Cray Next Generation Software Integration Options

Larry Kaplan, Kitrick Sheets, Dean Roe, Bill Sparks, Mike Ringenburg, Jeff Schutkoske
Cray Inc.
Seattle, WA
{lkaplan,kbs,roe,bsparks,miker,jjs}@cray.com

***Abstract*—The ability to integrate third-party software into Cray's systems is a critical feature. This paper describes the types of third-party software integrations being investigated for Cray systems in the future. One of the key goals is to allow a plethora of third-party integration options that take advantage of all parts of the Cray technology stack. As a position paper, it aims to be directional and provides a guide through design and architecture decisions. It should not be considered a plan of record. It is intended to enumerate the possibilities but does not attempt to prioritize them or consider additional effort to enable them, if any.**

Keywords-system management, operating systems, programming environment

I. INTRODUCTION

Cray systems designs are moving to include far more flexibility than in the past, both in hardware and software. One of the main goals of such flexibility is to allow Cray systems to be easily integrated into a wider variety of customer data centers. Another is to allow Cray systems to be easily customized using site selected software components at all levels of the software stack. This document provides a description of the types and levels of software integration that are possible with this design. It is not intended to be a commitment to support all of these integrations or make definite statements about the plan of record, but rather it is an exploration of these possible integrations to facilitate their discussion and ultimately a selection and prioritization of the specific integration capabilities.

II. SYSTEM CONTEXT

Cray systems provide a great amount of flexibility in both hardware and software. The hardware infrastructure is designed to flexibly support a variety of compute, external network, and storage technologies. In addition, these hardware components are moving to leverage standard, open control interfaces to allow the system to be integrated into a variety of different data center ecosystems. In the same vein, Cray is providing management software that supports common APIs for managing the hardware infrastructure. These system management APIs are designed to support the integration of Cray platforms into existing data centers and support the wide range of diverse software ecosystems that are emerging in the high-performance computing space.

Cray software is evolving to be highly flexible. Previous Cray software stacks were very monolithic and tightly integrated. While the tight integration provides some benefits, it often made the integration of third-party software difficult if not impossible, depending on the type of software. The future software stack is being designed in a more modular fashion to allow a larger number of viable integration points, while still maintaining the benefits of tight integration. The goal of this design is that the deployed software stack is not dependent on the cabinet or node type, nor are the integration points which are the subject of this document.

Roughly speaking, Cray software can be divided into three categories: Component Management, System (or Infrastructure) Management, and Managed (user-facing) Ecosystems. These categories and where they predominately run are shown in Figure 1. Their purpose is described in more detail next.

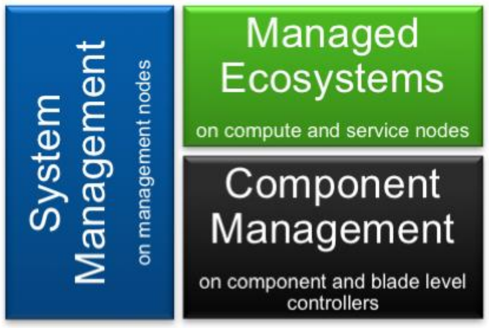

Figure 1: Software Categories

A. *Component Management*

Components here refer to individual hardware components as seen by a management system. This can include an entire node along with its associated controller, though usually there are sub-components within a node's domain that each have their own management capabilities. Some examples include the processor, the VRMs, and the network interface core/card (NIC). A significant part of the future strategy is to implement all component management interfaces using the Distributed Management Task Force (DMTF) Redfish® API. This is a developing industry standard.

B. *System Management*

This software is responsible for the configuration, operation, and monitoring of the entire system. It relies on the component level interfaces to perform its job. Much of the future system management software stack is derived from the Rhine System Management software that is part of

CLE6. However, many of the other components are new. Overall, System Management software makes use of RESTful interfaces to interact between its components.

System management performs tasks such as booting, dumping, logging, image management, configuration management, network configuration, etc. In addition, it is also responsible for managing anomalous behavior within the system and alerting responsible data center personnel of such events. In general, the role of the system management environment is to provide services and interfaces to support the efficient and effective use of the system infrastructure. Finally, it is a fundamental responsibility of the system management environment to provide secure authentication and authorization for access to management services as well as to ensure isolation of data transmissions between managed software stacks deployed on the system.

