Noname manuscript No.
(will be inserted by the editor)

Autonomic Clouds on the Grid

Michael A. Murphy · Linton Abraham · Michael Fenn · Sebastien Goasguen

the date of receipt and acceptance should be inserted later

Abstract Computational clouds constructed on top of existing grid infrastructure have the capability to provide different entities with customized execution environments and private scheduling overlays. By designing these clouds to be autonomically self-provisioned and adaptable to changing user demands, user-transparent resource flexibility can be achieved without substantially affecting average job sojourn time. In addition, the overlay environment and physical grid sites represent disjoint administrative and policy domains, permitting cloud systems to be deployed non-disruptively on an existing production grid. Private overlay clouds administered by, and dedicated to the exclusive use of, individual Virtual Organizations are termed Virtual Organization Clusters.

A prototype autonomic cloud adaptation mechanism for Virtual Organization Clusters demonstrates the feasibility of overlay scheduling in dynamically changing environments. Commodity grid resources are autonomically leased in response to changing private scheduler loads, resulting in the creation of virtual private compute nodes. These nodes join a decentralized private overlay network system called IPOP (IP Over P2P), enabling the scheduling and execution of end user jobs in the private environment. Negligible overhead results from the addition of the overlay, although the use of virtualization technologies at the compute nodes adds modest service time overhead (under 10%) to computationally-bound grid jobs. By leasing additional grid resources, a substantial decrease (over 90%) in average job queuing time occurs, offsetting the service time overhead.

1 Introduction

Cloud computing provides a means by which organizations may have dedicated customized computational environments without the associated hardware and infrastruc-

M. Murphy, L. Abraham, M. Fenn, S. Goasguen
School of Computing
120 McAdams Hall
Clemson University
Clemson, SC 29634-0974 USA
E-mail: mamurph@cs.clemson.edu

ture costs. Virtual Organization Clusters (VOCs), designed around the Virtual Organization Cluster Model (VOC Model) [1], provide a mechanism for the autonomic creation of cloud systems for Virtual Organizations [2], utilizing commodity grid computing resources as underlying physical hosts for virtual systems. VOCs provision themselves and adapt to changing resource demands within the cloud by automatically leasing physical systems to run VO-customized virtual machines [3]. These dynamic environments are ideally suited to computationally-bound jobs, such as bag-of-tasks applications [4], since the communications overheads introduced by current virtualization technologies can be significant [5,6].

A key benefit of VOCs is their inherent transparency to both end users and non-participating entities. VOCs are explicitly designed to use existing grid middleware systems at the user-facing job submission endpoint, thereby eliminating the need for users to install additional software to utilize the cloud systems. Users simply submit computational jobs to the cloud systems exactly as they would submit the same jobs to existing physical sites. Furthermore, VOCs remain completely transparent to grid entities that choose not to utilize VOC technology, permitting clouds to be deployed in a manner that is non-disruptive to the existing grid infrastructure. With the addition of private overlay networking technologies, VOs that choose to participate in VOC technology may control access to private resources, perform their own scheduling and resource allocation functions, and prioritize user jobs according to site-independent policies. As illustrated in figure 1, physical resources are obtained for hosting VOC nodes by means of pilot jobs, which are autonomically managed by the VOC middleware.

One direct result of leasing physical resources simply as platforms for hosting virtual systems is that the conceptual distinction between different grid sites becomes less sharp when a grid system is primarily comprised of machines with the same instruction set architecture. With the addition of an overlay networking system to join widely distributed virtual machines into a single virtual cluster, the collections of hardware resources across different grid sites can be conceptually treated as a single resource collection, similar to a large Condor [7] pool. By leasing these resources autonomically and instantiating overlay environments, VOCs transparently adapt commodity grid resources to the needs of user communities, instead of forcing users to adapt applications to the capabilities of available resources. Moreover, since all physical grid resources effectively become equal under the virtualization assumptions, the need to queue jobs to wait for compatible execution environments to become available is reduced. Even with virtualization overheads increasing the service (execution) time of jobs run within virtual machines, the decrease in waiting time can be significant enough to minimize impacts to the average job sojourn time (time from submission to completion, or the sum of the waiting and service times) across the grid. To achieve these results, the autonomic system must adapt simultaneously to changing workloads and changing resource availability [8] through the use of a "watchdog" daemon, which dynamically adjusts the size of each VOC by starting and stopping virtual machines.

The purpose of this paper is to extend the existing Virtual Organization Cluster dynamic provisioning system [3] to enable private overlay scheduling within a virtual cluster, where the virtual machines comprising the virtual cluster are themselves scheduled on leased physical resources. Through the addition of the IPOP overlay network [9], VOCs can span different administrative domains, allowing physical resources to be leased from different providers and joined into a single cluster. As demonstrated through simulation results, grid-wide deployment of VOC technology should not nega-

tively affect aggregate performance for computationally-bound grid jobs, even though the addition of virtualization overhead increases execution time.

The remainder of this paper is organized as follows: related work is discussed in section 2. Background material is presented in section 3, including a brief description of the VOC Model in section 3.1, the IPOP overlay network in section 3.2, and policy options for autonomic adaptation of VOCs in section 3.3. Section 4 describes the results of a prototype implementation test using a single grid site. For scalability evaluation, grid-wide simulation results based on actual job traces from the Enabling Grids for E-sciencE (EGEE) production system are presented in section 5. Finally, conclusions follow in section 6.

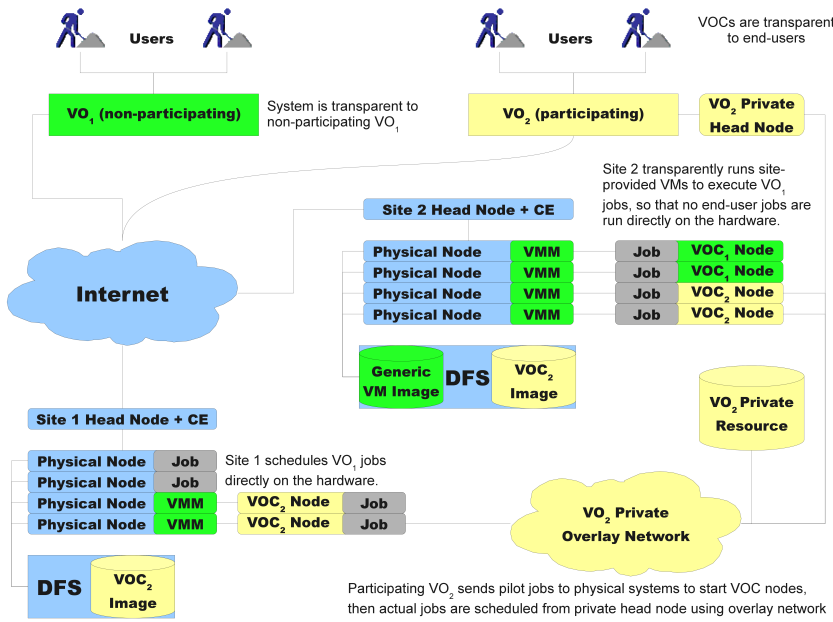

Fig. 1 Virtual Organization Clusters provide a mechanism that enables participating Virtual Organizations to develop private cloud environments, while simultaneously remaining transparent to end users and non-participating entities.

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 physical computer, known as the "host" system, to run an additional "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. [10] Virtualization of grid systems, first proposed in [11], 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. [11] Higher-level system components may also be virtualized; examples include virtual networking systems such as ViNe [12], VNET [13], VDE [14], and IPOP[9]. Virtualization at the application layer has been realized in systems such as In-VIGO [15].

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. [16] 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 [17]. By leasing multiple resources simultaneously, Virtual Workspaces may be aggregated into clusters [18].

