SCHEDULING IN VIRTUAL INFRASTRUCTURE FOR HIGH-THROUGHPUT COMPUTING

Chalapathi Valupula[1], R. Lakshman Naik[2], Sunitha Muppa[3]

[1]UPC-Barcelona Tech, Barcelona, Spain
[2]Dept. of CSE, BITS, Warangal, AP, India
3Dept. of CSE, JITS, Warangal, AP, India
lakshman432@gmail.com

ABSTRACT

For the execution of the scientific applications, different methods have been proposed to dynamically provide execution environments for such applications that hide the complexity of underlying distributed and heterogeneous infrastructures. Recently virtualization has emerged as a promising technology to provide such environments. Virtualization is a technology that abstracts away the details of physical hardware and provides virtualized resources for high-level scientific applications. Virtualization offers a cost-effective and flexible way to use and manage computing resources. Such an abstraction is appealing in Grid computing and Cloud computing for better matching jobs (applications) to computational resources. This work applies the virtualization concept to the Condor dynamic resource management system by using Condor Virtual Universe to harvest the existing virtual computing resources to their maximum utility. It allows existing computing resources to be dynamically provisioned at run-time by users based on application requirements instead of statically at design-time thereby lay the basis for efficient use of the available resources, thus providing way for the efficient use of the available resources.

KEYWORDS

Scientific Applications, Virtualization, Grid computing, Cloud computing.

1. INTRODUCTION

A common goal of computer systems is to minimize cost while maximizing other criteria, such as performance, reliability, and scalability, to achieve the objectives of the user(s). The introduction of highly distributed computing paradigms, such as Grid Computing and Cloud Computing, has brought unprecedented computational power to a wide area of the scientific community of the world [1] [2]. An important research effort has been devoted to effectively deliver these raw processing resources to the scientific applications. The characteristics of the distributed environment, such as its heterogeneity or dynamism, hinder the efficient use of the infrastructure for the scientific community of the world.

In Grid computing and Cloud Computing , a scalable way to harness large amounts of computing power across various organizations is to amass several relatively inexpensive computing resources together. However, a growing heterogeneity hinders the development of large scale Grid and Cloud infrastructures. These resources do not only differ in their hardware but also in their software configurations (operating systems, libraries, and applications) [17]. This heterogeneity increases the cost and length of the application development cycle, as they have to be tested in a great variety of environments where the developers have limited configuration capabilities. Therefore, some of the users are only able to use a small fraction of the existing resources.

Coordinating these distributed and heterogeneous computing resources for the purposes of perhaps several users can be difficult. In such an environment, resource users have several varying, specific, and demanding requirements and preferences for how they would like their applications and services to leverage the resources made available by resource providers. Resource providers must ensure the resources meet a certain quality of service (e.g. make resources securely and consistently available to several concurrent users).

In the past, control over the availability, quantity, and software configurations of resources has been limited to the resource provider. Virtualization has emerged as a promising technology to tackle the previous problems by decoupling the services from the physical hardware [2]. With virtualization, it becomes possible for resource providers to offer up more control of the resources to a user without sacrificing quality of service to other resource users. Users (resource consumers) can more easily create execution environments that meet the needs of their applications and jobs within the policies defined by the resource providers. Such a relationship, enabled by virtualization, is both cost-effective and flexible for the resource producer and user. Moreover, the introduction of a new virtualization layer between the computational environments and the physical infrastructure makes it possible to adjust the capacity allocated easily [3] [9].
Virtual machines add a new abstraction layer that allows partitioning and isolating the physical hardware resources. They also offer a natural way to face a highly heterogeneous environment. At the same time induce an overhead which has to be minimized for harvesting the maximum advantage of the virtualization technology [4] [5]. The main problem with the Virtualization technology is the amount of the overhead induced by the virtual machines, Virtualization induces nearly sixty percent overhead on data intensive jobs so it is very important to reduce the overhead as much as possible. Thus scheduling the virtual infrastructure in an efficient way to reduce the cost induced by the virtualization overhead and achieve efficient performance from the available resource is very important. Different distributed computing environments can form a large virtual computing system offering a variety of resources [7] [9]. The users of this grid can be organized dynamically into a number of virtual organizations, each with different policy requirements. These virtual organizations can share their resources collectively as a larger grid for executing their jobs (e.g. High-Throughput Computing Jobs).

