Elastic Management of Cluster-based Services in the Cloud *

Rafael Moreno-Vozmediano, Ruben S. Montero, Ignacio M. Llorente
Dept. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid
28040 Madrid, Spain
{rmoreno,rubensm,llorente}@dacya.ucm.es

ABSTRACT

In this paper we analyze the deployment of generic clustered services on top of a virtualized infrastructure layer that combines a VM manager (the OpenNebula engine) and a cloud resource provider (Amazon EC2). The use of this virtualization layer between the service and the physical infrastructure extends the classical benefits of VM platforms to distributed infrastructures. Additionally, the integration of the cloud in this layer allows us to give additional capacity to the services using an external provider, thus complementing the local infrastructure without notice from the users or affecting the service workload. This flexible approach, which separates the resource provisioning from the service management, provides important benefits: elastic service capacity to adapt it to its dynamic workload; physical infrastructure partitioning to isolate it from other running services; and support for heterogeneous configurations tailored for each service class. The feasibility of the proposed approach is analyzed for two different clustered services: a classical computing cluster and a web server.

Categories and Subject Descriptors

K.6.4 [**Management of Computing and Information Systems**]: System Management; C.2.4 [**Computer - Communication Networks**]: Distributed Systems

General Terms

Management, Performance

*This research was supported by Consejería de Educación de la Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BIOGRIDNET Research Program S-0505/TIC/000101, by Ministerio de Educación y Ciencia, and through the research grant TIN2006-02806, and by the European Union through the research grant RESERVOIR Contract Number 215605

ACDC'09, June 19, 2009, Barcelona, Spain.

Keywords

Virtualization, Cloud Computing, Cluster-based Services

1. INTRODUCTION

Recently, virtualization has brought about a new utility computing model called *cloud computing*, for the on-demand provisioning of virtualized resources as a service. The Amazon Elastic Compute Cloud (Amazon EC2, see [1]) is probably the best example of this new paradigm for elastic capacity provisioning. Thanks to virtualization, the clouds can be used efficiently to supplement the capacity of local services with outsourced resources.

This resource provisioning model can be seamlessly integrated with the in-house physical infrastructure when it is combined with a virtual machine (VM) management system. A VM manager is responsible for the efficient management of the virtual infrastructure as a whole, by providing basic functionality for the deployment, control and monitoring of VMs on a distributed pool of resources. Usually, these VM managers also offer high availability capabilities and scheduling policies for VM placement and physical resource selection.

The use of this virtualization layer between the service and the physical infrastructure extends the classical benefits of VM platforms (isolation, encapsulation, hardware independence, support for heterogeneous software stacks, etc.) to distributed infrastructures, thus enabling new extra capacities, such as on demand resource provisioning, server consolidation, workload balance, dynamic partitioning and resizing of the physical infrastructure, etc.

One of the main goals of this work is to show the benefits of this architecture, which totally decouples the infrastructure management from the service management, and enables the dynamic provisioning of virtual resources on demand, to adapt the infrastructure to the service requirements. This approach is fully transparent for the service itself, and independent of the type of service. Furthermore, this provisioning model can be integrated with external cloud providers, to provide additional elastic capacity to the virtual infrastructure when the service demands increase or to satisfy peak demand periods.

In order to validate and prove the feasibility of the proposed architecture, we analyze the deployment of two cluster-based services of a disparate nature on top of this virtualization layer, namely: a batch-processing computing cluster, and a transactional web server platform. Cluster-based architectures are usually used to deliver high-performing, scalable and fault-tolerant services [2]. The appearance of cloud

computing provides a potential avenue for adding an extra flexibility to these cluster-based services. Clustered services can be scaled on-demand by adding worker nodes from the cloud. By managing the redundancy on local and cloud resources high availability can be increased. Since the system is built out of cloud resources using an utility cost model the cost-effectiveness of the clustered service is greatly improved.

In Section 2 we briefly describe the main components and characteristics of the proposed architecture for the elastic management of cluster-based services. Then in Section 3 we detail the experimental environment used in this paper. We also analyze in Section 4 the performance of the virtual service in the execution of representative benchmarks. Section 5 reviews related work, and the paper ends with some conclusions in Section 6.