Another lease-oriented mechanism for grid system virtualization is Shirako [19], which allows resource providers and consumers to negotiate resource allocations through automated brokers. Back-end cluster provisioning is provided by Cluster-On-Demand (COD) [20], an automated system that performs rapid reinstallation of physical machines or Xen virtual machines [21]. An application of Shirako, called the Grid Resource Oversight Coordinator (GROC), permits physical resources to be leased by individual Virtual Organizations for the purpose of deploying completely virtualized grids, which are accessed by end users through existing Globus middleware. Each physical grid site hosts a self-contained virtual cluster belonging to the same VO, without spanning of individual clusters across grid sites. [22]

Autonomic virtualization overlay systems have been devised for the purpose of extending local clusters to utilize idle resources available on other clusters within the same local area. VioCluster is based on the concept of dividing each cluster into a physical and a virtual domain, where the virtual domain may be borrowed and administered by another group within the same entity. Virtual domains are transparently and autonomically resized by means of a broker application, which trades machine allocations between groups by following explicitly configured policies. This brokering process is transparent to end-users, permitting the virtualized cluster to be used as if it were a regular physical cluster. [23] By utilizing the Violin overlay network, virtual domains on several different physical clusters may be combined into a single execution environment with a private network space [24].

Dynamic Virtual Clustering allows clusters of virtual machines to be instantiated on a per-job basis for the purpose of providing temporary, uniform execution environments across clusters co-located on a single research campus. These clusters comprise a Campus Area Grid (CAG), which is defined as "a group of clusters in a small geographic area ... connected by a private, high-speed network" [25]. Latency and bandwidth properties of the private network are considered to be favorable, thereby allowing a

combination of spanned clusters to function as a single high-performance cluster for job execution. However, the software configurations of the different component clusters may differ, as the component clusters may belong to different entities with different management. DVC permits Xen virtual machines with homogeneous software to be run on federated clusters on the same CAG whenever the target cluster is not in use by its owner, thereby allowing research groups to increase the sizes of their clusters temporarily. [25]

Virtual Organization Clusters [1] differ from explicit leasing systems such as Globus Nimbus and Shirako, in that virtual clusters are leased autonomically through the use of pilot jobs, which provide for dynamic provisioning in response to increasing workloads for the associated VO [3]. VOCs remain transparent to end users and to non-participating entities, while enabling participating VOs to use overlay networks, such as IPOP [9], to create virtual environments that span multiple physical domains. Unlike Campus Area Grids and VioCluster environments, however, these physical domains are grid sites connected via low-bandwidth, high-latency networks. These unfavorable connections, coupled with overheads introduced by the addition of virtualization systems, make VOCs better suited to high-throughput, compute-bound applications than to high-performance applications with latency-sensitive communications requirements [6].

3 Background

Extending the prototype implementation of Virtual Organization Clusters to enable virtual clusters to span multiple physical domains requires the addition of an overlay network to join virtual compute nodes to a VOC head node, which is intended to be a persistent system that is publicly addressable. Since VOCs need to scale to a large number of nodes over a wide area, while simultaneously changing size on a frequent basis, the IP Over P2P (IPOP) overlay networking system [9] has been chosen for use in the prototype VOC described in section 4. Autonomic scheduling of the VOC is performed by a watchdog daemon running on the dedicated VOC head node and utilizing an algorithm shaped by an autonomic adaptation policy. These components are detailed in sections 3.1, 3.2, and 3.3.

3.1 Virtual Organization Clusters

Virtual Organization Clusters (VOCs), described more thoroughly in [1], enable the creation of virtual cluster environments that are compatible across sites, 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. Since VOCs are constructed from virtual machines (VMs), and VM instances can be spawned from a single image, VOC environments are nominally homogeneous (and therefore software compatible) across grid sites. If each grid site utilizes a central distributed filesystem store such as PVFS [26], VOC nodes can be booted directly from the shared filesystem, without staging the multi-gigabyte image files to each physical compute node. Once operational, VOCs remain completely transparent to the end user, since the virtual environments are autonomically managed without explicit resource reservation requests. VOCs also remain transparent to entities that

choose not to deploy them, allowing VOC implementations to be added to existing production grids without disrupting the operational infrastructure. Different technologies can be utilized to implement VOCs, since a VOC is simply an implementation of a system that conforms to the specifications presented in the VOC Model.

A key specification of the VOC Model is the explicit separation of administrative domains. Each physical site on the grid is a unique Physical Administrative Domain (PAD), which is managed by local administrators. Components of each PAD include the physical computing resources, networking interconnections, and all associated infrastructure, including power distribution and cooling. Each hosted VOC is a separate Virtual Administrative Domain (VAD) that is managed by the owning VO or an agent thereof. This explicit administrative access permits each VAD to have a customized software environment. Moreover, the separation of administrative domains implies a separation of policy authority. Local site owners and VOC owners may implement their own resource allocation, scheduling, and management policies. These policy decisions are all independent: no coordination is required between the VOC administrators and the site administrators. VOCs are explicitly a "best-effort" system and will execute on any physical site willing to provide VM hosting services.

As illustrated in figure 1, jobs are submitted directly to a VOC through a dedicated, grid-facing head node. This cluster head node contains the necessary grid interconnection software and appears as a compute element on the grid. As the size of the job queue on the private head node increases, pilot jobs are submitted to grid sites to obtain physical resources. In turn, these pilot jobs start virtual machines, which are dynamically configured, or "contextualized" [27], at boot time to start the overlay network and join the scheduler pool on the private head node. User jobs are then executed on the virtual cluster as provided by the scheduling policies implemented in the VOC.

3.2 Overlay Networking System

In order to permit Virtual Organization Clusters to span multiple grid sites to utilize additional resources, it is necessary to provide a mechanism by which VOC nodes on different physical sites can communicate directly with each other and with the private head node. Since physical sites may be connected to the Internet via an edge router with Network Address Translation (NAT) to a local subnetwork containing the compute nodes, an overlay network capable of traversing NAT boundaries is required. One such overlay system is Internet Protocol Over Peer-to-peer (IPOP). IPOP is built upon the Brunet Peer-to-Peer (P2P) library, which uses Distributed Hash Tables for structured P2P routing and scalable object storage [28]. Unlike systems designed around virtual switches and other virtual network hardware, IPOP utilizes publicly addressable "bootstrap nodes" that act as Session Traversal Utilities for NAT (STUN) servers and provide initial routing for new nodes that join the overlay. VOC nodes join the IPOP pool by connecting to these bootstrap systems, using peer-to-peer technology both for communications and to distribute system state across the entire network. This system is scalable to large networks of thousands of nodes distributed across tens or hundreds of domains, at a cost of reduced network throughput and increased latency. [9] However, since virtualization overheads constrain VOCs to compute-bound processes with low sensitivity to network latency and minimal bandwidth requirements [6], network performance is not a major design concern.

Since IPOP provides an isolated private network to the VOC, both job scheduling and administration tasks can be performed over this network. The Condor scheduler [7] can be configured to service a pool comprised of IPOP-connected nodes, enabling private job scheduling within the virtual cluster. In addition, the VOC head node can send control instructions through the overlay network to the compute nodes, including instructions to terminate idle compute nodes when the size of the VOC is to be reduced. Expansion of the VOC is accomplished by submitting pilot jobs to physical grid sites, which start virtual machines when executed. These virtual machines are configured to contact the IPOP bootstrap infrastructure upon startup, thereby joining the private overlay network. Decisions to shrink or expand a VOC can be made within the Virtual Administrative Domain of the VOC itself, using policies configured by the Virtual Organization. Unlike prior related work [29], it is not necessary to trust the virtual systems to terminate themselves correctly, since the physical systems retain the ability to terminate the pilot jobs used to start the virtual machines. When the pilot jobs are terminated by the local site, the virtual machine instances are immediately killed.