Some problems in the field of scientific research require weeks or months for solving them the people in such fields' require large amount of computing resources over a long period of time. Such an environment is called a High-Throughput Computing (HTC) environment. High-Throughput Computing (HTC) [14] environments deliver limited resources over a long period of time. Whereas High-Performance Computing delivers large amount of resources in a short span of time HPC environments are often measured in terms of Floating point Operations per Second (FLOPS). The best example for a HTC environment is Condor system and for a HPC is a Supercomputer. Moreover the HTC environment is cost effective which can be established fairly at a minimal cost comparably with HPC which require large amounts of investment which the low level research organizations cannot afford a HTC environment can be realized by using the existing infrastructure in the organization by utilizing the available computing resources during the off office hour for the execution of scientific applications [16].

Condor is one such system which takes the wasted computation time and puts it to good use. Condor collects all such unutilized resources and matches them with the jobs. We choose Condor for the implementation of the project. Condor, "is a specialized workload management system for compute-intensive jobs" [11].Condor currently abstracts the resources of a single physical machine into virtual machines which can run multiple jobs at the same time [12]. A "universe" is used to statically describe the execution environment in which the jobs are expected to run. This approach assumes the resources (whether real or virtual) have to all be allocated in advance. While there is support for adding more resources to an existing pool via the Glide-in mechanism, the user still has to dedicate the use of these other physical resources [16].

The purpose of this paper is to describe how a Condor execution environment (universe) can be dynamically created at run-time by users to more flexibly and cost-effectively use and manage existing resources using virtualization. The implementation details of the work performed for a Condor virtual universe are provided along with performance tests results. Future enhancements are included for making this work-in-progress more robust.

While virtualization has a number of applications for business computing and software development and testing, the work outlined in this paper most directly applies to technical computing, including Grid computing, clusters, and resource-scavenging systems [10].

This work we propose a new concept of Pre-Staging to harvest the Life Cycle of Virtual Machine and scheduling the Virtual Machine in such a way to extract the maximum performance from it. The work consists of five sections, from here the second section describes about the proposed solution to solve the problem. The third section describes the Implementation part, fourth section speaks about the experimentation set and results obtained, section five is about the conclusions and the future work which can be done to improve the proposed work.

2. PROPOSED ARCHITECTURE.

In order to achieve our objective we develop a system pool using two systems where Condor job scheduler runs and we also install Virtual Machine Monitor (VMM) or Hypervisor which is supported by the Condor on both the systems to execute our jobs directly using Condor Virtual Machine Universe also by our proposed[8][9]. Pre-Staging Model. Once the system is established we first prepare our Virtual Machine Images and also choose the jobs to run on them, the Virtual Machine Images should have the following characteristics.

2.1. Image Compatibility

The Virtual Machine image must be in a format usable by the hypervisor software in use at the execute machine.

2.2. Architecture Compatibility

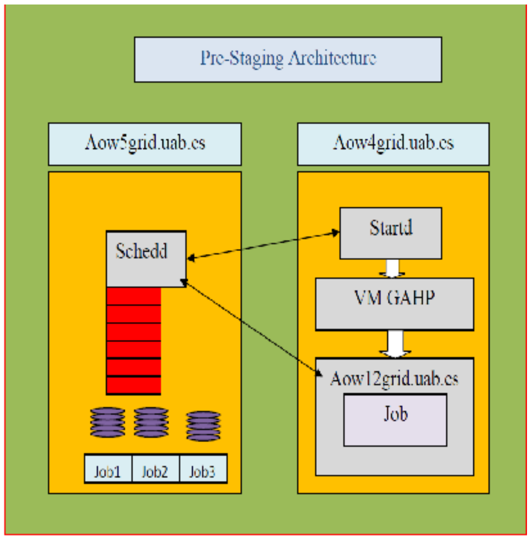
Operating system running in Virtual Machine must be compatible with the system architecture exposed by the hypervisor [10].

