PerfCloud: Performance-Oriented Integration of Cloud and GRID

Valentina Casola[1], Massimiliano Rak[2], and Umberto Villano[3]

[1] Dipartimento di Informatica e Sistemistica,
Università degli studi di Napoli Federico II
casolav@unina.it
[2] Dipartimento di Ingegneria dell'Informazione,
Seconda Università di Napoli
massimiliano.rak@unina2.it
[3] Dipartimento di Ingegneria,
Università del Sannio
villano@unisannio.it

Abstract. Cloud Computing and GRID computing are two different but similar paradigms for managing large sets of distributed computing resources, and there have been many efforts that aim at integrating them. The *cloud on GRID* approach should provide to final users a simple way to manage their resources and to interact with the offered services. This paper proposes the *PerfCloud* architecture, which offers a set of services able not only to create Virtual Clusters (VCs) that become part of the GRID, but also to predict by simulation the performance of user applications. It also presents the `PerfCloudClient`, a user-friendly client with graphical interface to the *PerfCloud* services.

Keywords: Cloud Computing, GRID, Performance.

1 Introduction

Cloud computing, widely known after the success of the EC2 Amazon project [1], is an emerging paradigm, which is steadily spreading in the e-business world. In essence, cloud computing is based on the use of distributed computing resources that are easily allocated, de-allocated, migrated and possibly re-allocated on user request. As such, it relies heavily on the use of virtualization technologies (e.g., [2,3]), able to offer an almost unlimited amount of computing resources. Thanks to virtualization, which controls the access to physical resources in a transparent way, it is possible to offer computational resources that final users can configure as administrators, without any restriction.

On the other hand, GRID computing is basically a paradigm that aims at enabling access to high performance distributed resources using a service-oriented standardized approach. As such, it is widely diffused in the e-science world. In practice, GRID is born with the Globus project, and currently the Globus toolkit [4] and gLite [5] are the most relevant implementations available. In

D.R. Avresky et al. (Eds.): Cloudcomp 2009, LNICST 34, pp. 93–102, 2010.

GRIDs, users can compose complex stateful services in order to build up complex and typically computation-intensive tasks. This is obtained by means of a middleware paradigm: every host has a GRID interface, and developers adopt middleware-dependent APIs for building up their applications.

In fact, cloud and GRID computing paradigms have many points in common: both adopt large datacenters , both offer resources to users, both aim at providing a common environment for distributed resources. The integration of the two environment is a debated issue [6]. At the state of the art, there are two main approaches for their integration:

- **GRID on Cloud:** a cloud IaaS (Infrastructure as a Service) approach is adopted in order to build up and to manage a flexible GRID system [7]. As in this context the GRID middleware runs on a virtual machine, the main drawback of this approach is performance. Virtualization inevitably entails performance losses as compared to the direct use of physical resources.
- **Cloud on GRID:** the stable GRID infrastructure is exploited to build up a cloud environment. This solution is usually preferred [8], because the cloud approach mitigates the complexity of the GRID. In this case, a set of GRID services is offered in order to manage (create, migrate ...) virtual machines. The use of *Globus workspaces* [8], with a set of GRID services for the Globus Toolkit 4 is the prominent solution, as in the Nimbus project [9].

Both approaches have positive aspects but also serious problems for overall system management, as the environments are very complex and managed through thin command-line based clients. In this paper, we essentially propose to use a Cloud on GRID approach, adopting the Virtual Workspaces GRID services to build up a Cluster on Demand (CoD) system. In other words, our system can create Virtual Clusters (VCs) on user request. These VCs are natively provided with support for high performance application development (HPC compilers, MPI, OpenMP, ...). The newly created VCs are directly accessible through the Globus middleware (they contain a preconfigured Globus container) and so they contribute resources to the GRID environment.

