White Paper:

An Interoperable Server Mgmt Software Stack

5/19/05

Abstract: This paper proposes the construction of a stack of interoperable and highly functional software systems for the management of servers that plays a key role in the delivery of the computer industry's vision of Utility Computing.

Intended Audience: Information Technology Vendors

Intended Use: This paper is intended to coordinate/guide work on industry standards to ensure that multi-vendor products interoperate and to ensure that these products provide superior functionality when integrated into solutions.

Non-Goals: This paper does not address the broader question of a complete architecture for Utility/Grid Computing Services. This work is underway at the GGF and the EGA.

Authors/Return Comment: DMTF Utility Computing WG/RReich@VERITAS.com

Conclusion/Recommendations

The stack of server management products discussed in this paper exists in a limited capacity inside enterprise datacenters today. Through the ubiquitous deployment of interoperable and highly functional management interfaces between these management systems, the computer industry faces an opportunity to expand the market for these products while simultaneously delivering increased customer value. Coordinated development across the standards in this stack is key to delivering this opportunity and is an objective. In specific, work groups inside the DMTF who are building these standards will need to coordinate the development of object models (profiles), related opensource, specification content, interoperability testing/conformance programs, and early product development. The successful deployment of an interoperable and highly functional stack of server management software will also create an essential foundation for the construction of "higher level" management and automation systems that are ultimately required to fulfil the industries vision of utility computing. The Tuscany specification, under development in the Utility Computing Work Group, is intended to serve as a coordination/integration point for this vision.

Introduction

Successful industry wide deployment of the Utility or Grid computing customer vision is to a large part predicated on the deployment of more functional and interoperable management interfaces between the components/products of Utility Computing solutions. More specifically, customers require:

1. The ability to more broadly/flexibly share server resources across the breadth of applications in a datacenter thus, increasing asset utilization.
2. Substantially reduced administration/management costs associated with high availability application services delivery. Today enterprise datacenters employ expensive staffs who principle value is to manage multivendor configurations and ensure proper "up-time".

3. Multi-vendor interoperability between products (to avoid vendor "lock-in"). As a metaphor, a consumer in the process of purchasing electricity has no knowledge of the original generation source or transmission path for that electricity (they simply receive a highly available and cost effective resource).

Utility Computing Management

Figure 1 illustrates a broader architecture for the necessary foundations of Utility Computing and the relationship of a stack of Server Management Software within the context of this architecture. The purpose of this diagram is to illustrate the breadth of clients that exist in a datacenter for a highly functional server management software stack as well as to illustrate the progress made in an orthogonal market (i.e., storage) with highly functional and interoperable management standards.

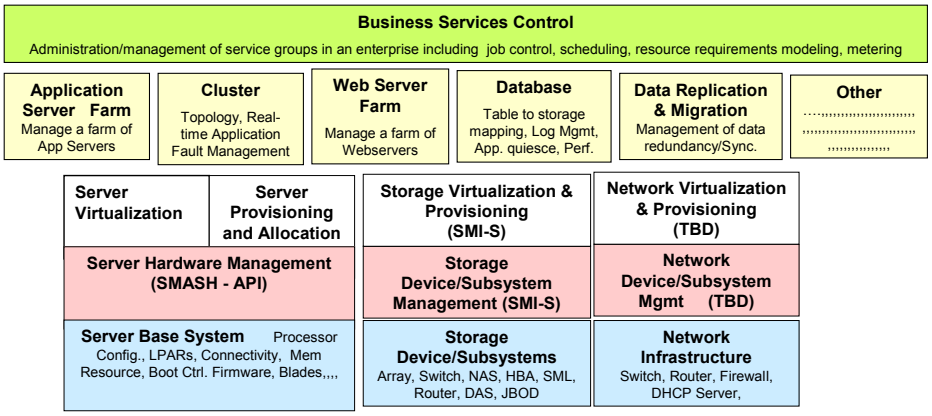


Figure 1, Utility Computing Architecture Foundation

Server Management Software Stack

Figure 2 illustrates CIM based server management interface standards efforts that have been launched at the DMTF as well as the potential dependencies between them in a highly functional and well integrated multi-vendor environment.