Cray's system management software is delivered as a set of layers, from basic hardware control interfaces, through basic infrastructure services, to full support for capability-class CLE software environment deployment and configuration. Each layer builds on the previous and exposes APIs for the control and maintenance of the hardware and software environment of the system.

Figure 2 shows some of the software components used for system management, arranged in the layers described above. The different Customer Platforms describe the different types of customers and how they might choose to integrate to (or "consume") the system. A Hardware Consumer is only interested in Cray providing the hardware itself along with the component level management (level 1). Such a customer would provide not only their own managed ecosystems, but also their own management system. A Dynamic Infrastructure Consumer is interested in using some of the lower level management services provided by Cray (level 2) but uses them to deploy their own managed ecosystem and platform support tools. A Production Class Consumer is looking for Cray to provide the production level system management and platform support. They are also more likely to use the Cray provided managed ecosystem (CLE below) but that is not required.

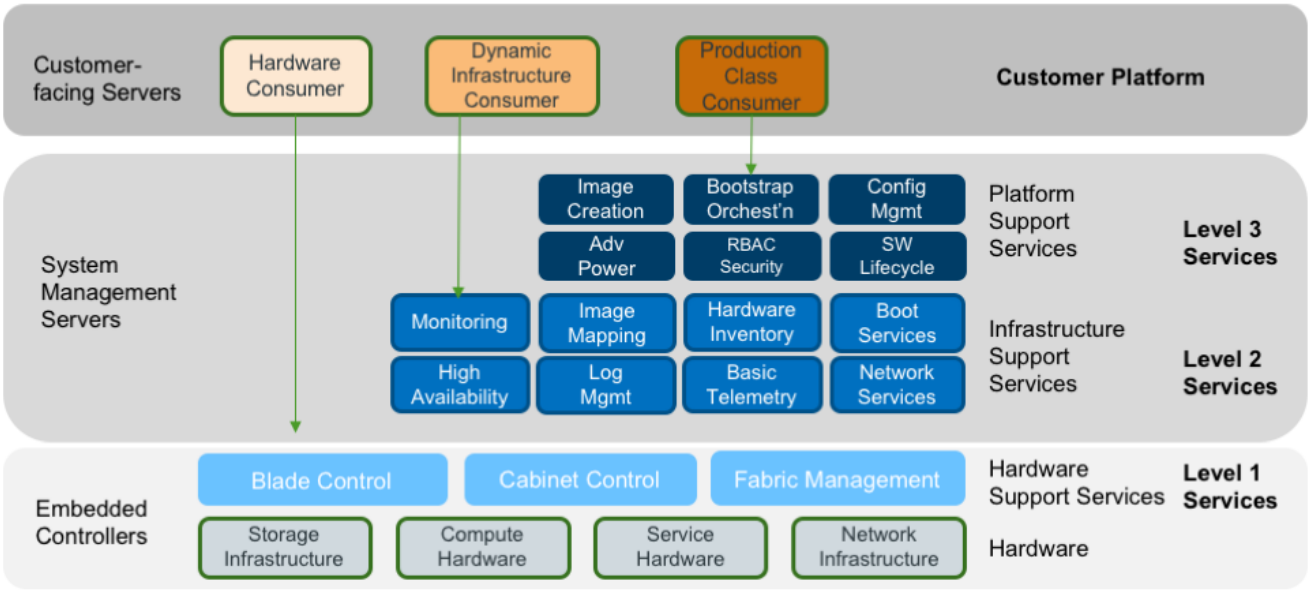

Figure 2: System Management Components

C. *Managed Software Ecosystems*

The traditional Cray example of this level of software is the Cray Linux Environment (CLE). CLE consists of the operating system and services used on compute and service nodes. CLE includes a rich set of supporting services, such as distributed network file systems, programming environment tools, and application launchers that are deployed primarily on service nodes. These services support the execution of user-originated application codes that are launched onto compute nodes within the infrastructure. CLE supports Cray's capability-class application execution framework for its traditional supercomputing customers.