PerfCloud, the architecture we are developing [10] and that is the object of this paper, offers a set of services able not only to create VCs on user request, but also to predict by simulation how fast the target application will run on the newly created system. This is an original approach, that can help the user to re-modulate the resources requested for his VC in order to meet his performance expectations. Alternatively, the performance predictions obtained through *PerfCloud* can be used for optimizing the application to be executed in the VC. For simulation purposes, target applications are described in a high-level description language (MetaPL); the performance predictions are obtained by a simulation environment named HeSSE [11].

From the user point of view, the use of *PerfCloud* is very simple. By invoking the GRID service `VCService`, it is possible to create the Virtual Cluster, to obtain an IP address to access it, and to build automatically a configuration file that will be successively used for simulation. An additional service (`BenchService`)

runs a set of predefined benchmarks to characterize the performance of the new VC and measures the timing parameters needed by the simulator. Finally, the `SimulationService` accepts the high-level description of the application, runs the simulations, and returns the predicted response time of the given application on the previously created VC.

When a cloud is created on the top of a GRID, user access to services exploits underlying GRID access services. Moreover, all the security features of the cloud environment are implemented through the GRID infrastructure. Most state-of-the-art GRIDs, being oriented to HPC, offer only simple command line-based interfaces, and are not particularly user-friendly. We have implemented a client for *PerfCloud* that offers a simple interface to the virtualized resources. We will also present here the `PerfCloudClient`, an extensible metaclient component that makes it possible to invoke generic GRID services, together with specific performance-oriented *PerfCloud* services. `PerfCloudClient` is provided with graphical interface and is accessible through a tray icon on the host desktop. A small framework for writing new services makes it possible to define their graphical interfaces and to include them into the metaclient.

The remainder of this paper is structured as follows. In Section 2 we will illustrate the *PerfCloud* architecture. Sections 3 and 4 describe the main components of the architecture that enable the integration of the cloud and GRID environments, whereas Section 5 introduces the client that offers services to manage the infrastructure and provides graphical utilities for end-users. In Section 6 related work is briefly reported. Finally, the conclusions are drawn and our future work is sketched.

2 The PerfCloud Architecture

PerfCloud is a framework that provides performance prediction services in an e-science cloud. The design relies on the adoption of a set of grid services able to create a Virtual Cluster (VC) and to predict the performance of a given target application on that particular VC.

As mentioned in the introduction, *PerfCloud* builds a IaaS (Infrastructure as a Service) cloud environment upon a GRID infrastructure. The *PerfCloud* model of the infrastructure is a collection of clusters, each of which is composed of a front-end node (FE) and a set of computing nodes in a private network. Both the nodes and the network can be physical or virtual.

The clusters managed by *PerfCloud* participate in the underlying GRID and offer their computational resources to the GRID infrastructure. Their FEs host a Globus container and are certified within the GRID Virtual organization. The FEs also host job schedulers (such as PBS or Condor) to distribute the workload on their computing nodes.

Figure 1 describes the overall architecture of *PerfCloud*. The *PerfCloud* application client resides on a user machine (which has access to the GRID environment) and interacts with the *PerfCloud* system through invocation of GRID services. Furthermore, it manages GRID connections, also providing utilities for

end-users as, for example, performance analysis services. The architecture provides different GRID services that enable the user to build up a new cluster as a GRID Virtual Workspace [8] with full access rights. The GRID services of *PerfCloud* also offer other performance evaluation services (simulation, tuning and benchmarking) that can be invoked to simulate and to predict the performance of the environment just built. In order to help user interaction with the clusters, *PerfCloud* offers a tunneling grid service that lets the users execute commands on the target clusters. Moreover, *PerfCloud* offers a set of virtual machine pre-configured images which can be adopted to set-up virtual clusters. The images are ready-to-use cluster configuration enriched with all the software needed to execute HPC applications (compilers, MPI and OpenMP platforms, Globus containers, job schedulers, ...).