2. ELASTIC MANAGEMENT OF CLUSTER-BASED SERVICES

Typically, a cluster-based service consists of a front-end server, which usually acts as a service end-point and workload balancer, and a variable number of back-end servers (or *worker nodes*), interconnected with a LAN, that serve the user requests. This architecture has been applied to implement a wide range of services, from network services (e.g. search engines) to batch processing services like computing clusters. However, these cluster based services present several limitations.

Successful Internet-based business models are totally dependent on the efficiency of modern web platforms. One of the main difficulties these web platforms is their adaptation to variable user demands, what usually leads to an over-provisioning of the cluster, in order to satisfy peak demand periods. That entails a strong investment on back-end computers, which are most of the time underutilized, a large power consumption, and many administrative efforts.

Computing clusters have been used for decades as the primary computational platform for scientific applications. Although cluster architectures and related tools have been considerably improved, these system present some inherent difficulties, namely: (i) support for heterogeneous configurations (e.g. different versions of the same program or library); (ii) isolation of different workloads; and (iii) performance partition of the cluster.

The above pathologies of these two cluster-based services can be mitigated if the underlying cluster is virtualized. A cluster can be easily virtualized by putting the front-end and worker nodes into VMs. In this work, the functionality needed to deploy, control, and monitor these VMs is provided by the OpenNebula [3, 4] virtual infrastructure engine. OpenNebula provides a uniform management layer regardless of the underlying virtualization technology.

In this way, OpenNebula can be easily integrated with cloud services by using a specific Amazon EC2 plug-in. The plug-in assumes that a suitable Amazon machine image (AMI) has been previously packed and registered in the S3 storage service, so when a given VM is to be deployed in EC2 its AMI counterpart is instantiated. The EC2 plug-in then converts the general requests made by OpenNebula core, such as deploy or shutdown, using the EC2 API. Figure 1 depicts the virtual cluster components, OpenNebula and its interaction with the Amazon cloud.

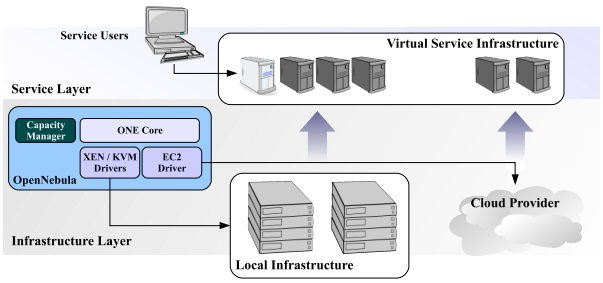

Figure 1: Distributed virtual infrastructure running on top of a geographically distributed physical infrastructure consisting of resources from the private infrastructure and external cloud providers.

The separation of resource provisioning, managed by OpenNebula, from service management, provides the following benefits:

- *Elastic cluster capacity.* The capacity of the cluster can be modified by deploying (or shutting down) virtual worker nodes on demand, either in local physical resources or in remote EC2 resources.
- *Cluster partitioning.* The physical resources of the data center could be used to execute worker nodes bound to different cluster-based services, and thus isolating their workloads and partitioning the performance assigned to each one.
- *Heterogeneous configurations.* The virtual worker nodes of a service can have multiple (even conflicting) software configurations. The service images can be maintained with a minimal operational cost, following an *install once deploy many* approach.

The above management is absolutely transparent to the user or the service workload that is being executed in the virtual cluster, as they are unaware of the physical resource (and its location) that is hosting each VM. So, the users and services preserve their uniform view of all the virtual worker nodes.

3. EXPERIMENTAL ENVIRONMENT

The physical infrastructure used in this work consists of five hosts (Host0 to Host4), which are interconnected by a Gigabit Ethernet LAN. Each physical host node has a dual 2.0 GHz Xeon processor and 8GB of RAM. The remote VMs, deployed on Amazon EC2, are based on an EC2 small standard instance, equivalent to 1.0-1.2 GHz Xeon processor.

The Host0 acts as the front-end of the physical pool and it is also connected to the Internet. This host runs the OpenNebula engine, which has the ability to deploy, manage and monitor local VMs on any host from the physical pool (using the XEN hypervisor) and also remote VMs on Amazon EC2.

The deployment of VMs (either local or remote) by OpenNebula can be controlled manually or can be done automatically by the scheduler module. In this case, the scheduling policy limits the number of VMs per physical host to a given

threshold. When this limit is reached and the cluster needs to grow, OpenNebula will deploy on-demand remote VMs, hosting new worker nodes on Amazon EC2.