3.3 Autonomic Adaptation

Virtual Organization Clusters are designed to expand to utilize available grid resources whenever the number of user jobs in the VOC scheduler queue exceeds the size of the VOC itself. Conversely, as the virtual cluster becomes under-utilized due to jobs completing, the virtual machines are terminated so that the computational resources may be returned to resource pool for others to use. The policy that is used to effect expansion and shrinkage is ultimately linked to the private scheduling queue and thus falls within the Virtual Administrative Domain of the VOC. Since the principle of administrative domain separation results in a complete separation of policy between the VAD and the underlying Physical Administrative Domains of the host sites, policy coordination might not exist between the host sites and the VOC. In such a case, the VOC relies on host site scheduling to obtain resources, and host sites may enforce any desired policy against the Virtual Organization to limit resource usage or share physical systems among different VOs. Since VOC nodes are started via regular pilot jobs, the host systems need only provide a virtualization system and a means for starting and stopping VMs. Provided the pilot jobs are authorized to run on all grid sites, and assuming that the grid consists largely of machines with a compatible instruction set architecture, the disparate resources on different grid sites may be conceptually viewed as a single pool of collected resources onto which the virtual machines are scheduled. Once several machines have been started for a single Virtual Organization, the overlay network binds them into a private Virtual Organization Cluster.

A significant challenge in managing virtual clusters arises from the need to determine when the cluster size should be changed in response to changing workloads and changing resource availability [8]. As the total workload in terms of the number of jobs waiting to utilize a VOC increases, it is generally desirable to increase the size of the VOC by adding additional virtual machines leased from additional physical resources. However, the VOC must also be careful not to over-subscribe the physical resources, in order to prevent queuing of the pilot jobs and an attendant decrease in aggregate performance. The initial “watchdog” daemon implemented to perform this management [3] utilizes a naive greedy algorithm that responds quickly to workload changes but assumes that the total workload will not result in over-subscription of the

physical resources. Once a grid system approaches capacity, however, it is necessary for the watchdog to adjust both to workload demands and resource availability, in order to optimize resource allocation. Performing this optimization in an environment without policy collaboration between the virtual and physical administrative domains is the subject of ongoing research. For the purpose of testing of overlay scheduling behavior, a simplifying assumption is made that there is sufficient excess physical capacity in the grid to permit scheduling based solely upon workload demands.

4 Prototype Implementation Results

Although the overhead of adding virtualization to grid systems has previously been evaluated [6], and the overhead of using the IPOP network overlay has been independently studied [9], the combination of both overheads in an overlaid scheduling environment with Virtual Organization Clusters has not been measured. Since the overhead of virtualization would primarily affect job service time in a grid system with sufficient physical resources to handle all jobs concurrently, a chief concern was the amount of latency that might be added by the overlay scheduling system, which necessitated the use of an overlay network. In order to ensure that this overlay overhead would not have unexpected detrimental impacts on compute-bound jobs running within the virtual machines, tests were conducted using an Open Science Grid (OSG) [30] site configured to support virtualization. The OSG grid site was configured with 16 dual-core compute nodes, each with an Intel Xeon 3070 CPU and 4 binary gigabytes (GiB) of Random Access Memory (RAM), with the Kernel-based Virtual Machine (KVM) [31] hypervisor running within a 64-bit installation of CentOS 5.2. Virtual machine images were configured with 32-bit CentOS 5.2 and located on a shared Parallel Virtual FileSystem (PVFS) [26] store. Virtual machine instances were booted directly from the image located on the shared filesystem, without first staging the image to the local compute nodes, using the "snapshot" mode of KVM. These shared images were made available in a read-only configuration, with non-persistent writes redirected to a temporary file on local disk storage at each compute node. Internet connectivity for the test site was provided by an edge router using Network Address Translation (NAT), with the physical compute nodes isolated in a private IPv4 subnetwork. OSG connectivity was provided through the standard OSG software stack including Globus [32].

A VOC head node was constructed using a virtual machine hosted by an offsite laboratory workstation. Condor [7] was installed on the head node to serve as a scheduler, and synthetic workload jobs were submitted directly to the VOC local pool. A watchdog daemon process, using the naive greedy watchdog algorithm with added support to maintain a minimum number of running VMs at all times, was run on the same head node. This watchdog created pilot jobs, which were submitted through a Globus client to the OSG test site, in response to the arrival of synthetic workload jobs in the VOC Condor queue. The pilot jobs started virtual compute nodes, which joined an IPOP network anchored at the workstation by contacting its bootstrap service. Once connected to the private IPOP network, the virtual compute nodes joined the Condor pool created by the collector on the VOC head node (the workstation-hosted virtual machine, also joined via IPOP), and Condor scheduled and executed the actual test jobs. Whenever the watchdog daemon determined that an excess number of pilot

Table 1 Observed makespan lengths and system throughputs for 10 overlay experiment trials

Test	Without Overlay	With Overlay	Change (Absolute)	Change (Relative)
Minimum Makespan (s)	2706	2714	8.000	0.2956 %
Median Makespan (s)	2709	2720	11.00	0.4061 %
Maximum Makespan (s)	2710	2737	27.00	0.9963 %
Mean Makespan (s)	2708	2722	13.50	0.4985 %
Makespan Standard Deviation (s)	1.414	7.735	6.321	447.0 %
Minimum Throughput ($\text{Jobs} \cdot \text{s}^{-1}$)	1.845×10^{-2}	1.827×10^{-2}	-1.820×10^{-4}	- 0.9865 %
Median Throughput ($\text{Jobs} \cdot \text{s}^{-1}$)	1.846×10^{-2}	1.839×10^{-2}	-7.467×10^{-5}	- 0.4045 %
Maximum Throughput ($\text{Jobs} \cdot \text{s}^{-1}$)	1.848×10^{-2}	1.842×10^{-2}	-5.447×10^{-5}	- 0.2948 %
Mean Throughput ($\text{Jobs} \cdot \text{s}^{-1}$)	1.846×10^{-2}	1.837×10^{-2}	9.000×10^{-5}	- 0.4954 %
Throughput Standard Deviation	9.644×10^{-6}	5.207×10^{-5}	4.243×10^{-5}	439.9 %

jobs were running in comparison to the size of the Condor queue, the pilot jobs were instructed to terminate, causing the virtual machines to be terminated.

4.1 Overlay Scheduling and Networking

To measure the relative performance difference of using a VOC with overlay scheduling and IPOP overlay networking, two sets of tests were conducted. A synthetic workload consisting of a 10-minute sleep procedure was devised, in order to approximate compute-bound jobs without incurring potential variations in service times that could result from running an actual compute-bound job within a virtual machine. In the control trials, a batch of 50 sleep jobs was submitted directly to the local scheduler on the physical grid site head node. For the experiment trials, the same batch of 50 jobs was submitted directly to the Condor central manager running within the VOC. Total makespan times were collected for both sets of trials, and each trial was repeated 10 times to reduce the effects of random variation in observed makespan lengths. Descriptive and relative statistics were computed for the makespan times. Throughput measures in jobs per second were also computed.

Results of the trials, summarized in table 1, indicated a slight increase in average makespan time (less than one half of one percent) for jobs submitted through the overlay scheduling system, compared to jobs submitted directly to the physical cluster scheduler. This increased makespan length corresponded to a similarly small decrease in job throughput resulting from the addition of the overlay. In the worst case observed in all trials, the maximum makespan and minimum throughput were affected by less than one percent. Variations in makespan and throughput observations between trials was substantially increased by over 400% when the overlay scheduler and network were added, likely due to the addition of a second layer of interval scheduling with the additional Condor pool overlaid on top of the physical Condor pool. Plotted traces of mean observations (figures 2 and 3) further confirmed the minimal overhead of the VOC overlay system.

