes occur.

As its name implies, HSM is responsible for tracking the state of system hardware necessary for the management of the system – if it is currently on, off, behaving abnormally, missing from the system, or otherwise inaccessible. In addition, HSM assembles a detailed inventory of system hardware – nodes, memory modules, network cards and other field replaceable units. HSM also tracks the historical location of these physical components, enabling administrators to view the historical composition of the system at any point in time and retain detailed information about every component both past and present.

Hardware state manager also functions as the source of truth on system hardware during interactions with the rest of Cray's software ecosystem and bootstraps many important system services. For example, hardware cannot be booted until it the information required to do so is available to the boot service via HSM. Another example is HSM providing detailed node inventory information to support job scheduling.

To assemble this inventory, HSM performs inventory discovery on system hardware by interrogating BMCs, nCs, and other embedded management controllers whenever endpoint discovery finds one. It also continues to monitor and update the state of these components, and repeats the inventory process periodically in response to system events, such as when a piece of hardware may have been swapped with another. As an example, the inventory discovery process for Redfish endpoints involves HSM learning about a controller and then walking the entirety of the Redfish tree, extracting information about whatever hardware that controller manages. This includes detailed processor, memory and network interface properties, important identifying information such as MAC addresses, serial and part numbers, as well as a variety of other data. Upon completion, the collected state and inventory information is available to the rest of the system via HSM.

B. IDEALS

IDEALS (Ideal Design of Equipment, Architecture and Layout Service) defines how the system as a whole is intended to be constituted. Shasta systems have a huge number of possible layouts and configurations. For flexibility, Cray does not impose a system model (e.g.: a 2-dimensional grid) on Shasta systems. IDEALS represents the system based on existing sources like manufacturing specifications. The data provided by IDEALS provides the scaffolding that allows endpoint and inventory discovery to work.

IDEALS details the physical locations of network hardware, compute nodes and cabinets. Further, it stores information about the network, such as which port on which switch should be connected to each compute node. IDEALS does not store the details of the actual hardware (e.g.: hardware identifiers). IDEALS thus does not need to change as hardware within the system is replaced, only as the system constitution changes.

IDEALS presents a simple to use HTTP API for querying the stored information. Cray does not anticipate customers needing to interact with IDEALS frequently or in a significant manner, unless making changes to system hardware that were not planned for at system creation. However, interaction with IDEALS is required if the system setup changes – for example, if system cabling is altered or if the system is expanded or reduced.

C. REDS

The River Endpoint Discovery Service (REDS) manages Endpoint Discovery, Initialization, and Geolocation of River hardware. Because it interacts with commodity off-the-shelf (COTS) hardware, it is relatively more complex than the other services. REDS only interacts with the rest of the system when an "unknown" compute node attempts to boot on the network. A compute node is unknown if it is not already listed in hardware state manager.

Customers should have to interact with REDS only to correct error conditions.

REDS consists of two major components: a daemon hosted in the management plane and a discovery image that is booted on compute nodes. These two cooperate to gather core information about compute nodes and network addresses. This core information is then used to identify the node and send information about its BMC to HSM (which then performs detailed hardware discovery).

First, the discovery image. The image is a standard PXE boot image. When a node attempts to PXE boot and is not known to HSM, it is told to boot the discovery image. This image contacts the REDS daemon to get configuration information (such as authentication credentials for the BMC) and configures the BMC. It then gathers information about the BMC (such as the IP address assigned to the BMC and the MAC address of the BMC interfaces) and sends this to the daemon.

