Universitat Autònoma de Barcelona

Departament d'Arquitectura de Computadors i Sistemes Operatius

Màster en Ciència i Enginyeria Computacional

Scheduling in Virtual Infrastructure

Memoria del Trabajo de ´´ Iniciación a la Investigación. Trabajo de Fin de Máster ´´ del ¨Máster en Ciencia e Ingeniería Computacional ¨, Realizada por Chalapathi Valupula bajo la dirección de Elisa Heymann. Presentada en la Escuela de Ingeniería (Departamento de Arquitectura de Computadores y Sistemas Operativos).

2011

Iniciación a la Investigación.

Trabajo de fin de Máster

Máster en Ciencia e Ingeniería Computacional

Curso 2009-11

Scheduling in Virtual Infrastructure

Realizada por Chalapathi Valupula en la Escuela de Ingeniería en el Departamento de Arquitectura de Computadores y Sistemas Operativos. Universidad Autónoma de Barcelona.

Dirigida por: Elisa Heymann.

Firmado:

Directora	Autor
Elisa Heymann	Chalapathi Valupula

Acknowledgements

Foremost, I would like to thank my supervisor, Prof. Elisa Heymann, who shared with me a lot of her expertise and research insight. She quickly became for me the role model of a successful researcher in the field. I also like to express my gratitude to Arindam Choudary, whose thoughtful advice often served to give me a sense of direction during my Project.

I am indebted to my Professorsand student colleagues of Departamento de Arquitectura de Computadores y Sistemas Operativos. Universidad Autónoma de Barcelona, for providing a stimulating and fun environment in which to learn and grow.

I am grateful to the secretaries of Escuela de Ingeniería, Universidad Autónoma de Barcelona for all the help they extended me.

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.

Resumen

En la ejecución de aplicaciones científicas, existen diversas propuestas cuyo objetivo es proporcionar entornos adecuados de ejecución que oculten la complejidad de las infraestructuras distribuidas y heterogéneas subyacentes a las aplicaciones.

Recientemente, la virtualización ha emergido como una prometedora tecnología que permite abstraer los detalles del hardware, mediante la asignación de recursos virtualizados a las aplicaciones científicas de altas necesidades de cómputo. La virtualización ofrece una solución rentable y además permite una gestión flexible de recursos. Este nivel de abstracción es deseable en entornos de Grid Computing y Cloud Computing para obtener una planificación adecuada de tarea (aplicaciones) sobre los recursos computacionales. Este trabajo aplica el concepto de virtualización al sistema gestor dinámico de recursos Condor, mediante la utilización de Condor Virtual Universe para conseguir una máxima utilización de los recursos computacionales virtuales. Además, permite que los recursos de cómputo existentes sean proporcionados dinámicamente en tiempo de ejecución por los usuarios, en función de los requisitos de la aplicación, en lugar de mantener la definición estática definida en tiempo de diseño, y así sentar las bases del uso eficiente de los recursos disponibles.

Palabras Clave: Virtualización de Aplicaciones Científicas, Grid Computing, Cloud Computing.

Resum

En l'execució d'aplicacions científiques, existeixen diverses propostes amb l'objectiu de proporcionar entorns adequats d'execució que amaguin la complexitat de les infraestructures distribuïdes i heterogènies subjacents a les aplicacions.

Recentment, la virtualització ha sorgit com una prometedora tecnologia que ha de permetre abstraure els detalls del hardware, mitjançant l'assignació de recursos virtualitzats a les aplicacions científiques amb altes necessitats de còmput. La virtualizatzació ofereix una solució rentable i a més permet una gestió flexible de recursos. Aquest nivell d'abstracció es desitjable en entorns de Grid Computing i Cloud Computing per a obtenir una planificació adequada del treball (aplicacions) sobre els recursos computacionals. Aquest treball aplica el concepte de virtualització al sistema gestor dinàmic de recursos Condor, mitjançant la utilització de Condor Virtual Universe per aconseguir una màxima utilització dels recursos computacionals virtuals. A més, permet que els recursos de còmput existents siguin proporcionats dinàmicament en temps d'execució pels usuaris, en funció dels requisits de l'aplicació, en lloc de mantenir la definició estàtica definida en temps de disseny, i així assentar unes bases per l'ús eficient dels recursos disponibles.

Paraules Clau: Virtualització d'Aplicacions Científiques, Grid Computing, Cloud Computing.

Table of Contents

Chapter 1 Introduction

1.1 Introduction of Problem

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. 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). 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. With virtualization, it becomes possible for resource providers to offer up more control of the resources to a user without sacrificing quality of serviceto 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.

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.

The main problem with the Virtualization technology is the amount of the overhead induced by the virtual machines, every time a job is submitted the virtual machine boots and once the job is completed the virtual machine goes down.

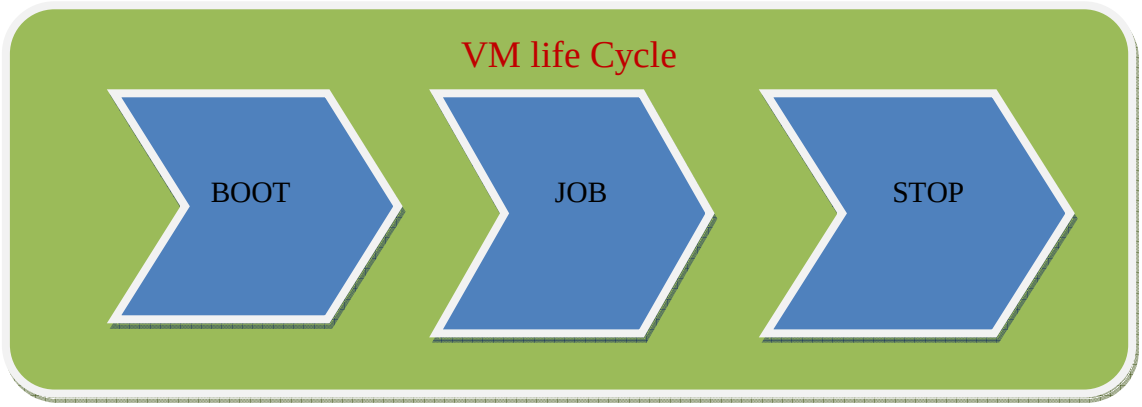

Fig 1.1Virtual Machine Life Cycle.

In order to reduce the overhead induced by virtualization, we propose to use the same virtual machine for performing number of jobs instead of booting the virtual machine every time a job is submitted.

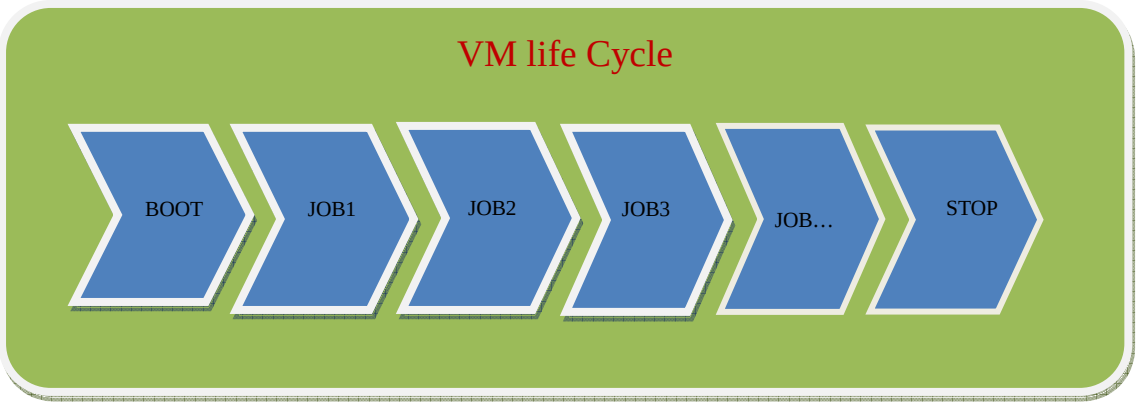

Fig 1.2 Proposed Pre-Staging Model.

1.2 Objective

The main objective of the Project is to harvest the life cycle of the virtual machine and use the same virtual machine for executing multiple successive jobs.

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.

Virtualization inducesnearly sixty percent overhead on data intensive jobs so it is very important to reduce the overhead as much as possible by re-using the same virtual machine this overhead is considerably reduced as there is no need to transfer the virtual machine image every time a job is submitted. More ever the boot and shutdown times of the Virtual Machine are also saved. As we can use the same virtual machine the time to boot the virtual machine is also saved. Thus reduce the cost of job execution to a large extant.

1.3 Structure of Memory

This work presents 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 six chapters, from here the second chapter describes that State of Art in the field, the third chapter describes about the proposed solution to solve the problem.

The fourth chapter describes the Implementation part, fifth chapter speaks about the experimentation set and results obtained, chapter sixth is about the conclusions and the future work which can be done to improve the proposed work and at last is about the references utilized for the work.

Chapter 2 State of Art

2.1 What is virtualization?

The term virtualization broadly describes the separation of a resource or request for a service from the underlying physical delivery of that service.Virtualization is a computer system abstraction, in which a layer of virtualization logic manages and provides virtualized resources to a client layer above it. The client accesses resources, the virtualization layer manages the real resources and possibly multiplexes them among more than one clientVirtualization technologies provide a way to separate the physical hardware and software (OS and applications) by emulating hardware using software.

The virtualization layer resides at a higher privilege level than the clients, and can interpose between the clients and the hardware. Today, virtualization can apply to a range of system layers, including hardware-level virtualization, operating system level virtualization, and high-level language virtual machines Virtualization, the use of hypervisors or virtual machine monitors to support multiple virtual machines on a single real machine, is quickly becoming more and more popular today due to its benefits of increased hardware utilization and system management flexibility, and because of increasing hardware and software support for virtualization in commodity platforms.Virtualization not only makes sense economically, but also environmentally. Recycling, biking and other green trends will definitely make an impact, but not quite as much as an impact if your IT virtualizes its infrastructure. Green IT is growing in importance and virtualization is easiest way to get your company on board with the movement [1][46].

For these reasons, virtualization technology has become more prominent, and is expanded at a rapid pace into various real world systems. Today virtualization is used in enterprise systems, service providers, home desktops, mobile devices, and production systems, among other systems [1].

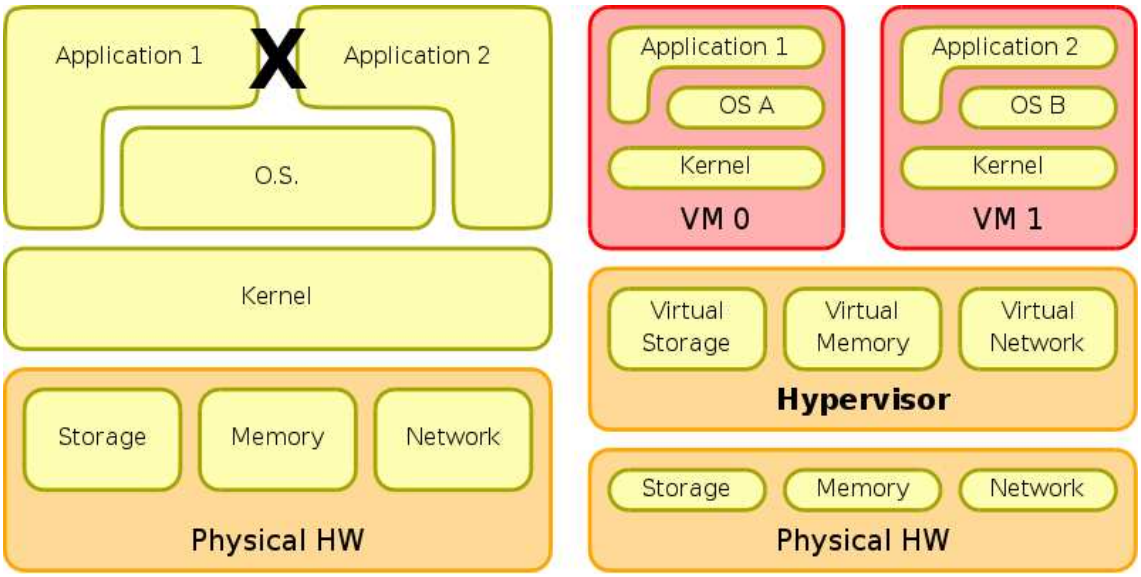

Fig 2.1 Virtualization Overview [27].

2.1.2. Basic Virtualization concepts

2.1.3. Interfaces

Interfaces offer different levels of abstraction which guest use to access resources. Virtualization technology presents an expected interface which accesses the physical resources behind the scenes, for example to access a virtual disk is just accessing a file on the physical disk [2][3].

2.1.4. Instruction Set Architecture (ISA)

In a typical system, ISA is related to programming, including the native data types, instructions, registers, addressing modes, memory architecture, interrupt and exception handlers, and external I/O. An ISA includes a specification of the set of op-codes (machine language), and native commands, implemented by a particular CPU design.

The ISA is the lowest level instruction interface that communicates directly with hardware.Some of the ISA can be used directly by applications, but another part of the ISA is only available to the higher-privileged operating system. If unprivileged software attempts to use a restricted portion of the ISA, the instruction will trap to the privileged operating system.

2.1.5. Device drivers

Device driver is an interface allowing operating system to interact with a hardware device. When a calling program invokes a routine in the driver, the driver issues commands to the device. Once the device sends data back to the driver, the driver may invoke routines in the original calling program. Device driver often reside in the operating system kernel and run at high privilege, but as they are not always written with ideal security or robustness, yet times they a dominant source of operating system errors [4].