4.2 VOC Adjustment Policy

A second experiment was performed to evaluate the behavior of the watchdog daemon when monitoring the private scheduler queue and adjusting the size of the Virtual

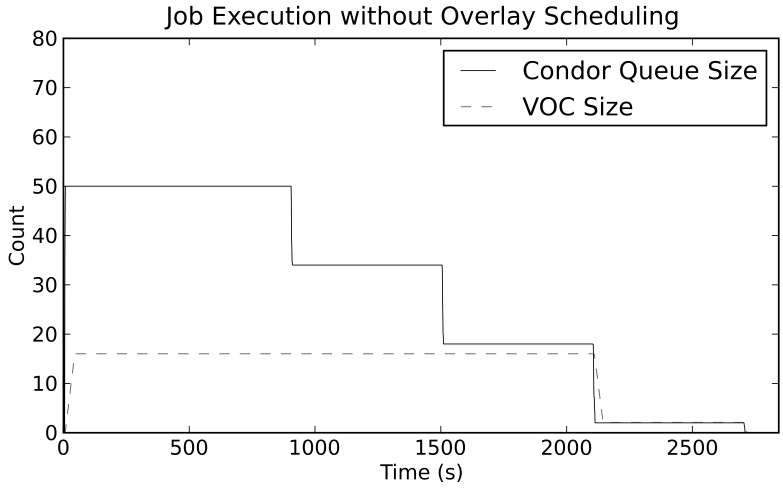

Fig. 2 Autonomic VOC size adjustment behavior when executing long jobs without an overlay network (average of 10 repetitions of the experiment).

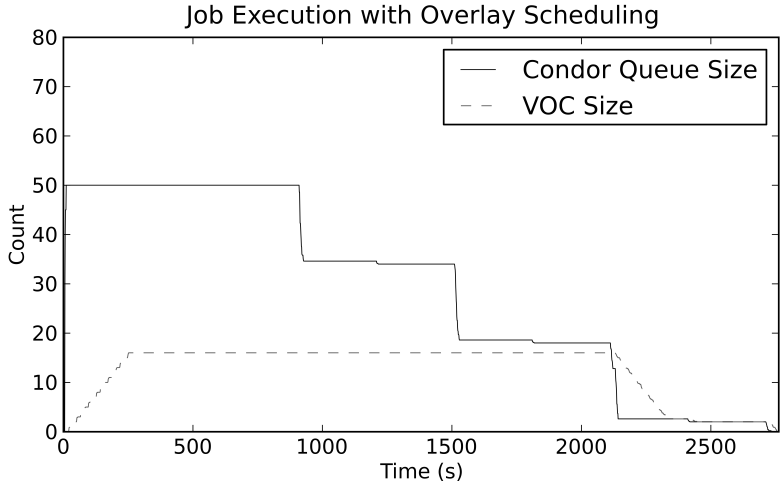

Fig. 3 Autonomic VOC size adjustment behavior when executing short jobs privately scheduled using the IPOP overlay network (average of 10 repetitions of the experiment).

Organization Cluster. As illustrated in figure 4, the simple greedy watchdog algorithm proved to be over-responsive when batches of microbenchmark (10-second) jobs were submitted to the scheduler. The VOC was rapidly expanded to use all 16 available processor cores at the first watchdog interval. A delay of approximately 60 seconds was observed while the VOC nodes booted, after which the short user jobs quickly ran to completion. Even though additional jobs arrived in the queue while the VOC nodes were booting, all jobs from the first two batches had completed within 140 seconds. At this time, the size of the VOC was shrunk to zero, causing the virtual machines to vacate the physical systems completely. When another batch of short jobs arrived

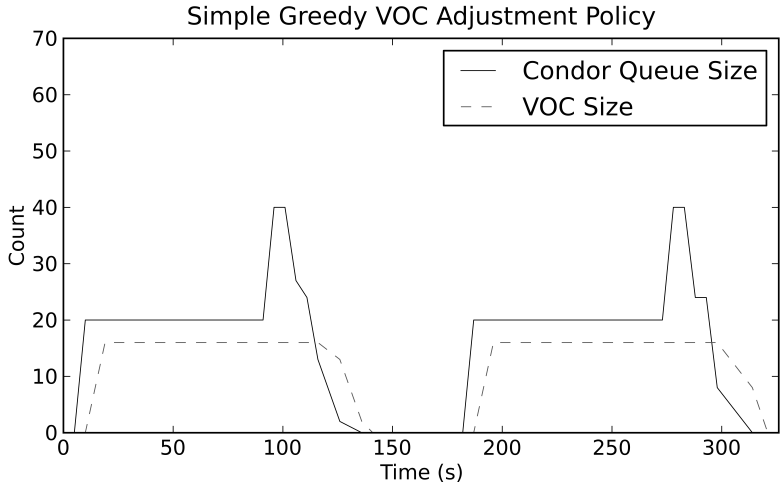

Fig. 4 Simple Greedy Algorithm for autonomic VOC size adjustment: VMs are started whenever there are excess jobs in the scheduler queue and free physical nodes available. Once the jobs complete and the queue size decreases, VMs are terminated quickly.

at 180 seconds into the test, all 16 virtual machines had to be restarted, resulting in another boot delay.

To provide a buffer against excessively short jobs, a Delayed Response adaptation algorithm was devised. This policy resulted in the immediate creation of a pair of VOC nodes to remain active at all times for the handling of instantaneously short (by design or by failure) jobs. In addition, the VOC was expanded by only one node at a time, and expansion only occurred if the size of the Condor scheduler queue exceeded the VOC size for at least 10 watchdog intervals. Similarly, the VOC was decreased by one node at a time, to a minimum size of two nodes, only when the size of the VOC exceeded the size of the Condor queue for at least 10 watchdog intervals. The results of submitting two batches of short jobs have been illustrated in figure 5, which shows a slow increase in the number of VOC nodes in response to the first batch of jobs. A slow decrease in the number of VOC nodes was observed between batches, followed by another slow increase as the second batch of jobs arrived. Once all jobs from the second batch completed, the VOC size slowly declined to the minimum size (two) specified by the policy.

5 Simulation Results

Although tests on the prototype, as presented in section 4, suggested the addition of both overlay networking (IPOP) and overlay scheduling (Condor) would have minimal impact on job execution, the results did not account for either the behavior of such a system at scale or the additional overhead of operating system virtualization, which was not tested by the microbenchmark sleep processes. In order to ensure the viability of VOCs over a real production grid, simulations were required. Therefore, a simulation system was developed, called the Simulator for Virtual Organization Clusters (SimVOC) [33]. Following the taxonomy provided by Sulistio et al. [34], SimVOC was

