

vManage: Coordinated Management in Virtualized Systems

Sanjay Kumar, Vanish Talwar*, Partha Ranganathan*, Karsten Schwan

College of Computing, Georgia Institute of Technology Atlanta, GA 30332
{ksanjay, schwan}@cc.gatech.edu

*HP Labs, Palo Alto CA 94306
{vanish.talwar, partha.ranganathan}@hp.com

1. INTRODUCTION

With rising complexity and scale in today's enterprise data-center, managing these systems has become complex and costly. Additional issues are created by the adoption of virtualization in enterprise systems which enables new levels of flexibility like dynamic VM migration and other runtime mappings between virtual and physical resources. Key challenges for management solutions in these contexts include (1) the lack of coordination across different management domains, including across hardware and software, across different levels of abstraction, and across multiple hosts, (2) the virtual machines' (VMs) lack of privileges to access management hardware, (3) the conflicting management actions of VM-level management because of the autonomous nature of VMs, and (4) the lack of visibility into the service level agreements (SLAs) and performance requirements of the VMs' applications at the platform management layer leading to inefficient management actions.

Lifecycle management of VMs illustrates these problems. Lifecycle management refers to management of VMs throughout their lifecycle, from establishing policies for virtual machines, to request and approval, to provisioning, to runtime management and decommissioning. From the systems research perspective, provisioning and runtime management of VMs are important for the overall utility and efficiency of the data-center. VMs may have different requirements from the host machine, e.g., in terms of resources (e.g., CPU, memory, network bandwidth etc.), reliability, trust, etc., depending on their workload. Hosts on the other hand may have their own requirements, e.g., in terms of power and thermal budgets, and they may have certain values for available resources, reliability and trust attributes. In such a scenario, a problem is to determine which VMs should run on which hosts so as to satisfy both VMs' as well as hosts' requirements. Termed the VM placement problem, this problem also includes dynamic provisioning of VMs and hosts so that if any of the VMs or host requirements or attributes changes at runtime (e.g., a VM's cpu requirement or host reliability may change over time), it may be necessary to re-provision the VMs (e.g., by increasing the CPU reservations for the VM or by migrating the VM to a more reliable host) to ensure that the requirements of both VMs and hosts remain satisfied.

Power management provides a concrete example of dynamic nature of lifecycle management and the numerous platform- and application-level attributes contributing to effective management. Utilization based power management uses dynamic voltage and frequency scaling (DVFS) to manage CPU requirement of the VMs. For example, the Linux *ondemand* power governor increases the CPU frequency whenever the CPU utilization increases beyond a certain threshold (because of the increase in the load in VMs) and decreases the CPU frequency when the CPU utilization goes below

certain threshold. The increase in CPU frequency, however, leads to increased power consumption by the host which may cause it to violate its power budget. Reducing the power consumption (by reducing the CPU frequency) may cause the VMs' CPU requirements to be violated. Hence, a VM's CPU requirement and a host's power requirement may be in conflict with each other if the system is over-provisioned. In such cases, it may be necessary to migrate some of the VMs to less loaded hosts so as to satisfy VM's as well as host's requirements.

VM placement may also introduce stability issues. Since VM migration is a costly operation, any solution must implement methods to reduce the number of migrations over time. In other words, the decision to run (or migrate) a VM on a particular host should lead to a more stable system so that the newly migrated VM doesn't cause further violations on the new host and doesn't in turn cause more migrations. Finally, it may not be suitable to migrate VMs in the face of some transient SLA or power violations. Hence, deciding when and where to migrate are very important challenges in the lifecycle management of a virtualized data-center.

This paper presents a coordination architecture termed *vManage*, to simplify and improve the manageability of virtualized enterprises [1] and to provide dynamic provisioning and runtime management of VMs. The architecture consists of two building block abstractions: a communication abstraction, termed *M-channels*, and a coordination abstraction, termed *M-brokers*, privileged management VMs (MVM) and Cluster Leaders (CL). Compared to existing solutions, this architecture exploits coordination among VMs, MVM, host hardware, and CL to implement integrated management where both hosts' and VMs' requirements and multiple metrics (resources, reliability, trust etc.) can be managed simultaneously. We instantiate this architecture under Xen, and evaluate its benefits by implementing various power management policies (e.g., SLA-aware power management and power budgeting) for server class systems. The results show greater flexibility in implementing these policies while reducing the complexity of power management [1].

2. VMANAGE: SOFTWARE ARCHITECTURE

The vManage architecture divides a data-center into smaller management clusters, each with a Cluster Leader (CL) that is responsible for overall management of the cluster. The CLs coordinate among each other to implement data-center wide policies. Such a multi-level management hierarchy helps in the scalability of the vManage architecture for larger configurations. Figure 1 depicts the vManage architecture of one such cluster wherein each of the nodes is a virtualized system and one of the nodes' virtualized