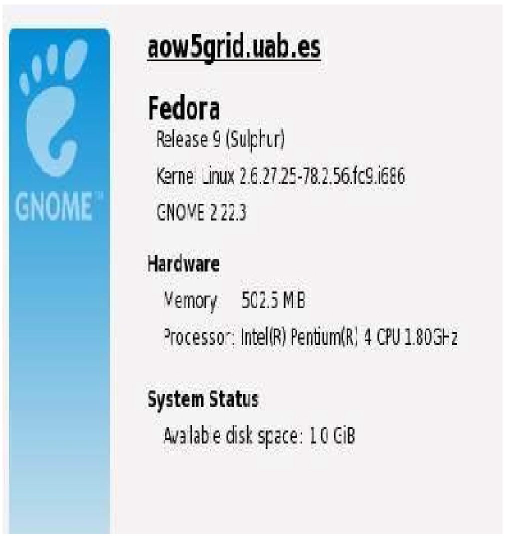
2.3. Dynamic Reconfigurability

The guest system inside the VM must be able to have certain properties, such as its MAC address, IP address, hostname, and Condor job scheduler set at boot time.

2.4. Pre-Staging Model.

We submit the Condor VM (**aow12grid.uab.es)** job from submit machine (**aow5grid.uab.es)** to the execute machine (**aow5grid.uab.es)** the virtual machine boots on the execute machine and joins the condor pool. Once the virtual machine is on the pool we submit jobs from the submit machine to the virtual machine directly.





aow5grid.uab.es

Fedora
Release 9 (Sulphur)
Kernel Linux 2.6.27.25-78.2.56.fc9.i686
GNOME 2.22.3

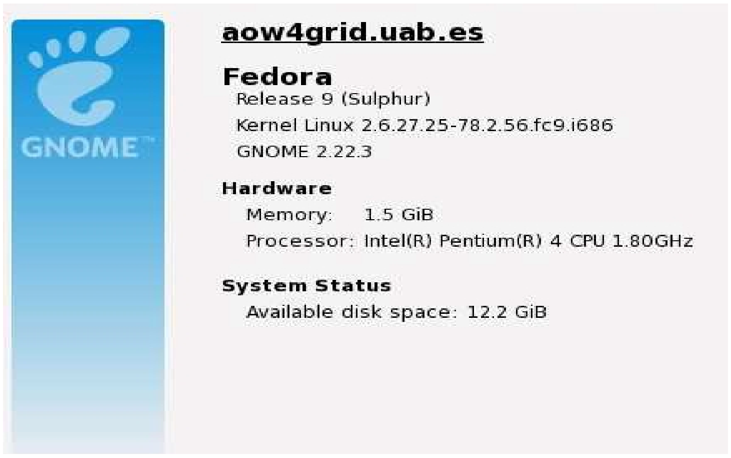
Hardware
Memory: 502.5 MiB
Processor: Intel(R) Pentium(R) 4 CPU 1.80GHz

System Status
Available disk space: 1.0 GiB

Fig.1: Pre-Staging Architectural Design. Fig.2: Submit Machine Configuration.

For designing the architecture we use the following systems for hardware purpose we use aow5grid.uab.es and **aow4grid.uab.es** with the following hardware configurations.
This system is used as a Manager as well as a job submitter and is used to submit our jobs to the condor pool.

Aow4grid.uab.es



aow4grid.uab.es

Fedora
Release 9 (Sulphur)
Kernel Linux 2.6.27.25-78.2.56.fc9.i686
GNOME 2.22.3

Hardware
Memory: 1.5 GiB
Processor: Intel(R) Pentium(R) 4 CPU 1.80GHz

System Status
Available disk space: 12.2 GiB

Fig.3: Execute Machine Configuration.

This system is used as a execute machine, where we run our jobs.
Now that we have the required hardware design we choose the following software for our design.

We choose Condor High-Throughput Computing framework for implementations of our design, as it is open source software which can be downloaded and implemented free of cost and more ever the Condor Team provides an excellent support to the problems that we encounter.