The virtual computing cluster consists of a front-end node and a variable set of worker nodes, as shown in figure 2. Job submission and execution within the cluster is managed by Sun Grid Engine (SGE) software [5]. The virtual cluster front-end (SGE master host) has been deployed locally in the Host0, since it needs to have Internet connectivity to be able to communicate with Amazon EC2 virtual machines. This cluster front-end acts also as NFS and NIS server for every worker node in the cluster.

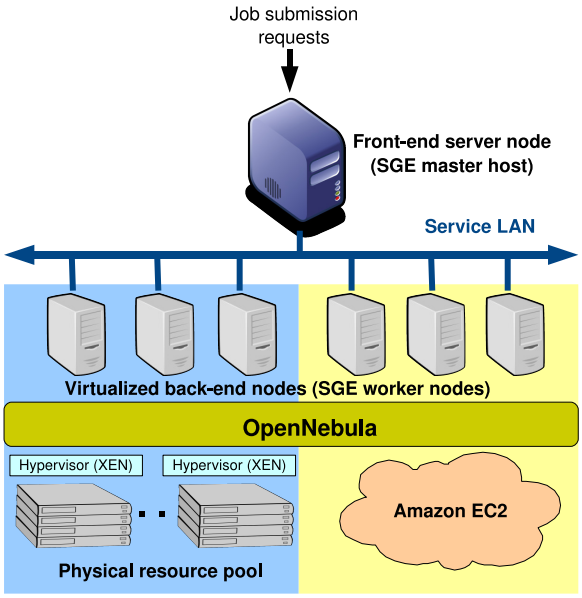

Figure 2: Virtual computing cluster infrastructure

On the other hand, the virtual web server cluster consists of a server front-end that runs the Nginx reverse proxy software and distributes the user HTTP requests among the different virtual back-end servers, which run the Nginx web server, as shown in figure 3. Nginx is a lightweight open source web server, which provides efficient low-overhead HTTP and reverse proxy handling. Nginx includes a simple built-in load balancer module (`upstream` module) for distributing request among back-end servers based on a simple round-robin algorithm. However, there are other third party modules for Nginx that provide more sophisticated load balancing techniques.

Figure 4 shows the network infrastructure used for implementing both virtualized cluster services. Every virtual back-end node communicates with the front-end trough the service LAN. The local back-end nodes and the front-end are directly connected to this network by means of a virtual bridge configured in every physical host. On the other hand, the remote back-end nodes (deployed on Amazon EC2) are connected to the service LAN by means of a virtual private network (VPN) tunnel, using the OpenVPN software [6]. This tunnel is established between each remote node (OpenVPN client) and the cluster front-end (OpenVPN server). It is obvious that the VPN software can introduce some extra latencies in the communication between the front-end and

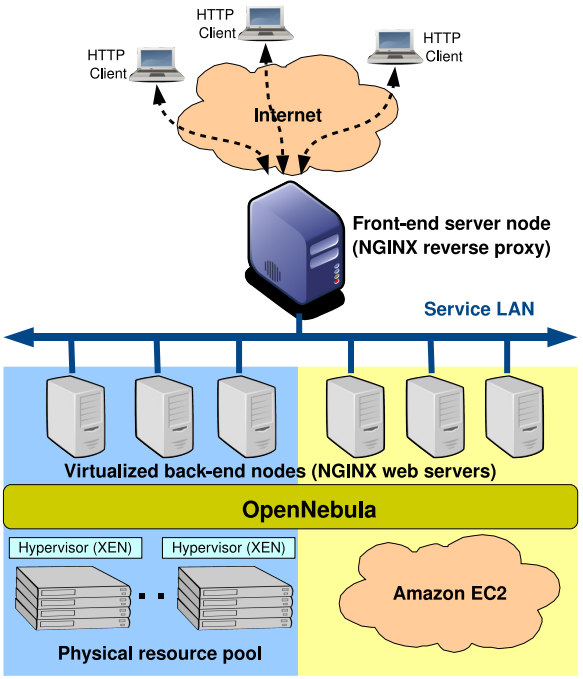

Figure 3: Virtual web server cluster infrastructure

the remote back-end nodes, however, it also involves important benefits. First, all back-end nodes (either local or remote) are accessed in a similar way through the private local area network, which provides higher transparency to the cluster architecture. Second, although virtual nodes deployed on Amazon EC2 have a public network interface, they can be configured to only accept connections through the private interface implemented by the OpenVPN tunnel; this configuration provides the same protection degree to the remote back-end nodes as to the local ones, since the front-end can apply the same filtering and firewalling rules to prevent them from unauthorized or malicious access. Finally, communications between the front-end and the remote back-end nodes are encrypted inside the OpenVPN tunnel, so privacy is guaranteed through the public Internet connection.