2.1.6. Application Binary Interface (ABI)

A low-level interface between an application program and the operating-system, between an application and its libraries, or between component parts of the application. The ABI typically consists of system calls. Through system calls, applications can obtain access to system resources mediated by the operating system. An ABI differs from an application programming interface (API) in that an API defines the interface between source code and libraries, so that the same source code will compile on any system supporting that API, whereas an ABI allows compiled object code to function without any changes to a system using a compatible ABI.

2.1.7. Application Programming Interface (API)

API defines the interface between sourcecode and libraries, so that the same source code will compile on any system supporting that API, This abstraction can facilitate a common interface for applications not only across different hardware platforms, but also across different operating systems, APIs can be built on top of other APIs, making it at least possible that only the lower-level APIs will have to be re-implemented to be used on a new operating system. all software is executed through the ISA in the end meaning that any API or application will have to be recompiled, even if it doesn't have to be re-implemented, as it moves to a new platform [2][19].

2.1.8. Types of virtualization

The most important basic types of virtualization are process virtualization and system virtualization. The other concepts of virtualization are binary translation, paravirtualization, and previrtualization, as well as containers, a more lightweight relative of system virtualization. These concepts illustrate basic types of virtualization currently in use [2][19].

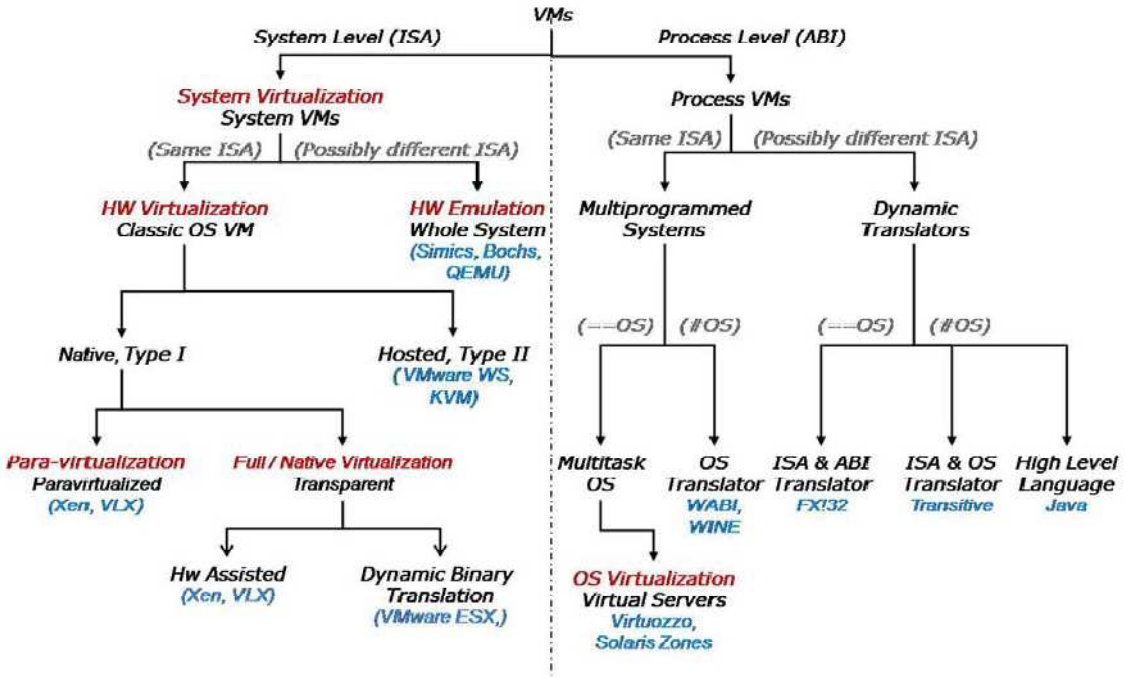

Fig 2.2 Virtualization Taxonomy [19].

2.1.9. Process virtualization

Process-level virtualization is a basic concept in virtually every modern computer system. An operating system virtualizes the memory addressspace, CPU, CPU registers, and all other system resources for each runningprocess. Each process interacts with the operating system using its own virtual interface, unaware ofthe activities of other processes running on the system.

The operating systemhas a scheduling algorithm to ensure that every process gets a fair share of CPU time thereby each process thinks that it has the sole access to the CPU. Thus the operating system implements the concept of process virtualization.

2.1.10. System virtualization

System virtualization is a kind of virtualization that enables several virtual machines to run over a software virtualization layer. System virtualization can involve the same ISA (hardware virtualization) or a different ISA (hardware emulation). System virtualization has the goal of logically replicating physical resources. Each logical replica is named a virtual machine. The virtualization layer can be implemented in different ways. It might run on the bare hardware (Xen hypervisor), or it might require a host operating system (Vmware hypervisor).

In system virtualization an entire system is virtualized, enabling multiple virtual systems to run isolated alongside each other. A hypervisoror Virtual Machine Monitor (VMM) virtualizes all the resources of a physical machine, creating a virtual environment known as a Virtual Machine (VM). Software running in the virtual machine behaves as if it is running in a real machine, and has access to all the resources of a real machine through a virtualized ISA.

The Virtual Machine Monitor (VMM) manages the real resources, and provides them to the virtual machines. The Virtual Machine Monitor (VMM) may support one or more virtual machines.

Virtual Machine Monitor (VMM) will divide the system resources in different ways. For example, if there are two CPU cores and two virtual machines are running on the system, it may allocate one specific core to specific VMin a fixed manner or it may assign and unassigns the two cores to the two VMs asneeded by them. The same goes for memory usage.

Virtualization of this standard type has been around for long from 60’s, and is increasing quickly inpopularity today, thanks to the flexibility and cost-saving benefits it offers to individuals and organizations, as well as due to commodity hardware support. Today it isexpanding from its traditional ground the of data center and into newer areas such as security and Mobile/embedded computing application fields [2][19].

2.1.11. ISA translation

If the guest and virtualization host utilize the same ISA, then no ISA translation is necessary.Clearly, running the host and guest with the same ISA and thus not requiring translation issimpler, and better for performance. In case the guest uses adifferent ISA than the host. In such cases, the host must translate the guest's ISA. Both processand system virtualization layers can translate the ISA.

In some systems where the hardware is not virtualization friendly Binary translation is used, the Virtual Machine Monitor can translate unsafe instructions from a Virtual Machine into safeinstructions.

2.1.12. Paravirtualization

Paravirtualization allows multiple operating systems to run on hardware at the same time by making more efficient use of system resources, such as processors and memory, through effective resource sharing. Unlike full virtualization where a whole system is emulated (BIOS, disk, processor, NIC, etc.), the paravirtualization's management module (a hypervisor or virtual machine monitor) operates with an operating system that has been modified to work in a virtual machine[46][16].

The Denali system uses paravirtualization in support of a lightweight, multi-VM environment suited for networked application servers [17].

The abstraction created with paravirtualization generally means operating systems will perform better than a full virtualization model where all elements must be emulated. However, this efficiency does come at the cost of flexibility and security. Flexibility is lost because the operating system must be modified to run with the paravirtualization, which means that a particular OS or distribution may not be readily available for the solution. But in cases of well-maintained, open software (such as Linux) paravirtualized software distributions may be conveniently available.

Paravirtualization can also serve in situations where underlying hardwares not supportive of virtualization. Besides better performance, paravirtualization's efficiencies can also lead to better scaling and also faster than other forms of virtualization.

2.1.13. Pre-virtualization

Pre-virtualizationattempts to bring the benefits of both binary translation and paravirtualization. Pre-virtualization is achieved via an intermediary between the guest operating system code and the Virtual Machine Monitor.

Pre-virtualization aims to decouple the authoring of guest operating system from the usage of a Virtual Machine Monitor platform, and thereby retains the security and performance enhancements of paravirtualization, Pre-virtualization is a newer technique which has been introduced recently and needs to be proved [1].

2.1.14. Containers

Containers are an approach to virtualization that runs above a standard operating system like Fedora Linux but provides a complete, lightweight, isolated virtual environment for collections of processes [16].An example is the OpenVZ project for Linux [8].

Applications running in the containers must run natively on the underlying operating system, this type of virtualizationdo not promote heterogeneous environments. Containers can pose a less resource intensive path to system isolation than traditional virtualization.

Container virtualization system is not a minimal trusted hypervisor, but instead running as a part of operating system, hence any security problems in the container system architecture and the isolation mechanisms must be studied.

2.1.15. Non-standard systems

Exokernel [9] are one of the other non-standard virtualization systems. Exokernels take a totally different approach instead of trying to abstract and replicate a system with higher and higher level interfaces, exokernels provide high access to resources and allow applications to work out the details of resource safety and management for themselves. This yields much more control and power to the application developer, but is more difficult and dangerous to deal with.

2.1.16. Hypervisors

A hypervisor is thin software layer in charge of logically replicating the physical system. Each logical replica is called a virtual machine .Within each virtual machine, a guest operating systemruns independently of other guest operating system running in other virtual machines.Facilitating the use of the virtual machine as a system abstraction as illustrated in Figure 5

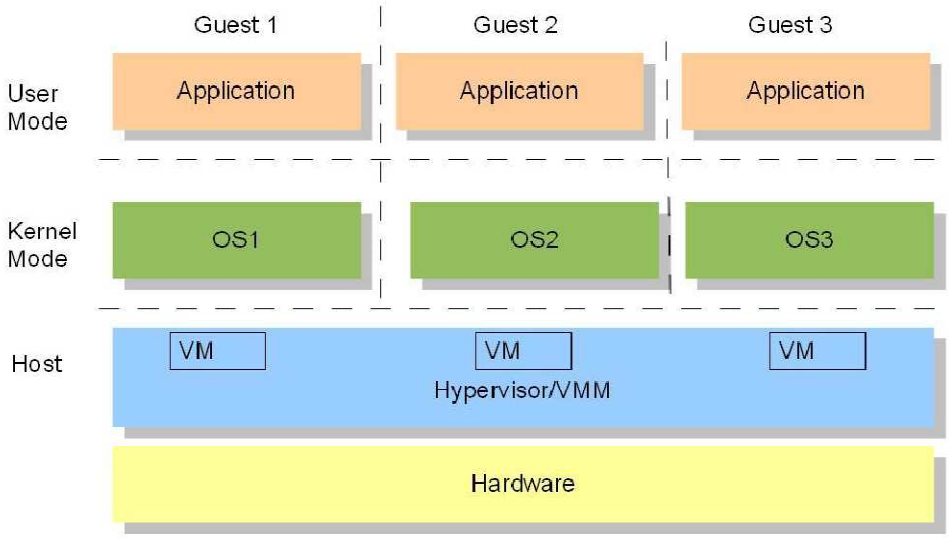

Fig 2.3 Typical Hypervisor [1].

2.1.17. Bare-Metal hypervisors

A Bare-Metal hypervisor is also traditionally referred to as a Type I hypervisor, it is a hypervisor that runs directly on the hardware and that hosts guest operating systems. A type 1 hypervisor is a pure hypervisor. It runs on its own, not inside a host operating system and provides services to the virtual environments. Vmware ESX Server is the best known example. The Wind River hypervisor is also a type I.

Hypervisors, such as Xen [10] and Vmware ESX [11], run on the bare metal and support multiple virtual machines. This is the classic type of hypervisor, dating back to the 1970s, when they commonly ran on mainframes. A Bare-metal hypervisor must provide device drivers and any other components or services necessary to support a complete virtual system and ISA for its virtual machines. For this to function hypervisors

may use paravirtualization, as Xen does, or binary translation, as Vmware ESX does, or a combination of both.

The Xen hypervisor originally required paravirtualization, but can now support full virtualizationif the system offers modern virtualization hardware support. Xen alsodeals with device drivers in an interesting way. Instead of having all the device drivers includedin the hypervisor itself, it instead uses the device drivers running in the operating system found in the specialhigh-privilege Xen administrative domain.

2.1.18. Hosted hypervisors

Hosted hypervisor is also called type II, some hosted hypervisors such as Virtual Box [20] or VMware Workstation, VMware Player [21], runs on the top of a standard operating system and supports multiple virtual machines. The hypervisor runs as a user application, and therefore so do all the virtual machines. Performance is preserved by having as many VM instructions as possible run natively on the processor. Privileged instructions issued by the VMs must be caught and virtualized by the hypervisor, so that VMs don't interfere with each other or with the host. One of the main advantage of the hosted approach is that existing device drivers and other services in the host operating system can be used by the hypervisor and virtualized for its virtual machines reducing hypervisor size and complexity [2].Hosted hypervisors also support useful networking configurations (such as bridged networking, where each VM can in effect obtain its own IP address and thereby network with each other and the host), as well as sharing of resources with the host (such as shared disks). Type II hypervisors provide an easy way for desktop users to take advantage of virtualization.

2.1.19. Advantages of System Virtualization

The main advantage of virtualization is usuallycost saving. For many companies, the largest benefit of server virtualization, which allows multiple operating systems to be installed on a single server, is in reducing the amount of hardware that is required to run all the software needed by the business. Consolidating servers using a virtualization process not only provides savings in terms of how many physical machines must be bought

and maintained, but also potentially reduces the amount of physical space that a company needs for its servers. Apart from this there are many more advantages which are explained below.

2.1.20. Isolation

The main advantage of virtualization is isolation between the virtual machines enforced by the hypervisor. Each virtual machine behaves as if it is a complete physical system and is isolated from other virtual machines that are running on the hypervisor. This leads to robustness and security.

2.1.21. Minimized trusted computing base

A user application mainly depends on all the software running beneath it. A compromise in any software beneath it on can compromise the application itself. Where software often runs with administrative privileges, a compromise of any piece of software can result in total machine compromise and leads to compromise other software running on the machine.

Virtualization addresses this problem by using a trustworthy hypervisor. Software can be partitioned into virtual machines that are trusted and untrusted, and a compromise of an untrusted virtual machine will have no effect on a trusted virtual machine, since each machine functions as an independent identity.