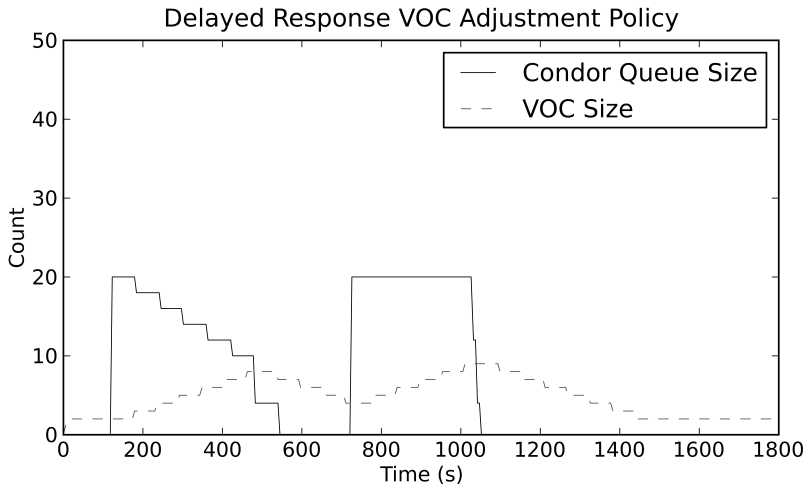

Fig. 5 Delayed Response Algorithm for autonomic VOC size adjustment: a minimum of 2 VMs are kept in operation at all times, and VMs are started and stopped in response to queue size trends over a time series of samples.

designed to be a dynamic, trace-driven, discrete-event simulation system with continuous output, hybrid grid model, and a serialized (single-threaded) kernel. As utilized in the experiments described here, SimVOC was operated via Python driver applications in a completely deterministic execution mode. For simplicity, the simulation system did not model cross-site authentication and authorization. Instead, it was assumed that each physical grid site permitted jobs from users affiliated with any Virtual Organization observed in the trace inputs. For the purpose of simulating VOCs, a synthetic VO named "_pilot_" was created, and it was assumed that every grid site would accept virtualization jobs from the "_pilot_" VO.

As introduced in the previous section, the choice of watchdog algorithm was found to impact the aggregate behavior of the grid whenever VOCs were in use. An adaptation to the Delayed Response Algorithm was made for simulation purposes, in which the minimum level of virtual machines – called the "target level" – was achieved by starting the target number of VMs whenever the first user job arrived for a particular VOC. This minor optimization aided in the comparative analysis of results when compared to the naive greedy algorithm. A second adaption, which permitted the total number of machines for a single VOC to be limited to a fixed upper bound, was not utilized in these simulation experiments.

5.1 Trace-Driven Simulations

In order to use a realistic distribution of job metrics, trace inputs to the simulation system were acquired from the Grid Observatory [35] and converted to a format usable by the simulator. The traces consisted of a dynamic map of the Enabling Grids for E-sciencE (EGEE) grid; the set of jobs, including submission times, lengths, and VO relationships, observed on EGEE by the gLite middleware [36] from 00:00 UTC on December 8, 2008, to 23:59 UTC on May 10, 2009; and a dynamic list of Virtual

Organizations found on the grid over the same time period. An aggregation function of the simulation system was used to calculate the total number of jobs running on grid sites and waiting in scheduler queues grid-wide.

Since the available data sets listed the Computing Elements (CEs), or grid sites, present on EGEE and not the actual cluster systems, it was necessary to adjust the CPU core count. Multiple grid sites frequently share the same underlying hardware, resulting in a substantial over-count of actual CPU cores. Since all site names were given as fully-qualified domain names, a set of heuristics was devised to reduce the over-count by detecting sites on the same domain with the same CPU count, then merging the CPUs of all sites on the same domain into a single cluster. The size of the merged cluster was set to the largest observed CPU count for all CEs on the same domain. Without these heuristics, the total number of CPU cores on the EGEE grid at 00:00 UTC on December 8, 2008 was reported to be 314,547, which was determined to be an unrealistic overestimate. After application of the heuristics, the total number of CPU cores on EGEE on the same date was reduced to 107,396.

An additional data issue resulted from the relative timing of job arrival and Virtual Organization registration, relative to the grid map data. Since the job and VO data were derived from the job traces, while the map data were extracted from a different data set, some discrepancies were found in the timing. In particular, jobs could not always be matched to target sites or to affiliated VOs, resulting in jobs receiving a zero execution length and an error status, since the simulator was unable to find the resources on which the jobs were supposed to run.

5.2 Traditional Grid Systems

To check the non-scheduler portion of the simulated grid model, a control simulation was effected, in which the actual job start and finish times from the EGEE job input trace were used to start and stop jobs on sites. As illustrated in figure 6 and table 2, most jobs executed on the actual EGEE grid were compelled to wait in the queue for some period of time, averaging 2,740 s, before starting execution. Combining the wait time and service time, jobs experienced an average sojourn time (time from submission to completion) of 11,700 s. However, the median sojourn time was only 402 s. This time discrepancy, along with the substantially smaller median service time compared to the mean service time, indicated that the distribution of job lengths was right-tailed, with an abundance of short jobs. Issues with the trace data also resulted in an absence of jobs between December 29, 2008 and January 18, 2009, as evidenced between hours 500 and 1000 in figure 6.

Significant differences were observed between the simulator and the EGEE data once the scheduling portion of the simulated grid model was enabled. Results of this "standard" simulation, illustrated in figure 7 and summarized in table 2, indicated greatly increased job queuing behavior, with an order of magnitude increase in observed job waiting time and an attendant increase in job sojourn time. The greatly reduced performance of the simulated grid in this case was determined to have been caused by a naive setting of the scheduling interval (300 s) for those sites using the Condor scheduler, resulting in a mean wait time increase of nearly half an hour. Grid sites using the Condor scheduler were modeled using an interval scheduler optimized for High-Throughput Computing (HTC) applications, whereas sites specifying other schedulers were modeled using a zero-interval scheduler optimized for High-Performance Com-

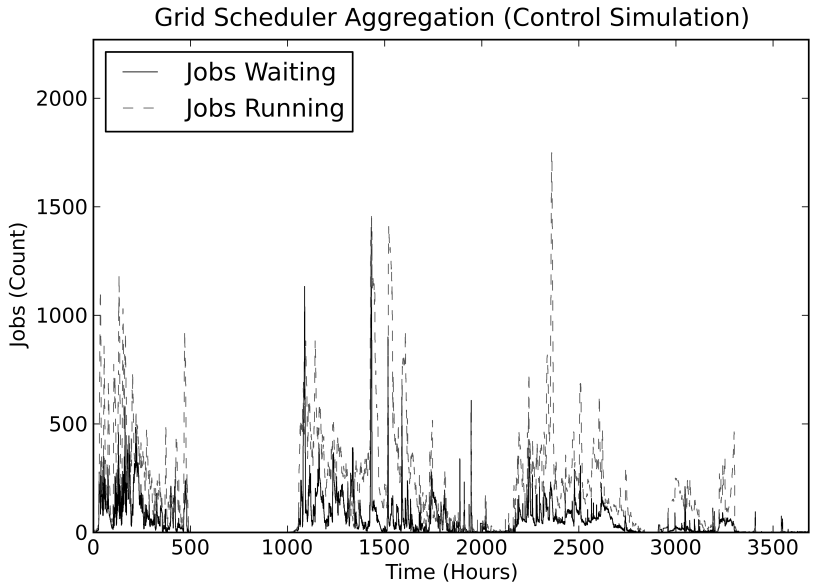

Fig. 6 Control Simulation: Jobs were started and stopped in the simulator according to the timestamps present in the raw input data, without running the simulated schedulers.

puting (HPC). Thus, the median wait time was reduced to zero as a large number of jobs targeted at sites with non-Condor schedulers started instantly. Nevertheless, the reduced performance of the simulated Condor scheduler resulted in a larger amount of aggregate queuing, indicating poorer performance than was actually observed on the physical EGEE grid.