While CLE is a critical managed framework for Cray systems, CLE is not the only possible managed ecosystem in the future. The software stack is designed to also support existing and emerging HPC software frameworks. This is made possible by better defining the interfaces between the managed ecosystem and the management system. By having well defined (and versioned) interfaces, other managed ecosystems can be deployed. Some examples include the Tri-Lab Operating System Stack (TOSS) and OpenHPC. Deployment of complete analytics ecosystems could also be done at this level (though these are currently layered on top of CLE using containers).

III. Major Integration Points

A major integration point is one in which Cray software components are fully (or mostly) replaced with third party components. Given the flexibility described above, this can be done in several ways, including replacing one or more layers of the Cray management stack, shown in Figure 2, or deploying a managed software stack other than CLE.

Note that full use of Cray proprietary hardware requires some Cray software components in the management and managed stacks. These components are not required in environments without such hardware.

A. System Management

A major integration at the management stack level represents the replacement of most or all of Cray's system management components with third-party management components. The future system management environment is designed to enable the integration of the system into a wide range of environments. For example, customers may wish to leverage pure Cray hardware and integrate into their existing datacenter. Other customers may want to leverage Cray's at-scale infrastructure management components but deploy HPC stacks of their choosing. Finally, some of Cray's existing capability-class customers will want to take advantage of the familiar Cray CLE software stack on the Cray platform. For this reason, the future system management environment is delivered as a set of layers with increasing capabilities. Customers can select the level of capabilities that best fits their use cases.

These levels include:

- **Level 1: Hardware as a Service (HaaS)**
  At this level, the Cray platform is delivered with open control interfaces for management of the hardware components including network infrastructure and fabric management. Redfish interfaces allow customers to leverage their existing data center management stacks through open and standard APIs.

- **Level 2: Infrastructure as a Service (IaaS)**
  At this level, the system management environment adds services and APIs to support the delivery of customer-supplied software content into the Cray infrastructure. Services supporting image management, scaled node bootstrap, and utility storage management support efficient and flexible use of the infrastructure to deliver the customer's chosen platform software stacks.

- **Level 3: Platform as a Service (PaaS)**
  This level delivers the system management infrastructure to support Cray's capability-class software stack, CLE. Included here are services to allow customers to build bootstrap and supporting images based on Cray-delivered software content, efficient, at-scale configuration of the CLE environment, advanced CLE reliability support interfaces, advanced power monitoring and management, advanced debugging and system support capabilities, etc. This level of system management support provides the features and functions that are familiar to existing Cray XC customers but also provides enhanced scalability, resiliency, and security.

As a major integration, customers may choose to replace most or all of the Cray provided management components at and above the chosen level of integration. Partial replacement is described later in section IV.A on minor integrations for system management.

To support these diverse system usage models, the future system management environment is designed to expose open and familiar APIs that can be consumed based on the administration model required by the customer. In the case of pure hardware integration, Redfish interfaces are exposed to enable the use of open and third-party management tools.

B. Managed Ecosystems

As noted above, the design and separation of the management software stack from the managed ecosystems allows one to more easily deploy managed software stacks other than the standard Cray-supplied CLE environment. Some customers may wish to leverage the Cray management stack's ability to deploy software at scale to stand up their existing managed software stacks. Note that it is also possible to add other co-resident managed ecosystems side-by-side with CLE. At the infrastructure level, APIs to support the management of bootable images, network services, and utility storage allocation are provided. Sites could plug their own managed content into this workflow and leverage these basic infrastructure management controls. It is envisioned that other ecosystems can be deployed on bare metal with sufficient integration support from Cray, likely in the form of a kernel module or two (for Cray proprietary hardware) and the use of well-defined APIs to communicate with the management system.

The future system management environment supports the ability to flexibly deploy concurrent software stacks into various portions of the system. In addition, on networks that implement sufficient support, the network management interfaces provide the ability to control various forms of isolation, from VLAN-style access isolation to traffic class-based performance isolation, allowing the software environments to get the most from the network.

IV. Minor Integration Points

A minor integration point is one in which Cray components are only augmented or partially replaced. The Cray provided software stack is still used for most of its normal duties. Minor integration can happen both in the System Management stack and in the Managed Ecosystems.