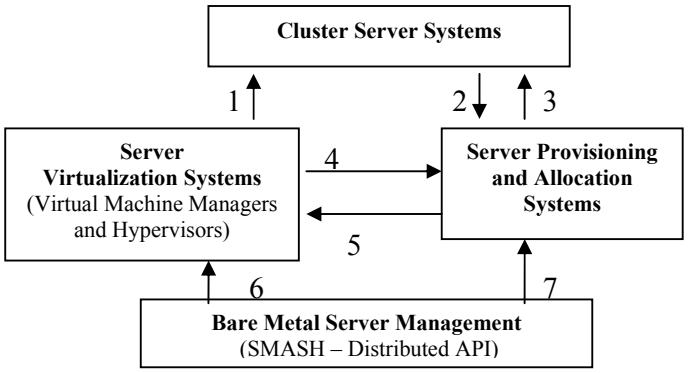


Figure 2: Server Management Software Systems

- Clusters Servers (improves the availability of applications in the presence of server or server resource failures). Examples: SunCluster, Parrallel SysPlex, VCS, HACMP, VMSClusters, TrueCluster
- Server Virtualization Systems (slices physical processors in virtual computers and enables the boot of multiple operating systems). Examples: Hypervisors like LPAR/DPAR managers, Prism as well as Virtual machine managers like Xen, VMware, IBM-VM. Today management models for Hypervisors and virtual machines are largely equivalent.
- Server Provisioning and Allocation Systems (automates the construction/configuration of bootable O.S.+Application images as well as permits server resources to be efficiently allocated across the breadth of consumers in an enterprise). Examples: N1 System Manager, N1 Service Provisioning Systems, OpForce, Tivoli TPM

- Bare Metal Server Management. Several popular interfaces exist in the market for the management of bare metal servers (IPMI, EMI, etc.). Using the SMASH profiles the market can receive the first distributed/secure/extensible/scaleable application programming interface (API) for server bare metal controllers.

Per Figure 1 (above) operational dependencies between these components of a management stack for servers are as follows. Note that number list below correspond to the numbers in Figure 1.

1. Temporarily freeze the operations of an O.S. Instance and relocate/restart that instance on another processor per cluster policies.
2. Request the state/names of nodes currently operating within a cluster.
3. Request that additional processors be provisioned with an O.S.+Apps and software and allocated for use in a cluster.
4. Request that additional processors be created with specific attributes and/or request the state of virtual processors with their mappings to physical processors.
5. Request information on the allocation of virtual processors within an enterprise.
6. Identify and manage physical processors.
7. Identify and manage physical processors.

Server Management and Related UC Standards

The Figure 3 illustrates the relationship of Utility/Grid computing standards under development in other standards organization working on UC/Grid computing and their relationship to Server management software stack proposed in this paper. This illustration is provided to stimulate cross standards organization coordination as well as to delineate the mountain of work ahead of the industry to deliver "open" utility/grid computing solutions to customers.