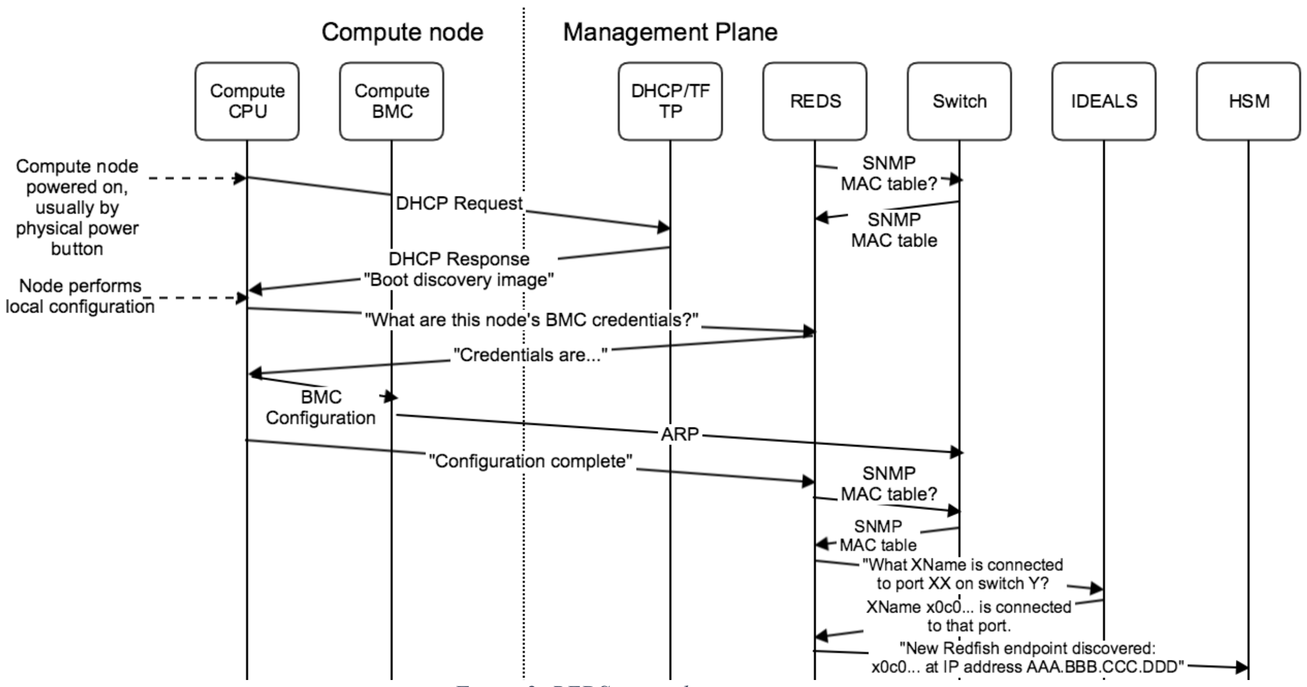
Compute node
Management Plane
Compute CPU
Compute BMC
DHCP/TFTP
REDS
Switch
IDEALS
HSM
Compute node powered on, usually by physical power button
DHCP Request
SNMP MAC table?
SNMP MAC table
DHCP Response "Boot discovery image"
Node performs local configuration
"What are this node's BMC credentials?"
"Credentials are..."
BMC Configuration
ARP
"Configuration complete"
SNMP MAC table?
SNMP MAC table
"What XName is connected to port XX on switch Y?
XName x0c0... is connected to that port.
"New Redfish endpoint discovered: x0c0... at IP address AAA.BBB.CCC.DDD"

Figure 2: REDS network communications

The daemon acts as a coordinator, taking in information from multiple sources and using it to perform identification of the nodes. Broadly speaking, the daemon associates an IP address to an xname (and thus a location) by associating the IP address to a MAC address in the discovery image. It then determines which switch and switch port that MAC address is connected to. Finally, it queries the switch and switch port in IDEALS to determine what xname is connect to that port.

For a visual representation of this process, see Figure 2. Figure 2 shows the process of performing endpoint discovery on a single node. It begins as the node is first powered on. The left-hand side is operations performed on the compute node while the right is operations performed in the management plane. Note that interaction between REDS and the switches happens asynchronously from interaction between REDS and the compute node.

The daemon has two major components: an HTTP module to handle interactions with compute nodes and an SNMP module to handle interaction with network switches.

The HTTP module exists to handle information flow to and from the discovery image booted on the compute nodes.

The SNMP module gathers information on which MAC address is plugged into which port on each switch. This information is combined with the MAC address discovered in the discovery image to allow a bridge between the network layout and the physical layout of the system. The SNMP module also determines when a node is no longer available on the network by when it disappears from the switch it is connected to.

The core of the daemon combines the information from the HTTP and SNMP modules. Once the daemon has an IP address, MAC address, username and password for a BMC, it notifies HSM that a new endpoint exists. In the case of a disappearing node, it determines which xname matches the MAC address that has disappeared, the notifies HSM that the node is no longer present.

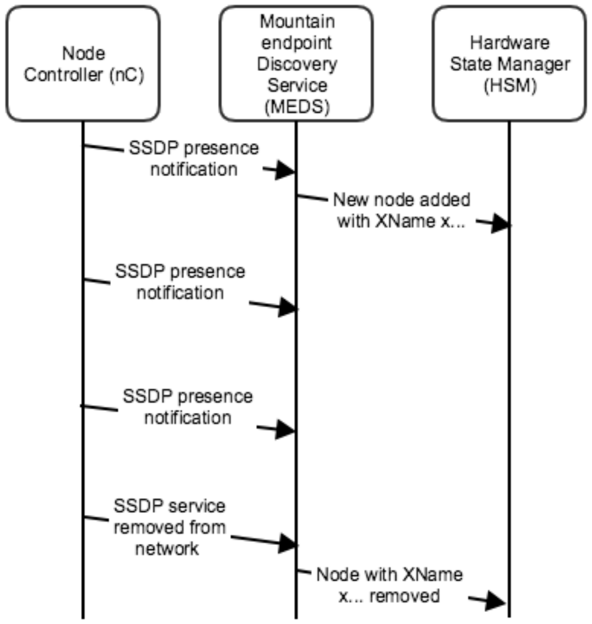
Node Controller (nC)
Mountain endpoint Discovery Service (MEDS)
Hardware State Manager (HSM)
SSDP presence notification
New node added with XName x...
SSDP presence notification
SSDP presence notification
SSDP service removed from network
Node with XName x... removed

Figure 3: Network operations of MEDS.