5.3 Adding Virtual Organization Clusters

Following the standard simulation, Virtual Organization Clusters were added to the simulated grid. One simulated VOC was constructed for each Virtual Organization observed in the input job trace, for a total of 50 VOCs. Each VOC utilized the Condor scheduler, again with a naive 300 s interval setting, and was given a unique grid Compute Element (CE) named with a prefixed version of the VO name. Jobs from the EGEE input trace were modified so as to be submitted to the unique CE corresponding to the job VO, instead of submission directly to a simulated grid site. Each VO CE was equipped with a simulated watchdog, and experiment sets were constructed using both a naive greedy VOC adjustment algorithm and a greedy algorithm with a minimum target level set to 1024. Although this target level was regarded as high, the total number of physical CPU cores that would be utilized at the target level for all 50 VOCs running simultaneously would have been 51,200, slightly under 50% of the cores present in the simulated grid. Each set of VOC experiments was repeated for both the KVM and Xen hypervisors, using 8.8% and 6.6% overheads (respectively) for compute-bound jobs as measured in prior work [6].

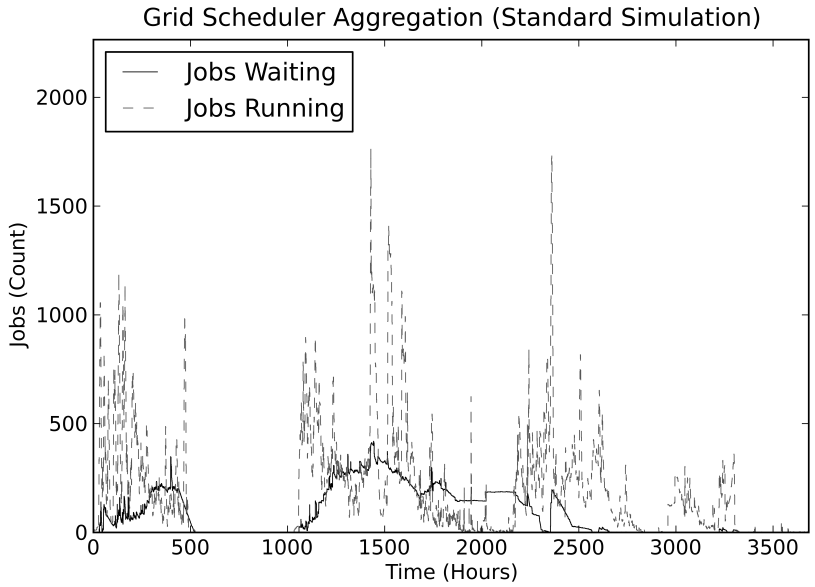

Fig. 7 Standard Simulation: The entire grid system, including the schedulers, was simulated, using the submission time, job length, and target grid site from the input data set.

As illustrated in figures 8 through 11 and summarized in table 2, the addition of Virtual Organization Clusters to the simulated EGEE grid model reduces job queuing substantially. Regardless of the choice of the KVM or Xen hypervisor, the addition of VOCs with the naive watchdog algorithm reduced mean job waiting times by 83% compared to the actual EGEE data from the control experiment, and 90% compared to the standard simulation experiment. When the target level was set to 1024, wait times were reduced by another 9% compared to the naive algorithm, resulting in a 91% reduction from the standard test. For both tests, all Condor intervals – both within the VOCs and on the simulated physical sites – were set at the same naive levels used for the poorly-performing standard simulation experiment.

The addition of virtualization overhead did increase both the mean and median job service (execution) times, as expected. However, the increase in service times was largely offset by the decrease in waiting time, resulting in the same mean job sojourn time for both the KVM VOC (without target levels) and standard architecture simulations. A slight decrease in sojourn times was observed when the Xen hypervisor was simulated. At scale, the addition of target levels demonstrated no improvement over the naive greedy watchdog algorithm, with an observed increase in mean sojourn times observed for both KVM and Xen. As noted in table 2, maximum sojourn times for any job decreased relative to the standard simulation whenever VOCs were in use. However, one job experienced an exceptionally long queuing delay when the Xen VOC test was conducted with target levels set.

To lease the simulated physical grid resources, pilot jobs were required. These jobs had the effect of doubling the utilization of the grid at any time at which user jobs were running, since there was a 1-to-1 correspondence between user jobs and pilot jobs when the naive greedy watchdog algorithm was used. Pilot job requirements for either

Table 2 Measured statistics for simulation experiments. Values are exact for job counts and to 3 significant figures for time measurements. All time measurements are in seconds. KVM-0 and Xen-0 refer to simulations without target levels set, while KVM-1024 and Xen-1024 refer to simulations with target levels set to 1024. Since jobs were recorded whenever jobs finished, and pilot jobs for target-level VOC simulations were left running when above the target level, pilot job counts for these simulations were not recorded.

Measure	Control	Standard	KVM-0	KVM-1024	Xen-0	Xen-1024
User Jobs	258,097	258,097	258,097	258,097	258,097	258,097
Pilot Jobs	0	0	149,888	N/M	150,189	N/M
User Jobs Discarded	101,524	101,293	9,151	120	8,939	115
User Jobs Executed	156,573	156,804	248,946	257,977	249,158	257,982
Min Wait Time	0.00	0.00	0.00	0.00	0.00	0.00
Median Wait Time	206	0.00	316	164	316	164
Max Wait Time	823,000	2,380,000	462,000	462,000	462,000	2,090,000
Mean Wait Time	2,740	4,470	457	417	457	418
Stddev Wait Time	15,100	62,800	6,240	6,400	6,240	7,610
Min Service Time	0.00	0.00	0.00	0.00	0.00	0.00
Median Service Time	111	111	885	1,000	878	982
Max Service Time	478,000	478,000	919,000	919,000	901,000	901,000
Mean Service Time	9,000	9,020	13,100	13,300	12,800	13,000
Stddev Service Time	23,200	23,200	31,900	32,300	31,400	31,700
Min Sojourn Time	0.00	0.00	0.00	0.00	0.00	0.00
Median Sojourn Time	402	111	1,190	1,200	1,180	1,170
Max Sojourn Time	1,040,000	2,430,000	920,000	919,000	901,000	2,090,000
Mean Sojourn Time	11,700	13,500	13,500	13,700	13,300	13,400
Stddev Sojourn Time	30,300	68,100	32,600	33,200	32,200	32,900

hypervisor were determined to be nearly identical based on the experiments (table 2), with approximately 150,000 jobs required in both cases. This number was less than the total number of user jobs executed in either case due to re-use of existing virtual machines whenever possible. Since the simulation system recorded job results only upon completion of the jobs, and the simulations ended without completion of the pilot jobs in cases where target levels were set, the total number of pilot jobs utilized was not measured when the target level watchdog algorithm was employed.

Another effect of adding Virtual Organization Clusters to the grid was that 58% more user jobs were able to run, as recorded in table 2. With VOCs, user jobs were not submitted to physical grid sites; instead, these jobs were submitted to virtual grid sites created for each Virtual Organization observed in the trace. As a result, discrepancies between the grid map data and job trace data did not result in job errors. However, timing issues between VO registration (which resulted in associated VOC head node creation) and job arrival did result in a small set of user jobs failing to execute (3.5% in the worst case, compared to 39% in the standard simulation). Since a greater number of jobs executed, while total queuing across the grid was substantially decreased, it was determined that the primary contributor to queuing on the actual grid system was a result of jobs targeting specific grid sites.