2.1.22. Architectural flexibility

Virtualization provides a great deal of architectural flexibility the virtual machines can be combined on a single platform arbitrarily to meet particular needs.

2.1.23. Simplified development

Virtualization can lead to simplified software development and easier porting. Use of virtual machines enables rapid deployment by isolating the application in a known and controlled environment. Unknown factors such as mixed libraries caused by numerous installs can be eliminated. Severe crashes that required hours of reinstallation now take moments by simply copying a virtual image.

2.1.24. Security

Virtualization of systems helps prevent system crashes due to memory corruption caused by software like device drivers. As the virtual machines are isolated so if a particular virtual machine crashes it does not influence the entire system thus providing greater reliability, security, and availability.

A hypervisor has great visibility into and control over its virtual machines, yet is isolated from them, and thus forms an apt base for security services of many and varied persuasions. An interesting aspect of virtualization-based security architecture is that it can bring security services to unmodified guest systems, including commodity platforms.

By using virtualization in the creation of secure systems, designers can reap not only the bounty of isolated virtual machines, but also provide a good amount of security to the system.

2.2.What Grid Computing is?

Grid computing has been around for over years now and its advantages are many. Grid computing can be defined in many ways but for these discussions let's simply call it a way to execute compute jobs across a distributed set of resources instead of one central resourceGrid computing can mean different things to different individuals. The grandvision is often presented as an analogy to power grids where users get access to electricity through wall sockets with no care orconsideration for where or how the electricity is actually generated. In this view ofgrid computing, computing becomes pervasive and individual users gain access to computing resources as needed with little or no knowledge of where thoseresources are located or what the underlying technologies, hardware, operatingsystem, and so on are , it just like you connect to the system and start reaping the benefits of the system.

This vision of grid computing can capture one’s imagination and mayindeed someday become a reality there are manyissues that need to be addressed.

Grid computing could be defined as a virtualizationalong a continuum. Where along that continuum we can say that aparticular solution is an implementation of Grid computing versus a relativelysimple implementation using virtual resources. But even atthe simplest levels of

virtualization, one could say that grid-enabling technologiesare being utilized [31].

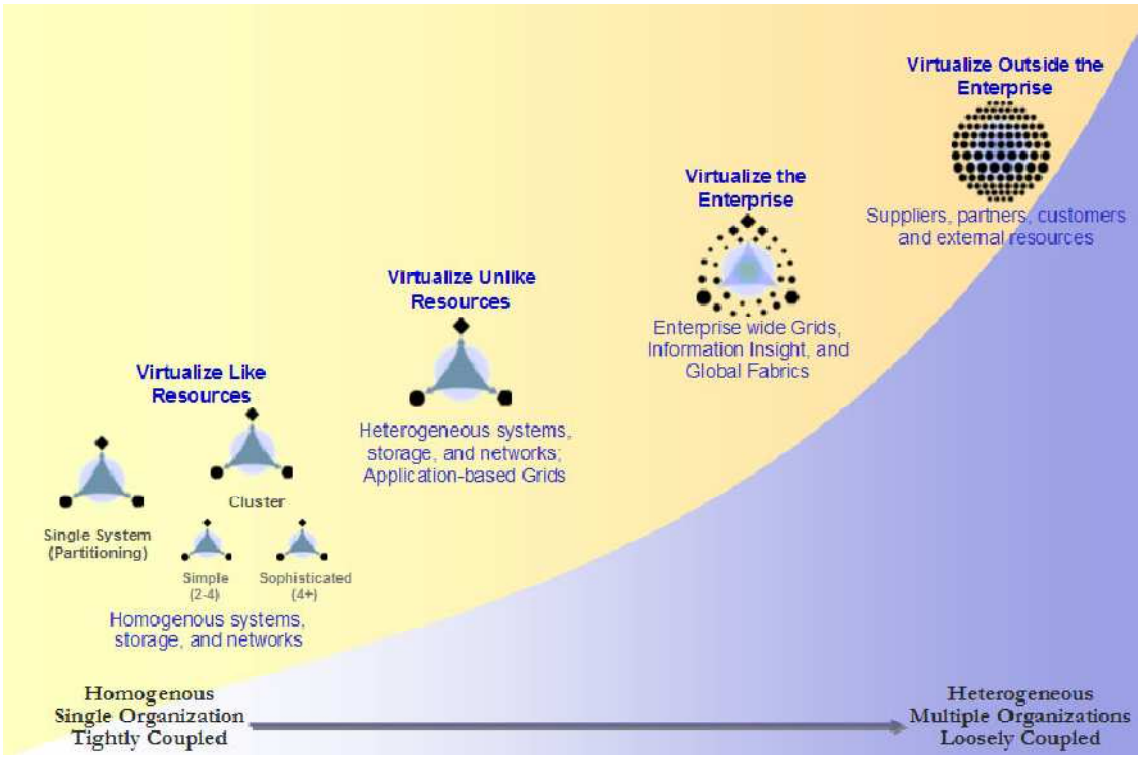

Fig 2.3 Grid continuum [31].

Starting in the lower left wesee single system partitioning. Virtualization starts with being able to carve up amachine into virtual machines. As we move up this spectrum you start to be ableto virtualize similar or homogeneous resources. Virtualization applies not only toservers and CPUs, but to storage, networks, and even applications.

Many research institutions are using some sort of grid computing to address complex computational challenges. The below figure give a typical overview of Grid architecture.

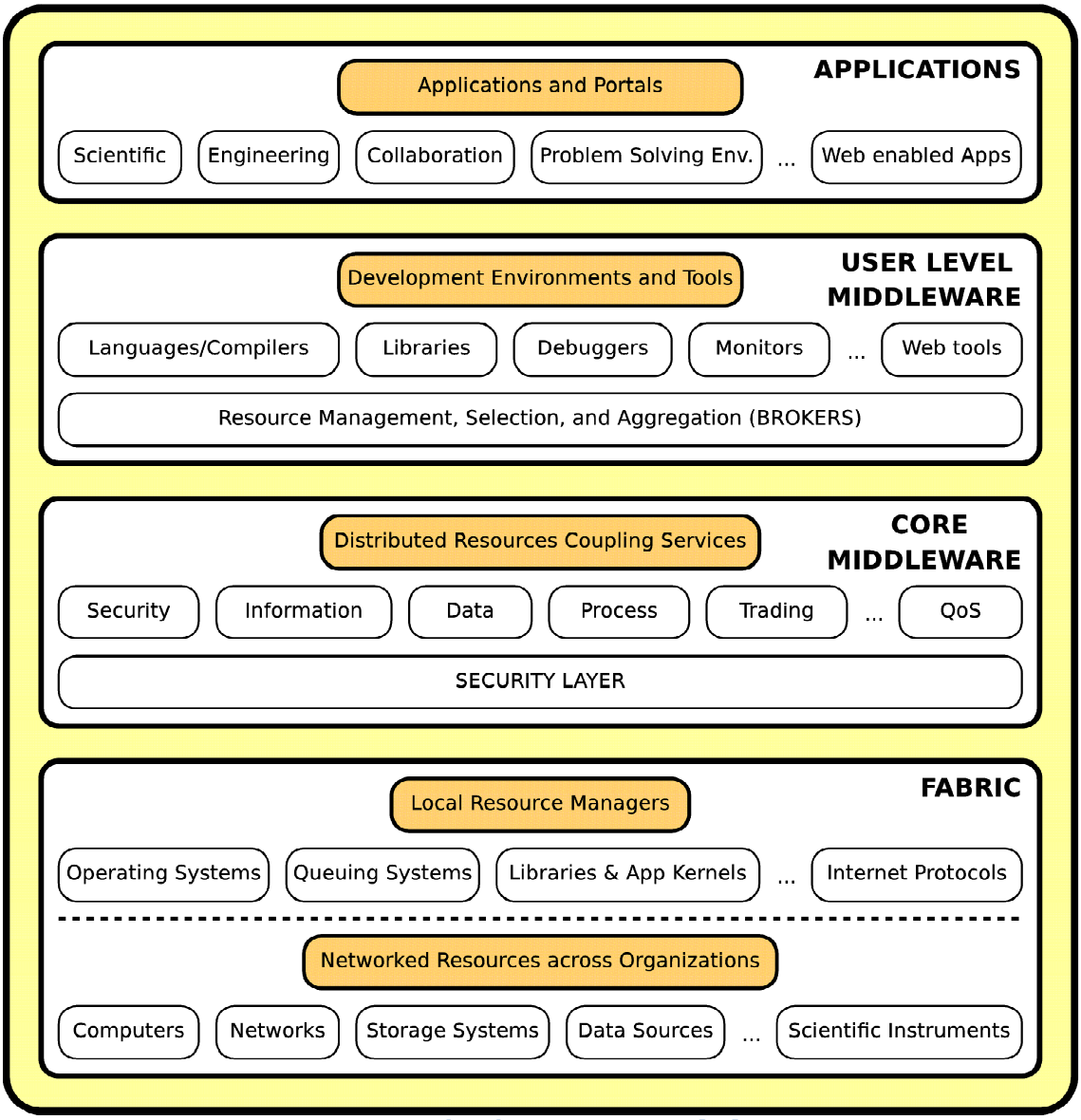

Fig 2.4 Grid Architecture Overview [27].

2.2.1.Benefits of Grid Computing

The capabilities of the Grid Computing are to be better understood to take the advantage of its resources. Here below we list few of the important capabilities of grid computing.

2.2.2.Exploiting underutilized resources

There are large amounts of underutilized computing resources in many organizations by utilizing the concept of grid computing all those underutilized resources can be used for the execution of user applications. Grid Computing provides a framework for exploiting these underutilized resources and thus has the possibility of substantially increasing theefficiency of resource usage [27][31].

More efficient use of idle resources. Jobs can be farmed out to idle servers or even idle desktops. Many of these resources sit idle especially during off business hours. Policies can be in place that allows jobs to only go to servers that are lightly loaded or have the appropriate amount of memory/cpu characteristics for the particular application.

2.2.3.Parallel CPU capacity

A CPU intensive application can be divided in to a number of subjobs and executed on different machines at the same time.Jobs can be executed in parallel speeding performance. Grid environments are extremely well suited to run jobs that can be split into smaller chunks and run concurrently on many nodes.

2.2.4.Virtual resources and virtual organizations for Collaboration

Different distributed computingenvironments can form a large virtual a large virtual computing system offering a variety of resources. The users of the grid can be organizeddynamically into a number of virtual organizations, each with different policyrequirements. These virtual organizations can share their resources collectivelyas a larger grid.

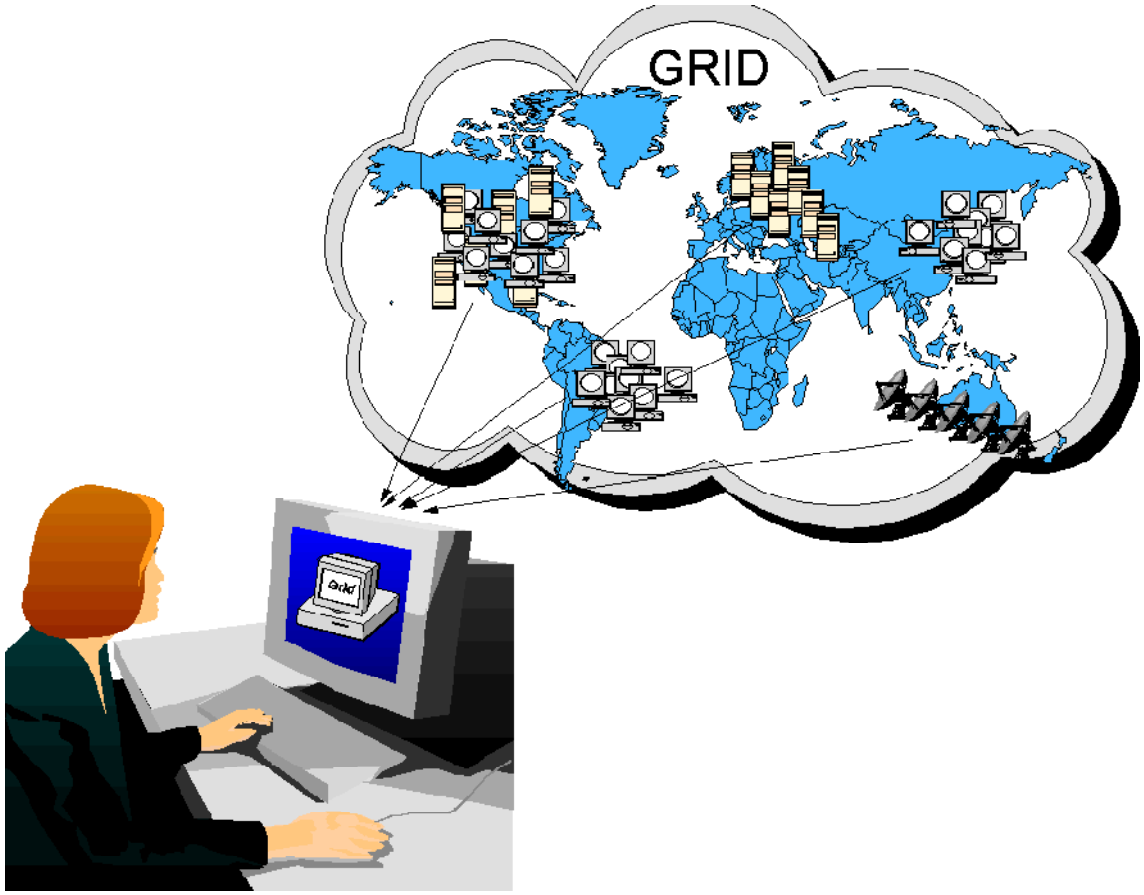

Fig 2.5The Grid virtualizes heterogeneous, geographically disperse resources [31].