A. System Management

Minor integration of third party components in the management stack can include managing the entire system with only partial use of Cray-provided services. In this case, a customer with an existing management environment may want to leverage some parts of their existing approach to ease the integration of Cray systems into their existing environment. For example, they may wish to leverage Cray's capability for deployment of software at scale but wish to manage their own content. In this case, the customer can use their existing tooling for managing image content and environment configuration but leverage Cray's management APIs for the deployment of this content into the system. This would be similar to how a customer may leverage infrastructure services such as Amazon Web Services (AWS) or Microsoft Azure. Like these environments, Cray exposes REST APIs that provide access to these infrastructure management services. Some of these environments may also have existing workflows in place to coordinate the integration of systems into their datacenters. These workflows can also be leveraged to manage Cray systems using these APIs. Figure 2 shows some of the components that can be individually replaced in layers 2 and 3 for this type of integration. Additional system management components can also be added. While the CLE environment is designed to leverage level 3 management interfaces, these level 3 interfaces are supported and documented such that other software platforms could leverage these capabilities as well.

1) *Network Management*

For Cray proprietary networks, network management requires some components from Cray. These can potentially be integrated into third party management systems even when replacing Level 2 and above components.

For non-Cray networks, it is expected that vendor and community components are used both by Cray software and by potential customer integrations. For example, Intel Omni-path network management is provided by Intel.

2) *Storage Management*

Storage management represents a special case of system management software. It has some similarities to component management software though is usually managing a fairly large set of components as one subsystem. Storage devices also often come with their own management software.

There are two distinct use cases for the storage infrastructure in a Cray system. The first, referred to as utility storage, is controlled and provisioned by the system management infrastructure and provides supporting storage for both management and managed software components within the system. Examples of management use of utility storage would be space consumed by bootstrap images, configuration data, etc. Within a managed stack, utility storage may be used for general operating system use, such as temporary storage space and swap partitions for diskless nodes, or other stateful consumers, such as database applications used in support of the managed software stack. Utility storage can be delivered in various forms. From an attached storage array, to a scalable storage cluster such as Ceph, or even via an NFS or other network file system provided by the customer's data center. This last example represents the potential to integrate a customer's storage system into future Cray systems for use as utility storage. In each of these cases, it is the role of the system management environment to manage the consumption of this storage within the system environment.

The second major storage use case within the system environment is high-performance storage utilized directly by the managed software environments and the applications that run within them. The role of this high-performance storage is to support the extreme storage throughput and capacity requirements of capability-class codes running on Cray high-end systems. High-performance storage is typically delivered via a specialized appliance attached to the high-speed networking infrastructure of the system. These appliances form the base of the high-performance storage infrastructure. In addition, the system may also be configured with specialized capabilities for application-managed storage.

Unlike utility storage, primary storage appliances typically contain their own management environment. This management software supports basic facilities to configure and provision file systems for use by the managed software stack. It also provides capabilities that support the ongoing management of the hardware within these devices. These capabilities include various monitoring and alerting capabilities designed to allow timely response to component failures within the system. In addition, it is typical for these appliances to provide telemetry streams that support sophisticated performance monitoring and metering of the storage infrastructure. Cray plans to continue the work started in the Caribou project to provide clean integration of storage telemetry with the broader system management ecosystem. This provides the customer flexibility in choosing their primary storage solution with the confidence that its management and monitoring can be integrated into the overall Cray system management.

In addition, it is possible to customize Cray storage hardware with third-party file systems and access methodologies. This can be done by using user-mode containers (section V.B.1) or type-2 virtualization (section V.B.2) on the storage server nodes to provision the desired alternative software stack while continuing to use portions of the Cray stack. Bare-metal or type-1 virtualization could also be used but would likely constitute a major integration (as it would completely replace this part of the Cray stack).

B. *Managed Ecosystems*

Minor integration in the managed ecosystems can take many forms. There are many software pieces that make up a managed ecosystem, many of them with their own well-defined interfaces. In addition, the various container technologies (both user-mode and kernel/hypervisor) can be used to facilitate integration.

Figure 3 shows some of the primary components of CLE which is Cray's primary managed ecosystem. Each of the boxes represents software that is supplied as part of CLE but could be replaced with third party software providing similar functionality. Standard interfaces are used to integrate each of these components, though the exact interface varies depending on the component. Some additional details on the kinds of integrations possible at this level are provided below. Two of the components, user-mode containers and virtualization, provide functionality that can ease the burden of some types of integrations and are described in more detail in section V.B.