6 Conclusions

The use of pilot jobs and IPOP overlay networking enables the provisioning of Virtual Organization Clusters with overlay scheduling, permitting each Virtual Organization

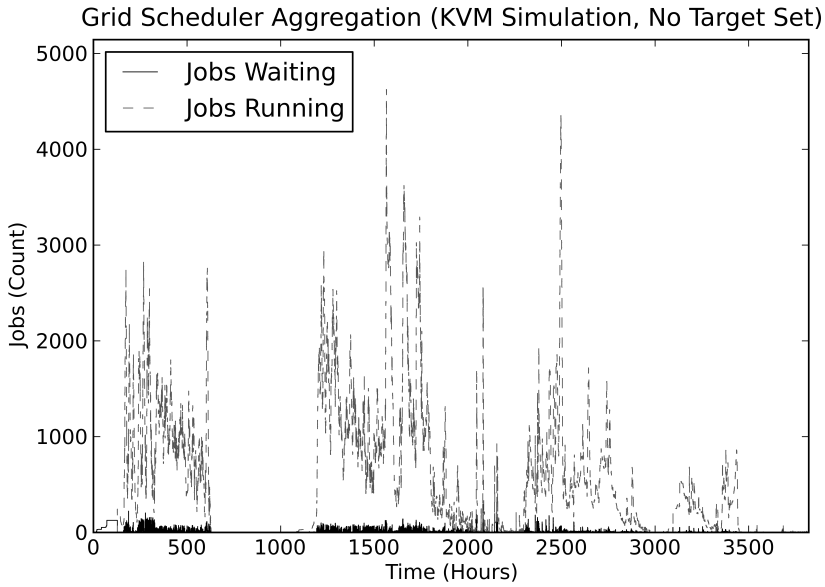

Fig. 8 KVM simulation without the target level set

to make resource allocation and job priority decisions within its private virtual environment. VOCs are similar to pilot job frameworks used by High-Energy Physics (HEP) experiments [37]. As demonstrated through tests using a prototype grid-connected system, the added overhead of the scheduling and network overlay is negligible for compute-bound grid jobs. Simulation results using actual trace data from the EGEE grid indicate that widespread VOC deployment on a grid system would not adversely affect the aggregate behavior of the grid, even though virtualization systems add execution overhead. VOCs reduce total aggregate queuing by making all jobs compatible with all sites composed of machines with the same instruction set architecture. Moreover, the virtual head node for each VOC creates a single submission point for all jobs affiliated with a particular VO, simplifying job submission for grid users.

Since Virtual Organization Clusters could be deployed system-wide on an existing production grid without causing performance problems, VOCs are a promising mechanism for delivering the benefits of grid virtualization systems, including environment customization, VO isolation, and legacy application support [11], to existing production grids without large-scale disruption. Through the use of overlay scheduling, individual VOs will be able to make resource allocation decisions for their members, allowing site and VO policies to be independent. Furthermore, the customization capabilities offered by virtualization will empower VOs to provide the software stacks required by their users, instead of forcing users to adapt applications to the available software environments installed by site administrators. As a result, existing computational grids can be made more useful for domain applications and more accessible to domain experts, without forcing users into system administration roles. VOCs thus provide a mechanism for leveraging existing infrastructure to provide new computational opportunities.

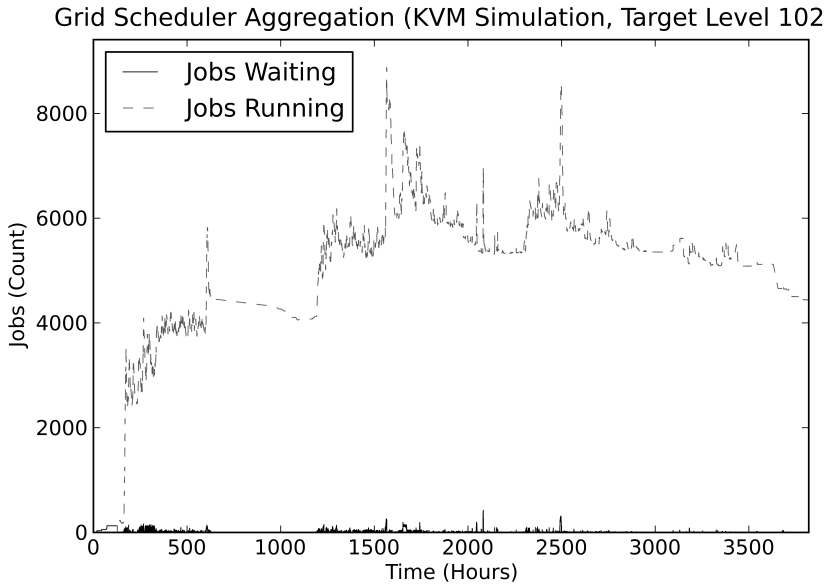

Fig. 9 KVM simulation with the watchdog target level set to 1024

Acknowledgments

The authors would like to thank Prof. Pradip Srimani of the Clemson University School of Computing for his helpful feedback for improvement of this paper. Simulation datasets used in this work have been provided by the Grid Observatory (www.grid-observatory.org). The Grid Observatory is part of the EGEE-III EU project INFSO-RI-222667. This material is based upon work supported under a National Science Foundation Graduate Research Fellowship. Additional support has been provided by the National Science Foundation and the United States Department of Energy via the Open Science Grid.

References

1. Murphy, M.A., Fenn, M., Goasguen, S.: Virtual Organization Clusters. In: 17th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2009), Weimar, Germany (February 2009)
2. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the Grid: Enabling scalable virtual organizations. International Journal of Supercomputing Applications **15**(3) (2001) 200–222
3. Murphy, M.A., Kagey, B., Fenn, M., Goasguen, S.: Dynamic provisioning of Virtual Organization Clusters. In: 9th IEEE International Symposium on Cluster Computing and the Grid (CCGrid '09), Shanghai, China (May 2009)
4. Beaumont, O., Carter, L., Ferrante, J., Legrand, A., Marchal, L., Robert, Y.: Centralized versus distributed schedulers for bag-of-tasks applications. IEEE Transactions on Parallel and Distributed Systems **19**(5) (May 2008) 698–709
5. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: Nineteenth ACM Symposium on Operating Systems Principles. (2003)

