SOA Based Control Plane for Virtual Clusters

Paolo Anedda[1,2], Simone Manca[1], Massimo Gaggero[1], and Gianluigi Zanetti[1]

[1] CRS4, Center for Advanced Studies,
Research and Development in Sardinia,
Parco Scientifico e Tecnologico,
POLARIS, Edificio 1, 09010 Pula (Ca), Italy
[2] DIEE (Department of Electric and Electronic Engineering),
University of Cagliari,
Italy
{paolo.anedda,simone.manca,
massimo.gaggero,gianluigi.zanetti}@crs4.it

Abstract. Virtualization is an essential enabling technology for the construction and control of computing facilities that can dynamically adapt available physical resources to transient tasks such as the temporary creation of a virtual computing center tailored to the needs of a virtual organization. In this paper we will describe our strategy for the creation of virtual computer clusters based on standard SOA and hosts virtualization technologies and we will report on our ongoing work on the application of the latter to the deployment and management of a research cluster with 140 dual core cpu. Our deployment mechanism, as well as the system management, is delegated to a control plane based on workflows of coordinated web services. The control plane is based on two logically independent modules, the first is responsible of the physical resources and the deployment on the hardware of virtual Xen hypervisor images, while the second manages operations on virtual clusters such as their creation, startup and control. Low level operations – e.g., the control of a running image on a given computational host – are directly provided by atomic web services, in this specific case a WSRF service running in the dom0 of each participating physical Xen host, while all logic above that level is implemented as BPEL scripts.

1 Introduction

Recent advances in virtualization technologies and the exponential growth in network bandwidth are introducing new dimensions for the computational facilities configuration space and opening interesting and effective new ways to deliver High Performance Computing (HPC) resources to applications. In this paper we will report on our ongoing work on experimenting with these technologies in the context of Cybersar, a new computational infrastructure for research being currently built in Italy.

1.1 Virtual Computing Facilities

Virtual machines, such as VMWare [1], Xen [2], and KVM [3], bring to HPC two main gifts. On one hand, they enable the specialization of operating systems to particular

L. Bougé et al. (Eds.): Euro-Par 2007 Workshops, LNCS 4854, pp. 154–163, 2008.
© Springer-Verlag Berlin Heidelberg 2008

tasks with hardware resources safely and transparently multiplexed by the vm hypervisor [4]. On the other hand, virtual machines allow the decoupling of the physical computational infrastructure management from the management of the user-visible virtual computing facility [5,6], making the former essentially homogeneous and agnostic with respect to the specific applicative context, while putting the latter in the hands of the virtual organization managing the virtual resource [7,8]. This separation removes many of the difficulties that oppose the effective sharing of computational resource between organizations that are not adhering to very strict common standards for their software stacks.

Virtualization guarantees a high level of flexibility in the dynamic configuration and use of "atomic" computational resource, which can then be organized and orchestrated to provide virtual clusters that, differently from standard, general purpose, HPC computing facilities, can be tailored to satisfy the needs of specific virtual organizations [9,8].

Virtual clusters can encompass computational resources that are distributed on different, potentially remote sites, current trends on ever increasing available network bandwidth [10] make it possible – at least when the structure of the applications run allows to trade latency for pipeline depth [11] – to tightly couple geographically distant computational resources in a single computational facility that appears as a coherent entity to its users. Such a constructions, however, requires a much closer coordination between computational and network resources with dynamic network configuration mechanisms, similar to the ones described in [12], being activate in parallel to the virtual cluster construction.

