A Personal Platform for Parallel Computing in the Cloud

Bruno Filipe Paredes Macedo

Thesis to obtain the Master of Science Degree in

Engenharia Informática e de Computadores

Examination Committee

Chairperson: Prof. Ernesto José Marques Morgado
Supervisors: Prof. João Nuno Silva
Prof. Luís Antunes Veiga
Member of the Committee: Prof. David Martins de Matos

October 2013

Abstract

There is an increasingly number of users with demanding computational requirements. A lot of jobs require the processing of large amounts of data. Processing this jobs in user's personal computers is practically impossible due to the time it would take to complete. There are job manager's applications, like Condor or Apache Hadoop that can be used to create infrastructures that give users the possibility of remote and parallel execution of such jobs. However users have to gain access to those infrastructures in order to execute their jobs. Also, in these infrastructure, users may not have the execution environments needed to execute their jobs, i.e. the software needed to run their programs. Users are limited to the execution software that exist on those environments.

In this work, it is shown a system, called Virtual Machine Manager that provides the ability to deploy a number of virtual machines across a pool of hosts. Besides the creation of virtual machines it has the automatic installation of software packages in those machines and an optimization mechanism that allows a faster provision of virtual machines. It is also shown a possible integration with a Condor infrastructure. The developed system proved to be a good solution to remove some limitations on the execution environments that can prevent user to execute their jobs. The deployment of the virtual machines revealed to be a fast process that can inclusive allow the usage of other resources that aren't an option without virtualization technology.

Keywords: virtualization, parallel execution, job managers, computing infrastructures, execution environments

Resumo

Existe cada vez mais um maior número de utilizadores com elevados requisitos computacionais. Muitos trabalhos requerem o processamento de grandes quantidades de dados e o processamento destes trabalhos nos computadores pessoais dos utilizadores é praticamente impossível devido ao tempo que demorariam a completar. Existem aplicações que gerem a execução de trabalhos, como o Condor, permitem criar infraestruturas para possibilitar a execução paralela de trabalhos. Os utilizadores necessitam de ter acesso a este tipo de infraestruturas para conseguirem executar os teus trabalhos. Mesmo utilizando estas infraestruturas os utilizadores podem não ter os ambientes de execução necessários para correr os teus trabalhos, ou seja, pode não existe o software necessário os executar. Os utilizadores estão limitados ao software existente nos ambientes de execução das infraestruturas. Neste documento é apresentado um sistema chamado Virtual Machine Manager, que oferece a possibilidade da criar um dado número de máquinas virtuais espalhas num conjunto de recursos. Para além da criação de máquinas virtuais, também possibilita a instalação automática de software e um mecanismo de optimização do processo de criação das mesmas. Neste documento também é mostrado uma possível solução para a integração desde sistema com uma infraestrutura Condor. O Sistema desenvolvido provou ser uma boa solução para remover algumas limitações que possam existir nos ambientes de execução e que venham a prevenir que os utilizadores executem os seus trabalhos. O processo de criação de máquinas virtuais revelou-se rápido e eficaz e pode inclusivamente permitir a utilização de outros recursos que sem isso não poderiam ser utilizados.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The advancement of science and technology makes the computational requirements much higher since users have more data to be processed and the number of users processing data is much bigger. Processing large amounts of data is now common in professional and domestic jobs. Rendering images, making digital models or photo enhancements are being made on everyday bases whether for work or leisure. With this, the range of users with remote and parallel execution of jobs has expanded and their computational requirements are constantly increasing. However, together with this new user requirements, hardware technology is also advancing and computers are also more powerful and thus increasing the computational power available.

In recent years there has been a significant improving on distributed systems, as we have viable systems that provide to anyone the deployment and execution of parallel tasks on remote computers. Although there are solutions which allow parallel execution in distributed systems, these solutions are not always available to users. This phenomenon occurs because that most current systems require an user experience, like programming or parametrizations, that most of common users don't have or institutions links with the owner of the infrastructure.

With the increasing number of personal computers available today, there is much processing power unused and the ideal would be to use these personal computers as processing nodes, along with other larger more diffuse distributed computing infrastructures (clusters, clouds, etc.) to solve such problems.

If we use a tool that allows an easy deployment of jobs in multiple environments, it would greatly improve the execution time and in some cases reduce the associated costs. Users who have contracted services to compute jobs, could take advantage of their local/available unused processing resources to reduce the cost in such services.

Due to the high demand of such jobs and the fact that they are more common, computing services emerged to dealt with these demands. The goal is to provide to all users, a simple and effective way to compute their work. These services make use of high performance computing infrastructures which otherwise are not accessible to common users. This kind of services can be found as Cloud Computing Services like Amazon EC2[1].They have an unit minimum payment (usually an hour) and as example, if the user could perform the tasks that take less than the minimum unit in another resource, they would not be wasting paid processing power and would reduce overall costs. On the other hand, common users who only use their personal computer to process these jobs, could easily use other available resources (LAN computers, institutional clusters or if needed contracted services) to decrease execution time.

With the increase need of processing power and the spread to a wider group of users, the long execution time of these jobs became a problem. There are a few ways to improve the performance of such jobs and match users demands, being one of them the use of infrastructures like super-computers and institutional clusters. However access to these resources is restricted and thus limited to a set of users that have some relationship with the organization that owns the infrastructures. Furthermore there are users in institutions or owners of clusters that do not take advantage of them.
In all cases it is necessary to use tools that allow the execution of parallel jobs on remote computers. For example, Bag-of-Tasks problems are made by a set of independent tasks with no need for communication between tasks, each node only needs to execute its task and return the results. So, to compute such jobs is necessary to spilt them into multiple tasks. A set of independent tasks may be obtained by using systems such *SPADE*[1] or *HTCondor*[2]. The first splits jobs that fit in the Bag-of-Tasks problems into independent tasks and the former allows parallelization of compute-intensive jobs. Such systems together with an environment with the necessary requirements (operating system and software) can easily execute independent tasks efficiently.
Depending on the infrastructure, to get the necessary environment, mechanisms to supply on-demand creation and execution of virtual machines can be used. After the set up, with the required environment, each virtual machine can contribute too the problem by providing computation cycles.
These solutions can indeed provide more options to get enough processing power to solve jobs in optimal time, however they do not take advantage of all execution environments and are to complex for some users. Solutions developed for these intensive jobs don't usually include execution

[1] http://aws.amazon.com/ec2/

in multiple environments like the use of clusters, personal computers and cloud infrastructures, so users can not use the resources they have available. As for the systems that include execution in some environments they require more knowledge than of the normal user.
A large number of users are left without viable options to execute this kind of jobs. They either lack the knowledge to use parallel computation systems or they lack the access to all available resources (from personal computers to dedicated infrastructures).

As mentioned previously, new classes of users are requiring fast processing data computation whether for work or leisure [1]. Some of these users have limited parallel programming knowledge or almost none, some have a great pool of resources while others don't. Due to the difference in knowledge and resources available, as shown in Table 1.1, it is necessary to take into account different types of users. Will be considered four types of users:

- Expert Users
- Programmers
- Regular Users
- Common Users

Users	**Computational Demand**	**Resources Available**	**Knowledge**
Expert	High	High	High
Programmers	Medium/High	Medium	Medium/High
Regular	Medium	Low/Medium	None/Low
Common	None	Low	None

Table 1.1: User's Computational demands, resources and knowledge.

Expert Users are those with experience and knowledge in parallel programming likely professionals in the area of parallel programming and usually have access to a large set of resources like institutional clusters or even High Performance Computing Infrastructures. As they have the knowledge and resources to compute demanding jobs, such users don't have much interest to this work.

Programmers have programming knowledge that may or may not be specific parallel programming and usually have available some amount of resources. Typically, this type of users are college students or researchers working in areas like statistics, computer science, physics or computational biology and have to perform heavy computations and simulations to be done. They may have personal clusters or even limited access to institutional clusters but the amount

of computation power needed to process their work, in optimal time, in most cases exceeds the computation power available in their resources.

Regular Users are the new emerging type of users that weren't included in the target population for distributed computing solutions. They have the ability to use advanced features and programs which are beyond the abilities of common users but don't have programming knowledge neither sufficient resources to optimal execution of distributed tasks. They are from distinct areas that don't focus on computer science but they need to execute heavy computational jobs like processing batches of images or video rendering. As example they can be professionals or hobbyist in areas like photography or 3D design.
Enthusiasts that pursue this types of activities in their spare time, the so called hobbyist, should not be neglected. Due to technological advancement, the difficulty and knowledge needed to perform these tasks as greatly decrease, thus increasing the number of such users.

Common Users are the normal end-users with no special knowledge, no computing resources and no computational needs. As they do not need to process data they are of no interest for this work.