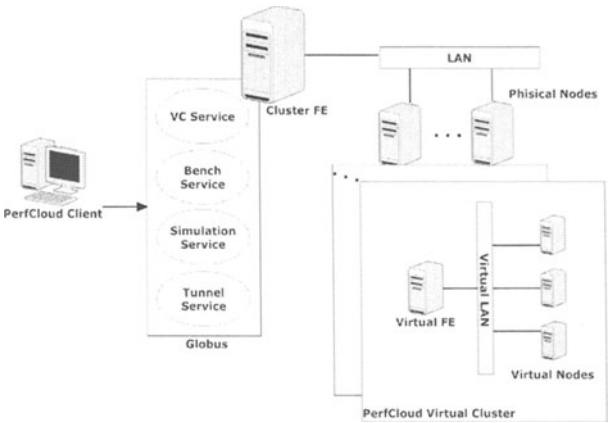


Fig. 1. The *PerfCloud* architecture

In light of the above, the *PerfCloud* architecture can be subdivided into three main components, as is shown in Fig. 1:

Services, which offer the *PerfCloud* functionalities to the GRID environment. The component implementation relies on a minimal set of four GRID services, named `VCService`, `BenchService`, `SimulationService` and `TunnelService`;

Images, which are the Virtual Cluster Node images, containing all the software needed to integrate the VC into the GRID environment (a GT container) and to offer services to the final user (a set of GS deployed on the VC container), along with other software needed for application development and execution (compilers, messaging libraries and run-time support, ...)

Client, which allows the final user to interact with the Cloud environment.

These components will be orderly dealt with in the next sections.

3 PerfCloud Services

The Services component is the core of the *PerfCloud* framework. It offers a minimal set of services, which add (virtual) cluster management capabilities to the GRID environment. The main service (`VCService`), which incorporates all the functionalities needed to manage the virtual clusters, has specific requirements (essentially, the Xen hypervisor) for the physical environment hosting the service. In the following, we will present a brief description of the service functionalities. The details about their implementation can be found in [10].

The `VCService` makes it possible to *design* a VC with the characteristics required by the user (number of virtual nodes, number of virtual CPUs for virtual node, network configuration, ...). It is important to point out that the physical system (usually a cluster) hosting this service has to be able to manage virtual machines, and so, in addition to the Globus workspaces, it requires the presence of the Xen hypervisor. The `VCService` service generates a file description that is used both for cluster creation and, possibly, for its successive simulation. It *creates* a VC, i.e., starts up on the cloud a set of virtual machine images, and allows to perform a performance evaluation of the newly created VC. This entails executing a set of benchmarks and storing their outputs, which are successively used for tuning the simulation model, evaluating the timing parameters typical of the VC created by the user. Since the information needed for building up the virtual cluster and the simulator configuration are similar, we defined an XML cluster description (see [10]). As shown in Figure 1, the VCs that are created and simulated in *PerfCloud* have the same organization of common physical clusters, i.e., they are composed of a Virtual Front-End (FE),which is the only node with a public IP address, and a set of nodes (slave machines) connected to the FE through a network (a private network built by means of Xen bridges).

The `BenchService` runs benchmarks on the virtual clusters and collects the results. The *PerfCloud* system provides a dedicated virtual image, which resides on one of the available physical machines (typically, on a machine not used for VCs). When this service is invoked, it starts up a wrapper Java runtime on the target virtual cluster, which launches the benchmarks. The results are collected on the Virtual FE of the virtual cluster, and successively returned to the service caller which stores the performance figures obtained for future use.

The `SimulationService` offers to the user a simple interface for predicting the performance of his application in a virtualized environment. The simulation package provides two main services: (a) `HeSSEService`, which accepts as input an application to be evaluated and returns the predicted response time; (b) `TuningService`, which tunes the simulator configuration to the target virtual cluster; it retrieves the simulator configuration from the VC resource, queries the DB for the benchmarking results and starts up the automatic tuning procedure to update the simulator configurations.

The `TunnelService` lets the user execute a given command on the target cluster resource, returning the standard output as result. Note that only a GRID-enabled user, i.e., a user owning a valid certificate, is able to execute the

command. This service can be used as a base class for building up services from existing commands.