1.2 Cybersar

Cybersar is a high performance computing initiative recently (March 2006) funded by the Italian Ministry of Research that has as its goal the development of a Cyberinfrastructure for research in Sardinia based on high speed networks interconnecting all main scientific computational facilities and research communities of the island. Cybersar network core is on dark fibers and it provides an optical network backbone that can support application driven dynamic creation of point to point multi-lambda, currently 1GbE to be soon upgraded to 10GbE per lambda, optical circuits. The optical backbone is linked to the DWDM (Dense Wavelength Division Multiplexing) Regional network (2.5Gbit/s per lambda) and via the latter to the Janna submarine optical cables connecting Sardinia to the Italian Mainland and to Sicily. The main Cybersar computational resources are hosted by physically separated (20 to 70 km apart), computing facilities based at CRS4, the University of Cagliari, Cagliari Astronomic Observatory and the University of Sassari. The total computing power directly available to Cybersar is of about 1200 AMD Opteron class cores with about 2.5TB of RAM and more than 200TB of available disk space. Besides being a source of computing power for the local research community, and to contribute to the National computing infrastructure, Cybersar main goal is to be an experimental platform for research on new approaches to high performance computing.

2 Generalized Computing Control Plane

The dynamic construction of specialized virtual clusters will be the main mechanism that will be used within Cybersar to support large scale applications spanning the resources of multiple sites. This will allow, as it has been discussed above, to insulate the system managers of the partecipating sites from the specifics of operating system and middleware configuration while, at the same time, supporting reasonably fast procedure for application instantiation, freeze and tear down. On the other hand, as we will discuss below, the control plane machinery that we are setting up has, at its most basic layer, general mechanisms to deploy complete systems starting from the bare hardware. Thus, if the overhead to be payed to the virtualization layer is too high (e.g., when one needs to drive specialized hardware such as Infiniband PCI-Xpress boards), we mantain the option of mapping specific applications, similarly to what it is done in Grid5000 [13], to dynamically build real clusters. The logic behind the creation and the management of a virtual clusters, that we can represent as a set of information work-flows of cooperating components, is demanded to the control plane. Making a parallel with the networking field, the latter can be seen as the abstraction layer to which all the logic for the setup of the virtual computing center is entrusted. It is composed by all the programs and system architectures that are required to successfully deploy a new virtual cluster following a cluster blue-print. We consider the control plane as divided in two main components. The first is responsible for the physical allocation of the computational nodes and network resources, while the second is in charge of creating and administering virtual resources. For the practical implementation of the system we choose to adhere to SOA [14] principles and, in particular, to the W3C style of implementation. The entire system is basically build upon web services and is implemented using standard languages and protocols. Following the work of [15], we use Xen as a virtualization layer and web services to control virtual hosts. Since the creation of virtual clusters can be seen as the results of a process of web services calls, all control logic, above low level operations e.g., the control of a running image on a given computational host is implemented as work-flows.

2.1 Physical Resources Management

To be able to deploy complete systems starting from the bare hardware, we have developed a new deployment system, called HaDeS . It meets two main requirements. The first is to be able to communicate with an orchestrator, using standard SOA mechanisms, to properly control the critical steps of the deployment process such as creation of a specific disk partitioning, filesystems, and the population of partitions with the operating systems. The second is to be agnostic with respect to the operating system to be deployed. This is a major requisite for two reasons: we want to be able to support a large number of OS types with different installation methods; we want to be able to deploy a pre-assembled image of an operating system that can't be installed with traditional methods. These are requirements that systems such as RedHat Kickstart, or Suse Yast2, or even Debian FAE cannot meet. While other deployment system, such as SystemImager, are not designed to negotiate with an orchestration system. HaDeS is similar in

spirit to kadeploy [16], albeit with a different implementation, but, differently from the latter, has been thought from the beginning to be integrated in a SOA architecture.

2.2 Virtual Resources Management

As reported in [6], there could be various levels or aspects of virtualization. In our work, when we talk about virtual resources, we refer to the possibilities offered by the so called virtualization software. Following this assumption, we define a virtual host as the abstraction of a real computing facility from which it inherits all the functionalities, running on top of a virtualization software. Going forward in this reasoning, a virtual cluster is composed by a set of cooperating virtual hosts connected by a virtual network.

Virtual Hosts Management. The lifecycle of a virtual host can be represented as a state diagram characterized by six states.