There are some details in the existing systems for parallel execution that need to be addressed in order to bring more and different type users to parallel computing.
One of the issues is the availability of infrastructures. Users may not have access to the existing infrastructures that they need to execute their jobs. When they do most of them have complex configurations that must be done in order to run the jobs or create the necessary environments for its executions. As for the execution environment, the software needed to execute the jobs is not always available and depending on the type of infrastructure/environment, users may not have the permissions to install the needed software. Even if they do, they have to manual install it, which is a very time consuming task.

1.1 Proposed solution

The proposed solution is based on the fact that users don't always have the execution environments needed to execute their jobs. The solutions consists on a system to provides an easy creation of virtual machines and the automated installation of software in those machines to allow the parallel execution of jobs. We further propose how such environments can be integrated to existing infrastructures.

The use of virtualization, will allow users to use resources that weren't possible without virtualization. For example, if the user wants to execute jobs that require Linux operating system but their personal computers only have windows installed they probably would not use them. Even if they do, they need to change operating system or use dual-booting techniques and both options are lengthy tasks.

The on-demand installation of the software will give users the necessary tools to execute their jobs. When users are using some type of cluster infrastructure they are limited to the existing software on the cluster. Even if they aren't, they must ask for its installation to the cluster owners or if they own the cluster or have permissions to do it, they must install it on nodes. In both cases the software needed will not be available immediately. The solution will allow for users to ask the installation of software on-demand when they are submitting their jobs, removing the issue of the software availability.

Since users can ask for software installations to create their execution environment, the solution also includes the optimization in the creation of such environments. In a large environment there are a lot of users in need of the same type of software to run their jobs. When using the on-demand installation of software there will be multiple requests for the same or similar sets of software by the users. This solution will use the users requests to create templates pre-installed with software that will allow a faster response to future users requests containing that software packages. These templates are created based on users requests and these requests will be made to the existing nodes on a infrastructure, which means that different templates will exist in different nodes. The system should allow, if possible, users to use nodes that already have templates containing their desired software.

Chapter 2

Related Work

In order to develop the proposed solution it is necessary to know where and how users can run their jobs in a parallel execution and what can we use to improve that execution. Here it will be describe the current computing environments, job managers and virtualization software options.
Computing environments or physical infrastructures that may be available to the user. The ideal solution should include all or most of the existing infrastructures for users to use all the resources available to them.
Job managers are applications that allow a parallel execution of jobs and are an important subject for this work. It is required to know what are the current options for users to execute their jobs and the requirements of each option.
Since one of main objectives of the proposed solution is to use virtualization technology to create the environments for execution of jobs, it is analysed some virtualization software options that can be used. The information about them and their features will help make the choice that best fits the solution.

2.1 Hardware environments

In the past, environments with processing power to execute data processing or simulations were only available in dedicated infrastructure found in data centers. These type of infrastructures were not within reach of everyone due to its high cost. Nowadays, due to technology evolution, we can create or obtain an environment with a high computational power with a much lower cost. In addition to these infrastructures there are services providing computation cycles to an almost unlimited scale. Of course these services have the associated cost, in which more processing

power means a greater cost. However we can obtain processing power in other sources like local clusters or even personal computers that are available to all users. If merged, these multiple environments make a large pool of resources to execute remote jobs in a efficient and cost-effective manner.

The following hardware environments will be taken into account:

- High Performance Computing Infrastructures or Institutional Clusters
- Personal Clusters
- Cloud Computing
- Personal Computers in the Internet
- Multiprocessors

2.1.1 High Performance Computing Infrastructures

High Performance Computing Infrastructures and Institutional clusters are dedicated and specialized infrastructure composed by several nodes with high processing power, high speed network links and hundreds gbytes of storage. Made by several medium/high-end servers with state of the art components, from multiple processors, huge amounts of memory and advance storages that are are connected through high speed links, like fiber channels, with raids (redundant array of independent disks) technology that can improve performance and/or give a certain level of redundancy. These infrastructures focus on high but scalable performance, maximum efficiency, optimized performance and high levels of availability. These environments are the top performance environments when it comes to processing power, making then ideal to execute high demanding computing jobs. However, due to its high cost, these infrastructures are available only to users with high and continuous demand of computation cycles, typically organizations or companies. An Institutional Clusters is much more powerful than a Personal Cluster but less than a High Performance Computing Infrastructures as it may use medium-end servers or less high-end servers than High Performance Computing Infrastructures (HPCI).

With time, these infrastructures are getting closer to the technology found in user's homes. If we look at the most known HPCI historic, that can be found in top500[3] web site, we can observe the technology changes in these infrastructures. For example, the use of a clustered architecture that was 0% in 1997, currently stands at about 80%, contrary to Symmetric multiprocessing (SMP) that was the most used architecture in 97 and as now a share of 0% in these kind of infrastructures. Another example is the use of Gigabit Ethernet, with 31.80% of system share, as

the most used technology for interconnect the systems. It was started to be used only in 2001 where SP Switch technology was the most used.
The interesting fact is that the top used technologies used a few years ago were not available to regular users but nowadays the top used technologies in HPCI is easily acquired or even found in the house of regular users.

2.1.2 Personal Clusters

Personal Clusters are typically made of few low/medium-end computers that join their resources, as processing power, memory and storage, to process demanding jobs. With the available hardware and its cost, is easy to create a small cluster with considerably processing power. With today's network technology, communication between nodes isn't a problem, as sufficient network speed for communication can be achieved with a simple gigabit switch that already exists in the home of most users. However the configuration and management of these clusters is not trivial. Tools and applications for clusters require significant knowledge from the user. Computers are configured as hypervisors, running some virtualization technology, like kvm, xen or vmware, to provide a flexible number of virtual machine that can be created or deleted according to the user needs. These computers can share resources like an external storage but in most personal clusters each machine as their own components. On top of that if user wants to have some high-availability mechanisms common in institutional clusters or HPCI, further and more advance configurations must be made. As example of personal clusters we have the Beowulf cluster[4]. A Beowulf cluster is a set of machines with identical hardware to personal computers working together to make a high-performance parallel computing cluster with the particularity of having a central node to coordination. They have became extremely popular as users can obtain a high performance cluster that can be created at a very low cost.

2.1.3 Cloud Computing

Emergence Cloud Computing has brought new solutions to computing problems. These solutions provide on-demand access to a pool of configurable computing resources like networks, servers, storage, applications and services. Users can install any operating system and software as they desire and the charged value depends on the execution time and on the selected hardware.
According to the definition of Cloud Computing [10] from the NIST[1], a cloud solution has the

[1] National Institute of Standards and Technology

following essential characteristics:

- On-demand self service;
- Broad network access;
- Resource pooling;
- Rapid elasticity;
- Measured service;

Cloud computing architecture can be divided, as shown in Figure 2.1, in multiple layers: infrastructure, platforms and applications [11]. Depending on the desired service is granted access to the corresponding layer. This causes that cloud services to be generally found as the following business models: *Software as a Service* (SaaS), *Platform as a Service* (PaaS) or *Infrastructure as a Service* (IaaS).

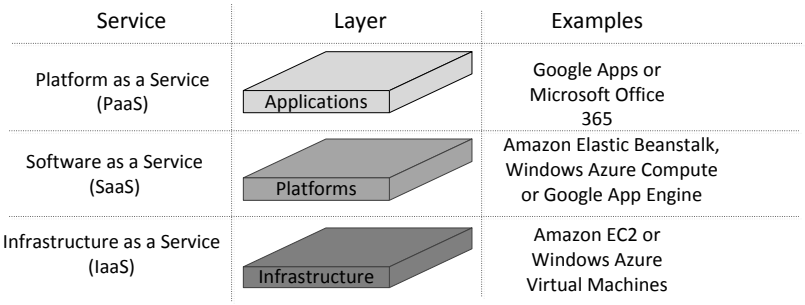

Figure 2.1: Cloud Architecture

1. Infrastructure as a Service: Is the most basic service, where the provider offer infrastructural resources (usually as virtual machines). Users can deploy their operating system images and applications software. They are responsible for their configurations and maintenance. Pools of high-end servers in the internal infrastructure give the ability to scale according to user's demands which are then charged according to the amount of resources used. Examples of IaaS service include Amazon EC2[12] or Windows Azure Virtual Machines[13].

2. Platform as a Service: This service providers an environment that includes operating system, programming languages execution environment, libraries, services and tools. Users

can deploy and develop their software without worrying about hardware resources and environment managing. Examples of PaaS service include Amazon Elastic Beanstalk[14], Windows Azure Compute[15] or Google App Engine[16].

3. Software as a Service: In Software as a Service model, users use the running applications on the cloud infrastructure offered by the provider. These applications are available through an interface which can be accessed via web browser or a program interface. Users don't have any control over the cloud infrastructure and platform. Examples of SaaS service include Google Apps[17] or Microsoft Office 365[18].