4. PERFORMANCE ANALYSIS

In this section we present a performance analysis for both cluster-based services: the batch-processing computing cluster, and the clustered web server platform.

4.1 Computing cluster performance

We present some application level benchmarks to study the behavior of the computing computing cluster from the application's point of view (see [7] for a detailed benchmarking of the Amazon Web Services). In particular, we will use the *Embarrassingly Distributed* (ED) benchmark from the *NAS Grid Benchmarks* [8, 9] (NGB) suite. The ED benchmark models a typical HTC (*High Throughput Computing*) application, which consists of multiple independent runs of the same program, but with different input parameters.

Let us first analyze the performance degradation introduced by the virtualization layer. Table 1 shows the results of running one iteration of the ED benchmark, for different

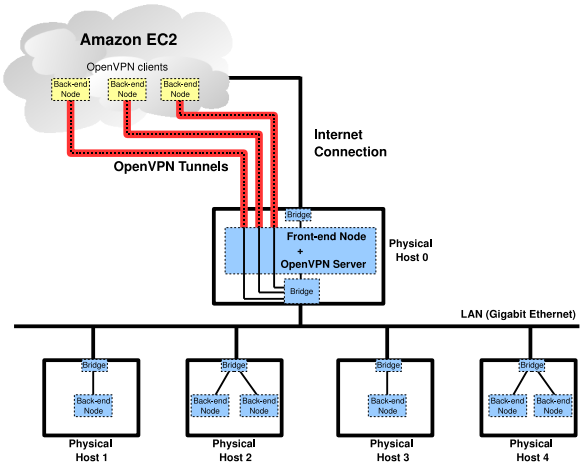

Figure 4: Network infrastructure for the virtual cluster.

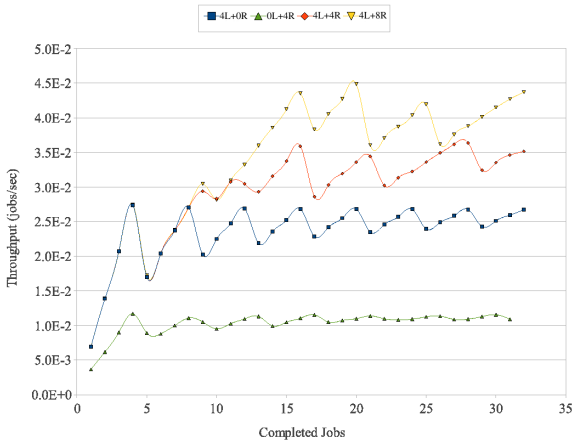

Figure 5: Experimental performance of the ED benchmark for a local cluster, an EC2 cluster, and for a hybrid cluster configuration.

problem sizes (classes A, B, and C), on a physical host and on a virtual machine deployed in the same physical host. As we can observe, the overhead of execution time due to virtualization is, in the worst case, around 15%.

Table 1: Execution times for the ED benchmark on physical and virtualized hosts

	Execution Time (sec.)		Virtualiz.
Benchmark	Phys. Host	Virt. Machine	Overhead
ED Class A	135	144	6.7%
ED Class B	585	637	8.9%
ED Class C	2432	2806	15.4%

Figure 5 shows the dynamic performance (jobs per second or throughput) of three different cluster configurations: (i) all the cluster nodes deployed in the EC2; (ii) all the cluster deployed locally; and (iii) an hybrid configuration with local and EC2 worker nodes. For each configuration, the benchmarking measurements consist of the execution of 32 jobs from the ED family, class A. Throughput is computed as the number of completed job per second over time, as more jobs get completed.

As expected, the EC2 cluster presents a lower performance, about half of the performance of the local cluster. This lower throughput is due to the lower performance profile of the EC2 instance compared to the local nodes. Additionally, these results show a sustained increment in the performance of the cluster with a growing number of EC2 nodes. It is interesting to note that no additional configurations in the cluster nor in the benchmark were introduced to perform these tests.