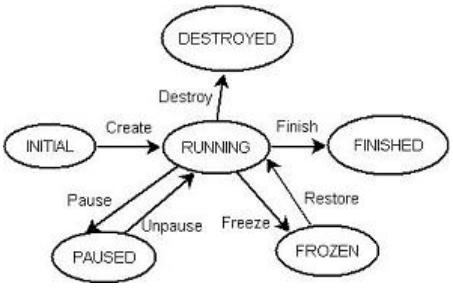

Fig. 1. The lifecycle of a virtual host

As depicted in the figure above, the lifecycle of a virtual host starts from the Initial state in which the virtual host is only a representation of a potential running host. When it is in this state, we call it a "Virtual Host Image" (VHI). A VHI is characterized by an operative system, a configuration and some representation of the initial conditions. We can represent it with a triple in the space of the virtual hosts:

$$(OS_i(t_0), C_i(t_0), Ic_i) \tag{1}$$

where we use OS_i, C_i and Ic_i to indicate, respectively, the operating system, its configuration and the initial state (initial conditions) of the virtual host.

When the resources controller decides to create a new virtual host, it turns a VHI into a running virtual host (VH) by instantiating it on a physical host. This transition is fired by a "create" event. After his creation, a virtual host evolves into the running state. A VH is represented by the same triple of the starting VHI at the time t, plus the reference to the VHI from which it was created:

$$(OS_i(t), C_i(t), Ic_i, Ref(VHI_i)) \tag{2}$$

From this state, a VH can be destroyed, paused or frozen. When it is paused all the VH's operations are suspended, while when it is frozen, the controller serialize it into

a file and release all the resources associated with that particular VH. From this state it can be instantiated again in the same machine or, after a migration procedure, in another one.

The very basic building block of the system is the Virtual Host Controller (VHC). The VHC is the component responsible to instantiate and control the VHs inside a real host. The process of creation of each VH is controlled by the Virtual Host Factory (VHF). The VHF asks the appropriate VHC to instantiate a new VH using a specific VHI. The VHF knows exactly where the VHCs are and how to communicate with them. It can choose a particular VHC using a special algorithm for the resources' optimization, or on a random basis.

All the VHIs are maintained by the Virtual Host Images Repository (VHIR), that stores all information regarding a particular VHI and make them available to all the others components.

The Virtual Host Manager (VHM) is responsible for the orchestration of all the components' activities.

The creation of a Virtual Cluster (VC) is managed by the Virtual Cluster Manager (VCM). All the operations to instantiate a new VC are performed by the Virtual Cluster Factory, which is responsible to dialog with the VHM for the VHs creation. Its responsibilities include also the managing of all information about a particular VC that is running.

2.3 Virtual Resources Implementation

Virtual Host Controller. As we mentioned before, the very basic building block of the entire architecture is the VHC. This is the component that physically controls the creation and the management of the virtual machines running onto the physical hosts, through a common interface that wraps the command line calls to the Xend hypervisor control program. Each physical node is equipped with a WSRF (Web Services Resources Framework, [17]) container which contains an implementation of the interface. This allows to export, through a standard web interface, the main functionality of Xen to the rest of the system.

Virtual Host Images repository. The VHIR is the component responsible to manage the information regarding the "potential" virtual hosts. It is the repository where all the information about all the VHIs are stored.

A VHI is the description of the components of a VH in his initial state. It is composed by an image of an operative system, an initial configuration and the initial conditions.

The initial configuration is the set of information regarding the network, the device drivers and the storage resources.

The initial conditions are a representation of the state of the VH at the moment of his first running. With this, we mean a representation of the virtual memory and the filesystem associated with it. That's because a running VH can be frozen and his image, with all the virtual memory and the filesystem, can be stored to be used as a new image for the instantiation of a new VH.

When the VHIR is asked to give the information about a specific VHI, it knows how to retrieve the physical image of the operative system and creates a reference to it. It also has a description about the configuration of the network and the storage resources.

These information are usefull to create the connections among the VHs and are set at the moment of the creation of the VHI.