2.1.4 Grid

Grid [5] [6] infrastructures is a collection of remote computers from multiple resources in order to provide computing support for a wide range of applications.
It is a type of parallel and distributed system that allows the remote access to high performance computing infrastructures to users outside the owner organization.
Grid infrastructures are often used to complete complicated mathematical or scientific calculations. However it is necessary for a user, in order to use it, to have an institutional relationship with a grid initiative participant. Grid middleware and protocols were developed mainly to aggregate, manage and share distributed computational resources /mainly clusters).

2.1.5 Personal Computer in the Internet

With the advancement of science, technology found in personal computers is similar to the existing infrastructure of high performance mentioned above. Personal computers are multiprocessor with high processing power, high amounts of memory and fast Internet network connections that are rarely fully used by its user. We can take advantage of this unused processing power and execute tasks from high demanding jobs like those from the Bag-of-Tasks type in these personal computers. Some users, like hobbyists or students, commonly use their own personal computer to execute heavy computational jobs. The idea of using personal computer in the internet to execute high demanding jobs is not new and there are known projects like SETI@Home[7], BOINC[19] or GIMPS[8] using this idea. SETI@Home[1] is a project from the from University of California, Berkeley that allows internet-based computers to act as volunteers to analyze radio

[1] Search for Extra-Terrestrial Intelligence

signals for signs of extra-terrestrial intelligence and GIMPS[1] is a project from Mersenne Research, Inc and is one of the largest distributed computing projects, and like SETI@Home, it uses volunteers PC's in the internet to search for Mersenne prime numbers[9].

2.2 Job Managers

After having access to hardware infrastructures, it is necessary to deploy jobs on them. Job managers allow users to do a parallel execution of their jobs across a pool of physical resources. Theses resources can vary from a HPCI to a personal computer. In most cases, users have to write a launching mechanism and job managers select the best available hosts to execute the tasks, manage their execution and return the results.

2.2.1 Condor

HTCondor[24] is made by the Center for High Throughput Computing in the Department of Computer Sciences at the University of Wisconsin-Madison (UW-Madison) and is a workload management system for compute-intensive jobs. Condor provides job queueing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management.
The job submission is made through the *condor_submit* command that takes as argument a description file. This file, as showned in Figure 2.2, has the basic information that Condor needs about to execute the job, the name of executable to run, the initial working directory or command-line arguments. Users submit their serial or parallel jobs that is placed into a queue and then system chooses when and where to run them based on a policy always monitoring their progress and then informing the user when its finished.

```
Executable   = foo
Universe     = standard
Log          = foo.log
Queue
```

Figure 2.2: Simple condor job description file

[1] Great Internet Mersenne Prime Search

Condor runs in Grid-style computing environments and allows to harness wasted CPU power from idle desktop workstations and when the machines leaves the idle state the job is migrated to a new machine as the owner of that machine has priority over its use. Users don't need to change their source code neither to make their data files available to the remote nodes or have any kind of authentication on them.

Condor is divided into five main processes: master, node, user, collector and negotiator:

1. The main process (daemon) is the master (*condor_master*), it coordinates the entire system and is responsible for starting all other condor processes or to restart them if they fail.
2. The processes that represents a node (*condor_startd*) is responsible for starting, suspending or stopping jobs and for enforcing the machine's owner policies.
3. The user process (*condor_schedd*) maintains and manage a persistent queue of jobs and contacts available machines to spawning waiting jobs.
4. Collector's process (*condor_collector*) runs in the master node, collecting information from all other processes in the system and responding to queries from them. It contains the pool's of machines/processes database. Each process send a periodic update to the collector.
5. The process corresponding to the negotiator (*condor_negotiator*) also runs in the master node and performs the matchmaking between available machines and jobs, i.e., It gets information from the collector about all available machines and idle jobs and trying to match them.

Those processes are running on the machines with Condor installed depending on the node function. On the central manager node, Condor has the following processes:

- condor_master, condor_collector, condor_negotiator ,condor_startd, condor_schedd.

On all other machines beloging to the pool:

- condor_master, condor_startd, condor_schedd.

As shown in figure 2.2, Condor allows users to choose a universe. The universe defines an execution environment. Condor supports the following universes: Standard, Vanilla,Grid, Java, Scheduler, Local, Parallel, VM. For example:

- The *vanilla* universe is the simplest universe. Condor finds a suitable computer, sends the necessary files (job executable and input), run the job and returns the output.
- *Standard* universe, users have to recompile their programs with a special Condor library and brings the abilities to checkpointing and use remote procedure calls.
- *Parallel* universe allows parallel programs, such as MPI jobs.
- *VM* universe allows execution of VMware, KVM and Xen virtual machines.

The VM universe of Condor allows the use of virtual machines. Users create a disk file that is a pre-installed virtual machine and Condor runs it. The virtual machine itself is the job that will be executed.
When using the VM universe the job description file is slightly different. Instead of a executable there is a disk image and the input, output and error commands do not apply. Users must also specify the *vm_type* field that may be vmware, xen or kvm and the *vm_memory* field that is the memory (in MBytes) of the virtual machine.
The job is completed when the virtual machine is shut down. Users must configure their virtual machines, by the means of a script, to shut down after the job execution, otherwise Condor will not notice that the job is finished.

2.2.2 SPADE

SPADE [1] is a middleware system that allows the multiple remote execution of computing tasks such as Bag-of-Tasks tasks that are usually run on a single computer. One of the main focus of this system is to provide simpler remote execution of jobs to all users, on multiple LAN connected PCs.
To do that, SPADE hides the usual complex configurations of clusters. Selection of the job execution location is made by the system and its hidden from user, configuration of the available applications is easy since users only need to give the locations and identifier of the application, creation of the available pool of resources is transparent as the system is able to discover nodes.
For SPADE jobs are a set of tasks that will be executed on remote computers. Users can easily submit jobs, using a graphical user interface, by stating the code to be processed by the tasks, files that will be processed, the application used and their arguments. The system uploads the files to the nodes, run the specified application with their respective arguments and finally downloads the results. This implies that the remote computers have the necessary environment for the execution of tasks.
In a brief explanation, the system consists, as shown in Figure 2.3, of a *Client* application and a

Daemon. Every computer with the system has both *Client* and *Daemon* running, meaning that all machines are processing nodes. The SPADE *Client* is where the user creates and submit jobs to be executed and the SPADE *Daemon* is where the user defines new applications.

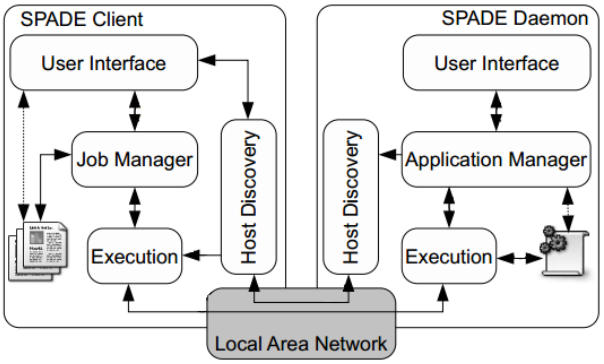

Figure 2.3: SPADE Architecture

The execution flow is the following:

1. The *Daemon* sends information about its available applications to the *Client* that keep records of the available *Daemons* and their applications.
2. When a user submits a job, through the Job Manager module, it must state how many tasks should be used. These tasks are then submitted to the Task execution module that chooses where to execute each task.
3. *Daemon* receives a task from the *Client* and contacts its Application Manager module to get the path to the needed application.
4. The SPADE *Daemon* executes the received task an return the result to *Client*.

SPADE can be used to accelerate, the execution of jobs by using idle remote computers or dedicated personal clusters. For example, a user can make manipulation of images by running it on several computers simultaneously as long as they have the required software installed.

2.2.3 BOINC

BOINC [19],[20] is a successful and best known platform for distribution and execution of parallel jobs on remote computers. BOINC provides communication, client management infrastructure

and data storage. To create a project, the project manager must use C++ or Fortran to develop an application that will execute on the client computers to process data.
By intalling a BOINC server, researchers create and spread their projects to the community that willing donate their personal computer's idle cycles and start processing data with the application downloaded from the server that was previously developed. This projects require solving a complex problem that would be difficult to solve using only the resources available to the project manager. Even though it is simple to install a BOINC server, there are some issues. The project manager must have the necessary programming knowledge on C++ or fortran to make the application that will execute the job and it is necessary to spread the information about project to the community in order to get clients to process the job.
The BOINC software acts like a single program but is a set of programs (figure 2.4).

