Future Generation Computer Systems 26 (2010) 1271–1281

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Virtual Organization Clusters: Self-provisioned clouds on the grid

Michael A. Murphy *, Sebastien Goasguen **

School of Computing, Clemson University, 120 McAdams Hall, Clemson, SC 29634-0974, USA

ARTICLE INFO

Article history:
Received 14 May 2009
Received in revised form
19 December 2009
Accepted 25 February 2010
Available online 6 March 2010

Keywords:
Grid computing
Cloud computing
Self-provisioning
Autonomic resource management

ABSTRACT

Virtual Organization Clusters (VOCs) are a novel mechanism for overlaying dedicated private cluster systems on existing grid infrastructures. VOCs provide customized, homogeneous execution environments on a per-Virtual Organization basis, without the cost of physical cluster construction or the overhead of per-job containers. Administrative access and overlay network capabilities are granted to Virtual Organizations (VOs) that choose to implement VOC technology, while the system remains completely transparent to end users and non-participating VOs. Unlike existing systems that require explicit leases, VOCs are autonomically self-provisioned and self-managed according to configurable usage policies.

The work presented here contains two parts: a technology-agnostic formal model that describes the properties of VOCs and a prototype implementation of a physical cluster with hosted VOCs, based on the Kernel-based Virtual Machine (KVM) hypervisor. Test results demonstrate the feasibility of VOCs for use with high-throughput grid computing jobs. With the addition of a "watchdog" daemon for monitoring scheduler queues and adjusting VOC size, the results also demonstrate that cloud computing environments can be autonomically self-provisioned in response to changing workload conditions.

1. Introduction

Grid systems, such as the Open Science Grid (OSG) [1], enable different entities to share computational resources using a flexible and secure framework. These resources enable users to run computational jobs that exceed the capabilities of the systems available at any single location. To mitigate compatibility issues that result from resource heterogeneity, computational grids could be designed as Service Oriented Architectures, in which grid-enabled applications do not interact with low-level systems and resources directly. Instead, these applications communicate with abstract service libraries, which are built on top of standard network communications protocols such as TCP and IP. The services layer acts as a high-level operating system for the underlying physical resources, providing both abstraction and arbitration functionality. Since co-scheduling physical resources across sites is a complex task, such grid services are well-adapted for High Throughput Computing (HTC) applications, which tend to be compute-bound and do not require completion deadlines [2].

Virtualization of grid systems has been proposed as a mechanism for providing custom environments to users on grid systems that expose low-level computational resources as opposed to services [3], thereby enabling private clouds to be constructed using grid computing resources as a hosting utility [4]. Virtualized grid systems to date have taken the approach that new middleware should be developed for the leasing of physical resources on which to run virtual containers. The most widely published systems – Virtual Workspaces [5], In-VIGO [6], and Shirako [7] – all require the addition of system-specific middleware at both the execution and submission endpoints. In some cases, these systems require outright replacement of entire middleware stacks. With such requirements imposed on both the hosting site and the user, these lease-oriented systems are not transparent and cannot be easily deployed in a non-disruptive fashion. In contrast, a completely autonomic system would adapt to changing workloads and resource availability, without requiring manual intervention by either the user or system administrators [8].

The system presented here is a clustering overlay for individual grid resources, which permits Virtual Organizations of federated grid users to create custom computational clouds with private scheduling and resource control policies. This system is designed according to a specification known as the Virtual Organization Cluster Model, which stipulates the high-level properties and constraints of Virtual Organization Clusters. This model is technology-agnostic, permitting the use of different virtualization technologies, networking systems, and computational grids. A prototype cluster designed according to the model demonstrates that a viable implementation of the VOC specification is possible.

* Principal corresponding author.
** Corresponding author. Tel.: +1 864 656 2838.
E-mail addresses: mamurph@cs.clemson.edu (M.A. Murphy), sebgoa@clemson.edu (S. Goasguen).