2.3. Cloud Computing

Cloud computing comes into focus only when you think about what IT always needs: a way to increase capacity or add capabilities on the fly without investing in new infrastructure, training new personnel, or licensing new software. Cloud computing encompasses any subscription-based or pay-per-use service that, in real time over the Internet, extends IT's existing capabilities.

Cloud computing is at an early stage, with large number of providers, large and small delivering a slew of cloud-based services, from full-blown applications to storage services.

Cloud Computing has been talked about [35], blogged about [36, 37], written about [38, 39, 40] and been featuredin the title of workshops, conferences, and even magazines. Here are few quotes from the IT industry people.

Oracle's CEO:

The interesting thing about Cloud Computing is that we've redefined Cloud Computing to include everything that we already do. . . . I don't understand what we would do differently in the light of Cloud computing other than change the wording of some of our ads.

- Larry Ellison, quoted in the Wall Street Journal, September 26, 2008

These remarks are echoed more mildly by Hewlett-Packard's Vice President of European Software Sales:

A lot of people are jumping on the [cloud] bandwagon, but I have not heard two people say the same thing about it. There are multiple definitions out there of "the cloud."

- Andy Isherwood, quoted in ZDnet News, December 11, 2008

Richard Stallman, known for his advocacy of "free software", thinks Cloud Computing is a trap for users ifapplications and data are managed "in the cloud", users might become dependent on proprietary systems whose costswill escalate or whose terms of service might be changed unilaterally and adversely:

It's stupidity. It's worse than stupidity: it's a marketing hype campaign. Somebody is saying this is inevitable and whenever you hear somebody saying that, it's very likely to be a set of businesses campaigning to make it true.

- Richard Stallman, quoted in The Guardian, September 29, 2008

2.3.1.What is Cloud Computing?

Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systemssoftware.

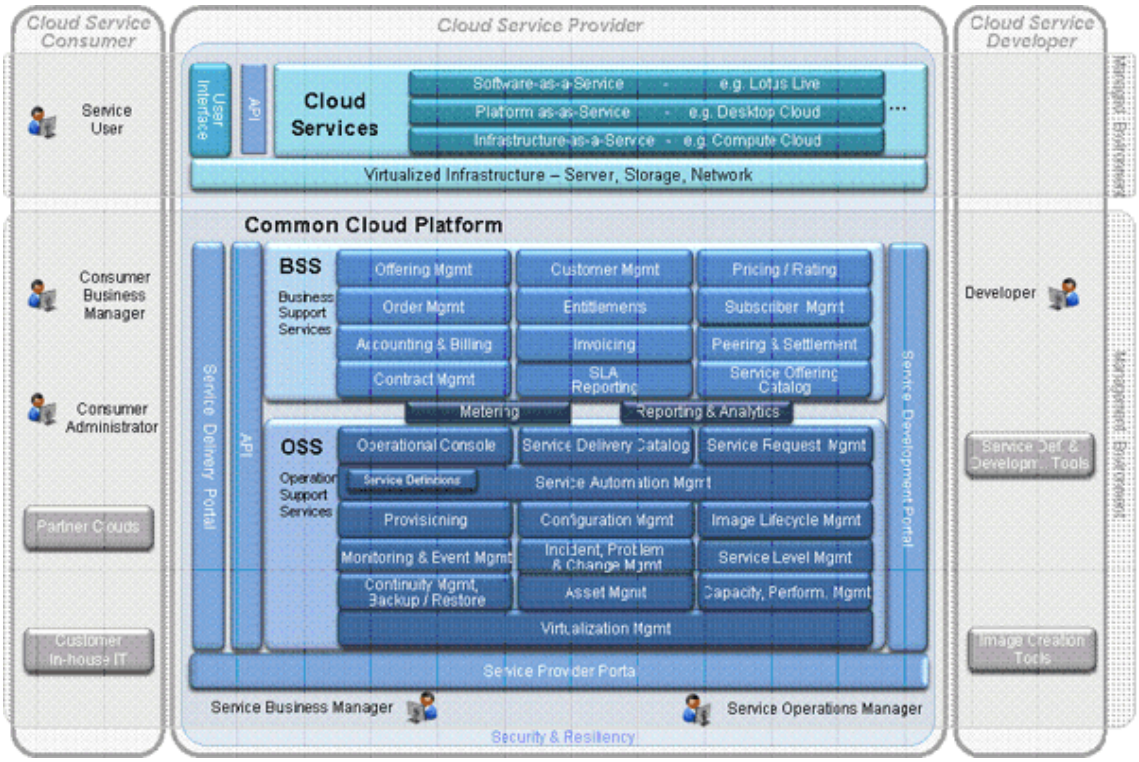

Fig 2.6 Cloud Computing Overview [38].

When a Cloud is made available to all people and is used in pay-as-you-use manner then that particular cloud is called a public cloud. Few examples for such type of clouds are Amazon Web Services, GoogleAppEngine, and Microsoft Azure.

When a particular cloud is used within a particular entity or organization then we call that cloud a private cloud because its usage is confined to an individual entity.

Figure 2.7 shows the roles of the people as users or providers ofthese layers of Cloud Computing, and we'll use those terms to help make our arguments clear.

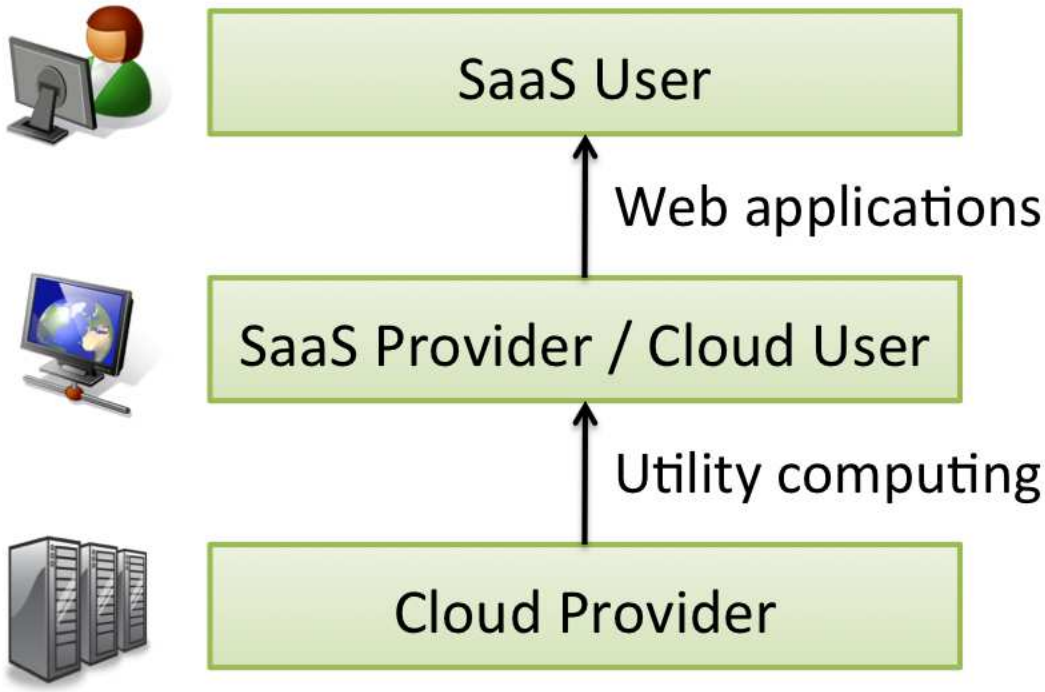

Fig 2.7 Roles of people as users and providers of resources [32].

The below figure shows some of the important players in the field of cloud computing.

Fig 2.8 Some important Cloud Providers [38].

2.3.2.What Cloud Computing is all about

2.3.3.SaaS

This type of cloud computing delivers a single application through the browser to thousands of customers using a multitenant architecture. On the customer side, it means no upfront investment in servers or software licensing; on the provider side, with just one app to maintain, costs are low compared to conventional hosting. Salesforce.com is the best-known example among enterprise applications.

2.3.4.Utility computing

This form of cloud computing is offered by Amazon.com, Sun, IBM, and others who now offer storage and virtual servers that IT can access on demand. The Amazon S3 is the best example for utility computing.

2.3.5.Web services in the cloud

Web service providers offer Application Programming Interfaces that enable developers to exploit functionality over the Internet. Google maps is one of such service [40].

2.3.6.Platform as a service

Platform as a service delivers development environments as a service. You build your own applications that run on the provider's infrastructure and are delivered to your users via the Internet from the provider's servers. Microsoft Azure is one of such type of Cloud Computing platforms.

2.3.7.MSP (managed service providers)

This is one of the oldest forms of Cloud Computing basically an application exposed to IT rather than to end-users, such as a virus scanning service for e-mail or an application monitoring service . McAfee online service is the best example for such type of Cloud providers [39].

2.3.8.Service commerce platforms

This cloud computing service offers a service hub that users interact with. They're most common in online trading environments the best example for this type of Cloud Computing is SAP online service.

2.3.9.Why Cloud Architectures?

Cloud Architectures address key difficulties surrounding large-scale data processing. When large amount of data is to be processed Cloud Architectures are of immense use, one such type of architecture is Eucalyptus System [32][34].

2.4.High-Throughput Computing

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) 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 [29].

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.

2.5. Distributed Resource Management

The main objective of the Grid Computing or the Cloud Computing is that the applications process large amount of data or use large amount of computing power without any bottlenecks, to obtain this objective we use various Distributed Resource Management systems

There are a number of such systems a few to mention among themare

2.5.1. OpenPBS

Portable Batch Systemperforms job scheduling. Its primary task is to allocate jobs among the available resources. It is often used in conjunction with Unix cluster environments.

OpenPBS was developed by NASA Ames Research Center, Lawrence Livermore National Laboratory, and Veridian Information Solutions Inc.

PBS is supported as a job scheduler mechanism by several Meta schedulers including Moab by Cluster Resources and GRAM (Grid Resource Allocation Manager), a component of the Globus Toolkit [47].

2.5.2. Oracle Grid Engine

Oracle Grid Engineis an open source batch-queuing system, was developed by Sun Microsystems. But presently it is named as Oracle Grid Engine or Grid Engine simply. Grid Engine is open source and free to use. The commercial version is also available from the Oracle [48].

Oracle Grid Engine is typically used on a cluster farm or high-performance computing (HPC) cluster and is responsible for accepting, scheduling, dispatching, and managing the remote and distributed execution of large numbers of standalone, parallel or interactive user jobs.

2.5.3. Condor

Condoris an open source high-throughput computing framework for computationally intensive jobs. It can be used to manage jobs on a dedicated cluster or to a pool out of idle system in an organization this method is called cycle scavenging. Condor runs on Linux, Unix, Mac OS X, FreeBSD, and contemporary Windows operating systems. Condor can easily integrate both dedicated resources like rack-mounted clusters and non-dedicated machines like cycle scavenging into one computing environment [28].

Developed by the Condor team at the University of Wisconsin–Madison and is freely available for use. It follows an open source philosophy and is licensed under the Apache License 2.0. It can be downloaded from their page.

2.6.Condor

The main arm of Condors system is to set up a provisioning system that overseas allof it computer resources and assigns jobs that have been submitted by users to them [28][29].

Some of the main functions that Condor offers are...

Job Queuing
Job Scheduling
Resource Monitoring
Resource Management

When users need to submit a job all that they need to do is specify in a small file,called a Submit file, the kind of environment theirjob needs and the Condor system manages the remaining part of the task and once the job is done the system places the results in the specified location.

Condor system makes use of all the wasted computing power that manyfacilities have tied up in idle workstations. Condor can manage these workstations andrun jobs on them while the mouse or keyboard has no input and if input is detected,store the jobs state, shift to an idle workstation and continue running again.

2.6.1.Universes

In Condor system there are several different environmentswhere user can choose to run his job according to the requirements of his job. These Condors universes allow user to specify even more about what type ofjob to run.

Standard Universe: For running jobs that you wish to be able to store their state and shift toa different machine if interrupted. Requires source code to be specially linked withcondor compile.

Vanilla Universe: An anything goes environment, if for some reason your code will not relinkunder condor compile you can run that job using the vanilla universe.

Parallel Universe: For running a number of jobs at the same time, or running a MPI job.

Grid Universe: Used to submit jobs onto remote grids.

Java Universe: Allows users to run jobs written for the Java Virtual Machine.

Virtual Universe: Allows users to run virtual machine jobs.

2.6.2.ClassAds

Condor manages such a diverse environment in a simple way by just matching the jobs with the machines those possess the resources that the job requires.

Condor's way of talking this problem is ClassAds. A ClassAd is simply as is sounds;the machines all have a ClassAd, and the job submit files all have a ClassAd. Condor'sjob is to match them appropriately.

Machine ClassAd - All Condor machines have a very verbose and highly configurableClassAd describing the entire machine. RAM, CPU, loads, usage hours, conditionsand many otherproperties are available in machines ClassAd the condor ClassAd can be configured by the machine owner according to his needs.

Job ClassAds - Every job that is submitted also has an associated ClassAd, and theuser may specify what type of architecture, operating system, amount of RAM andalso the type of Universe that they would like to run in.

Condor will then play the role of a matchmaker by continuously reading the entire jobClassAds and the entire machine ClassAds, matching and ranking job ads with machineads.