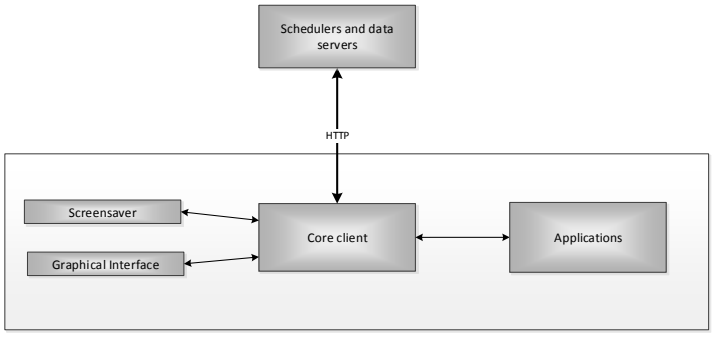

Figure 2.4: BOINC Software

The schedulers and data server programs are installed and manage by the project manager and represent the projects to which users will donate their computer cycles. The core client, the BOINC main application, communicates via HTTP to the schedulers and data servers go obtain tasks and report the results. The core client also runs and controls applications. Those applications are the programs made by the project managers that will compute the tasks. Users can run more than on application if their computers have more than one CPU. The graphical interface allows users to control the core client. They can add projects, suspend or resume tasks, view information about the tasks being executed. The screensaver program runs when users are away from the computer and show graphics corresponding to the project being executed (however, not all projects provide screensaver graphics).

2.2.4 Apache Hadoop

Apache Hadoop [21] is an Apache software Foundation open source project that can provide fast and reliable analysis of both structured data and unstructured data given its capabilities to handle large data sets.
The Apache Hadoop software library is essentially a framework that allows for the distributed processing of large datasets across clusters of computers using a simple programming model. Hadoop can scale up from single servers to thousands of machines, each offering local computation and storage.
Two of the main characteristics of Apache Hadoop are Hadoop MapReduce and Hadoop Distributed File System (HDFS) modules.
The Hadoop MapReduce model is a system for parallel processing of large data sets. As the name implies it uses the programming paradigm MapReduce [22], that allows for huge scalability across a hundreds of servers in a cluster. MapReduce paradigm consists in two functions, map and reduce. The map function takes a set of data and converts it into another set (called map) where elements are broken into tuples (a pair key/value). The reduce function takes the output from a map as input and combines those data tuples into a smaller set of tuples.
The Hadoop Distributed File System [23] is a distributed file system with high fault-tolerant capabilities designed to run on low-cost hardware. It is suitable for applications with large data sets and therefore used in the Apache Hadoop. HDFS allow the connection between nodes contained in a cluster and then access and store data files as one seamless file system. HDFS as some unique features and attributes such as:

- Write-once-read-many model that enables high-throughput access.
- Fault tolerance by detecting faults and applying automatic recovery.
- Data access via MapReduce streaming.
- Portability across heterogeneous hardware and operating systems.
- Scalability to reliably store and process large amounts of data.
- Simple and robust coherency model.

To use Hadoop users have to install and configure Hadoop on the hosts. On Linux and Mac OSX machines are able to install and run it with no additional software beyond Java, however to Windows users Hadoop requires the installation of cygwin (A linux-like environment that runs within Windows). Like BOINC, to create a new project the user must develop a JAVA or C++ program to execute their job. Besides the Java or C++ knowledge, users must be introduced to

the programming style for Hadoop. Their program must use the libraries from Hadoop framework and apply the MapReduce paradigm.
Apache Hapdoop is one of the powerful solutions for computing parallel jobs however to use it, users must have programming knowledge and learn the concepts of behind the HDFS and MapReduce.

2.3 Virtualization Software

Virtualization allows a single computer to run multiples operating system simultaneously on a single computer system. It allows companies to run different services with full isolation on a single server which enables to reduce the cost of managing more hardware and be more efficient in the usage of resources.
A virtualization software allows users to run multiple operating system without requiring new hardware or without using dual-boot or similar techniques. We can run multiple operating systems on a single computer without affecting the guest operating systems. Virtualization software's simulate a real machine to "deceive" the guest operating system into thinking its running on a real computer. Virtualization can be very useful to everyone from an enterprise usage, or a home user usage.
In order further analyse virtualization and it's software options, it is good to know some of the used terminologies:

- **Host:** Physical computer on which the virtualization software is installed.
- **Virtual Machine (VM):** Special environment created for the guest operating system. This environment emulates a real machine behaviour.
- **Guest OS:** Operating system that is running inside the virtual machine.

Virtualization can be classified into three different categories [25] for a particular:

- Full Virtualization.
- OS-Layer or ParaVirtualization.
- Hardware-Layer Virtualization.

In full virtualization, a virtual machine manager runs on the host operating system (just like user applications). This type of virtualization provides virtual hardware that allows the virtual machines and guest operating system to run on top of the virtual machine manager.

When using paravirtualization, various instances of the virtual guest operating system are created where the host and the guest operating system work together to achieve the fastest and most robust performance possible.
In Hardware- layer virtualization, the virtual machine monitor directly runs on hardware that controls the access of the guest operating system to the hardware resources of the host. It is a hight performance and isolation mostly used in virtualization for server technology.

In general virtualization provide several features or techniques that can be useful in multiple scenarios:

- *Multiple operating systems running simultaneously:* to maximize the usage of the real computer hardware, um can run a number of virtual machine on the same computer. Those virtual machines can have different virtual hardware specifications, according the requirements of that virtual machine. This allows the installation of old operating systems that are no longer supported by the real computer's hardware.
- *Easier software installations:* software vendors can provide a virtual machine with the full software configurations to their clients. Clients just have to run the virtual machine without worrying about complex configurations.
- *Testing environments:* a virtual machine be easily be duplicated making them perfect to test environments. If something goes wrong, users can just use another copy of that machine without requiring them to install all the environment again.
- *Disaster recovery:* virtual machines are stored in disk files that can be easly moved between hosts, furthermore they can have their execution paused, resumed, copied at any time. Most of, if not all, virtualization software's have the capability of taking snapshots. Snapshots can be used to save certain states that can be restored.
- *Greener infrastructures:* virtualization can significantly reduce energy costs and hardware. Most of the time, computers use only a fraction of their computational power. We can have several virtual machines running on the same computer, and thereby reducing the associated costs.

2.3.1 VirtualBox

VirtualBox [26] is a open source virtualization software from Oracle. As a cross-platform application it can installed on an Intel or AMD based computer and runs in Windows, Mac, Linux or

Solaris operating systems. VirtualBox is a very simple and powerful tools. It is used everywhere, from home user to datacenters or cloud environments.

Some of the key VirtualBox features are:

- **Portability:** Virtual box runs in multiple operating system and virtual machines can be moved between hosts regardless of their installed operating system.
- **No hardware virtualization required:** Virtual does not need features like Intel VT-x or AMD-V and so it can use older hardware. Intel VT-x or AMD-V (from Intel and AMD, respectively) are a set of hardware extensions for virtualization.
- Guest Additions: A software package that is installed on the the virtual machine to improve performance and to provide interaction with its host. For example, it allows the execution of commands from the host to the virtual machine or the transfer of files between them.
- **Vast hardware support:** Each virtual machine can have a number of different hardware configurations. Each virtual machine can have a maximum of 32 virtual CPU's regardless of how many exist in the host. VirtualBox also as vast hardware compatibility from USB devices, network cards, sound cards, hard disk controllers, etc.
- **Snapshots:** Users can take multiple snapshots of their virtual machines. These snapshots can be used to restore virtual machines to a specific state. These snapshots can be created or deleted even when the virtual machines are running.
- **VM groups** VirtualBox allows users to organize their virtual machines in groups to do a better management of those machines. Users can order the startup or shutdown to a group of virtual machines.
- **Command line interface:** Besides its graphical interface, it is provided a tool to control all the VirtualBox functionalities through command line.

2.3.2 Xen

Xen [27] is an open-source from the Xen Project and is a type-1[28] virtualization software, i.e, it runs directly on the host's hardware to control the hardware and to manage the guest operating system. Xen is widely used as server virtualization, infrastructure as a Service or desktop virtualization.
Xen run

Some of the main Xen feature are:

- **Small footprint and interface:** Due to its small footprint and interface, Xen is more secure and robust than other alternatives.
- **Multiple host operating systems support:** Xen can be installed in Linux, NetBSD or OpenSolaris.
- **Driver isolation:** Xen contains the used drivers so when they crash they can be restarted without affecting the rest of the system.
- **Paravirtualization:** Besides Hardware-Layer Virtualization it allows the use of paravirtualization to run guest system.

2.3.3 KVM

KVM (Kernel-based Virtual Machine)[29] is a virtualization software for Linux from Red Hat. KVM is included in the main Linux kernel and consist of a loadable kernel module that provides the core virtualization infrastructure and require a CPU with hardware virtualization extension (Intel VT-x or AMD-V).
KVM has native support for any guest operating system, it uses hardware-based virtualization which eliminates the need to modify guest operating system. By add KVM as a module to the kernel itsefl, allowed every virtual machine to became a standard Linux process that can operate at near bare metal speeds as a para-virtualized instance.

Some of its main feature are:

- **Qemu Monitor Protocol:** a JSON-based protocol that allows applications to control QEMU instances.
- **CPU hotplug:** CPU adding on the fly.
- **PCI hotplug:** PCI devices adding on the fly.
- **Virtual machines migration:** KVM allows the live migration of virtual machines between hosts.
- **Virtio devices:** Virtio[30] is the main platform for IO virtualization in KVM.
- **snapshots:** KVM allow its users to take snapshots of their virtual machines.