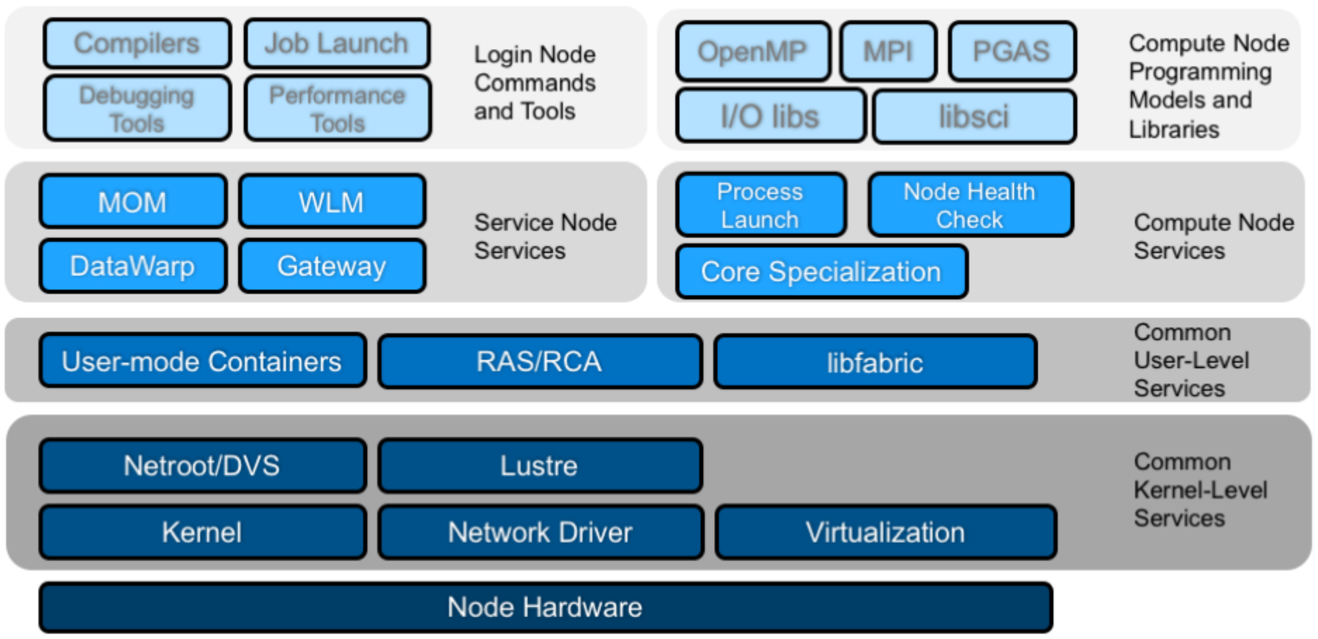

Figure 3: CLE Software Components (Managed Ecosystem)

C. *Other Managed Integrations*

As noted above, given the existence of well-defined interfaces, it is possible to use third-party software in place of Cray supplied software in many situations. One example is MPI. In the future, some of the expected networks use the industry defined libfabric interface. This allows other MPI implementations to run on these networks so long as those implementations use libfabric as their low-level network interface. The same is true for other network clients such as SHMEM. In addition, since Cray MPI is MPICH ABI compatible, applications can easily use a different compatible MPI without needing to modify their code.

Another integration example is the job launcher. While Cray primarily supports ALPS and Slurm for traditional HPC, and frameworks such as Mesos and Spark for analytics, customers may prefer using alternate software for application launch. Cray System Management enables customers to modify managed node image content to provide packages and configuration data tailored to their needs. While a site-specific job launcher may provide basic functionality, additional software features present on Cray systems (and available by default via ALPS and Slurm) could be utilized by extending the site job launcher to use APIs provided by Cray.

Customers can already choose from among several third-party workload managers (WLMs), such as SchedMD's SLURM, Altair's PBSPro, and Adaptive Computing's MOAB, and may also choose to provide their own. These are integrated using well-defined interfaces. The integration of both third-party WLMs and launchers is tied directly to the goals and options for unified orchestration discussed in section V.C.