D. MEDS

The Mountain Endpoint Discovery Service (MEDS) is an extremely simple service. Shasta Mountain hardware assigns itself a hostname, MAC and IP address based on its location, which it is able to determine by reading from an embedded cabinet controller. It also registers via SSDP upon power-on.

MEDS receives the SSDP registrations, which include all the information required by HSM to manage the node. In short, MEDS is simply a translation layer between the Mountain Hardware's SSDP broadcasts and HSM's API. Customers should interact with MEDS only to correct error conditions.

MEDS has a second functionality: determining when hardware has left the network. Hardware may leave the network gracefully (gives notice it is going to disappear) or un-gracefully (without notice). SSDP specifies that services should periodically broadcast while still available and should broadcast when shutting down. In the graceful case, MEDS simply translates the shutting down notifications for the hardware into notifications to HSM. In the un-graceful case, MEDS notes that it is no longer receiving heartbeat broadcasts for the hardware and notifies HSM that it has been removed.

The network operations of MEDS are shown in Figure 3. This diagram shows communication between the Mountain Node Controller (an out-of-band management interface), MEDS, and HSM. The node controller sends SSDP notifications periodically, as well as when the device will be removed from the network. This diagram does not display MEDS detecting that a node has left the network un-gracefully (i.e.: without sending a notification)

V. Workflows

Next, we will discuss the most common workflows administrators are expected to encounter while working with Cray's Shasta systems. We'll start with the smallest cases (changing single nodes) and work up to the more complex scenarios (system first boot).

A. *Adding a node*

In the adding a node case, we assume that the system is being expanded. This section will walk through a single node being added to the system for the first time. It assumes the hardware is new, but that the node is already in the system map managed by IDEALS.

1) *Mountain*

Mountain hardware self-configures on startup. It needs merely be plugged in to self-configure and join the system.

2) *River*

River hardware discovery is more complex than that of Mountain. It requires nodes to boot and run an image, rather than taking advantage of Cray-built and -controlled embedded controllers. It also has fewer available mechanisms to give feedback to the management software in the event of a failure during endpoint discovery.

For this reason, River hardware behaves in a very specific manner on first boot to give clear indicators to administrators on failure. If a River node successfully discovers after being powered on, it will power itself off. This gives a clear indicator of success. If, however, the node fails for any reason, it will remain powered on with the logs visible on the console to assist administrators in debugging the cause of the failure.

Thus, the workflow for adding a node to the system is:

1. Plug the node in to the appropriate connections
2. Power on
3. Verify the node powers itself off within 5 minutes

B. *Removing a node*

Both Mountain and River hardware should automatically detect when a node has been physically removed. To gracefully remove a node, administrators should first verify it has been drained in the workload manager and has been powered off. Unlike previous Cray systems, removing a node will not affect the system's networks.

C. *Replacing a node*

Replacing a node is a combination of the "removing a node" and "adding a node" cases.

D. *Initial Discovery*

This workflow focuses on the first power-on of the compute nodes. All of the compute hardware will need to run discovery when first powered on. Thus, for a new system, the first step in booting the system is simply to power it on. As none of the hardware has yet been discovered, it cannot yet be controlled by Shasta software. Much of the hardware will automatically power on when power is supplied; the hardware which does not (primarily River hardware) may be manually powered on in any order. Powering on River nodes may be delayed or may be done in parallel with other installation tasks.

After powering on, hardware will automatically be discovered. Any node that fails discovery should be noted as needing investigation to determine the cause of failure, but will not block the system boot process.

E. *Expanding/Contracting a system*

Expanding or contracting a system is a rare but important scenario. It is not uncommon for a system to be permanently altered in ways not accounted for when the system was designed. For example, a system could be expanded (new racks of hardware added) or contracted (racks of hardware removed).

In any case, the core of the process is an interaction with IDEALS. IDEALS manages the hardware that is expected to be present in the system. It therefore must be made aware of the changes. Cray will provide multiple ways of interacting with IDEALS, including a user-friendly utility and access to the RESTful API.

To alter a system, all changes to the system – including additional, moved, or removed switches and additional, moved, or removed cables – must be recorded with IDEALS. To add nodes to a system, they should first be added to IDEALS. Then the nodes should be installed and powered on, as in the "adding a node" workflow.