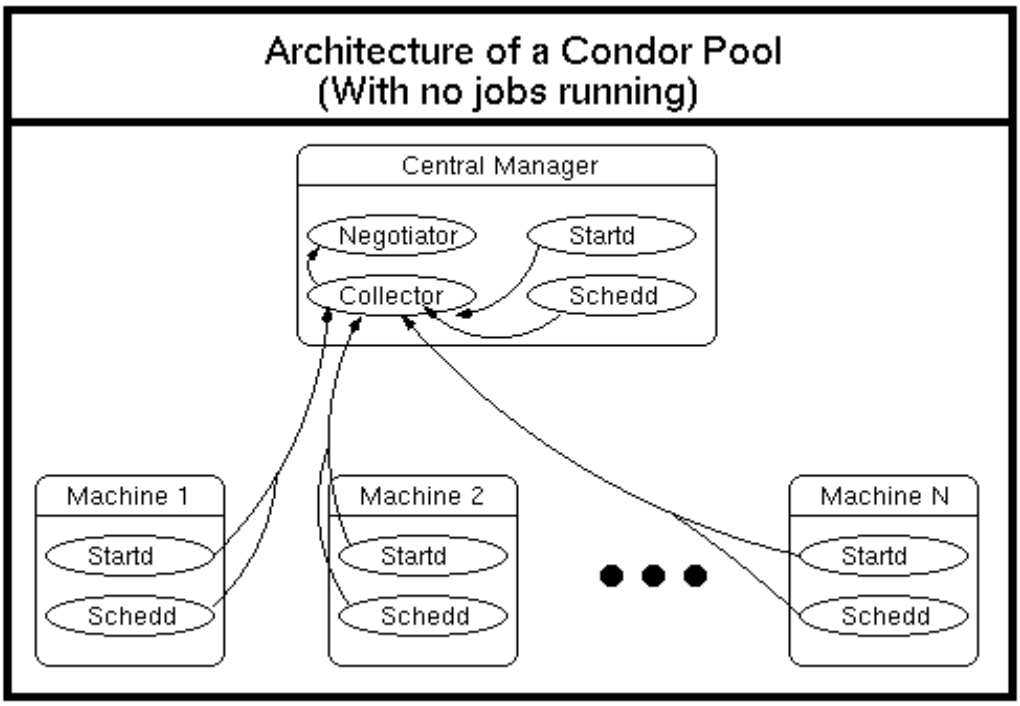

Fig 2.9 Condor Overview [29].

2.6.3.What happens when a job is submitted to Condor?

Every Condor job involves three machines. One is the submitting machine, where the job is submitted from. The second is the central manager, which finds an idle machine for that job. The third is the executing machine, the computer that the job actually runs on. But we canuse a single machine can perform two or even all three of these roles. In such cases, the submitting machine and the executing machine might actually be the same piece of hardware, but all the mechanisms described here will continue to function as if they were separate machines. The executing machine is often many different computers at different times during the course of the job's life. The jobs stays in the queue until and unless it the condor manager finds a execute machine with the resources required by that particular job.

When a match is made between a job and a machine, the Condor daemons on each machine are sent a message by the central manager. The schedd on the submitting machine starts up another daemon, called the "shadow". This acts as the connection to the submitting machine for the remote job, the shadow of the remote job on the local submitting machine. The startd on the executing machine also creates another daemon, the "starter". The starter actually starts the Condor job, which involves transferring the binary from the submitting machine. The starter is also responsible for monitoring the job, maintaining statistics about it, making sure there is space for the checkpoint file, and sending the checkpoint file back to the submitting

machine (or the checkpoint server, if one exists). In the event that a machine is reclaimed by its owner, it is the starter that vacates the job from that machine [28].

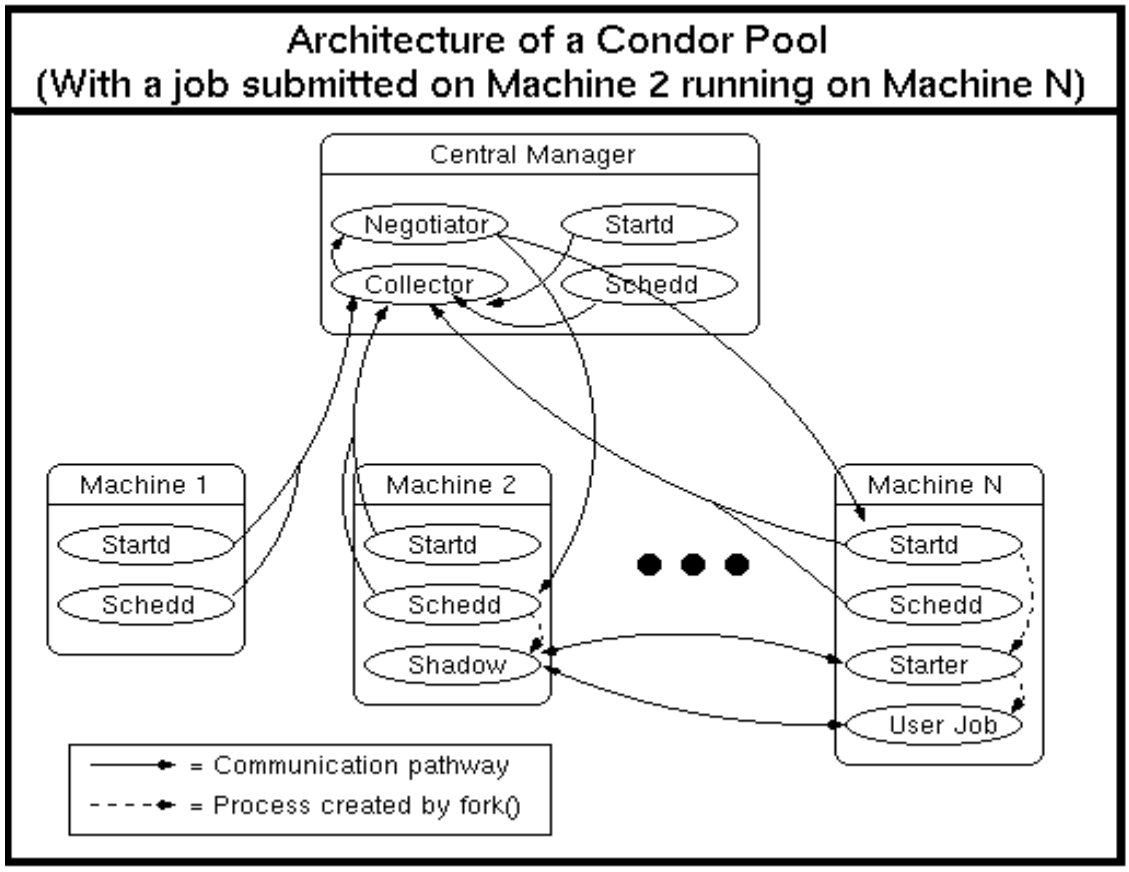

Fig 2.10 Condor job Overview[29].

2.6.4. Condor Virtual Machine Universe

Condor VM universe allows jobs to be Virtual Machines instead of simply executables. Virtual Machines allow for a greater flexibility with regards to the types of jobs users can submit. It allows a user to run applications written for one platform to be run on top of an arbitrary platform, without the need to port the original application to the new platform. VM universe supports several virtual machine applications, today we will be looking at VMware Server, but similar jobs can be run using Xen, etc[27][28].

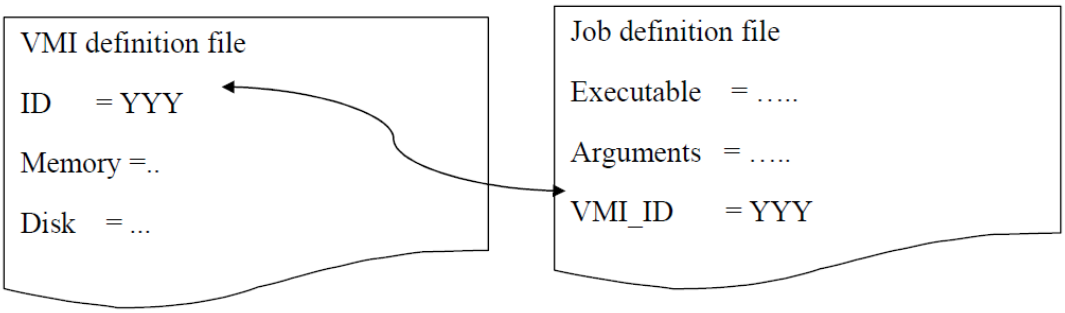

Fig 2.11 VM and Job definition files

2.6.5. Submitting a VM Job

The below fig shows a typical VM job submit file [30].

```
Universe                    = vm
Executable                  = any_name_you_like
Log                         = simple.vm.log.txt
vm_type                     = vmware
vm_memory                   = 64
vmware_dir                  = /home/condor/job/condorvm
vmware_should_transfer_files = Yes
Queue
```

Fig 2.12 VM job submit file

Chapter 3 Design of the proposed solution

3.1. Introduction

Now that we have seen the existing technology and the State of Art in our field of research we have to design the architecture to implement our project and experiment with the designed architecture.

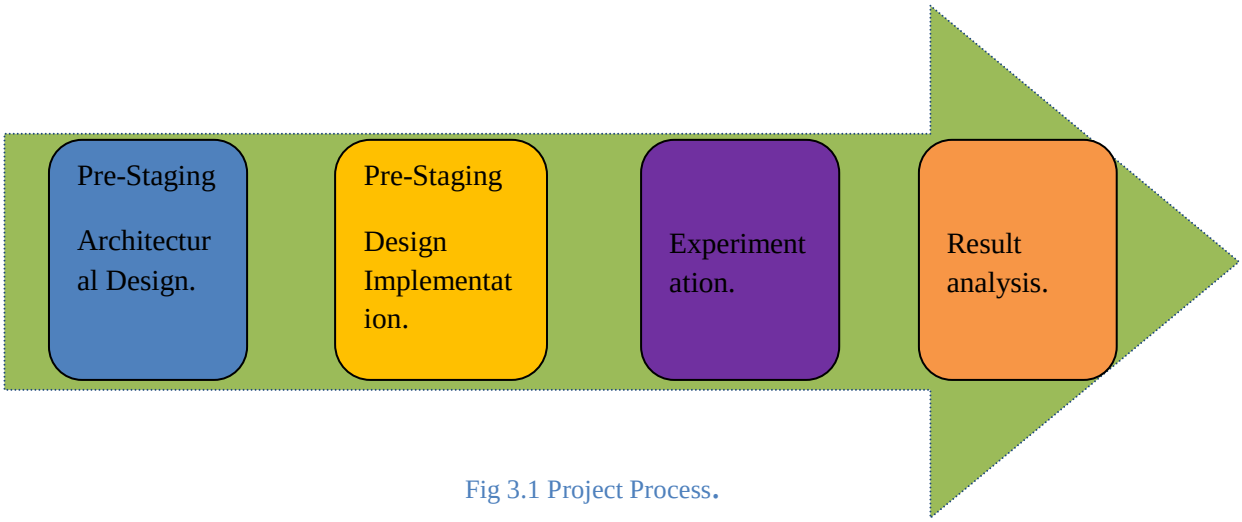

Fig 3.1 Project Process.

In order to achieve our objectivewe 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 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.

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

Architecture Compatibility The operating system running in the Virtual Machine must be compatible with the system architecture exposed by the hypervisor.

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.

The below figure shows our Pre-Staging Model Architecture.

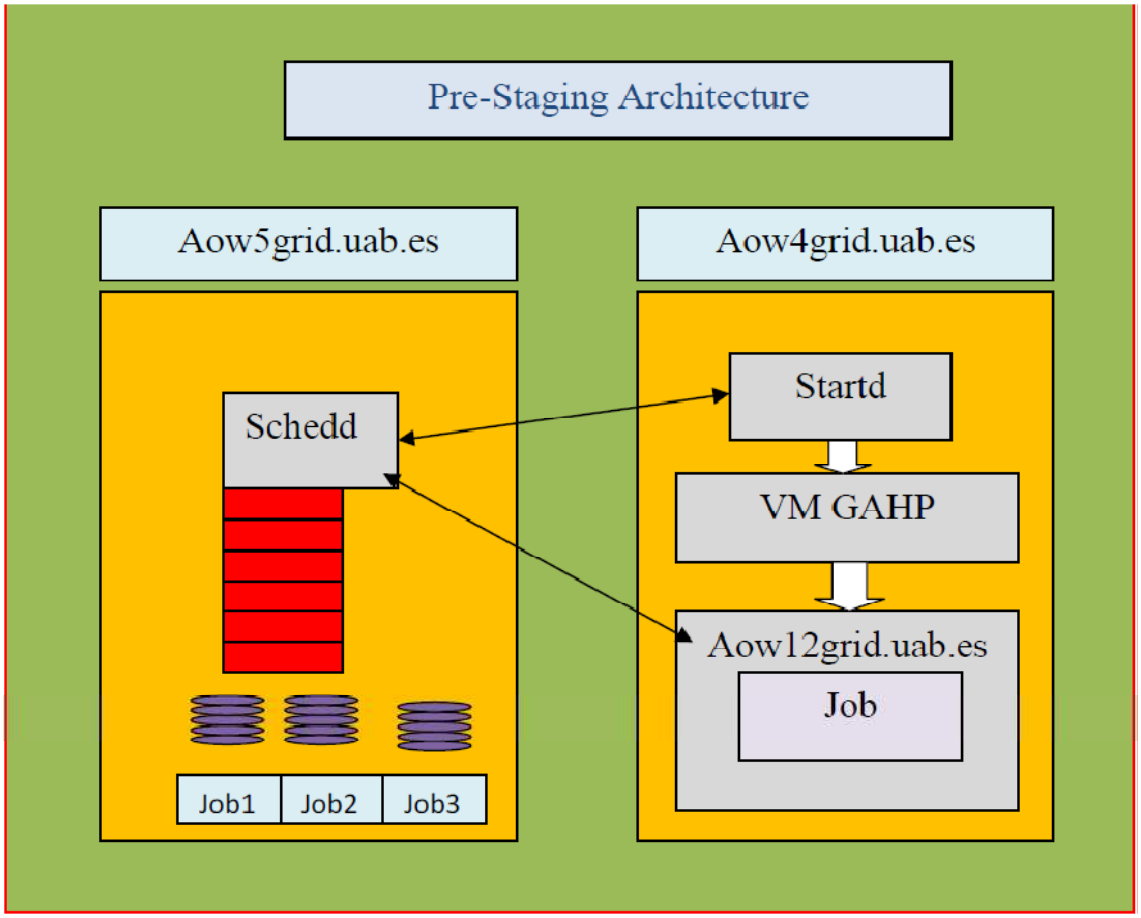

Fig 3.2 Pre-Staging Architectural Design.