0167-739X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2010.02.011

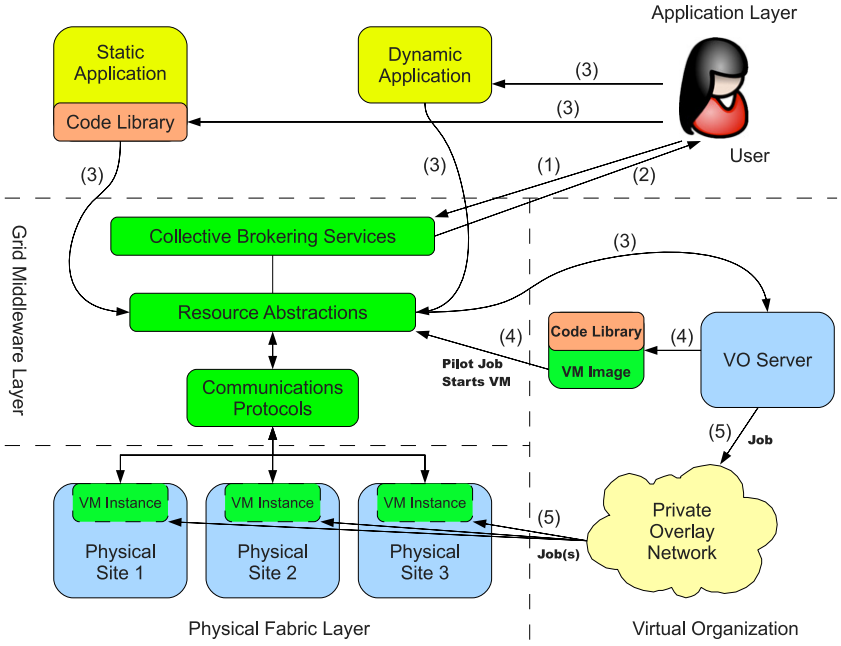

Fig. 1. A use case for Virtual Organization Clusters, in which pilot jobs are used to reserve physical resources to host virtual machines. User jobs are then privately scheduled to run in the virtual machines, using an overlay network to create a virtual cluster. It should be noted that the user submits her job(s) to a resource abstraction of the VO Server, which autonomically handles the creation and destruction of virtual environments. The user is unaware of the details of this environment management.

The motivation for the Virtual Organization Cluster Model and Virtual Organization Clusters (VOCs) built according to the model is to create a virtual cluster environment that is homogeneous across sites, autonomically self-provisioned, transparent to end users, implementable in a phased and non-disruptive manner, optionally customizable by Virtual Organizations, and designed according to a specification that permits formal analysis. Transparency is achieved by designing the system so that no submission endpoint middleware is needed, and VOC technologies can be added to execution endpoints with minimal disruption of existing middleware. Moreover, grid sites that choose to provide VOCs may provide them transparently, so that neither the user nor the VO is aware that execution is actually occurring on a VM. Conversely, a site and a VO may choose to permit the VO to supply the VOC image (Fig. 1), thereby allowing the VO to have optional administrative access for software stack and policy customization.

The remainder of this paper is organized as follows. Related work is presented in Section 2, after which the high-level Virtual Organization Cluster Model is described in Section 3. Section 4 describes a prototype implementation of a Virtual Organization Cluster and associated physical testbed, with test results following in Section 5. Finally, Section 6 presents the conclusions and describes future work.

2. Related work

Virtualization at the Operating System (OS) level, originally developed for IBM mainframe systems in the late 1960s, permits an operating system installed directly on the computational metal, known as the "host" system, to run a second "guest" operating system or "appliance" inside a virtual container. Virtualization systems allow architecture-compatible software systems to be decoupled from the underlying physical hardware implementation, thereby allowing computation to be location independent [9]. Virtualization of grid systems, first proposed in [3], offers substantial benefits to grid users and system administrators. Users of virtualized systems can be granted administrative access rights to their virtual machines, thereby allowing end-user customization of the software environment to support current and legacy applications. Since the hardware administrators retain control of the Virtual Machine Monitors (VMMs) or hypervisors, coarse-grained resource controls can be implemented on a per-VM basis, allowing hardware resources to be shared among different VMs [3]. Higher-level system components may also be virtualized; examples include dynamic service overlays [10], cloud storage frameworks [11], and virtual networking systems such as ViNe [12], VNET [13], VDE [14], and IPOP [15]. Virtualization at the application layer has been realized in systems such as In-VIGO [6].

Globus Virtual Workspaces, implemented as part of the Globus Nimbus toolkit, provide a lease-oriented mechanism for sharing resources on grid systems with fine-grained control over resource allocation. A Virtual Workspace is allocated from a description of the hardware and software requirements of an application, allowing the workspace to be instantiated and deployed on a per-application basis. Following the construction of the environment, the Workspace must be explicitly deployed on a host site, after which it can be directly accessed to run computational jobs [5]. By utilizing a leasing model, Virtual Workspaces can provide high-level, fine-grained resource management to deliver specific Quality of Service guarantees to different applications using a combination of pre-arranged and best-effort allocations [16]. By leasing multiple resources simultaneously, Virtual Workspaces may be aggregated into clusters [17].

Selection and final customization of VM appliances to match the requirements of individual jobs are accomplished via "contextualization," a process by which job requirements are used to make minor configuration changes, such as network and DNS settings, to an existing appliance [18,19]. Contextualization is done once per VM invocation, just prior to boot time, and involves injecting configuration data into the VM image prior to initialization of the instance. A centralized "context broker" provides an XML description of the configuration parameters to be set on a per-appliance basis [20]. This contextualization process may occur during workspace scheduling [21].

For performance reasons, workspace jobs should have a sufficient run length so as to make the overhead of leasing and starting a workspace relatively small. Given the non-trivial data