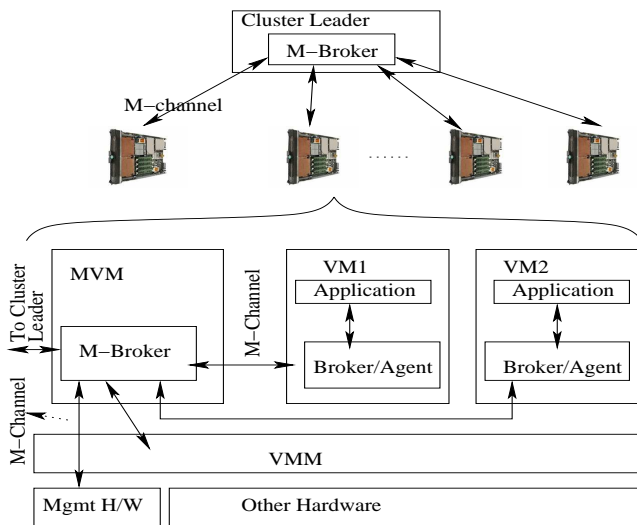


Figure 1: vManage Software Architecture

structure is shown in detail. Each host can have management controllers that perform hardware management. Each host has a special privileged Management VM (MVM) that can run host wide control and management software. The management and monitoring policies are run in containers termed M-brokers. M-brokers use M-channels to communicate with other management entities in the system. The next few sections describe each of these entities.

M-Channels: The M(angement)-channels are channels used for transferring commands and associated information items between MVM and other VMs, between MVM and management hardware, and between MVM and CL. M-channels provide standard APIs found in a typical middleware system like operations on channels, e.g., open, close, read, write etc. In addition, the M-channels also transparently handle the virtualization layer (apart from hiding hardware details) and VM migration (by supporting dynamic disconnection and reconnection between VMs and MVM). M-channels can be implemented using any underlying channel, e.g., shared memory communication for inter-VM communication and network sockets for communication over the network.

M-Brokers: M-brokers execute coordination policies by communicating with other M-brokers and monitoring agents using M-channels. The M-brokers in MVM gather monitoring data from hardware, VMM and VMs of the host, process the data and send this host information to the CL. For example, the M-broker can monitor Machine Check Events (MCEs) and IPMI sensor information from the host hardware and use some reliability models to calculate the current reliability of the host and update this information to the cluster leader.

Management VM: The MVM is a privileged VM which is the dedicated point to control all management tasks on a single host. MVM coordinates between VM's requirements and hosts requirements with the help of coordination architecture and inputs from VM's applications. Having a separate MVM to manage the host has several significant benefits compared to running the management inside the VMM. (1) Reduction in VMM's TCB: since most of the per-host management code (including various management models, policies and device drivers to access management hardware) run inside a separate VM as opposed to inside the VMM, it reduces the trusted computing base (TCB) of the VMM and in turn improves its security. (2) Virtual Appliances: Having a separate virtual machine for management simplifies its deployment and

upgrades in the form of a virtual machine appliance, since in this form, MVM can be independently deployed from the VMM, i.e., it can be dynamically added or removed from a system.

Cluster Leader: The Cluster Leader (CL) is responsible for the overall management of the whole cluster. It gather information about the hosts and VMs inside the cluster and implements the dynamic provisioning of the VMs and hosts. It runs an M-Broker that coordinates with other M-Brokers in MVMs over M-channels and decides on the initial placement of the VMs and dynamic VM migration according to the initial VM requirements and changes in the VMs and host attributes respectively.

2.1 Example: SLA-Aware Power Management

We have implemented novel SLA-aware power regulation with power budgeting using the vManage management architecture. As the host machines boot they contact the CL and inform it about their attributes (resources, reliability etc.). The CL then allocates a power budget for the hosts depending on the overall cluster budget. The MVM monitors the current power consumption of the host by querying the management hardware (e.g., in this case iLO processor in HP blades) and ensures that the power budget violations are handled automatically. It continuously monitors the current resource and power consumption of the host and updates the CL periodically. When a new VM needs to be deployed, the CL searches the list of hosts to find a host which has enough available resources to satisfy the VM's requirements. The MVM monitors the current SLA statistics of the VM applications using monitoring agents inside VMs. Whenever the SLA of VMs is violated, the notification is sent to the MVM, which increases the CPU frequency using DVFS. Whenever the power budget of the host is violated because of this increase in CPU frequency, the MVM reduces the CPU frequency to bring the power consumption within limits. The MVM also monitors the frequency of power budget violations and if the frequency crosses certain threshold, it requests the CL to migrate one or more VMs to bring down the load on the system. The CL finds a less heavily loaded host whose power consumption is well within budget limits and migrates VMs to bring down the power consumption on the loaded host.

3. CONCLUSIONS AND FUTURE WORK

This paper presents a management architecture termed *vManage*, for coordinated management in virtualized enterprises. The architecture provide generic interfaces for implementing diverse management policies, for dynamic VM placement and runtime management of the cluster. It provides integrated management of VMs and hosts, satisfying their requirements by using VM migration. We have shown the usefulness of this architecture by implementing it in Xen environment and using it to realize coordinated power management policies [1].

This remains an active work in progress, with ongoing work to implement additional management applications, such as reliability and trust, using this architecture. Further, we are working on an integrated management solution where VMs and host platforms can be managed by managing a variety of metrics, e.g., SLA, power, reliability, trust etc. in coordination with each other, compared to existing solutions where these metrics are managed in isolation from each other. The current MVM implementation is also being generalized to become a virtual appliance, suitable for dynamic deployment and upgrades in realistic enterprise systems.

4. REFERENCES

- [1] S. Kumar, V. Talwar, P. Ranganathan, R. Nathuji, and K. Schwan. M-Channels and M-Brokers: New Abstractions for Co-ordinated Management in Virtualized Systems. Technical Report GIT-CERCS-07-20, Dec. 2007.