Another integration example is Mercury POSIX, an I/O function shipping project. Cray systems utilize DVS for I/O forwarding capabilities, but a site preferring Mercury POSIX may include it in their node images in addition to DVS. While Mercury POSIX should integrate easily for standalone file system mount points, customers would be required to utilize Cray APIs if integration with services like DataWarp and the System Management image distribution is desired. In addition, integration with the Cray Dynamic RDMA Credentials (DRC) API would be required to enable shared network access. Note that DVS is a fundamental part of CLE and its network-based root filesystem service (netroot). As such, it cannot be replaced when running CLE, but that does not prevent other solutions such as Mercury from being used for user level I/O to the primary or other application level file systems.

The Cray Programming Environment (PE) provides compilers, libraries, and tools to support application development and improve application performance and programmability. Cray PE generally supports building libraries for three compilers per target architecture. For Intel processors, for example, Cray supports the Cray Compiling Environment (CCE), GCC, and Intel compilers. For ARM Cray PE supports CCE, GCC, and ARM Clang/Flang. In this context, support for third party compilers means building Cray developed libraries from the Message Passing Toolkit (MPT) and LibSci, some third-party libraries (HDF, NetCDF, and FFTW), and interoperability with the Cray performance tools (CrayPat and Apprentice2). While Cray PE provides a significant number of Cray and third-party packages, customers may wish to augment those capabilities. Additional third-party programming environment packages can be provisioned by extending node image content with Cray System Management tools as previously described. In addition, Cray provides APIs, many based on established standards, to enable access to both processor and network registers and performance counters, which allows for integration with additional performance analysis software.

Another significant integration example is that of open-source software frameworks for Analytics and AI/Deep Learning. Cray provides a set of commonly used frameworks (e.g., Spark, the PyData ecosystem, and TensorFlow) through Docker-based container images, which are imported into Shifter and/or Singularity. These components run using the user-mode container functionality provided by CLE (discussed below) but could also be plugged into alternate site-supplied container runtimes, provided they support importing standard image formats. Micro-services-based container runtimes also make it easy for sites to extend or replace portions of the Cray-supplied analytics and AI stack. In addition, Cray's preferred vehicle for providing differentiating capabilities in Analytics and AI is via software plugins. Many popular frameworks support this model, including Spark's pluggable shuffle manager and TensorFlow's user-defined operations. A software plugin delivery model decouples Cray's (or customer's) code from the core framework code, making it easier to plug optimizations into user-supplied versions of frameworks, or potentially into entirely new frameworks.

V. Execution Objects, Isolation, and Orchestration

The integration possibilities described above can be facilitated through the use of appropriate software technologies for the packaging, isolation, and orchestration of the various pieces of software. This section describes some of those technologies and their applicability.

A. Containers and Virtualization for Major Integrations and System Management

Virtualization can help ease the deployment of managed ecosystems and provide the ability to make the most of the computational and communication capacity of nodes in the system. There are two primary types of virtualization that are relevant here, with variants within each type.

The first is network virtualization or Ethernet-style VLANs, where network communication is isolated to only those endpoints (or nodes) that have been attached to a particular network segment (e.g., VLAN ID). This not only enables communication security between network segments, but it also enables the ability to manage traffic priorities and quality of service within the network. This provides the ability to give a third-party ecosystem the appearance of running on a "private" set of hardware resources, including the network, from the point of view of security and performance guarantees, even though the ecosystem is being deployed on a shared network resource. Depending on the network implementation, this feature could prevent network packets from ever reaching a NIC that is not configured as part of a particular VLAN, providing solid security even when running untrusted third-party managed ecosystems.

The second important type of virtualization is node (or server) virtualization, which allows the ability to stand up multiple virtual nodes, each running a distinct software OS and runtime environment, on a single physical node within the system. Server virtualization allows the complete isolation of runtime environments while allowing full use of a node's hardware resources. It differs from user mode containers in that alternate operating system kernels can be deployed on these virtual nodes with this technology. Additional use cases for node virtualization are described in section V.B.2).

Server and network virtualization are typically used in concert to connect services or other supporting runtime content with a broader software ecosystem. Server virtualization allows for use of shared node resources, while network virtualization provides secure separation of communication traffic. This combination of capabilities is common within existing cloud environments. Integration of these capabilities for future Cray systems supports a much more flexible usage model than provided previously.