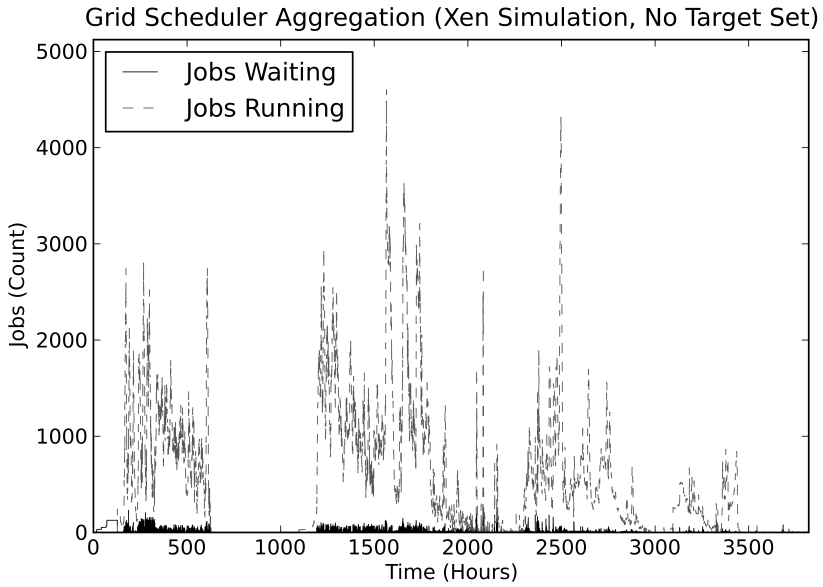

Fig. 10 Xen simulation without the target level set

6. Fenn, M., Murphy, M.A., Goasguen, S.: A study of a KVM-based cluster for grid computing. In: 47th ACM Southeast Conference (ACMSE '09), Clemson, SC (March 2009)
7. Tannenbaum, T., Wright, D., Miller, K., Livny, M.: Condor – a distributed job scheduler. In Sterling, T., ed.: Beowulf Cluster Computing with Linux. MIT Press (October 2001)
8. Xu, D., Ruth, P., Rhee, J., Kennell, R., Goasguen, S.: Autonomic adaptation of virtual distributed environments in a multi-domain infrastructure. In: 15th IEEE International Symposium on High Performance Distributed Computing (HPDC'06), Paris, France (June 2006)
9. Ganguly, A., Agrawal, A., Boykin, P.O., Figueiredo, R.: IP over P2P: Enabling self-configuring virtual IP networks for grid computing. In: 20th International Parallel and Distributed Processing Symposium (IPDPS 2006). (2006)
10. Figueiredo, R., Dinda, P.A., Fortes, J.: Resource virtualization renaissance. Computer **38**(5) (May 2005) 28–31
11. Figueiredo, R.J., Dinda, P.A., Fortes, J.A.B.: A case for grid computing on virtual machines. In: 23rd International Conference on Distributed Computing Systems. (2003)
12. Tsugawa, M., Fortes, J.A.B.: A virtual network (ViNe) architecture for grid computing. In: 20th International Parallel and Distributed Processing Symposium (IPDPS 2006). (2006)
13. Sundararaj, A.I., Dinda, P.A.: Towards virtual networks for virtual machine grid computing. In: Third Virtual Machine Research and Technology Symposium, San Jose, CA (May 2004)
14. Davoli, R.: VDE: Virtual Distributed Ethernet. In: First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (Tridentcom 2005), Trento, Italy (February 2005)
15. Adabala, S., Chadha, V., Chawla, P., Figueiredo, R., Fortes, J., Krsul, I., Matsunaga, A., Tsugawa, M., Zhang, J., Zhao, M., Zhu, L., Zhu, X.: From virtualized resources to virtual computing grids: the In-VIGO system. Future Generation Computer Systems **21**(6) (June 2005) 896–909
16. Keahey, K., Foster, I., Freeman, T., Zhang, X., Galron, D.: Virtual workspaces in the Grid. In: 11th International Euro-Par Conference, Lisbon, Portugal (September 2005)
17. Sotomayor, B., Keahey, K., Foster, I.: Combining batch execution and leasing using virtual machines. In: 17th International Symposium on High Performance Distributed Computing (HPDC 2008). (2008)

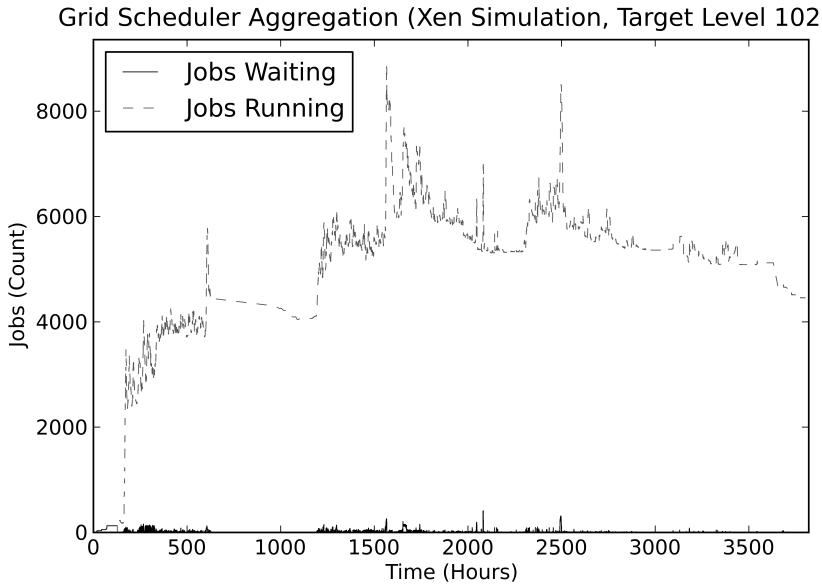

Fig. 11 Xen simulation with the watchdog target level set to 1024

18. Foster, I., Freeman, T., Keahey, K., Scheftner, D., Sotomayor, B., Zhang, X.: Virtual clusters for grid communities. In: 6th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2006), Singapore (May 2006)
19. Irwin, D., Chase, J., Grit, L., Yumerefendi, A., Becker, D., Yocum, K.: Sharing network resources with brokered leases. In: USENIX Technical Conference, Boston, MA (June 2006)
20. Chase, J.S., Irwin, D.E., Grit, L.E., Moore, J.D., Sprenkle, S.E.: Dynamic virtual clusters in a grid site manager. In: HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing. (June 2003)
21. Grit, L., Irwin, D., Yumerefendi, A., Chase, J.: Virtual machine hosting for networked clusters: Building the foundations for 'autonomic' orchestration. In: First International Workshop on Virtualization Technology in Distributed Computing (VTDC '06), Tampa, FL (November 2006)
22. Ramakrishnan, L., Grit, L., Iamnitchi, A., Irwin, D., Yumerefendi, A., Chase, J.: Toward a doctrine of containment: Grid hosting with adaptive resource control. In: 19th Annual Supercomputing Conference (SC '06), Tampa, FL (November 2006)
23. Ruth, P., McGachey, P., Xu, D.: VioCluster: Virtualization for dynamic computational domains. In: IEEE International Conference on Cluster Computing, Boston, MA (September 2005)
24. Ruth, P., Jiang, X., Xu, D., Goasguen, S.: Virtual distributed environments in a shared infrastructure. Computer **38**(5) (2005) 63–69
25. Emeneker, W., Stanzione, D.: Dynamic virtual clustering. In: 2007 IEEE International Conference on Cluster Computing. (2007)
26. Carns, P.H., Ligon, W.B., Ross, R.B., Thakur, R.: PVFS: A parallel file system for Linux clusters. In: ALS'00: Proceedings of the 4th annual Linux Showcase and Conference. (2000)
27. Keahey, K., Freeman, T.: Contextualization: Providing one-click virtual clusters. In: 4th IEEE International Conference on e-Science, Indianapolis, IN (December 2008)
28. Boykin, P.O., Bridgewater, J.S.A., Kong, J.S., Lozev, K.M., Rezaei, B.A., Roychowdhury, V.P.: A symphony conducted by Brunet. Online (September 2007)
29. Keahey, K., Doering, K., Foster, I.: From sandbox to playground: Dynamic virtual environments in the Grid. In: 5th International Workshop on Grid Computing (Grid 2004), Pittsburgh, PA (November 2004)

30. Pordes, R., Petravick, D., Kramer, B., Olson, D., Livny, M., Roy, A., Avery, P., Blackburn, K., Wenaus, T., Würthwein, F., Foster, I., Gardner, R., Wilde, M., Blatecky, A., McGee, J., Quick, R.: The Open Science Grid: Status and architecture. In: International Conference on Computing in High Energy and Nuclear Physics (CHEP '07). (2007)
31. Red Hat: Kernel-based Virtual Machine, http://www.linux-kvm.org
32. Foster, I., Kesselman, C.: Globus: A metacomputing infrastructure toolkit. International Journal of Supercomputing Applications **11**(2) (1997) 115–128
33. Clemson Cyberinfrastructure Research Group: SimVOC, http://cirg.cs.clemson.edu/software/simvoc
34. Sulistio, A., Yeo, C.S., Buyya, R.: A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. International Journal of Software: Practice and Experience **34**(7) (June 2004) 653–673
35. EGEE: Grid observatory, http://www.grid-observatory.org
36. Enabling Grids for E-sciencE: gLite, http://glite.web.cern.ch/glite/
37. Sfiligoi, I., Quinn, G., Green, C., Thain, G.: Pilot job accounting and auditing in Open Science Grid. In: 9th IEEE/ACM International Conference on Grid Computing (Grid '08). (2008)