We base our Pre-Staging Model on Condor where a Machine submits jobs to the pool, the manager matches the job to the available execute machines.

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.

3.2.Pre-Staging Architectural Design

For designing the architecture we use the following systems for hardware purpose we use the laboratory of area 51 of our University and the computers utilized for this project are **aow5grid.uab.es** and **aow4grid.uab.es** with the following hardware configurations.

Aow5grid.uab.es

Fig 3.3 Submit Machine Configuration.

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**

Fig 3.4 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 forimplementation 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 in the second chapter the various virtualization software that is available today like Xen, KVM, Vmware etc. we choose the Vmware Server version 1.0.1 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.

Chapter 4 Implementation

4.1. Pre-Staging Model 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.

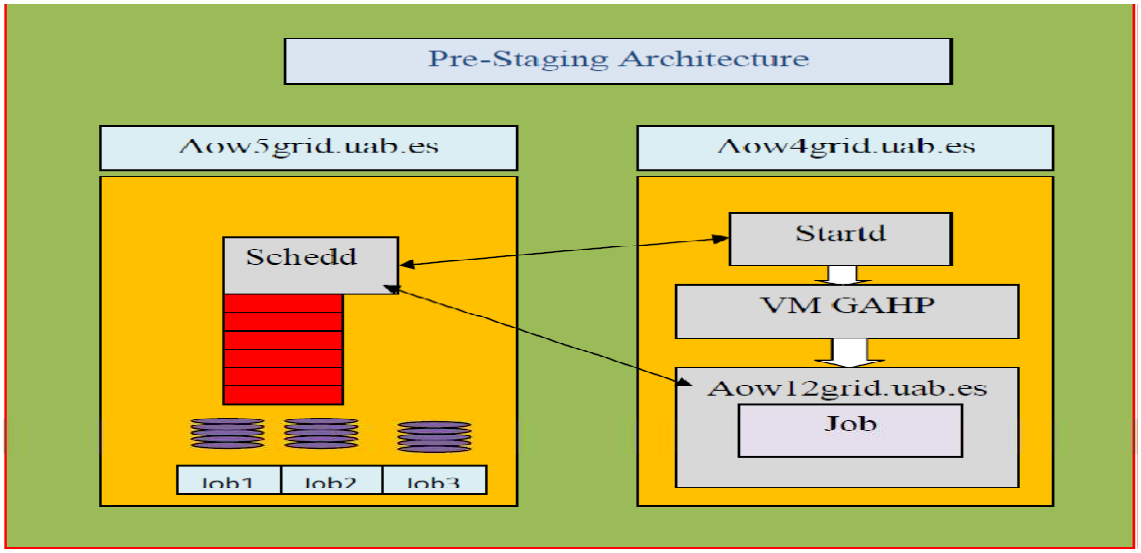

Fig 4.1Pre-Staging Implementation.

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.

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.

Fig 4.2 Virtual Machine Image Configuration.

4.2. Experiment Design

We leverage Condor's existing ability to schedule jobs, advertise resource availability and match jobs to resources and introduce a flexible extension for dynamically describing, deploying, and using virtual execution resources in the Condor virtual universe.

In Condor, one or more machines (resources) along with jobs (resource requests) are part of a collection, known as a pool. The resources in the pool have one or more of the following roles: Central Manager, Execute, and/or Submit. The Central Manager collects information and negotiates how jobs are matched to available resources. Submit resources allow jobs to be submitted to the Condor pool through a description of the job and its requirements. Execute resources run jobs submitted by users after having been matched and negotiated by the Central Manager.

We extend the responsibilities of each of these three different roles to incorporate virtualization into Condor. Each Execute resource describes

the extent to which it can be virtualized (to the Central Manager) and is responsible for hosting additional (virtual) resources. The Submit resource(s) takes a workflow of jobs and requirements and initiates the deployment of the virtual resources plus signals its usage (start/stop) to the host/execute machine. The Central Manager is responsible for storing virtual machine metadata used for scheduling. For this implementation, we use two machines one for the Central Manager, Submit, and the other for Execute roles.

The virtualization capabilities for a particular Execute resource can be published to the Central Manager via authorized use of *condor_advertise*. Attributes about the virtual Execute resources, such as the operating system (and version), available memory and disk space, and more specific data about the status of the virtual machine are included. Currently, the "host" Execute resource invokes *condor_advertise* for each "guest" or virtual Execute resource it anticipates hosting at start-up. This approach allows virtual resources to appear almost indistinguishable from real physical resources and will be included in Condor's resource scheduling. Note that real resources are running while the virtual resources are not.

4.3. Experimentation Set: 1

As we havediscussed Pre-Staging Model in the previous Chapter 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.

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.

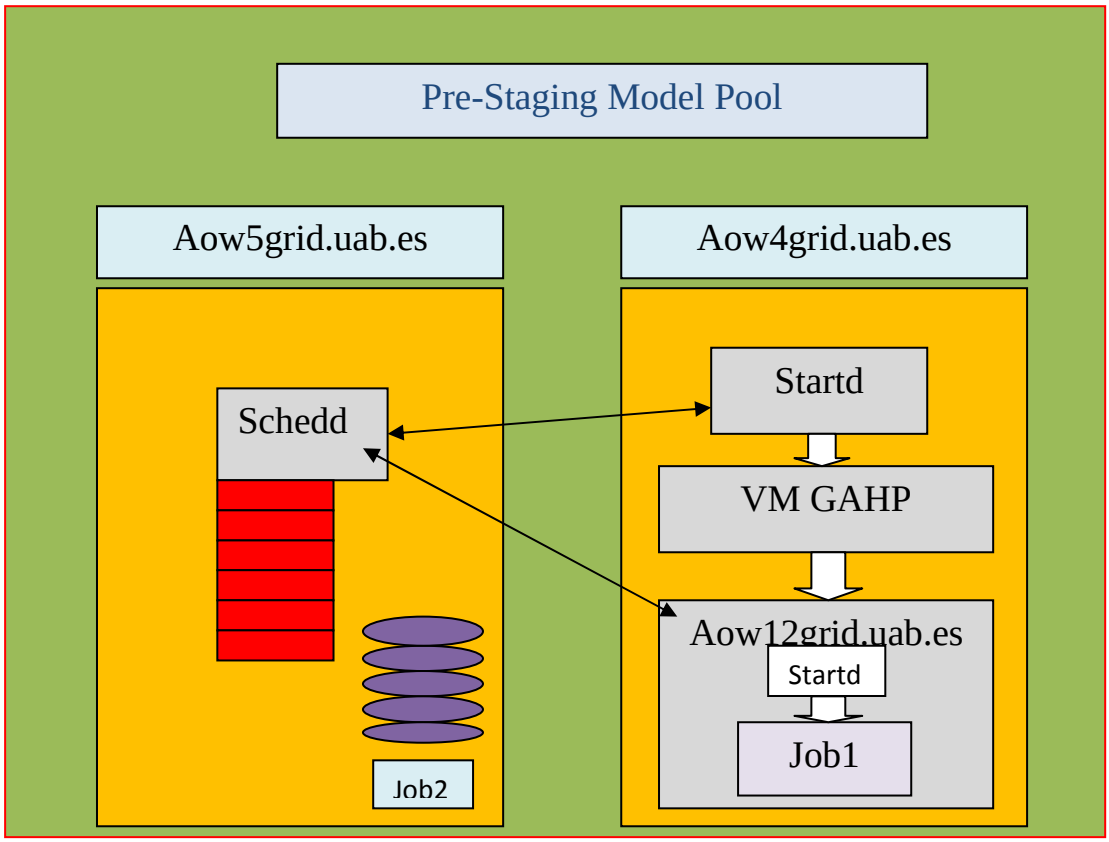

Fig 4.3 Pre-Staging Model Virtual Machine with a Job.

4.4. Experimentation Set: 2

In the second experimentation 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.

Chapter 5 Results

5.1. Experiment: 1

Objective: The main objectives of this experiment are to show that the Pre-Staging Model is more efficient than Condor Virtual Machine Universe.We use the first experimentation set mentioned in the fourth chapter for carrying out this experiment. We submit the VM using Condor VM universe without any job to do when the VM boots up, it joins the existing Condor pool.

```
Universe                        = vm
Executable                      = anything
Log                             = simple.vm.log.txt
vm_type                         = vmware
vm_memory                       = 160
vmware_dir                      = /home/condor/condor-job/Fedora
vmware_should_transfer_files = true
Queue
```

Fig 5.1 Virtual Machine Submit File.

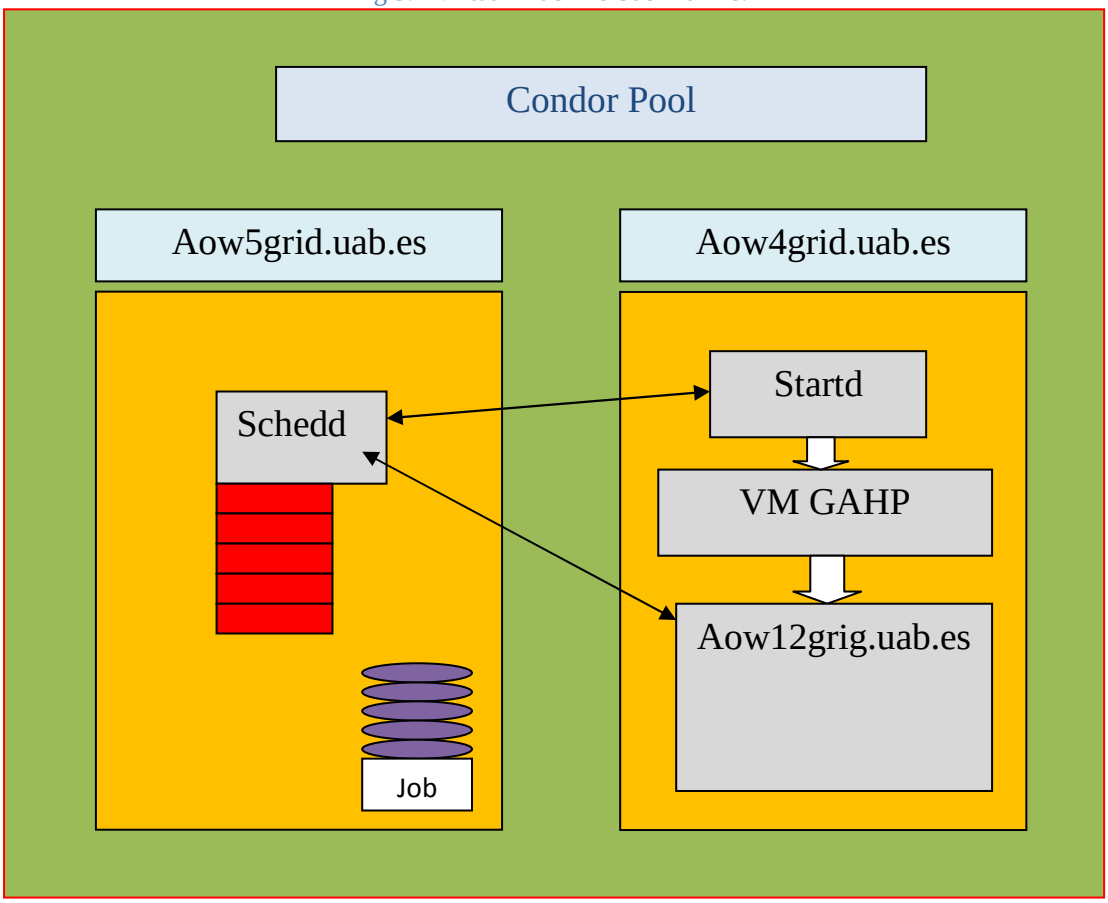

Fig 5.2 Shows Virtual Machine Joins the Condor Pool.

Once we assure ourselves that the VM is on the pool we submit jobs to the VM using the custom ClassAd attributes into a job via the submit file.