2.3.4 VMware Workstation

VMware workstation [31] is a virtualization software from VMware virtualization. VMware is considered the market leader in virtualization technology with its strong product features. VMware Workstation has a very intuitive interface and allows integration with other VMware products like VMware vSphere.

- User friendly interface.
- SSD Pass through.
- USB 3.0 support.
- Great 3D graphics support. Supports DirectX 9.0C and OpenGL 2.1 in both Linux and Windows virtual machines.
- Tablet sensors support. VMware workstation provides a virtual magnetometer, accelerometer and gyroscope in a virtual machine.
- snapshots.

2.3.5 Comparison

All the virtualization options mentioned above have their advantages and disadvantages. While important, the performance will no be a decisive factor in the choice of the virtualization software. Better performance usually brings more complexity or cost to the software. As we seek to bring more users to the parallel distribution, is preferable that the software is simple to use and with and the lowest possible cost.

Xen, being around for some time, is a robust and secure option that is intended to be used n an enterprise environment. The other three options are intended for personal or small-medium business environments. VMware has other options when it comes to a enterprise environment, VirtualBox has a commercial license that adds enterprise features and support and KVM has options like Redhat enterprise virtualization that ensures more stability and tools for critical virtualization systems.

The following table shows some comparisons between them:

The ease of use is an important feature and only VirtualBox and VMware Workstation have a graphical interface as they designed to be more user-friendly. KVM and Xen can have graphical

	VirtualBox	**Xen**	**KVM**	**VMware**
Virt. type	Full Virtualization	Hardware-Layer	Hardware-Layer and Paravirtualization	Full Virtualization
Pricing	free	free	free	paid
Host OS	Windows, Linux and Mac OSX	Linux	Linux	Windows and Linux
Gui	yes	no	no	yes
Cli	yes	yes	yes	yes

Table 2.2: Virtualization software comparison.

interfaces through installation of external software or additional modules. All the mention options have an command line interface, that is important to allow the integration with the proposed solution. Another important feature is the supported host operating system, VirtualBox allows its installation in the three most common operating system (Windows, Mac OSX and Linux) while KVM and Xen can only be installed on Linux. VMware has an extremely friendly interface and can be installed in both Windows and Linux, however it is a proprietary software that has an associated cost.On the other hand, VirtualBox brings together the easy usability, multi-platform support and low-cost.

Chapter 3

Architecture

The main objective of this work is to allow users to execute their jobs in a parallel computing environment, figure 3.5. To do a parallel execution of jobs, users need to have access to certain infrastructures,tools and resources. Or they have access to a parallel computing environment or they must, when have the means, create one.
The creation of a parallel computing infrastructure requires know-how and may not be to the reach of all users. Users have to install and configure a job manager like Condor, Apache Hadoop what is not a trivial action. Even when users have the know-how, it is there is an additional "problem". For every different type of job they have to configure the execution environments.
When using some existing infrastructure, usually associated to some institution or enterprise, users are limited to provided execution environment existing in that infrastructure.

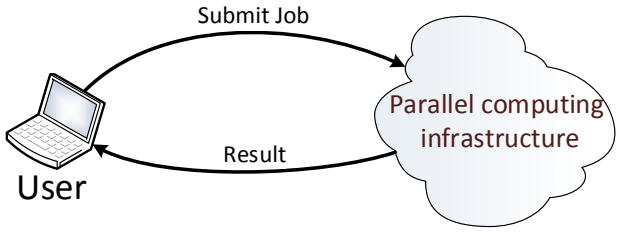

Figure 3.5: Job execution

The ideal solution should allow users to accomplish this execution without worrying about computational resources or environments configurations. Ideally users submit their jobs into a parallel computing infrastructure and receive the results. Removing any kind of difficulties users may find on these infrastructures will help in bringing more and different types of users to the parallel computation.

3.1 Virtual Machine Manager architecture

Here is presented the Virtual Machine Manager architecture to achieve the proposed objectives. This architecture is a virtual machine manager (VMM) that provides an automated way to create and deploy virtual machines, that will allow an easy and fast set up of the required environments to the parallel execution of jobs. Depending on the job being executed the software used to compute it changes. Therefore, the environment in each job that is going to be computed has to meet these changes. However the preparation or modification of these environments can be very time consuming. With the virtual machine manager the virtual machines will be prepared on-demand with the needed software.

VMM can act together with a job manager software, like HTCondor or SPADE, to produce an automated environment that not only allows the distributed execution of jobs but also manages (creates and deploys) the environments needed for these executions. The general view of VMM architecture is shown in figure 3.6.

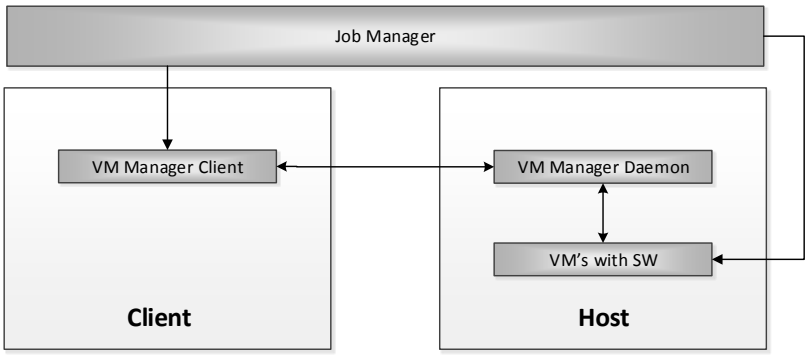

Figure 3.6: Architecture

On the client side, the user or job manager software tells the virtual machine manager how many machines are needed and which software has to be installed. The virtual machine manager will be responsible for the creation and deploy of those virtual machines in the available hosts. Virtual Machine Manager consists in two programs, the client, referred as VMMC, and the VMM daemon, referred as VMMD. As presented in figure 3.6, the VMM client runs on the user's computer or on the job manager machine and the VMMD on the hosts (resources) that will run the virtual machines.

From an external point of view, i.e. from a user or job manager application perspective, Virtual Machine Manager allows them for:

- Deployment of a given number of virtual machines across the available resources.
- Automatic installation of the named software packages into the virtual machines.
- Installation of a custom software or package into the virtual machines.
- Shutdown of virtual machines.

The input received by the Virtual Machine Manager client can be divided into two actions. Shutdown of virtual machines or deployment of a virtual environment. From the VMMC it is possible to order the shutdown of running virtual machines. This allows for a job manager application to shutdown the virtual machines after the jobs are completed.
For the deployment of a virtual environment, the VMMC program receives as input the desired environment to be created, i.e., the number of virtual machines to create, the software packages needed or the path to a custom software package. It will be passed to the virtual machines the software packages to be installed or uploaded a custom software.

From an internal view, VMM will have one more important task: the virtual machine migration. This process is not supposed to be visible to the user or job manager. When someone manually closes a running virtual machine on the host, VMM will migrate that virtual machine to another host. From the point of view of the user/job manager nothing happens, except the fact that the job being executed is going to take more time to be completed due to the migration process time.

3.1.1 Hosts management

The number of virtual machines allowed to run in each host, depends on their hardware specifications, mainly their processing power, memory and storage capabilities. The Virtual Machine Manager will take this into account through the VMMD, when creating the virtual machines. Besides the hardware limitations of the hosts, the Virtual Machine Manager will also take into account the optimization on the virtual machines creation.
As will be explained more detailed below, hosts will save previously virtual machines installations as templates. It's common for the same set software packages to be requested multiple times and having a template pre-installed with that software packages will allow a faster virtual machine creation. So when the Virtual Machine Manager has to create a virtual machine, it must take into account the existing templates in each host.

On a creation request, Virtual Machine Manager starts by connecting to the hosts requesting the creation and installation of those virtual machines. To decide which of the hosts should connect

first, the Virtual Machine Manager uses a priority system. This system uses the information about the existing templates in each host that is stored on the client.
Every time the VMMC makes a connection to the VMMD, it receives information about that host, i.e. existing base disks and the installed software they have. The next time VMMC is called, it will use this information and the information entered by the user, to calculate a priority number that will define the order in which VMMC will contact each host. If the VMMC as no information about the host, that host will have the lowest priority number. In other words, VMMC will attempt to create a virtual machine on a host that already as a template with the requested software (3.7)
Once a connection to the host is established, VMMD informs the client of the maximum number of virtual machines, based on the available hardware resources that its host can create. Virtual Machine Manager daemon uses information about the current running virtual machines, the host storage capability and its hardware configuration (hysical memory and number of CPUs) to calculate how many more virtual machines its host can have.

When it's not possible to create all the desired VM's in that host, VMM orders the creation of the remaining in the other hosts, according to their priorities.