4.2 Web server cluster performance

Regarding the virtual web server cluster architecture presented in the previous section, it is obvious that communication delays between the front-end and back-end servers will be significantly higher for remote back-end nodes deployed on Amazon EC2, than for local back-end nodes deployed on local resource pool, due to the intrinsic delay of the Internet, and the extra overhead introduced by the OpenVPN tunnel. These extra latencies can have a negative impact on cluster performance, hence it is essential to evaluate if scaling out the web cluster with external cloud resources actually gets a sustained and significant performance improvement. Otherwise, the solution presented would be useless.

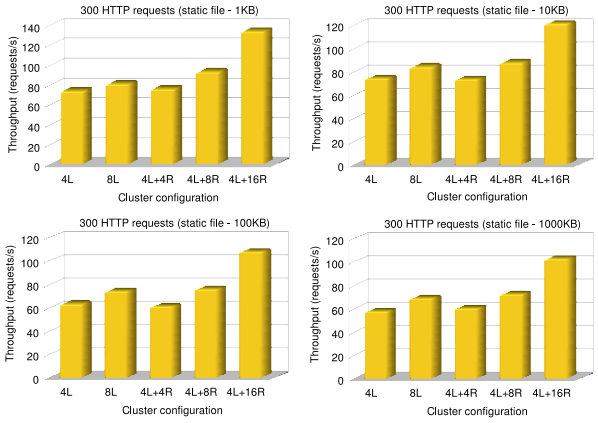

Figure 6: Web cluster throughput for 300 HTTP requests (different file sizes)

For this purpose, we have measured the cluster throughput (number of requests served per second) for different cluster configurations (see figure 6): 4 local nodes (4L), 8 local nodes (8L), and different combinations of local (L) and remote (R) nodes (4L+4R, 4L+8R, and 4L+16R respectively). For this experiment, we have achieved a fixed number of client HTTP requests (300 requests, with a concurrency of 10 simultaneous requests) of static files, sized from 1KB to 1MB.

We can observe that the 8L configurations outperforms the 4L+4R configuration in all cases. This is mainly due to the extra communication latencies of the remote nodes. However, the behaviour of the 8L configuration can be outperformed with four local nodes and eight or more remote

nodes (4L+8R and 4L+16R configurations). The graphics show clearly that, in spite of the extra communication latencies, we can obtain a sustained improvement in cluster throughput by adding an increasing number of remote nodes from the cloud provider to the cluster.

It is important to note that in these experiments we have considered only HTTP requests of static files. In a more general scenario, HTTP clients can request dynamic contents that can involve certain computational load on the back-end web servers. This computational latency affects local and remote nodes equally, so it can result in a better relative performance for those configuration that include remote nodes from the cloud provider.

5. RELATED WORK

Over the last few years, virtualization has been widespread adopted in many organizations and data centers as a solution to implement low cost and elastic server platforms. In particular, virtual cluster implementation is one of the most explored use cases in data center virtualization, specially in the case of computing clusters. Traditionally, these methods consist of overlaying a custom software stack on top of an existing middleware layer, see for example the My-Cluster Project [10] or the Falkon system [11]. These approaches essentially shift the scalability issues from the application to the overlaid software layer, whereas the proposed solution transparently scales both the application and the computational cluster. There are some interesting works that integrate a local resource management system with VMs to provide on a per-job basis a pre-configured execution environment, see for example [12]. A similar approach has been implemented at Grid level using the Globus GridWay Metascheduler [13].

Since Amazon EC2 started to popularize IaaS clouds, several solutions have been developed to create clouds. Some of these solutions, like Globus Nimbus [14] and Eucalyptus [15], fall under the category of cloud toolkits, offering turn-key solutions to transforming existing infrastructure into an IaaS cloud. Eucalyptus is compatible with Amazon's EC2 interface and is designed to support additional client-side interfaces. Globus Nimbus exposes EC2 and WSRF interfaces and offers self-configuring virtual cluster support. However, these tools do not support dynamic allocation and load balancing of computing resources among virtual machines, and the dynamic scaling of virtual clusters using resources from remote cloud providers.