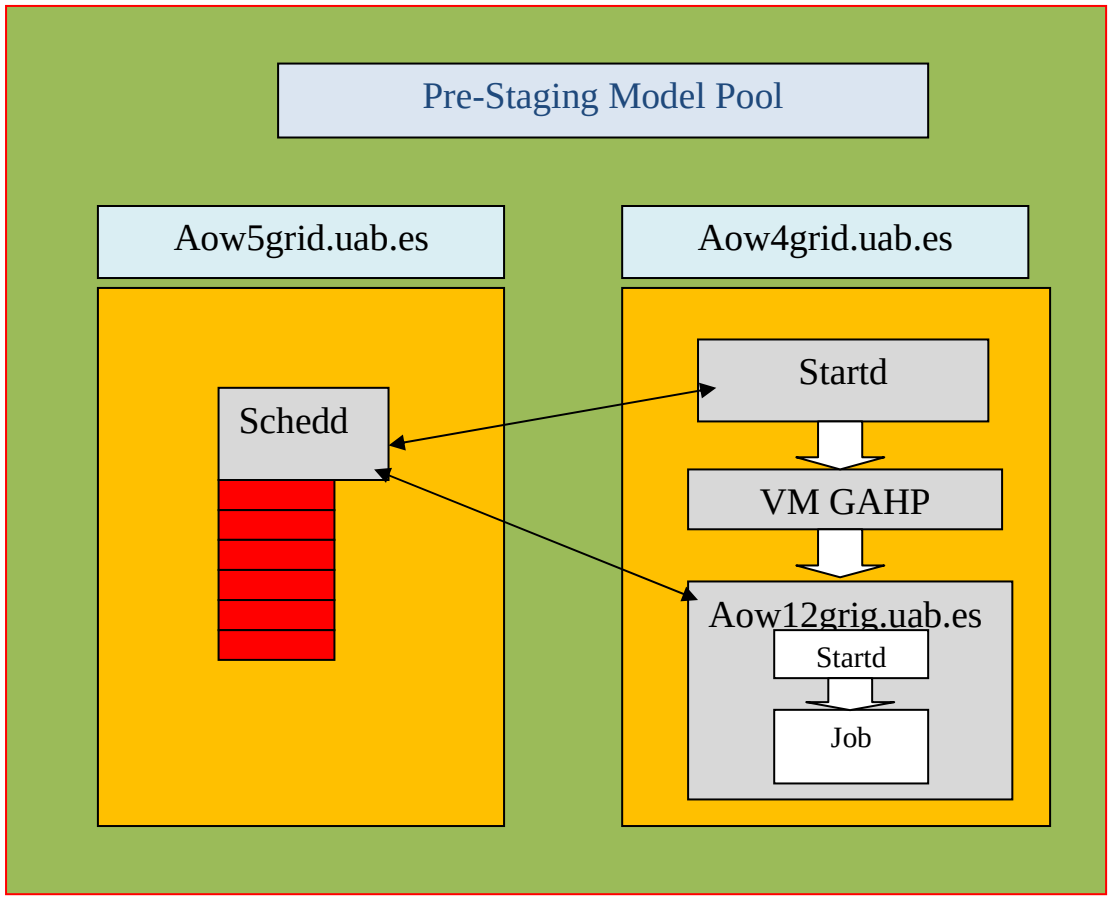

Fig 5.3 Shows the Job execution on the Virtual Machine.

After executing the jobs we want, we kill the VM by just killing the VM Universe job.

We repeat the experiment for a 15 times for each job and note the time taken for the VM for completing the job. At the same time we run the same job using Condor Virtual Machine Universe and note the timings obtained for various executions of the job and make an average of them to formulate our results.

We run the experiment for 15 times 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.

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

Table 5.1 Shows average execution times by using Pre-Staging and Condor VU methods.

C-Application:

For the C-language Application we repeat the experiment for 15 times on the Condor Virtual Machine Universe and also on our Pre-Staging model and note the execution times for each execution of job for all the 15 executions and make an average of them. The figure below shows them.

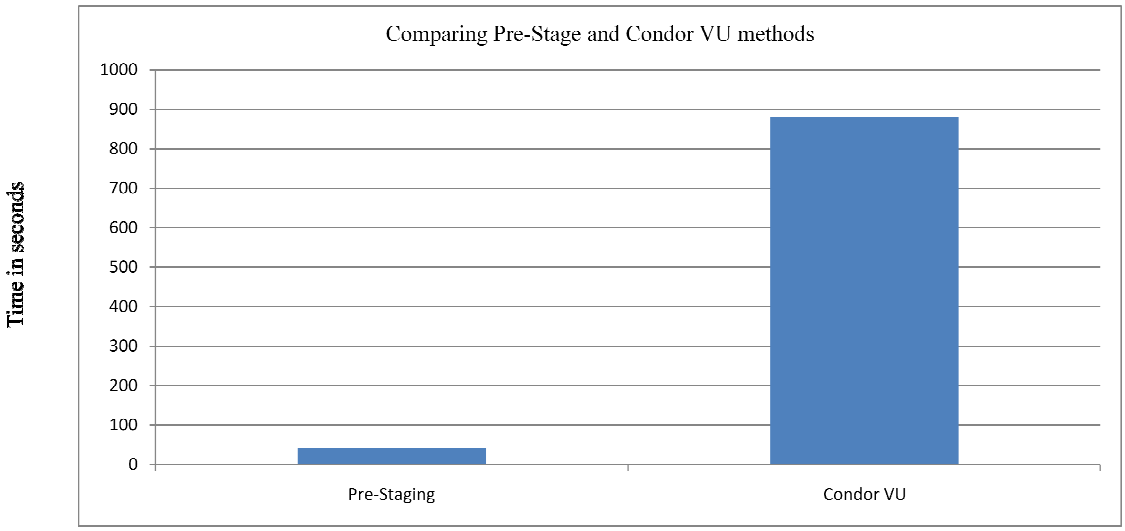

Fig 5.4 C-Application Job Completion using Pre-staging and Condor VU methods.

We can clearly observe that by using the Pre-Staging Model the application gains nearly 21 times in execution time. In Condor Virtual Machine Universe Method the time taken to transfer the virtual machine image from submitter to execute machine is 686 seconds which is 78% of the total job time, average virtual machine boot time is 67 seconds which is 7.1%of overall time of the job, the actual time taken to execute the application is 69 seconds which in turn is only 7.7% and the time taken by the virtual machine to shut down after the job completion is 58seconds which is 6.5% of total time.

Over all we can observe that the Condor Virtual Machine Universe Method consumes much time in other activities like file transfer, virtual machine boot and shutdown, when compared to job execution which is a small portion of the total time.

Java Application:

For the Java Application we repeat the experiment for 15 times on the Condor Virtual Machine Universe and also on our Pre-Staging model and note the execution times for each execution of job for all the 15 executions and make an average of them.
The figure below shows the comparison of execution times on both systems.

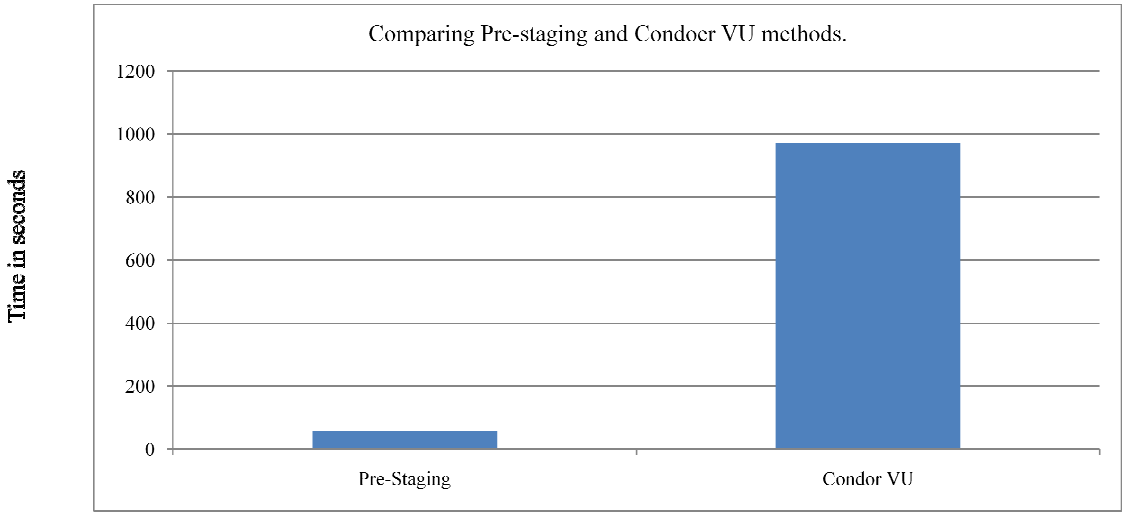

Fig 5.5Java Application Job Completion using Pre-staging and Condor VU methods.

From studying the above figure we can see that the application gains 17 times in comparison with Condor Virtual Universe by the Pre-Staging method.

For this application by Condor Virtual Machine Universe Method the time taken to transfer the virtual machine image from submitter to execute machine is 78.5% of the total job time, average virtual machine boot time is 7%of overall time of the job, the actual time taken to execute the application is only 8.5% and the time taken by the virtual machine to shut down after the job completion is 5.8% of total time.

MPI Application:

For the MPI Application we repeat the experiment for 15 times on the Condor Virtual Machine Universe and also on our Pre-Staging model and note the execution times for each execution of job for all the 15 executions and make an average of them.

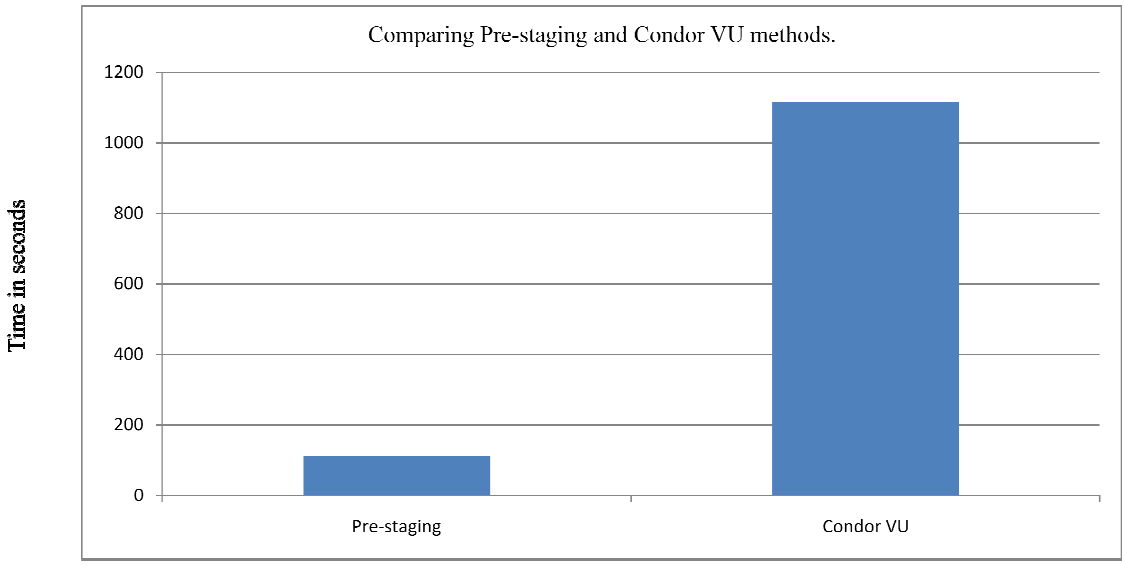

Fig 5.6MPI Application Job Completion using Pre-staging and Condor VU methods.

The figure shows that the application gains nearly 10 times in comparison to the Condor Virtual Machine Universe by the new method ,but we should also observe that the Pre-Staging Model efficiency decrease considerably for this application when compared to the above application this is due to the overhead induced by virtualization is around 60% for the High-Performance Computing applications.

For MPI Application by using the Condor Virtual Machine Universe Method we can examine that the amount of time taken to transfer the virtual machine image from the submit machine to the execute machine is76.5%of the total time taken, for this application time consumed to boot the virtual machine is 5.9%of the total time, the actual time required for the execution of the application is only 139 seconds which is only 12.5%of the overall time and finally 5.1%of the time is consumed for stopping the virtual machine after job completion.

5.2. Experiment: 2

We carry out the second experiment by using the experiment set 2 with a different Virtual Machine Image made up of Ubuntu10.10 for all the three types of applications and repeat the experiment for 15 times on both Pre-Staging and Condor Virtual Machine Universe models.

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

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

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

Table 5.2 Shows average execution times by using Pre-Staging and Condor VU methods.

C-Application:

For the C-language Application we repeat the experiment for 15 times on the Condor Virtual Machine Universe and also on our Pre-Staging model and note the execution times for each execution of job for all the 15 executions and make an average of them.

The below figure shows the average execution times for C-language Application.

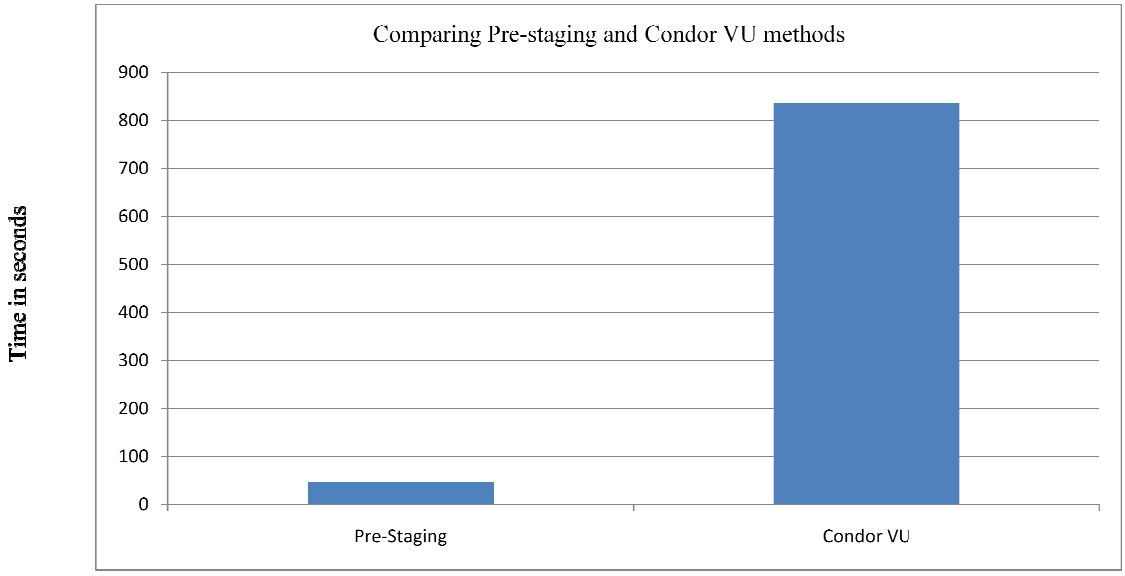

Fig 5.7 C-Application Job Completion using Pre-staging and Condor VU methods.