Virtual Host Factory. The VHF is the component responsible for the instantiation of a new VH. It receives the request, that indicates the type of the node to create and the resources associated to it, from the VHM. Then the VHF asks the VHIR for that particular image and evaluates the answer. If the request can be fulfilled, a VHI reference is sent from the VHIR to the VHF. At this point, because the VHF has a database with all the available VHCs and knows the state of every node, it asks a particular VHC to create a new VH passing the VHI reference to it. At this point the VHC creates a new VH and, if all goes well, starts it and returns the VH's reference to the VHF. The VHF updates his internal database with all the information regarding the new node and returns the new VH's pointer to the VHM.

Virtual Cluster Factory. The VCF is the component that is responsible for the creation of virtual clusters. When it receives a request from the VCM, it dialogs with the VHM for the creation of the VHs. For every VH specified in the VCM's request, it talks to the VHM.

It maintains an internal database of all the VCs available and of all the VHs belonging to each VC.

Work-flow control. The logic behind the creation and the management of virtual machines and its related network resources, can be expressed in terms of work-flows of business processes. A work-flow can be represented using a standard language. According to our philosophy to follow the SOA requirements, we choose the BPEL (Business Process Execution Language, [18]) language.

BPEL defines business processes that interact with external entities through Web Service operations defined using WSDL 1.1 (Web Service Definition Language, [19]). It defines business processes using an XML based language but do not define a graphical representation of processes or provide any particular design methodology for processes. Each business process has inputs, method and outputs and so, also the workflow exports a web-service interface. Workflows can be nested, so it is possible to call a workflow from inside another one. This allows us to define very complex execution processes.

Once is defined, a workflow is executed by a BPEL engine which is responsible for the instantiation of all the components defined and to coordinate their execution. The execution of a specific workflow is started by the invocation of a method defined by his interface.

The creation of VH can be represented as a sequence of coordinated actions performed by the components defined in the section above. Since the VHM is responsible for these operations, we decided to implement it as a workfow.

The process of creation of a new VH starts with the invocation of the create VH method of the VHM. This envent triggers the execution of the process depicted in figure 2. The VHM calls the VF with a request for a new VH. The VH asks the VHIR for the appropriate VI, according to the inputs from the VHM. The VHIR then returns a reference to the VI that meets the request's specifications. At this point, the VHF calls a VHC for the VH creation, passing the VI reference to it. The choice of a particular

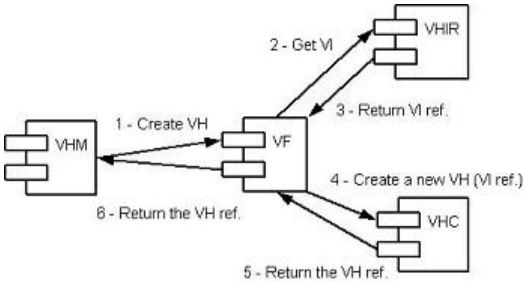

Fig. 2. The VH creation process

VHC, is made according to a specific workload policy. Once a new VH is created, the VHC returns a VH reference to the VF which sends it back to the VHM.

As we did for the VHM, we also implemented the VCM as a workflow. In this case the process of creation of a new VC, is represented by the VCM. When the VCM receives the request for a new VC, it invokes the create VC method on the VCF. The VCF then calls the VHM for the creation of a new VH as many times as the number of nodes defined in the VC creation's request. Once all the VHs are created, the VCF stores all the information regarding the VHs and returns to the VCM a reference to the new VC created.

Virtual Resources Definition. Following the approach of [15] and [9], to describe a VC we use an XML schema that defines all the details and the attributes of the corresponding computational resource associated to it. The XML describes the virtual cluster as a whole, in terms of general properties common to all the nodes; it also specifies the details of each node in terms of their parameters. So, a virtual cluster is defined by declaring a list of nodes. For each node, the name and the node type are specified. Each node can also contain the details of his implementation or a reference to a specific node configuration.