As we have seen earlier the virtualization software that is available today like Xen, KVM, Vmware etc [9]. We choose the Vmware Server version 1.0.1 [8] which is also free software which can be just downloaded and can be installed. More ever the Condor system also supports very well for Vmware Server.

3. IMPLEMENTATION

We implement our architecture based on the Condor Virtual Machine Universe where a submit machine submits the jobs and the execute machine executes the jobs submitted by the submit machine and returns the output back to the submitter.

With the aim of achieving the initial Objective of the project, we implement the architecture of the system in the following way.

We choose two computers aow5grid.uab and aow4.uab.es and establish a Condor pool by installing Condor version 7.4.4 on the systems, aow5grid.uab.es is configured as Condor Manager as well as a submit machine for our project. We also install the Vmware Server on this machine to create virtual machine mages which we are going to use to run our virtual machine jobs.

The second system aow4grid.uab.es is configured as a execute machine where we run our virtual machine jobs, this machine is configured in such a way that it supports the Condor virtual machine applications using the Condor_config file. Here too we install Vmware Server so as to support Condor Virtual Universe.

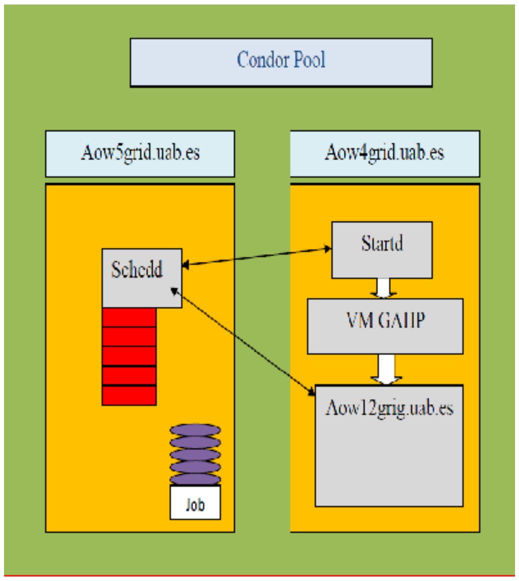
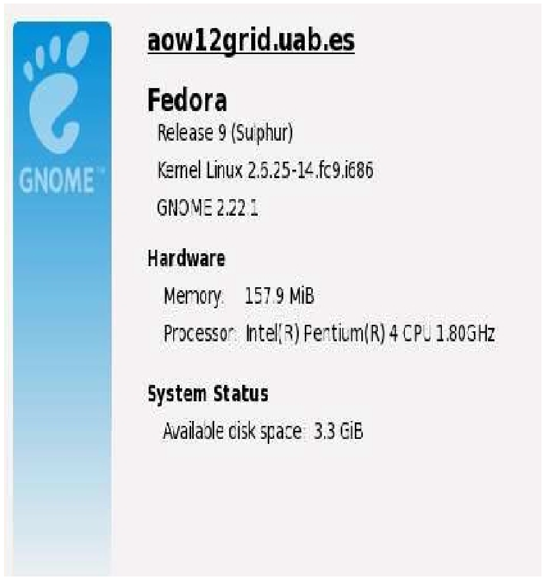


Fig.4: Pre-Staging Implementation.



aow12grid.uab.es

Fedora
Release 9 (Sulphur)
Kernel Linux 2.6.25-14.fc9.i686
GNOME 2.22.1

Hardware
Memory: 157.9 MiB
Processor: Intel(R) Pentium(R) 4 CPU 1.80GHz

System Status
Available disk space: 3.3 GiB

Fig.5: Virtual Machine Image Configuration.

3.1. Virtual Machine Image

The virtual machine image used in this project is a Fedora 9 with 157.9 MB of memory and 8 GB VMdisk with the Condor 7.4.4 installed with an independent IP address to that of the host, We install Condor 7.4.4 on the VM image and also pre-configure it using the custom ClassAd attributes in a machine ad via the config file once the VM image is ready.