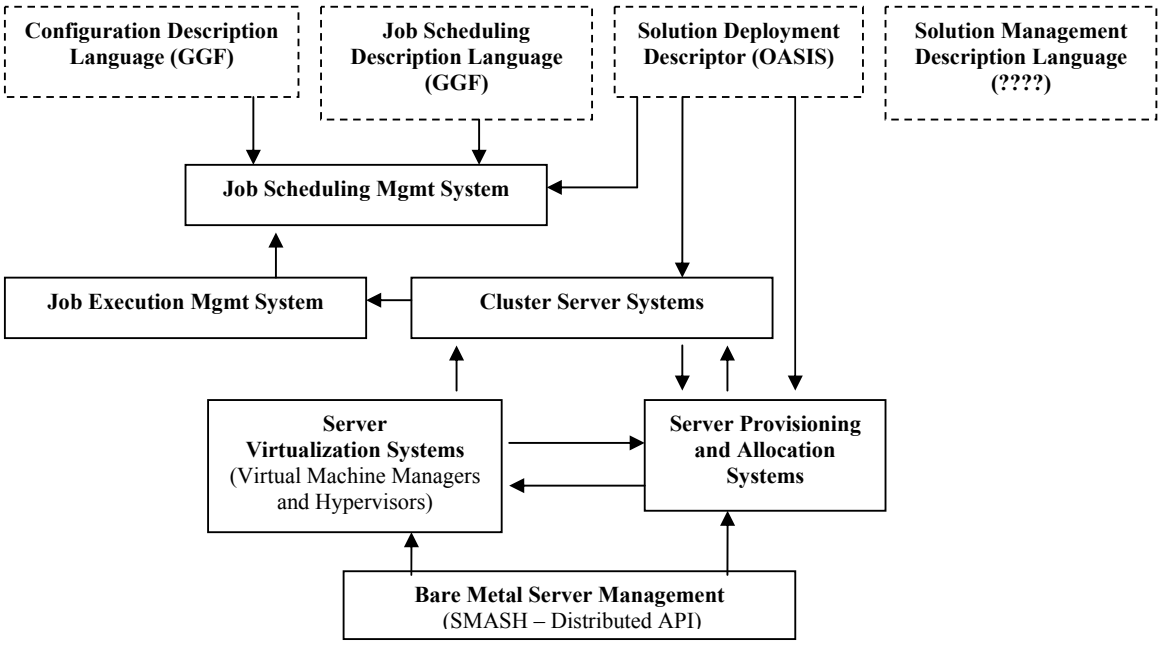


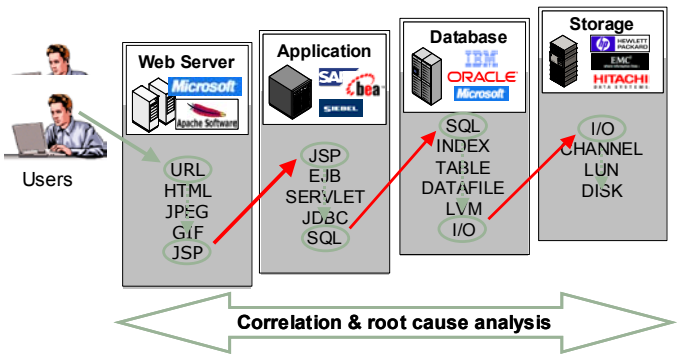
Figure 3: Server Mgmt and UC/Grid Standards

Management Interface Functional Requirements

The requirements documented below ensure that the integration of multi-vendor server management products produce more functional customer solutions (hence reducing administrative costs in the datacenter).

1. **Inherently Distributed:** Any management client can access any management server through a general-purpose network (preferably TCP/IP based protocol stack). Distributed access reduces the number of proprietary agents a management vendor must deploy across the systems/subsystems of a distributed enterprise. This reduces the customer's burden associated with qualifying, installing, and maintaining additional management product components. Examples exist today in SNMP and WBEM.
2. **Secure:** Authentication**:** Clients must successfully authenticate with a server before access. Similarly, servers must authenticate with clients before initiating communications. Privacy: Communications between the client and the server should be encrypted as provided by lower layers of the protocol stack where appropriate. Authorization: Sufficient semantics/content should exist to support role based authorization in the distributed system. That is, any given client may be restricted from performing some management actions on a given server based on the credentials of the client.
3. **Automated Discovery:** When a management server is configured on a network the server automatically identifies its presence to all management clients (i.e., clients are not required to manual configure connections to management servers). Additionally, when a client is first installed in a network, it may request that all servers identify themselves. A central directory agent may be required to improve the performance of discovery operations in a distributed system consisting of large numbers of clients/servers.
4. **Platform Independent:** A management client or server shall be capable of binding to the interface (its protocol) using any modern programming language (C, C++, Java, C#, VB) on any supporting operating system platform, and in doing so be able to utilize the data types and semantics inherent to the programming language. Platform independence is required for ubiquitous industry adoption.
5. **Interoperable:** For a given interface, a management client must be capable of seamlessly integrating (in the field) with management servers produced by multiple vendors. In support, a management server needs be able to certify conformance with the profiles/subprofiles and protocols that constitutes a management interface. The version numbers of all profiles, subprofiles, and protocols need to be available to the management client. Additionally, as components of the object model that define an interface are replaced with new technology, these legacy components need to be "deprecated" (i.e., preserved but discouraged for use in new implementation) in the specification such that legacy clients may continue to interoperate with new management servers.
6. **Scalable:** A management interface must include a specification for the minimum object model all clients may expect across multi-vendor implementations (a.k.a., a base **profile**). If the object model that expresses a standard management interface is resource intensive or includes functions not all vendors may choose to implement, that object model shall be capable of being decomposed into discrete components (a.k.a., ***subprofiles***) such that vendors may flexibly select (i.e., scale) the feature set expressed through an implementation of the interface. This flexibility permits computing resources (typically memory footprint) consumed by an implementation of the management interface to be scaled in support of low cost systems. Each subprofile in a hierarchy shall express a scalable and atomic unit of function. Subprofiles may be organized into a dependent hierarchy under a base profile.