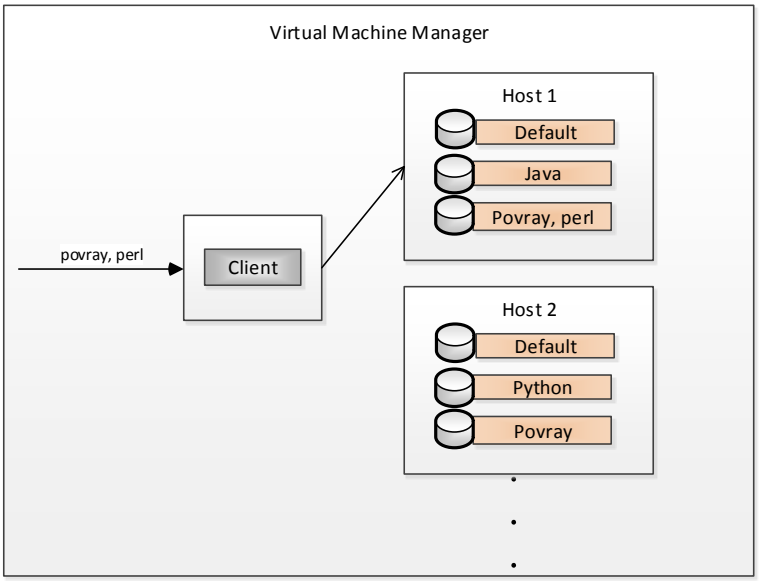

Figure 3.7: Hosts priorities

For example, if we want to create three virtual machine with the software packages *povray* and *perl*, the VMMC will contact the host with greater priority, if that host only has space for two virtual machines it will contact the second higher priority host to create the last one. If the hosts don't

have any template with the same software packages, VMMD will use the template that as more software in common with the request.

The Virtual Machine Manager client connection process to the hosts can be viewed in the follow figure 3.8:

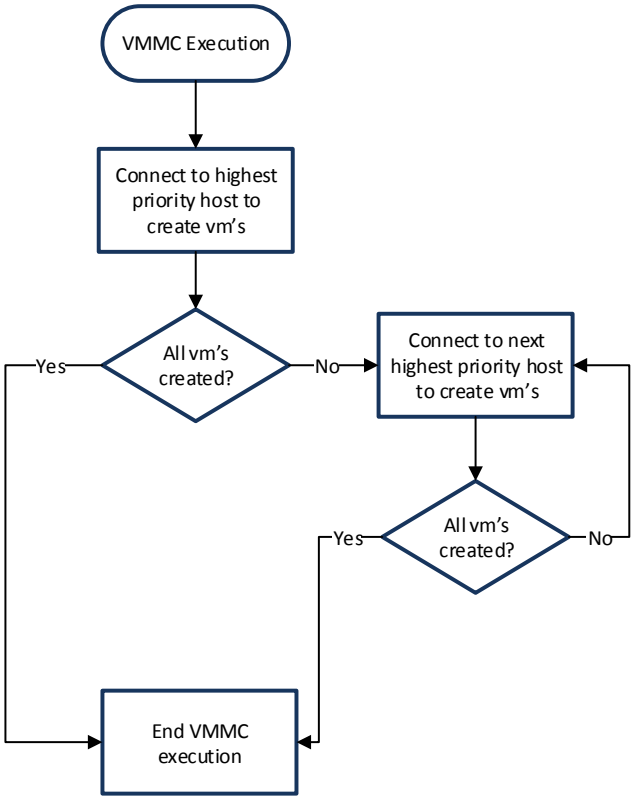

Figure 3.8: VMMC execution

VMMC will try to connect to all host until all the requested virtual machines are created or until its impossible to create more virtual machines in all available hosts. When the Virtual Manager client can create the virtual machines on the hosts with higher priorities it means that the all process will be faster as the creation of the virtual machines in that hosts will be faster.

3.1.2 Virtual Machine management

As we just see, the Virtual Machine Manager is responsible not only for the creation of virtual machines but also manage in which host will they be launched. As shown in figure 3.9 the virtual machines are created across the available hosts, depending of their hardware resources

and priority. For each job, VMM can launch multiple virtual machine in different computational resources (hosts), and if necessary a user or job manager can order the shutdown of all virtual machines associated with that job without the need to know where these virtual machines are. The virtual environments created from Virtual Machine Manager will allow users to execute their jobs without being limited to a particular software environment.

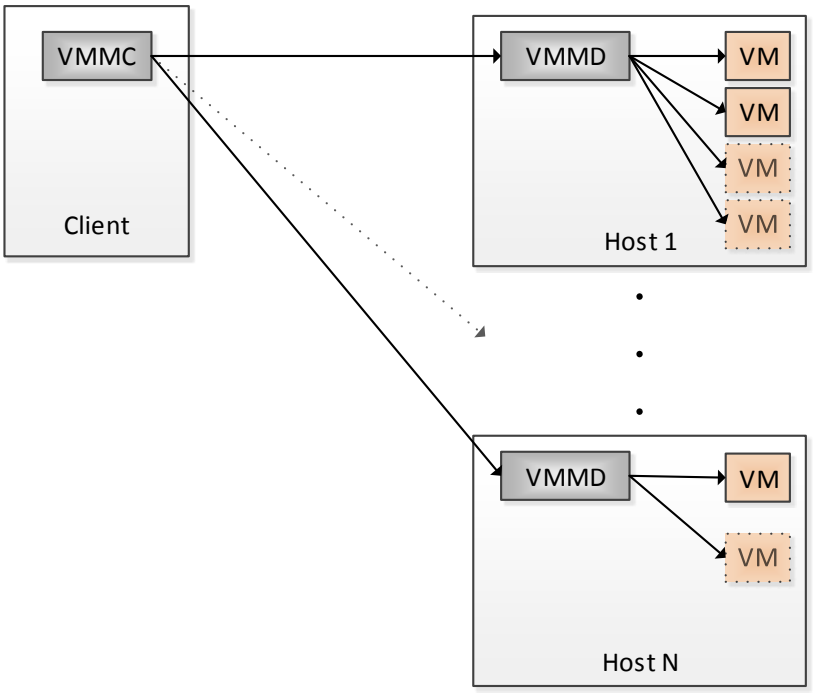

Figure 3.9: Virtual machine deployment

In this section, it will be explained how is the management of virtual machines is done.
The communication to the hosts will be made through the virtual machine manager daemon, VMMD, that be installed in the hosts.
The VMMD is a daemon running on the hosts that is waiting for requests from the VMMC. This daemon is responsible for creating the virtual machines and installing the software packages as well to manage the allowed number of running virtual machines.
Hosts have a limited number of running virtual machines based on their resources, namely, cpu cores, memory and storage. VMMD will, based on current number of existing virtual machines, decide if this host can create or run more virtual machines and replies to the VMM program accordingly. This reply is based on the maximum number of virtual machines disks allowed (defined by the available storage) and the maximum allowed number of running virtual machines (limited by the number of cpu cores and host memory).

To create virtual machines, hosts have a default virtual machine disk (called default base disk), pre-installed with only the operating system and the necessary tools for the VMM system to work, that will be used as a template to create virtual machines.

When creating a virtual machine, VMMD will create it based on one base disk plus the new software packages. The newly created disk will be saved to serve as a base disk to the creation of new virtual machines.

The creation of a new disk consist in the cloning process of an existing disk file (and therefore it will contain all the installed software that was in that disk file) and in the installation of the reaming software. This will create a new base disk.

After this process is completed the new base disk is cloned. This new clone will be used as the virtual machine disk and the original will be saved to be used as template. During the disk creation process, some snapshots are taken. After every cloning process and after the installation of new software is taken a snapshot. Those snapshots contain the differences between the base disk and the newly created disk. This will be used to allow the virtual machines migration, explained below.

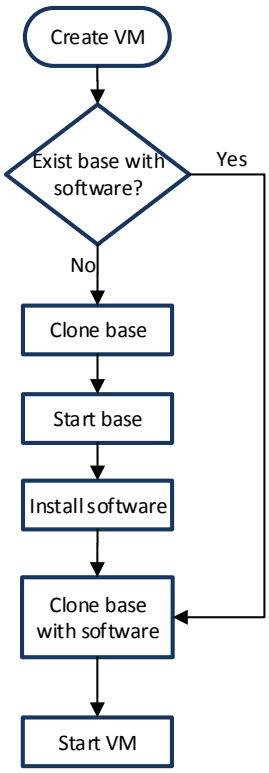

Figure 3.10: Virtual disk creation process

When the daemon receives the request to create a new virtual machine there are two scenarios (figure 3.11):

- If there is a base disk containing the needed packages, the new virtual machine will be created from that base.
- If not, Virtual Machine Manager daemon will create a new base disk with the software packages and then the virtual machine from the just created base.

In both cases, after the process the new virtual machine is started.