Table 1: Experiment summary

Experimentation Set: 1	**Experimentation Set: 2**
As we have discussed Pre-Staging Model in the previous section with the following configurations. **Submit Machine** Aow5grid.uab.es OS Fedora 9. Memory 502.5 MiB Processor: Intel(R) Pentium(R) 4 CPU 1.8GHz Condor version-7.4.4 Vmware server 1.0.1 **Execute Machine** Aow5grid.uab.es OS Fedora 9. Memory 1.5 GiB. Processor: Intel(R) Pentium(R) 4 CPU 1.8GHz Condor version-7.4.4 Vmware server 1.0.1 **Virtual Machine Image** Aow12grid.uab.es OS Fedora 9. Memory: 157.9 MiB. Processor: Intel(R) Pentium(R) 4 CPU 1.8GHz Condor version-7.4.4 **Jobs** We are going to run three types of jobs the first job is a C-language application, the second is a Java application and the last is a MPI application.	In the second experiment we change the Virtual Machine Image with Ubuntu10.10 with the following configurations. **Submit Machine** Aow5grid.uab.es OS Fedora 9. Memory 502.5 MiB Processor: Intel(R) Pentium(R) 4 CPU 1.8GHz Condor version-7.4.4 Vmware server 1.0.1 **Execute Machine** Aow5grid.uab.es OS Fedora 9. Memory 1.5 GiB. Processor: Intel(R) Pentium(R) 4 CPU 1.8 GHz Condor version-7.4.4 Vmware server 1.0.1 **Virtual Machine Image** Aow12grid.uab.es Ubuntu10.10. Memory:264 MiB. Processor: Intel(R) Pentium(R) 4 CPU 1.8GHz Condor version-7.4.4 **Jobs** We are going to run three types of jobs the first job is a C-language application, the second is a Java application and the last is a MPI application.

With the above mentioned Configuration we first start a Condor VM job form the submit machine which boots on the execute machine and joins the condor pool. Now we have two execute machines on the pool one is Aow4grid.uab.es and virtual machine Aow12grid.uab.es which we have submitted as a VM job now it is ready to accept our jobs.

4. RESULTS

We run the experiment for 15 times by using the first experimentation set for all the three types of jobs on the Condor Virtual Universe and also our Pre-Staging model and note the execution times for each type of job for all the 15 executions and make an average of them.

The table below shows the average execution times for three types of jobs on both the models.

Table 2: shows average execution times by using Pre-Staging and Condor VU methods.

Application	**Pre-Staging Method**	**Condor VM Method**
C-Language Application	42s	880s
Java Application	57s	972s
MPI Application	112s	1117s

The average execution times for the second experimentation set for all the three applications on Pre-Staging Model and Condor Virtual Machine Universe are demonstrated by the following table.

Table 3: shows average execution times by using Pre-Staging and Condor VU methods.

Application	**Pre-Staging Method**	**Condor VM Method**
C-Language Application	47s	837s
Java Application	68s	937s
MPI Application	132s	1289s

5. CONCLUSIONS AND FUTURE WORK

5.1. Conclusions

We can clearly observe that by Pre-Staging Model the performance increases more than 15 times for C-language Applications and is around 13 times for the Java Applications. But when it comes to the MPI Applications the performance improvement is reduced to approximately 10 times this is due to the overhead induced by the virtualization on High-performance Applications.

By using the Condor Virtual Machine Universe to execute jobs involving large Virtual Machine Images is tedious and complicated. Executing the jobs involving large Virtual Machine Images consumes much time and more ever the Virtual Machine cannot be reutilized. In Condor Virtual Machine Universe the job is to be placed as a boot script which is a bit complicated to general users, who are not familiar to programming. By using the new Pre-Staging method the VM can be re utilized to execute multiple jobs. The job consumes less time to be completed, at the same time the multiple jobs can be executed successively and is very easy to execute jobs by Pre-Staging Model to people who are not familiar to programming.