The schema definition has been splitted into two separate files. The former contains the description of the whole cluster while the latter the description of a single node. This was done to separate the level of details between the cluster definition and the nodes description. During the creation of a new cluster, the control plane, which is responsible for the new cluster deployment, processes the XML file and, for each node, it sends the corresponding XML fragment to a virtual host controller. The virtual host controller doesn't need to know about the whole cluster details; he only has to deal with the parameters necessary for the creation of a new virtual node.

A virtual node configuration contains the details of the disks partition, the network settings and also the boot parameters like the number of virtual processors or the amount of memory associated with the new host.

3 Preliminary Testbed

Following the approach described above, we develop the whole system using a preliminary experimental environment composed by three servers connected to the same

ethernet switch. Each server was equipped with two 2.2 GHz AMD Opteron CPUs, 2GB main memory, and two 200GB hard drives.

One server was used for the implementation of the control plane. We choose the ActiveBPEL server[20] as workflows manager. ActiveBPEL is an open source implementation of a BPEL engine, that follows the J2EE ([21]) specifications, entirely written in Java. We deployed it onto the Tomcat container ([22]).

The other two servers were equipped with the Xen software. The VHs are created through the call to a local web-service running inside a WSRF container. The implementation of the web services was written in Python, using the pyGridWare toolkit [23].

To create all the scripts for the activation of the container and his associated services, we used a template mechanism ([24]) entirely written in python that, starting from a definition of the methods written in XML, is able to automatically generate all the stub classes and the code that are necessary for the execution of the service's methods.

4 Target Testbed

The actual target testbed for the control plane software is currently being delivered at CRS4 site, and it is composed by 72 IBM System x3455 machines. Each machine contains 2 dual-core AMD Opteron processors of the latest "revision F" series (2218) operating at 2.6 GHz, with 64 bit extension and 1 MB of L2 cache per core. The amount of RAM is of 8 GB per machine, while the storage consists of two 200 GB SATA disks, managed by the Broadcom SATA/Raid controller. Two Broadcom 5704 Gigabit Ethernet allow each host to connect to the LAN while their IPMI integrated controller allows remote monitoring and remote management.

Twenty-four machines of the 72 are connected, through an Infiniband Interface, to a Cisco Infiniband Fabric Switch. All nodes have two free PCI Express slots that can be used for further addition of new peripherals.

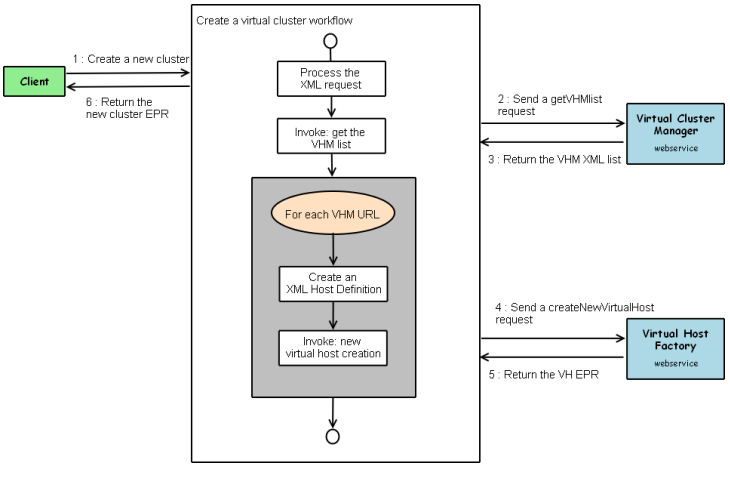

Fig. 3. The workflow of a new virtual cluster

5 Conclusions

We have described our strategy for the creation of virtual computer clusters based on standard SOA and hosts virtualization technologies and our ongoing work. Our deployment mechanism, as well as the system management, is delegated to a control plane based on workflows of coordinated web services. The control plane is based on two logically independent modules, the first is responsible of the physical resources and the deployment on the hardware of virtual Xen hypervisor images, while the second manages operations on virtual clusters such as their creation, startup and control. Low level operations – e.g., the control of a running image on a given computational host – are directly provided by atomic web services, in this specific case a WSRF service running in the dom0 of each participating physical Xen host, while all logic above that level is implemented as BPEL scripts.