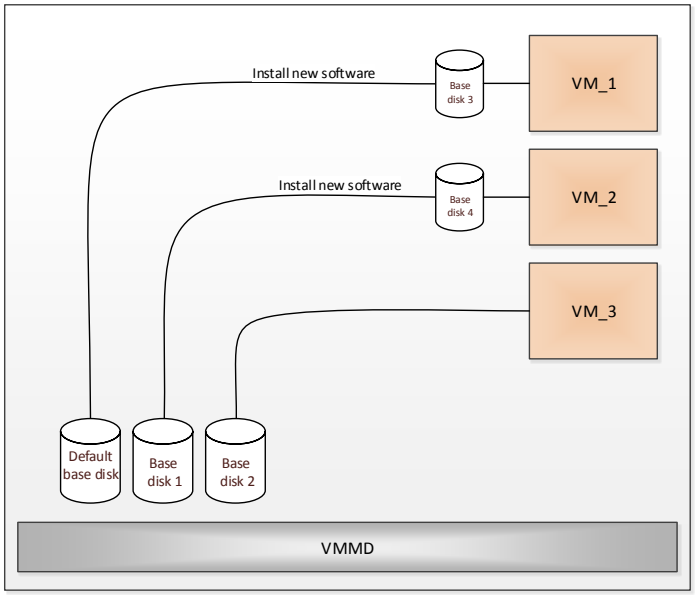

Figure 3.11: Base Disks creation

On the second case, where a new base disk is created, the system will choose the most suitable option to serve as template to the new base disk (the default base disk or another base disk). The most suitable option is based on the software packages wanted by the user and the installed on the existing disks. Virtual Machine Manager daemon will find which base disk that has more software in common with the request.

As part of the virtual machine management, it is given the option to shutdown virtual machines. This feature allows, for example, terminate all virtual machines associated with a job that finished (figure 3.12). When the VMMC receives the order to shutdown a virtual machine, it will contact all the hosts running that virtual machine to order the shutdown.

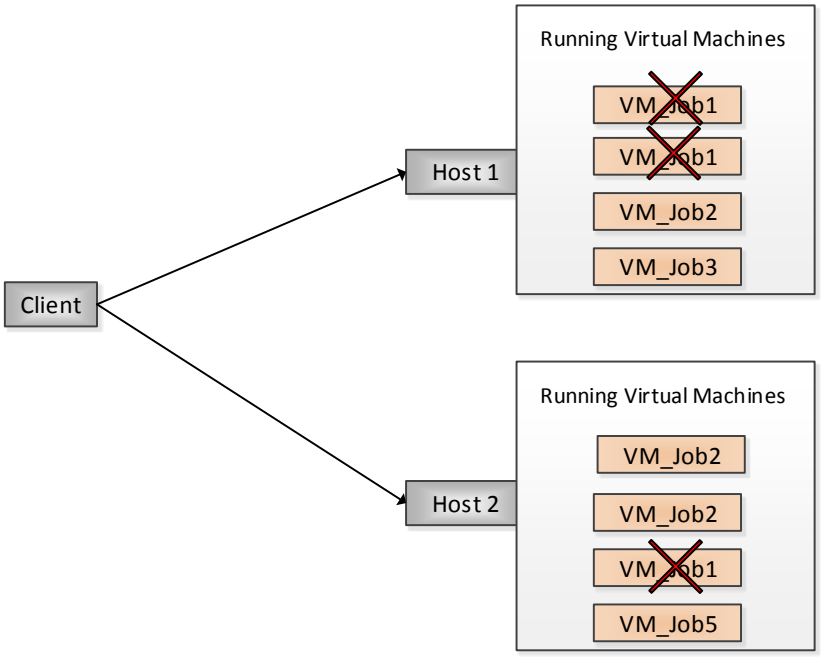

Figure 3.12: Virtual machines shutdown

Virtual machine manager daemon monitors every virtual machines execution, so that when a virtual machine is turned off it can take action. There are two actions that can happen when a machine is powered off:

1 - If the virtual machine was powered off by order of the Virtual Machine Manager client, it means that the tasks that were running on the virtual machine ended or was canceled and VMMD deletes the virtual machine.

2 - If the shutdown was was done by a user, ie someone manually closed the virtual machine, means that there is the possibility that the running tasks weren't finished. In this case, the VMMD will migrate the VM to another host.

When the VMMC gives the shutdown order of a virtual machine to a VMMD, after the shutdown, the virtual machines is deleted. This is done to optimize the storage capabilities on that host, otherwise the unused and powered off virtual machine would eventually take all the available space to store virtual machines. There is no need to save the powered off virtual machines as they will not be used again. Their job was completed and because it was created (if did not

already exist) a base disk with the same software of that machine, there is a fast away of deploy virtual machines with the same software.

3.1.3 Snapshot management

The set of snapshots that is sent, contains all the differences between the default base disk and the base disk used to create the virtual machine. As shown in figure 3.13, every base disk except the default have at least two snapshots.

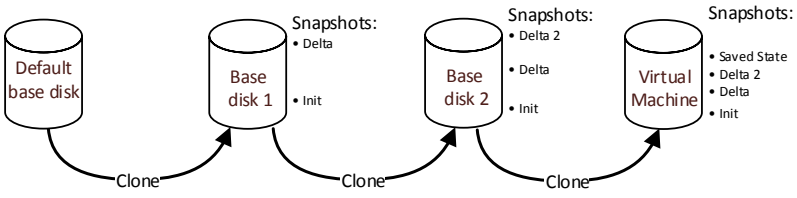

Figure 3.13: Disk snapshots

The idea beyond the multiple snapshots system is the following:
The *initial* snapshot is taken before any software installation and marks the state when the disk is equal to its parent (the base from where originated from). After the software installation is taken a snapshot, *delta*. This snapshot will contain all the differences between the initial and the current state, i.e. the installed software. If the base disk that originated a virtual machine was made from the default disk, we only need the default disk and the *delta* snapshot to create a similar VM.
If a base disk was created from another base disk that wasn't the default, we need the delta snapshot from all parents base disks. This doesn't mean that we have to get the snapshots from the parents, this would require to log all the inheritances and respective base disks parents and if one of those disks were deleted it would be impossible to achieve our goal. Therefore, when cloning, all disks inherit the snapshot of its parent.

3.1.4 Virtual Machine migration

The migration of virtual machines is accomplished by sending snapshots and the saved state of that machine to another host. Hosts will create a new virtual machine based on the default base disk with the received snapshots and the saved stated. When the machines are powered up, they

resume exactly where they were before the shutdown.
It is important to understand that the new host to where the virtual machine is being migrated, may not have the same base disk that was used to create the virtual machine. Base disks are created from requests, if the new host never had the same request, it will not have that base disk. When migrating a virtual machine we ill use the default base disk, the received snapshots and saved state.
When migrating a virtual machine, it is necessary to send to the new host that virtual machine existing snapshots and the saved stated. Those snapshots will be applied directly on the default base disk, as shown in figure 3.14.

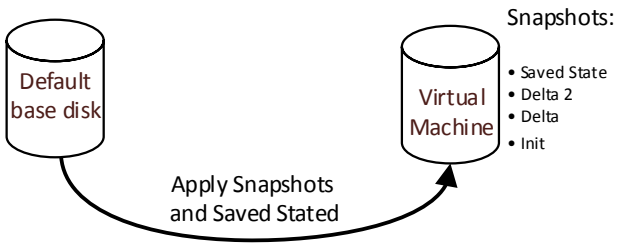

Figure 3.14: Create disk for migrated VM

For example, if we want to migrate the virtual machine from figure 3.13 it's necessary to send to the new host, the delta , delta2 snapshots plus the saved state. On the new host, the default base disk will be cloned and applied the snapshots. Due to the relation between snapshots and their original disks, applying a snapshot requires additional configurations that are explained in the implementation chapter.

3.1.5 VMM data structures and storage

As mention previously, the Virtual Machine Manager client receives information about the hosts. This information is stored on a local database and contains information about the existing base disks and running virtual machines of each host. When required, VMMC reads or writes directly from the database. For each virtual machine the following information is stored:

- Virtual Machine name.
- Host address.
- Base disk.

- Packages.

The *virtual machine name* field is the name the virtual machine and *host address* field is the hostname or ip address of the host where the virtual machine is running.
The field *base disk* is a flag (true or false) that indicates if this is running virtual machine or a base disk machine. If it is a base disk, means that is not a running virtual machine but a template that can be used to create other virtual machines. Only base disk machines entries are used in the calculation of host priorities.
The field *packages* is a list of the installed packages on the machine.

The VMMD also stores information on a local database similar to VMMC. The database of the daemon contains the following information:

- Virtual Machine name.
- Packages.

The Virtual Machine Manager daemon only stores information about the base disks as there is no need to save the running virtual machines since VMMD can, through the virtualization software, obtain the list of current running machines. The stored information is used to decide which disk is a better option to create a virtual machine (depending of the required software packages) and to send to VMMC so it can calculate the priority of this host in future requests.