4 PerfCloud Virtual Clusters

The GRID services provided by *PerfCloud* rely on the availability of a virtualization layer on the physical clusters. In our development environment, all the physical clusters making up the GRID are configured using Rocks, a widely-used cluster distribution based on Red Hat Linux.The latest version of *PerfCloud* was tested on Rocks 5.1 with the Xen roll. The GRID middleware adopted is the Globus Toolkit 4, with the customizations offered by the Rocks GRID roll and a dedicated OpenCA certification authority. To exploit the GRID environment as basis for the cloud system, we adopted the GRID Virtual Workspaces version Tp2.2.

In order to let the above described architecture create virtual clusters, we built a set of scripts able to manage (create, destroy, pause, ...) a set of virtual machine images, pre-configured in order to define a cluster environment. The previously described `VCService` accepts an XML description of the clusters and invokes the scripts in order to setup the Virtual Cluster. The description lets the user choose the virtual cluster configuration and the distribution of virtual nodes on the physical cluster nodes.

The virtual nodes images reside on a cluster FE repository. When an user asks for a new virtual cluster, the images are duplicated and assigned to him. From that moment on, he can fully manage the virtual cluster through the GRID Services and the `PerfCloudClient`.

The virtual clusters created by *PerfCloud* are Red Hat Linux systems, configured with a large set of common HPC tools (gnu compilers, MPI, OpenMP, PBS and Globus). The virtual clusters are configured in order to communicate with each other through a private (virtual) network based on xenbridge. Only the virtual cluster FE has a public IP. The virtual clusters are preconfigured with a Globus container, with a certificate valid for the *PerfCloud* Virtual Organization, and host the `TunnelService`.

5 The PerfCloudClient

The cloud approach aims at offering the services of the GRID infrastructure to a large number of users, not only to the specialized ones, as highlighted by Shantenu Jha et al [6]. These consideration led us to develop a simple graphical interface that makes the interaction with *PerfCloud* very user-friendly.

Nevertheless, the main requirement for such an interface is to be easily extensible, in order to manage the continuous growth of new services, which will be made available to end users. The `PerfCloudClient` is a simple metaclient, presented as a tray icon. It is written in Java and so it is highly portable. The `PerfCloudClient` offers many functionalities to access the GRID infrastructure

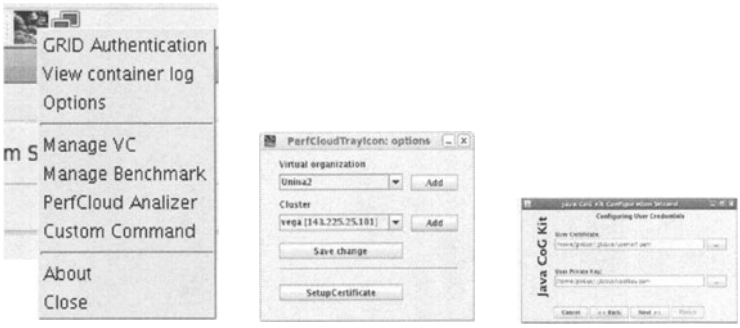
in a secure way (through the generation of a proxy certificate), to manage the connection, as well as further utilities.

According to the above notes, the scenarios are divided into three main use cases: Management of GRID Access and Connections, Management of *PerfCloud* Services and User Utilities.

5.1 Management of GRID Access and Connections

As we build up the Cloud environment on the top of GRID systems, we need to access to the GRID environment; the authentication procedure was developed by adopting the CoG Kit [12] and allows to generate a proxy certificate, as shown in Figure 2(a).

The access to a GRID environment is possible if the the environment has already been initialized and configured. As illustrated in Figure 2, the `PerfCloudClient` offers a setup procedure that enables the user to choose between the different GRID environments (the virtual organizations, top of the screenshot in Figure 2(b)), and possibly to launch the wizard for configuring the credentials (the `SetupCertificate` button and the wizard in Figure 2(c)). When a GRID environment is available, it is possible to choose the cluster to be accessed and to invoke the services.