Acknowledgments. This research is partially supported by the Italian Ministry of University and Research, project PON-Cybersar 2006.

References

1. Warren, S.S.: The VMWare Workstation 5 Handbook. Charles River Media, Hingham, MA, USA (2005)
2. Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Pratt, I., Warfield, A., Barham, P., Neugebauer, R.: Xen and the art of virtualization. In: Proceedings of the ACM Symposium on Operating Systems Principles (October 2003)
3. Qumranet: Kernel based virtual machine., `http://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine`
4. Mergen, M.F., Uhlig, V., Krieger, O., Xenidis, J.: Virtualization for high-performance computing. SIGOPS Oper. Syst. Rev. 40(2), 8–11 (2006)
5. 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)
6. 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 Gener. Comput. Syst. 21(6), 896–909 (2005)
7. Figueiredo, R.J., Dinda, P.A., Fortes, J.A.B.: A case for grid computing on virtual machines. In: ICDCS 2003: Proceedings of the 23rd International Conference on Distributed Computing Systems, Washington, DC, USA, p. 550. IEEE Computer Society Press, Los Alamitos (2003)
8. Ramakrishnan, L., Irwin, D., Grit, L., Yumerefendi, A., Iamnitchi, A., Chase, J.: Grid allocation and reservation—toward a doctrine of containment: grid hosting with adaptive resource control. In: Löwe, W., Südholt, M. (eds.) SC 2006. LNCS, vol. 4089, p. 101. Springer, Heidelberg (2006)
9. Foster, I.T., Freeman, T., Keahey, K., Scheftner, D., Sotomayer, B., Zhang, X.: Virtual clusters for grid communities. In: CCGRID, pp. 513–520. IEEE Computer Society, Los Alamitos (2006)
10. Foster, I., Grossman, R.L.: Data integration in a bandwidth-rich world. Commun. ACM 46(11), 50–57 (2003)

11. Smarr, L., Chien, A.A., DeFanti, T.A., Leigh, J., Papadopoulos, P.M.: The optIPuter. Commun. ACM 46(11), 58–67 (2003)
12. Lehman, T., Sobieski, J., Jabbari, B.: Dragon: A framework for service provisioning in heterogenous grid networks. IEEE Communications Magazine 44(3) (2006)
13. Cappello, F., Caron, E., Dayde, M., Desprez, F., Jeannot, E., Jegou, Y., Lanteri, S., Leduc, J., Melab, N., Mornet, G., Namyst, R., Primet, P., Richard, O.: Grid 5000: a large scale, reconfigurable, controlable and monitorable Grid platform. In: Grid 2005 Workshop, Seattle, USA, November 13-14, 2005, IEEE/ACM (2005)
14. Hégaret, P.L.: Web services and soa., `http://www.w3.org/2003/Talks/1211-xml2003-wssoa`
15. Zhang, X., Keahey, K., Foster, I., Freeman, T.: Virtual cluster workspaces for grid applications. In: Cluster Computing and the Grid, 2006. CCGRID 2006. Sixth IEEE International Symposium (2006)
16. Kadeploy team: Kadeploy, `http://kadeploy.imag.fr/`
17. Foster, I., Czajkowski, K., Ferguson, D., Frey, J., Graham, S., Maguire, T., Snelling, D., Tuecke, S.: Modeling and managing state in distributed systems: The role of OGSI and WSRF. Proceedings of the IEEE 93, 604–612 (2005)
18. BPELSource: Bpelsource, `http://www.bpelsource.com/`
19. World Wide Web Consortium: Web services description language (wsdl) 1.1., `http://www.w3.org/TR/wsdl`
20. ActiveBPEL: Activebpel., `http://www.activebpel.org/`
21. Sun: Java platform, enterprise edition., `http://java.sun.com/javaee/`
22. Apache: Apache tomcat., `http://tomcat.apache.org/`
23. Boverhof, J.: pygridware: Python web services resource framework
24. Rudd, T.: Cheetah - the python-powered template engine