1) Micro-services, Virtualization, and Containers for System Management

The system management services are delivered through a set of well-defined and documented APIs. These APIs define the supported interfaces for each service. Clearly defining the capabilities of individual services through well-defined interfaces provides the basis for building an extensible micro-service architecture. Building services with this approach provides a significant amount of flexibility in the support of a resilient, scalable, and extensible system management infrastructure.

With service interfaces cleanly separated, it is possible to package, deliver, and instantiate these services independently. This modular approach to service delivery has a significant impact on the ability to build a scalable and resilient system management infrastructure. Leverage of container technologies allows service updates to be applied efficiently with minimal disruption to the active management service infrastructure, thereby significantly increasing system availability.

Virtualization is another key technology that is leveraged by the future system management environment to increase the flexibility of the infrastructure. Virtualization allows management services to be deployed in environments that may be modest in size, where dedicated management server hardware is not warranted or in environments where a customer may want to leverage existing servers in their data center to perform certain management functions for their system.

B. Containers and Virtualization for Minor Integrations

Minor integrations use these technologies somewhat differently in that parts of the Cray stack are still present. These cases are now described in more detail.

1) User-mode Containers

Containers have become one of the most interesting and versatile alternatives to virtual machines for encapsulating applications. Containers enable the user to run an application by packaging not just the application, but also all of its dependencies and configurations. By encapsulating the application and its dependencies, containers can execute on a wide variety of software platforms. The Docker technology, now synonymous with containers, simplifies the deployment and management of containers on a variety of platforms and offers a user interface to facilitate DevOps in the enterprise environment.

What is needed by the HPC system managers is a secure way to allow containerization that would also allow access to the advanced networking and other hardware on their systems. To accommodate the HPC container use case, Cray systems deploy other execution engines better suited to scalable container execution. These scalable container execution engines integrate with the system orchestration and schedulers to provide the user with scale-out container and resource management that addresses the needs of the HPC and scientific user. Singularity[1] is an example of this.

Given a wide range of scientific communities, each with varying needs, Cray supports both types of user communities. It provides a standard environment that addresses the needs of the enterprise users and applications, but also provides support for the science community. With containers, even the most niche software could be installed, and further, the user is empowered to do this without administrative overhead.

2) *Node (Server) Virtualization*

If more than user-mode software is needed to support a particular integration, virtualization may be used. This technology allows kernels other than those supplied by Cray to be run on the nodes. There are two main types of virtualization: type-1 utilizes native bare-metal hypervisors (e.g. Xen) and type-2 utilizes hosted hypervisors (e.g. VirtualBox). While alternate OSs can also be run on bare metal, doing so with virtualization allows some portions of the Cray software stack to still be utilized and can potentially ease some of the integration burdens.

As an example, KVM is a popular virtualization product integrated into the Linux kernel. Turning on KVM does have an effect on the resulting kernel, and so has some implications, but it is a customer decision on whether to use virtualization enabled kernels or not.

Type-1 environments do not have access to most features provided by the CLE but can support alternate OS kernels that completely provide the customer's desired functionality. Type-2 environments can rely on CLE for the host functionality and therefore do have access to CLE features, but also allow the addition of other features including those supported by the guest kernel.

C. *Orchestration and Scheduling*

Orchestration provides central management of resources, scheduling, workflow management, and interactions of various processes running across heterogeneous systems. The goal of orchestration is not just to schedule and execute tasks, such as long running services or parallel jobs, it is also to provide mechanisms for monitoring and to offer management features to these tasks, such as elastic scaling, failover, and service control (stop, start, status).

[1] http://singularity.lbl.gov

One important driver of complexity in this area is the hardware and workload heterogeneity that is commonplace today. Cray systems are able to execute all types of workflows and workloads, from general science, HPC, analytics and long running services, and ad hoc applications. These workflows require information on resources which may be external to the framework in which the workloads are running. In support of this breadth of workloads, Cray system management must deploy different schedulers and resource management systems, and as such provide access to system information, enabling frameworks to allocate and monitor system resources.