(a) (b) (c)

Fig. 2. Virtual Organization Setup procedure

It is important to point out that the `PerfCloudClient` is able to manage connections to multiple GRID environments (that offer *PerfCloud* services), and it is able to create virtual clusters on all of them. Moreover, once a VC is created, it appears in the list of the available clusters for the virtual organization in which it has been created.

5.2 Management of PerfCloud Services

The graphical interfaces that manage services are based on a simple template. This is composed of a set of buttons on the top of the window to invoke the services methods, and of a text box which reports the services output.

(a) (b)

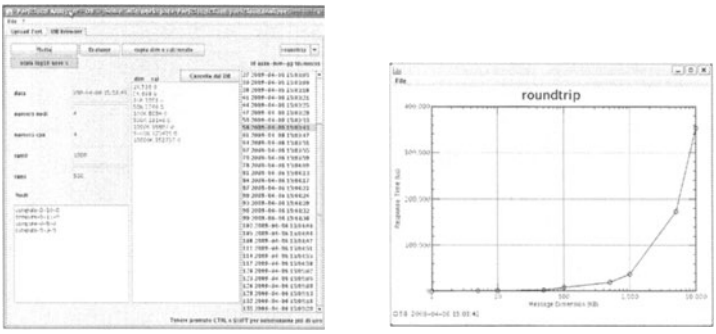
Fig. 3. Management of *PerfCloud* Services

As an example, let us consider the `TunnelService`, which allows a final user to execute a command on the selected cluster resource. Figure 3(a) shows the execution of the `ls -al` command on the target resource. Figure 3(b) shows the results of a `Log Viewer` command that invokes a customized version of the tunnel services and visualizes the log file of the Globus container.

5.3 User Utilities

Finally, the `PerfCloudClient` offers some user utilities that can be executed offline. Useful tools can be the graphical analyzer for performance evaluation, or the graphical tool for the definition of virtual cluster configurations.

At the state of the art, the Performance Analyzer is the only user utility available. It lets the user to build easily up graphical reports of the benchmarks performed on the virtual clusters (see Figure 4).



(a) (b)

Fig. 4. Analyzer User utility

6 Related Work - Cloud Technologies

The cloud paradigm appeared on the computing scene in 2005 with the Amazon Elastic Compute Cloud (EC2) [1]. Then a large set of related technologies has

been developed. In commercial contexts, it is worth mentioning the IBMs Blue Cloud, the Sun Microsystems Network.com, the Microsoft Azure Services Platform, the Google App Engine and the Dell Cloud computing solutions. Most of these commercial systems adopt proprietary solutions (such as the virtualization engine by VMWare), and relatively few details are available on the adopted architectures. In the academic world, and especially in the HPC area, cloud computing is in "competition" with the GRID model, as outlined in [6].

The idea of GRID-Cloud integration and the adoption of virtualization techniques in GRID infrastructure was explored in research projects as Reservoir [13], and in technologies as openNebula [14] and virtual workspaces [8,15], with the derived cloud toolkit Nimbus[9].

At the state of the art, examples of e-science clouds are beginning to emerge [16,9,17,18]. They are based on the above-mentioned technologies and have architectures similar to the one presented in this paper, even if, at the best of the authors' knowledge, none of them provides performance evaluation and prediction tools as services integrated in the architecture.

As regards the user interfaces, both Nimbus [9] and openNebula [14], the prominent solutions for building e-science clouds, offer powerful clients. However, these clients are command line-based and do not provide any graphical interface.

7 Conclusions and Future Work

In this paper we have presented the architecture of *PerfCloud*, which offers cloud-on-GRID functionalities integrated with a simulation environment able to predict user application performance on the newly instantiated Virtual Clusters. The architecture of *PerfCloud* makes use of existing GRID and virtualization technologies to manage at low-level the virtual clusters, and integrates them in the existing GRID, also providing a dedicated set of services able to offer performance prediction functionalities. A client with graphical interface presented as a tray icon on the desktop makes interactions with users more straightforward and user-friendly than in any other existing cloud-GRID integration environment.