To remove nodes from a system, start by using the removing a node workflow. The hardware may then be removed from the system and IDEALS updated.

VI. Comparison to XC

As Shasta is intended to replace Cray's XC systems, it is important to compare both systems. In the realm of discovery and hardware maintenance, the differences are quite significant – but for the better. Discovery and hardware maintenance should be significantly more simple in Shasta than they were in XC systems.

A. Initial Discovery

XC systems did not have a pre-generated map of the system. Instead, when first booted they would discover the hardware that was available. This was done with a tool called xtdiscover [6]. xtdiscover collected all system information in a single session, essentially regenerating the system database. Furthermore, collecting this information required all system components, even existing ones, to be successfully power cycled – a time consuming process that was often further lengthened if any blades required repeated attempts. It performed discovery in a "single shot" and required manual reconciliation if there were changes between runs. Further, each run required about two hours. While xtdiscover worked well for smaller systems where it is possible for all hardware to be present and functional at once, it is extremely problematic in Cray's larger systems.

Larger XC systems are often built up over a period of weeks as the hardware arrives at the final install location. Further, during this time, the customer wants to be using (or at least testing) the portions of the system that are available. Dedicating two or more hours to running xtdiscover significantly interferes with customer use and testing of the system.

In contrast, hardware discovery on Shasta is designed with ease-of-use as a priority. IDEALS holds a pre-generated mapping of the entire system. This means there is no single step where the entirety of the system must be present and hardware may be added to the system as it is available.

Further, REDS and MEDS (which perform the endpoint discovery to determine the hardware is added to the system) are ongoing processes. Both examine hardware as they receive indicators it is present and add it if it matches hardware listed in IDEALS. Neither needs manual triggering, so as soon as new hardware is in the system and powered on, REDS or MEDS will attempt to discover it. In the case of removal, both will use secondary information (e.g.: reading slot state in Mountain) to ensure they are not removing components due to mere malfunction. This ensures that in the case of hardware malfunction, components may be reintroduced to the system with a minimum of effort.

Finally, while REDS and MEDS operate with minimal intervention, the level of intervention required is configurable. They may be configured to require human intervention before making any changes, if the administrator desires.

B. Hardware Replacement

Hardware replacement in XC required rerunning a warm-swap command to locate new hardware once a failed blade was replaced. Because Shasta moves to the continuous discovery model, this manual effort is no longer required. Replacement hardware need merely be installed (and in the case of River hardware, powered on) in the system to be discovered and integrated. At that point, it will assume the identity of the hardware it was replacing, including the OS image.

VII. Conclusion

Cray's Shasta system should significantly reduce the effort required of systems administrators in performing hardware discovery and maintenance. Shasta prioritizes ease of use and independence of each component from the others. Further, it has, as much as possible, ensured hardware tasks (such as replacing a failed component) do not require software intervention on the system.

Hardware discovery and maintenance involves four independent services: IDEALS (which manages a "map" of the system as it was designed), HSM (which maintains information about the current and past state of the system), MEDS (which locates and configures Mountain hardware), and REDS (which locates and configures River hardware). Of these services, users should only have reason to interact with HSM and IDEALS.

This paper summarized the anticipated workflows to be used with this software, with a focus on how these workflows are simpler than in XC systems and significantly reduce the effort required of administrators.

References

[1] DMTF, "Redfish Scalable Platforms Management API Specification," 13 December 2018. [Online]. Available: https://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.6.1.pdf. [Accessed 15 March 2019].

[2] D. Harrington, R. Presuhn and B. Wijnen, "RFC 3411: An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," Internet Engineering Task Force, 2002.

[3] Intel Corporation, Hewlett-Packard Company, NEC Corporation, Dell Inc., "Intelligent Platform Management Interface Specification Second Generation," 2013.

[4] Y. Y. Goland, T. Cai, Y. Gu and S. Albright, "Simple Service Discovery Protocol/1.0 Operating without an Arbiter," Internet Engineering Task Force, 1999.

[5] Cray Inc., "XC™ Series System Administration Guide (CLE 7.0.UP00) S-2393," [Online]. Available: https://pubs.cray.com/content/S-2393/CLE%207.0.UP00/xctm-series-system-administration-guide/physical-id-for-cray-xc-series-systems. [Accessed 18 April 2019].

[6] Cray Inc., "XC™ Series Software Installation and Configuration Guide (CLE 6.0.UP04) S-2559 Rev B," [Online]. Available: https://pubs.cray.com/content/S-2559/CLE%206.0.UP04/xctm-series-software-installation-and-configuration-guide-cle-60up04-s-2559-rev-b/bootstrap-hardware-discovery. [Accessed 3 April 2019].