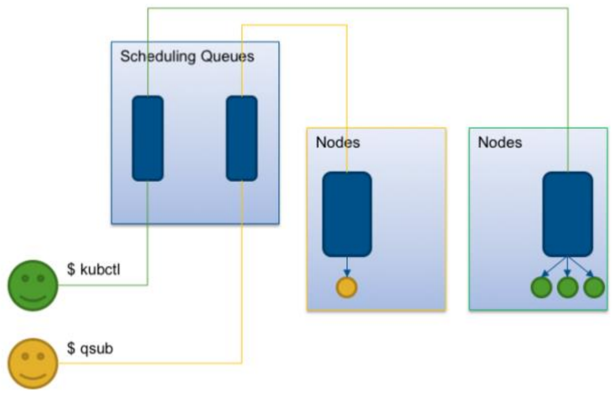

Figure 4: Separately Managed Frameworks

Considering the diversity of workloads, resource managers, and site preferences, Cray systems must accommodate a wide variety of vendor orchestration and scheduling solutions. Using vendor defined APIs and plugin infrastructures, Cray provides integration between different schedulers, providing a job execution framework which is able to provide HPC and alternate workload management. This can be achieved by supporting different deployment paradigms, from bootstrapping individual scheduler frameworks on dedicated resources (Figure 4), to supporting multiple co-existing scheduler frameworks across the entire system (Figure 5), where the system executes peer schedulers bound to a global concurrent state. Cray currently supports traditional workload managers such as SchedMD Slurm and Adaptive Computing Moab/Torque plus analytics frameworks such as Mesos and Yarn, though they currently operate independently. Future possibilities include Kubernetes and Grid Engine and integration and coordination of these schedulers and frameworks is also being investigated. Note that neither of these lists are exhaustive.

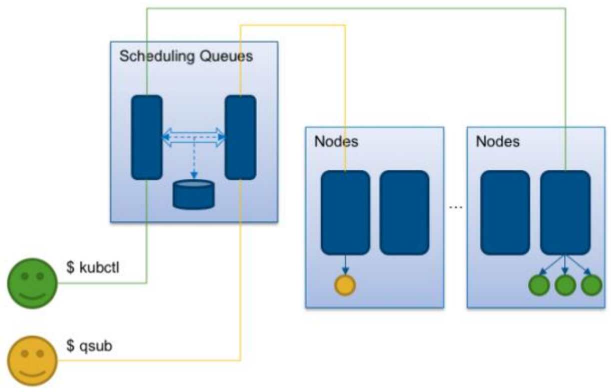

Figure 5: Coexisting Frameworks

By supporting multiple scheduler frameworks and workflows, from traditional HPC batch syntax, to Big Data job service specification, language and APIs become particularly important because data analysis applications are often written in rapid prototyping/scripting languages such as Python, Java, Scala, and R, which are not typical HPC programming languages. Most of big data schedulers can execute jobs in all of the major programming languages. However, with big data scheduler APIs, some languages are favored over others. Because many data analysis applications are often written with API calls to the scheduler itself (i.e. golang), language support can be very important to data analysis applications. Future Cray systems accommodate these different user models by supporting a variety of language interfaces and interpreters.

VI. Conclusion

This document has provided a description of the types and levels of software integration that are possible with the future software design. It provides a very high level of flexibility, both in supported hardware and software and in the ability to integrate third-party software into the system. Table 1 gives a summary of some of the integration possibilities and relevant APIs.

Table 1: Example Integrations and APIs

Producer	**Interface Description**	**Example APIs**	**Consumer**
PaaS, Node OS	Kernel Console	Syslog, REST	HaaS, IaaS, PaaS, and OS Debug
HaaS, IaaS	Status Events (HW/SW)	REST, Msg Bus	IaaS, PaaS, and related services
HaaS	HW Errors	REST, Redfish	HaaS, IaaS, and PaaS
HaaS	Power Mgmt Data	Redfish, PAPI	IaaS, PaaS, ProgEnv
HaaS	HW Inventory	REST	IaaS, PaaS
Fabric Mgmt	Network Topology	REST	IaaS, PaaS
ALPS	Interface Layer (BASIL)	Xml	WLM
PaaS	Image Mgmt and Repository Interface	REST	PaaS, Boot
PaaS	Image Configuration	REST	PaaS
PaaS	Boot Interface	PXE	IaaS, PaaS
IaaS, PaaS	DataWarp		PaaS, Services
IaaS	Network Access	libfabric	PaaS
Analytics Frameworks	Pluggable Enhancements	TensorFlow, Spark Shuffle	Applications
PaaS	Container Runtime	Docker, Shifter, Singularity	Analytics/Open Source/Customer Software