The main contribution of our work is undoubtedly the possibility to evaluate on-the-fly the performance of a given application on the particular VC received from the cloud. This is of great importance in the HPC world, where there is skepticism about the adoption of virtualization techniques because of the introduced overheads. Our research aims at making the resulting performance loss *predictable*. However, we think that the use of simple mechanisms to interact with the GRID/cloud is also an added value, as it may contribute to a wider diffusion of clouds in scientific and production environments.

As regards the evolution of our work, we will design services able to build up VCs tailored to the user performance requirements. In other words, the user will provide the application and the requested response time, and the system automatically will build up a suitable cluster. This will make it possible for the cloud to offer guarantees about the quality of service and to negotiate SLAs.

Acknowledgement. We wish to thank Raffaele Lettiero and Angelo Santillo for the technical efforts. This work has been supported by *LC3 -Lab. Pubblico-Privato di ricerca sul tema della comunicazione delle conoscenze culturali-* Nat. Project of MIUR DM1791 and by *Magda una piattaforma ad agenti mobili per il Grid Computing*, L.R. Campania n. 05 28/03/2002.

References

1. Amazon Inc.: Elastic compute cloud (2008), `http://aws.amazon.com/ec2`
2. Barham, P., et al.: Xen and the art of virtualization. SIGOPS Oper. Syst. Rev. 37, 164–177 (2003)
3. WMWare Staff: Virtualization overview (White Paper), `http://www.vmware.com`
4. Foster, I.T.: Globus toolkit version 4: Software for service-oriented systems. J. Comput. Sci. Technol. 21, 513–520 (2006)
5. Laure, E., et al.: Programming the Grid with gLite. Technical Report EGEE-TR-2006-001, CERN, Geneva (2006)
6. Jha, S., Merzky, A., Fox, G.: Using clouds to provide grids with higher-levels of abstraction and explicit support for usage modes. Concurr. Comput.: Pract. Exper. 21, 1087–1108 (2009)
7. Cherkasova, L., Gupta, D., Vahdat, A.: Optimizing grid site manager performance with virtual machines. In: Proc. of the 3rd USENIX Workshop on Real Large Distributed Systems, WORLDS 2006 (2006)
8. Keahey, K., Foster, I.T., Freeman, T., Zhang, X.: Virtual workspaces: Achieving quality of service and quality of life in the grid. Scientific Progr 13, 265–275 (2005)
9. University of Chicago: Nimbus project (2009), `http://workspace.globus.org/clouds/nimbus.html`
10. Mancini, E.P., Rak, M., Villano, U.: PerfCloud: GRID Services for Performance-Oriented Development of Cloud Computing Applications. In: Proc. of WETICE 2009, pp. 201–206. IEEE, Groninger (2009)
11. Mancini, E., Mazzocca, N., Rak, M., Villano, U.: Integrated tools for performance-oriented distributed software development. In: Proc. SERP 2003 Conf., USA, vol. 1, pp. 88–94 (2003)
12. von Laszewski, G., Foster, I.T., Gawor, J., Lane, P.: A java commodity grid kit. Concurrency and Computation: Practice and Experience 13, 645–662 (2001)
13. Reservoir Consortium: Reservoir project (2009), `http://www03.ibm.com/press/us/en/pressrelease/23448.wss`
14. Distributed Systems Architecture Research Group: Opennebula project. Technical report, Universidad Complutense de Madrid (2009), `http://www.opennebula.org`
15. Keahey, K., Foster, I.T., Freeman, T., Zhang, X., Galron, D.: Virtual workspaces in the grid. In: Cunha, J.C., Medeiros, P.D. (eds.) Euro-Par 2005. LNCS, vol. 3648, pp. 421–431. Springer, Heidelberg (2005)
16. Purdue University: Wispy project (2009), `http://www.rcac.purdue.edu/teragrid/resources/#wispy`
17. Masaryk University: Kupa project (2009), `http://meta.cesnet.cz/cms/opencms/en/docs/clouds`
18. Wang, L., Tao, J., Kunze, M., Castellanos, A.C., Kramer, D., Karl, W.: Scientific cloud computing: Early definition and experience (2008)