4.1. Future Work

The future work consists of implementing the Pre-Staging Model on a large scale to form Virtual Organization Clusters, which we were unable to do due to time and resource constrains. Finding a way to place the Virtual Machine on the Condor pool without using independent IP address because when the Pre-Staging Model is implemented on a cluster basis it requires a IP address for each of the Virtual Machine Images which is a bit difficult and also reducing the complexity regarding the custom MachineAds which are a bit difficult for the people to execute jobs using this method who are not familiar Condor. This method has to be tested by the other Virtualization software like Xen KVM Virtual Box and also using other types of applications which were left out due to time constrains.

REFERENCES

[1] Secure Virtualization and Multicore Platforms State-of-the-Art report By Heradon Douglas and Christian Gehrmann.

[2] James E. Smith and Ravi Nair. The Architecture of Virtual Machines. IEEE Computer, 38(5):32-38, 2005.

[3] Kirk L. Kroeker. The Evolution of Virtualization. Communications of the ACM, 52(3):18--20, 2009.

[4] L4Ka.org.L4Kapre-virtualizationpage. http://l4ka.org/projects/virtualization/afterburn/.

[5] VMWare. VMWare ESX and ESXi product page. http://www.vmware.com/products/esx/index.html

[6] Steven J. Vaughan-Nichols. New Approach to Virtualization is Lightweight. IEEE Computer, 39(11):12-14, 2006.

[7] Andrew Whitaker and Marianne Shaw and Steven D. Gribble. Denali: Lightweight Virtual Machines for Distributed and Networked Applications. Proceedings of the USENIX Annual Technical Conference, 2002.

[8] VMWare. Transparent Previrtualization info page.
http://www.vmware.com/interfaces/paravirtualization.html.

[9] Virtualization: State of the Art Version 1.0, April 3, 2008 Copyright © 2008 SCOPE Alliance.

[10] Management of Virtual Machines on Globus Grids Using GridWay A.J. Rubio-Montero, E. Huedo, R.S. Montero2 and I.M. Llorente

[11] Condor manual 7.4.4 pdf http://www.cs.wisc.edu/condor/.

[12] Extending a Desktop Computing Grid with Cloud Resources Purdue University and Cray, Inc.

[13] Condor custom MachineAds https://nmi.cs.wisc.edu/node/1469

[14] An Introduction to the Condor HTC Framework S.Hosking Parallel Programming, 159735, Massey University May 2009

[15] CondorVirtualJobs http://citi.clemson.edu/files/condor/cidays/CIDaysVM.htm.

[16] Exploring Virtual Workspace Concepts in a Dynamic Universe for Condor Quinn Lewis 2006

[17] http://www.cs.wisc.edu/condor/manual/v6.1/2_3Condor_Matchmaking.html#fig:CondorStatus.

[18] IBM Red book Introduction to Grid Computing 2005

[19] Above the Clouds: A Berkeley View of Cloud Computing http://radlab.cs.berkeley.edu/ February 10, 2009.

[20] Cloud computing: state-of-the-art and research challenges Qi Zhang • Lu Cheng • Raouf Boutaba.

Authors

Chalapathi Valupula received his M.S. in Computational Science and Engineering from Universitat Autònoma de Barcelona, Spain.and currently pursuing Ph.D UPC-BarcelonaTech, Barcelona Spain. He also served as a Technical Consultant ABAP/4 (SAP) at Stratesys Consulting (Barcelona) and Technical Jr. Consultant ABAP/4 (SAP) at PROsap (Madrid). His current research interests are in the fields of Cloud Computing, Virtualization and Ubiquitous Computing.

R. Lakshman Naik received his B.Tech. in Electronics and Communication Engineering from SIET, JNTU, Krishna, A.P, India and M.Tech. in Computer Science and Engineering from BITS, JNTU, Warangal, A.P, India. He also served as a Systems Engineer in Wipro Technologies. He has Publications more than 9 papers in area of computer networks, Data mining and neural networks.

Sunitha Muppa received her B.Tech. in Computer Science and Engineering and M.Tech. in Software Engineering from Jayamukhi Institute of Technological Sciences (JNTU) , Warangal, A.P, India. She has more than 2 years of experience in teaching. Her current research interests are in the fields of Cloud Computing,