7. **Automated Feature Discovery:** After a server is discovered in a network, a client shall be capable of programmatically discovering the ***subprofiles*** and ***packages*** unique to the implementation of each management server. This will enable a vendor to upgrade their management server (in the field) to include new subprofiles or packages without forcing the customer to upgrade their client management software. Version identification is key to discovery capability.
8. **Vendor Unique Extensibility:** Vendors need to be able to ship functionality in their products using the interface in advance of a standards body releasing a canonical sub-profile. Vendor extensions may take the form of a complete subprofile, additional object classes associated to the classes in an existing profile/subprofile, or additional properties/methods added to classes already existing in a profile/subprofile. Hence, vendors do not need to wait on the functional completeness of the new standard to release products and replace their legacy API's. To facilitate this function a methodology need be available that uniquely identifies each vendor's management server.
9. **Automated Discovery of Vendor Extensions:** A sufficiently rich set of intrinsic functions should exist in the protocol (like association traversal and enumeration) such that a suitably enabled client can dynamically recognize vendor unique functionality that has been implemented above the functionality defined in standard profiles and subprofiles. Hence, if a vendor adds a new performance counter (or an object), this information can be automatically displayed in a suitably designed management client without the release of a new version of software (analogous to the capabilities of an object browser). This capability to some degree decouples the release of management clients from management servers. To facilitate this capability, a standard set of datatypes needs be established across the protocol that constitutes the interface.
10. **Durable Reconfiguration:** Naming conventions need to exist for key resources like Logical Storage Volumes, processor platforms, and Tape Drives that allows these resources to be reconfigured in a datacenter without being misidentified after the reconfiguration.
11. **Correlatable Naming, multiple access points:** Multiple access paths to a given management server and its objects must result in each path seeing the same 'name' for a given instance of an object (yes, this is hard).
12. **Cross Server Correlatable Names:** If the management of a system/solution is comprised of multiple management servers, a management client shall be capable of reliably traversing object models across servers for the purpose of performing health and fault management root cause analysis (for example). In support of this, associators and addresses across the profiles/subprofile of different servers need be clearly documented and provide referential integrity.



13. **Out-of-Band Access:** Accurate monitoring and diagnosis of a distributed system requires that the transports used for management have the option to function independently of transports used to communicate data between elements of the distributed system.
14. **Mixed In-band and Out-of-band Access:** In some distributed systems, the transport used for management must (in some regions of the network) tunnel or coexist on the same transport used to communicate data in the distributed system.
15. **Indications/Events:** The interface drives vendors to uniformly implement indications support (sometimes called "events"). There need be two classes of events "lifecycle" and "alert". Lifecycle indications provide the ability for a given client to register for and receive specific

groups of events related to changes in the condition of the object model representing an interface (for example, the addition of a object, object instance, or property). Alert indications provide the ability for a given client to register for and receive specific groups of events that describe changes in state for a given object instance of an object. This will free client products from routinely polling the devices in a network to obtain updated status. This will improve the currency of information as well as reduce the performance impact of running management clients in a storage network. Event registration should be highly granular to reduce network traffic as well as the performance load on clients and servers.

16. **Policy:** The interface should provide a common and consistent language for communicating compound policy directives and related return status between clients and servers. This will enable a management client to dispatch a single compressive policy directive across multiple vendor products and ensure that the policy statement is carried out consistently.
17. **Internationalization:** The interface should provide a standard way for servers to return vendor unique error messages, log entries, and notifications in local languages. This will improve the customer experience in foreign countries.

About the DMTF and Disclosures