The integration of local physical resources with external resources from a cloud provider to grow specific services is being explored in several projects. Those only use virtual machines for the cloud part of the service as a way to provide service elasticity. For example, the BioTeam [16] has deployed the Univa UD UniCluster Express in an hybrid setup, which combines local physical nodes with virtual nodes deployed in the Amazon EC2. Our approach, which was introduced in [17], additionally includes virtualization in the local site, so providing a flexible and agile management of the whole infrastructure, that may include resources from remote providers.

Finally, regarding to the deployment of web severs using cloud resources, Amazon EC2 has proposed recently the Scarl hosting environment [18], which allows network administrators to create virtual server farms, using prebuilt component, for auto-scaling web sites.

6. CONCLUSIONS

In this work we have analyzed the deployment of two different cluster-based services (a computing cluster and a web server cluster) on top of a virtualized infrastructure, with the capacity of integrating external cloud resources. This flexible approach, which separates the resource provisioning from the service management, provides important benefits: elastic cluster capacity to adapt the cluster to its dynamic workload; cluster partitioning to isolate it from other running services; and support for heterogeneous configurations tailored for each application class.

Performance results show that, in spite of the observed communication overheads, in both cases we can obtain a sustained performance increment when adding a growing number of remote nodes from the cloud provider to the cluster. That proves the feasibility of the proposed architecture and provisioning model, and its capacity to support service elasticity.

7. ACKNOWLEDGMENTS

We would like to thank Javier Fontán and Tino Vázquez for their support to the development of the present work.

8. REFERENCES

[1] Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2.

[2] A. Fox, S. D. Gribble, Y. Chawathe, E. A. Brewer, and P. Gauthier. Cluster-based scalable network services. *SIGOPS Oper. Syst. Rev.*, 31(5):78–91, 1997.

[3] OpenNebula. http://opennebula.org.

[4] B. Sotomayor, R. Montero, I. Llorente, and I. Foster. Capacity Leasing in Cloud Systems using the OpenNebula Engine. In *Workshop on Cloud Computing and its Applications (CCA08).*

[5] Sun Grid Engine. http://gridengine.sunsource.net/.

[6] OpenVPN. http://openvpn.net.

[7] S. Garfinkel. An Evaluation of Amazon's Grid Computing Services: EC2, S3, and SQS. Technical Report TR-08-07, Center for Research on Computation and Society, Hardvard University, 2007.

[8] R. F. Van der Wijngaart and M. A. Frumkin. NAS Grid Benchmarks Version 1.0. Technical Report NAS-02-005, NASA Advanced Supercomputing (NAS), 2002.

[9] M. A. Frumkin and R. F. Van der Wijngaart. NAS Grid Benchmarks: A Tool for Grid Space Exploration. *J. Cluster Computing*, 5(3):247–255, 2002.

[10] E. Walker, J. Gardner, V. Litvin, and E. Turner. Creating personal adaptive clusters for managing scientific jobs in a distributed computing environment. In *Proceedings of the IEEE Challenges of Large Applications in Distributed Environments*, 2006.

[11] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, and M. Wilde. Falkon: a Fast and Light-weight tasK executiON farmework. In *Proceedings of the IEEE/ACM SuperComputing*, 2007.

[12] W. Emeneker, D. Jackson, J. Butikofer, and D. Stanzione. Dynamic Virtual Clustering with Xen and Moab. *Lecture Notes in Computer Science*, 2006.

[13] M. Rodriguez, D. Tapiador, J. Fontan, E. Huedo, R. Montero, and I. Llorente. Dynamic Provisioning of Virtual Clusters for Grid Computing. In *Proceedings of the 3rd Workshop on Virtualization in High-Performance Cluster and Grid Computing (VHPC'08), in conjuction with EuroPar*, 2008.

[14] T. Freeman and K. Keahey. Flying Low: Simple Leases with Workspace Pilot. In *Proceedings of the EuroPar*, 2008.

[15] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus Open-source Cloud-computing System. In *Proc. of Cloud Computing and Its Applications*, 2008.

[16] BioTeam. Howto: Unicluster and Amazon EC2. Technical report, BioTeam Lab Summary, 2008.

[17] I. Llorente, R. Moreno-Vozmediano, and R. Montero. Cloud Computing for on-Demand Grid Resource Provisioning. In *Advances in Parallel Computing, IOS Press (in press)*, 2009.

[18] J. Fronckowiak. Auto-Scaling Web Sites Using Amazon EC2 and Scalr. In *Amazon EC2 Articles and Tutorials*, 2008.