We can clearly observe that by using the Pre-Staging Model the application gains nearly 17 times in execution time.

In Condor Virtual Machine Universe Method the time taken to transfer the virtual machine image from submitter to execute machine is 664 seconds which is 79.3% of the total job time, average virtual machine boot time is 59 seconds which is 7.1%of overall time of the job, the actual time taken to execute the application is 61 seconds which in turn is only 7.2% and the time taken by the virtual machine to shut down after the job completion is 53seconds which is 6.3% of total time.

Java Application:

For the Java Application we repeat the experiment for 15 times on the Condor Virtual Machine Universe and also on our Pre-Staging model and note the execution times for each execution of job for all the 15 executions and make an average of them.

Java Application execution times are represented by the below figure

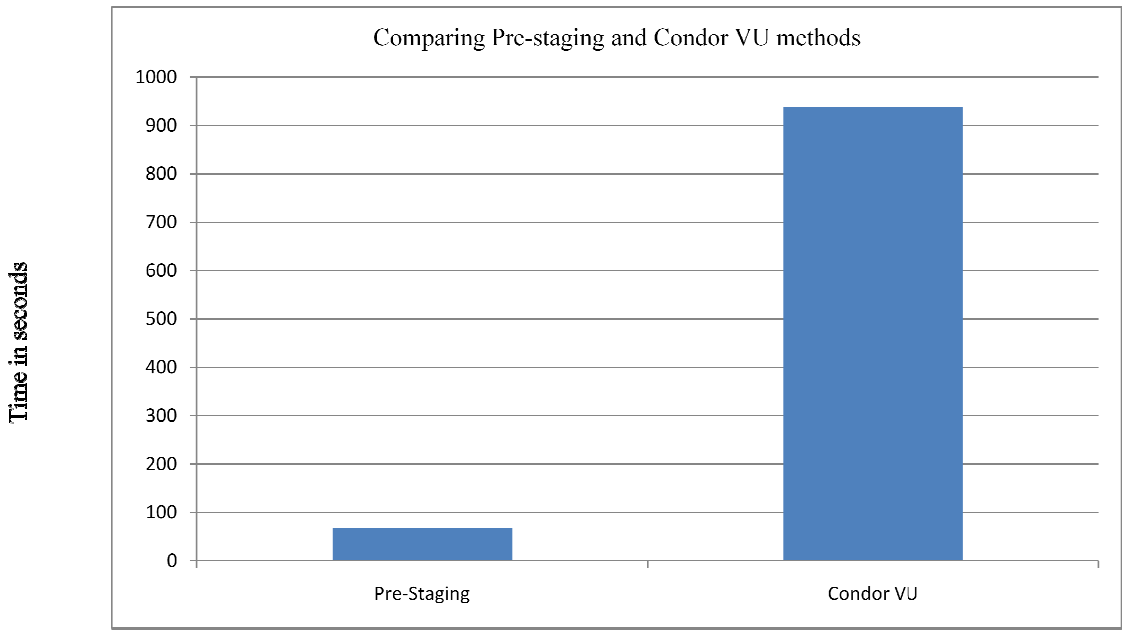

Fig 5.8 Java Application Job Completion using Pre-staging and Condor VU methods.

The application gains performance improvement of nearly 14 times in comparison to Condor Virtual Machine Universe by Pre-Staging Model.

For this Java application by Condor Virtual Machine Universe Method the time taken to transfer the virtual machine image from submitter to execute machine is 78.7% of the total job time, average virtual machine boot time is 6.1%of overall time of the job, the actual time taken to execute the application is only 9.4% and the time taken by the virtual machine to shut down after the job completion is 5.6% of total time.

MPI Application:

For the MPI Application we repeat the experiment for 15 times on the Condor Virtual Machine Universe and also on our Pre-Staging model and note the execution times for each execution of job for all the 15 executions and make an average of them. The figure below shows them.

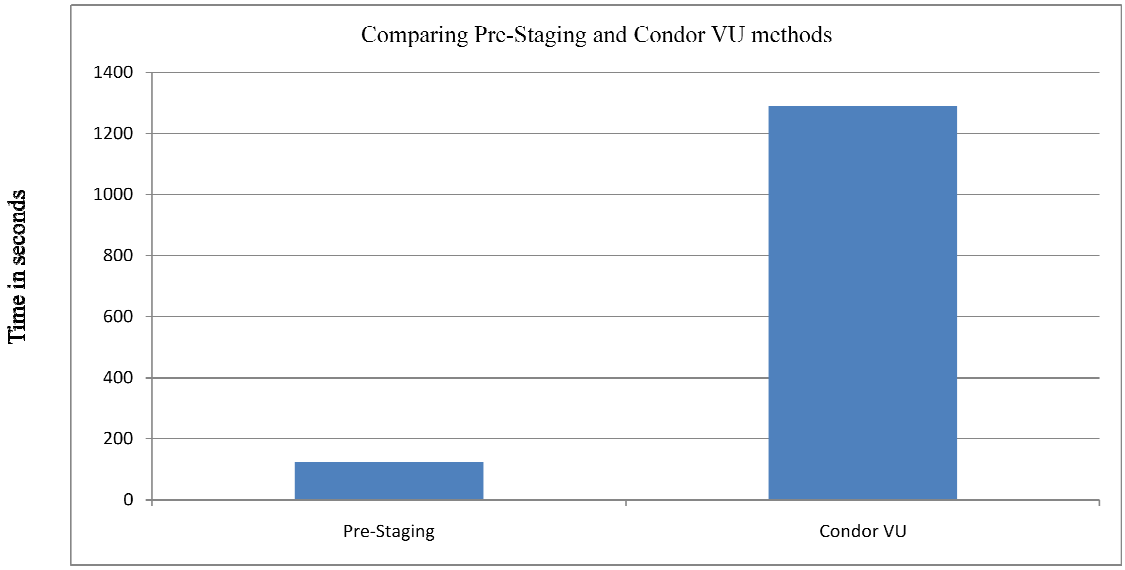

Fig 5.9 MPI Job Completion using Pre-staging and Condor VU methods.

Here we can observe that the performance improvement is just 10 times only when the above applications have much more, this performance degradation is due to the overhead induced by the virtualization on High-performance applications.

For MPI Application by using the Condor Virtual Machine Universe Method we can examine that the amount of time taken to transfer the virtual machine image from the submit machine to the execute machine is79.1%of the total time taken, for this application time consumed to boot the virtual machine is 4.5%of the total time, the actual time required for the execution of the application is only 154 seconds which is only 12%of the overall time and finally 4.3%of the time is consumed for stopping the virtual machine after job completion.

Chapter 6 Conclusions and Future Work

6.1 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.

6.2 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] James E. Smith and Ravi Nair. Virtual Machines: Versatile Platforms for Systems and Processes. Morgan Kaufmann/Elsevier, 2005.

[4] Andy Chou and Junfeng Yang and Benjamin Chelf and Seth Hallem and Dawson Engler. An Emprical Study of Operating System Errors. Proceedings of the 18th ACM Symposium on Operating System Principles (SOSP), 2001.

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

[6] Vasanth Bala and Evelyn Duesterwald and Sanjeev Banerjia. Dynamo: A Transparent Dynamic Optimization System. Proceedings of the ACM SIGPLAN 2000 Conference on Programming Language Design and Implementation (PLDI), Vancouver, British Columbia, Canada, 2000.

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

[8]OpenVZproject.OpenVZWikiMainPage.
http://wiki.openvz.org/Main_Page.

[9] Dawson R. Engler and M. Frans Kaashoek and James W. O'Toole. Exokernel: Operating System Architecture for Application-Level Resource Management. Proceedings of the 15th Symposium on Operating System Principles (SOSP 1995), 1995.

[10] Paul Barham and Boris Dragovic and Keir Fraser and Steven Hand and Tim Harris and Alex Ho and Rolf Neugebauer and Ian Pratt and Andrew Warfield. Xen and the Art ofVirtualization. Proceedings of the nineteenth ACM symposium on Operating systems principles in Operating Systems Review, pages 164--177, New York, 2003. ACM Press.

[11] VMWare. VMWare ESX and ESXi product page.

http://www.vmware.com/products/esx/index.html

[12] Michael D. Schroeder and Jerome H Saltzer. A hardware architecture for implementing protection rings. Proceedings of the 3rd ACM Symposium on Operating System Principles (SOSP '71), Palo Alto, CA, USA.

[13] L4HQ.org. L4Linux info page. http://l4linux.org/.

[14] Gerwin Klein and Kevin Elphinstone and Gernot Heiser and June Andronick and David Cock and Philip Derrin and Dhammika Elkaduwe and Kai Engelhardt and Rafal Kolanskiand Michael Norrish and Thomas Sewell and Harvey Tuch and Simon Winwood. seL4:Formal Verification of an OS Kernel. Proceedings of the 22nd ACM Symposium on OSPrinciples (SOSP '09), Big Sky, MT, USA, 2009. available athttp://ertos.nicta.com.au/publications/papers/Klein_EHACDEEKNSTW_09.pdf .

[15] Leonid Baraz and Tevi Devor and Orna Etzion and Shalom Goldenberg and AlexSkaletsky and Yun Wang and Yigal Zemach. IA-32 execution layer: a two-phase dynamic translator designed to support IA-32 applications on Itanium®-based systems. Proceedings of the 36th Annual IEEE/ACM International Symposium on Micro architecture, pages 191-201, 2003.

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

[17] 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.

[18] VMWare. Transparent Previrtualization info page.

http://www.vmware.com/interfaces/paravirtualization.html.

[19] Virtualization: State of the Art

Version 1.0, April 3, 2008 Copyright © 2008 SCOPE Alliance. All rights reserved.

[20] John Watson. VirtualBox: Bits and Bytes Masquerading as Machines. Linux Journal,2008. Available at http://www.linuxjournal.com/article/9941.

[21] Jeremy Sugerman and Ganesh Venkitachalam and Beng-Hong Lim. Virtualizing I/ODevices on VMware Workstation's Hosted Virtual Machine Monitor. Proceedings of the2001 USENIX Annual Technical Conference, Boston,MA,USA,2001.Availableat http://www.vmware.com/vmtn/resources/530.

[22] TU Dresden. The L4 μ-Kernel Family. http://os.inf.tu-dresden.de/L4/.

[23] Vasanth Bala and Evelyn Duesterwald and Sanjeev Banerjia. Dynamo: A Transparent Dynamic Optimization System. Proceedings of the ACM SIGPLAN 2000 Conference on Programming Language Design and Implementation (PLDI), Vancouver, British Columbia, Canada, 2000.

[24] Takahiro Shinagawa and others. BitVisor: A Thin Hypervisor for Enforcing I/O Device Security. Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual Execution Environments (VEE '09), Washington, D.C., USA, 2009.

[25] Condor manual 7.4.4 pdf

http://www.cs.wisc.edu/condor/.

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

[27] Execució de treballs sobre màquines virtuals personalitzades -Pau.

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

[29] An Introduction to the Condor HTC

Framework S.Hosking Parallel Programming, 159735, Massey University

May 2009

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

[31] IBM Red book Introduction to Grid Computing 2005

[32] Above the Clouds: A Berkeley View of Cloud Computing

Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz,Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia UC Berkeley Reliable Adaptive Distributed Systems Laboratory http://radlab.cs.berkeley.edu/

February 10, 2009.

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

[34] Cloud Architectures June 2008 Jinesh Varia Technology Evangelist Amazon Web Services (jvaria@amazon.com)

[35] BECHTOLSHEIM, A. Cloud Computing and Cloud Networking. talk at UC Berkeley, December 2008.

[36] CARR, N. Rough Type [online]. 2008. Available from: http://www.roughtype.com.

[37] HAMILTON, J. Perspectives [online]. 2008. Available from: http://perspectives.mvdirona.com.

[38] CHENG, D. PaaS-onomics: A CIO's Guide to using Platform-as-a-Service to Lower Costs of Application Initiatives While Improving the Business Value of IT. Tech. rep., LongJump, 2008.

[39] RANGAN, K. The Cloud Wars: $100+ billion at stake. Tech. rep., Merrill Lynch, May 2008.

[40] SIEGELE, L. Let It Rise: A Special Report on Corporate IT. The Economist (October 2008).

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

[42]http://etutorials.org/Linux+systems/cluster+computing+with+linux/Part+III+Managing+Clusters/Chapter+15+Condor+A+Distributed+Job+Scheduler/15.1+Introduction+to+Condor/

[43]https://www-auth.cs.wisc.edu/lists/condor-users/2009-June/msg00075.shtml

[44]http://www.mppmu.mpg.de/computing/condor/condor-V6-Manual/node14.html

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

[46] http://searchservervirtualization.techtarget.com/tip/Paravirtualization-explained

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

[48] An elasticity model for High Throughput Computing clusters

Ruben S. Montero, Rafael Moreno-Vozmediano_, Ignacio M. Llorente Distributed Systems Architecture Research Group (dsa-research.org), Dept. de Arquitectura de Computadores y Automática, Universidad Complutense de Madrid,28040 Madrid, Spain.

[49] Dynamic Provision of Computing Resources from Grid Infrastructures and Cloud Providers Constantino Vazquez Eduardo Huedo Ruben S. Montero Ignacio M. LlorenteDepartamento de Arquitectura de Computadores y Automática Facultad de Informatica, Universidad Complutense de Madrid, Spain.

[50] Condor custom ClassAds

https://nmi.cs.wisc.edu/node/1469