Disk files are stored on host disk and are access directly or through the virtualization software. When migrating virtual machines VMMD moves the snaphots to the correct location on the file system but when cloning a disk, the process uses the virtualization software tools.
On VMMD installation it is necessary to indicate the available space to store virtual machine. VMMD will use this information to control the number of virtual machines that can be created.

3.2 Network architecture

One of the main features of using parallel computation for the execution of jobs is the fact that we can use multiple type of computational environments. This resources can be found across multiple locations but all of them will contribute to the executions (figure 3.15). Essentially the computational infrastructures, that can be dedicated infrastructures or personal computers , are located on:

- Local area network (LAN).
- Internet.

Dedicated infrastructures are used to be exclusively located on the local network of the job manager but with the emergence of cloud environments, like Amazon EC2, and the internet connections improvements we can have dedicated infrastructures on the internet.
Even when users don't have access to these kinds of infrastructures, there are solutions like BOINC, that allows users to obtain computational resources from the internet.

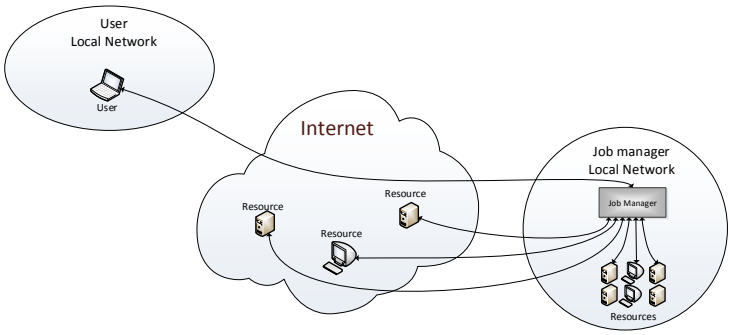

Figure 3.15: Resources locations

Depending on the job manager used, the possible locations of the resources may change. However most of the job managers allow resources from everywhere, from their local network or spread across the internet.
The developed work, should allow the usage of resources independently of their location. Limiting the network location of the resources would make an huge disadvantage to users.
Virtual Machine Manager can be used in multiple environments, for example:

- **Local clusters:** in a local cluster, use can use VMM on the hosts that with SPADE installed on the virtual machines. VMM will be in charge of managing the virtual machines while SPADE does all the work regrading the job execution.
- **EC2:** on a cloud computing environment like EC2, instead of VirtualBox, VMM would use the EC2 virtual machine system and with SPADE installed on the virtual machines we have a environment similar to the local clusters.
- **Condor:** to use Condor, it should executes VMM as a job that creates,configures the virtual machine and execute the job and return the results.
- **Boinc:** Similar to Condor, but instead of Condor we use BOINC to execute VMM.

3.3 Condor Integration

As previously stated, Virtual Machine Manager can be integrated with a job manager to provide a more automated system to execute jobs in multiple environments. Here is presented a proposed architecture for integration with Condor.

Condor allows the execution of jobs in remote machines. These machines belong to a pool of resources known to Condor and jobs can be executed with some requirements. These requirements can vary from operating system, CPU architecture or available software. For example, if a job needs Java, users must specify on the job description file a requirement to the Java availability. However, this only tells condor to find hosts that meet those requirements, if none is found the the job cannot be executed.

With the integration between VMM and Condor, we want to remove this problem. When users submit their jobs, instead a requirement for some software they will specify the software needed. VMM will be in charge of creating the respective execution environment.

When a user submits a job, Condor will find the suitable machines to run the jobs, this architecture consist in execution of Virtual Machine Manager when users submit their jobs to provide suitable machines. Using the information on the job description file, Condor will use VMM to create the virtual machines with the necessary software to execute the job.

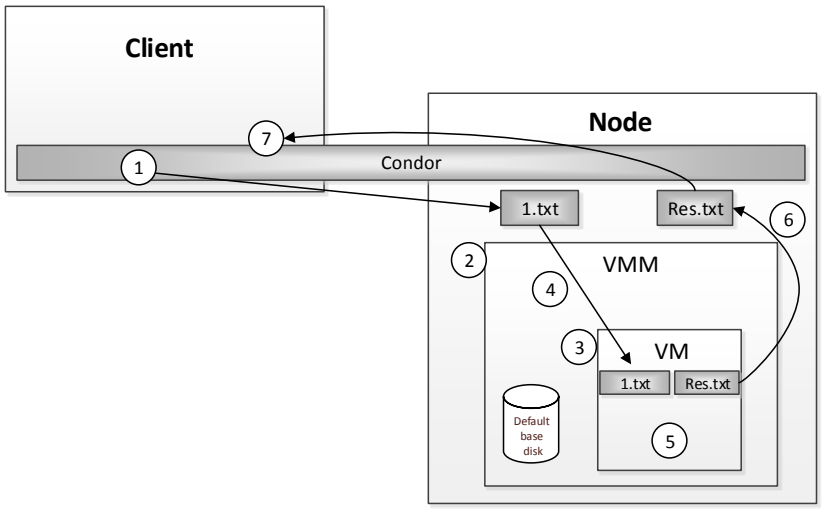

Figure 3.16: VMM + Condor Architecture

The Virtual Machine Manager will be installed on the nodes (both client and daemon) and as usual Condor is also running on the login/master node (where clients submit their jobs) and on the nodes. Users will submit their jobs in the same way they usually do with only a few syntax

differences, these differences are shown in the implementation chapter, but include as arguments the job executable, the software packages needed and the input and output files.
Instead of running the user job executable, Condor will execute a new VMM program that: calls VMMC to create a virtual machine, runs the job in that virtual machine and then return the result. From a Condor perspective nothing changes because it will run a program that returns a result as normal.

The flow observed in figure 3.16 describes the VMM+Condor behaviour:

1- When users submit their jobs, the input file is sent to the remote node.

2- On the remote machine Condor will execute the new VMM program.

3- The new VMM program will run VMMC to create a virtual machine using the mechanisms previously described. It will pass to the VMMC the arguments from Condor (i.e. the software packages and the job execution command).

4- VMM sends to the created virtual machine, the input file.

5- VMM starts the job execution and monitors its execution.

6- The virtual machine executes the job and return the result that is caught by the VMM new program and returned to be caught by Condor.

7- Condor sends, as usually, the result to the client.

The new VMM program will serve as an interface between Condor and the VMMC and will behave like a normal job execution to reduce the impact on Condor. When the job is completed the new VMM program will call VMMC again to shutdown the virtual machine and, according to the behaviour described before, it will be deleted.

Chapter 4

Implementation

4.1 Implementation Options

Here are presented the implementation options that were made to the development of this work.

4.1.1 Programming Language

The Virtual Machine Manager module was developed in Java so it can be used in the largest number of computer architectures and operating systems. As a widely used language, Java allowed a faster development time due to vast documentation and the large existence external packages. The persistence of data was achieved through the use of the HSQLDB system, a relational database system written in Java that provided an easy and efficient way of saving and retrieving persistent data.

4.1.2 Virtualization Software

VirtualBox was chosen to be the virtualization software supported by the VMM module. This decision was made after considering other options, like Xen or KVM. Note that were only taken into account open-source solutions.
In the choice of this solution were taken into account the following features: price, host operating system diversity and usability.
VirtualBox is an open source software, free of charge that runs in multiple operating systems and is very easy to install and to use. It can be found in enterprise environment but also in home

environments.
Unlike Xen or KVM, VirtualBox runs on hosts with Windows and Mac OSX. It is also much more user friendly which its more likely to be used in home environments. VirtualBox provides a command-line interface named VBoxManage that allows completely control VirtualBox and its virtual machines. This tool is used by the VMMD to create and monitoring the virtual machines on the hosts.

4.1.3 Virtual Machines Operating System

The virtual machines operating system, which was installed in the default base disk, is Debian. Debian is one of the most used and stable Linux distributions, that serve as base to many other distributions. It has large repositories and most of the existing software's are made available for this distribution. The default base disk should be a minimal installation of Debian with the VirtualBox Guest Additions installed. If the Guest Additions package is not installed, VMMD will not be able to run commands in the virtual machines, which means that it will be impossible to install packages on the virtual machines.

Before installing the Guest Additions it is necessary to install is the required packages.
This should be done by running the following commands:

```
# apt-get install build-essential module-assistant bzip2
# m-a prepare
```

After the required packages are installed, the Guest Addition image should be mounted through the VirtualBox interface into the virtual machine and then run the following commands:

```
# mount /media/cdrom
# sh /media/cdrom/VBoxLinuxAdditions.run
```

The Guest Addition package can be installed from other source such as repositories, however it is recommended the usage of this method to assure that the Guest Additions installed is compatible with the vBoxManage tool of the host.
Other OS could be used with the relevant changes regarding their package manager.

4.1.4 Network

To make the connection between the virtual machine manager client and daemon were used sockets. Sockets in Java are very easy to use and with a lot of documentation available, this option allowed a faster development.
Both client and daemon programs have a configuration file, config.properties, where the users can specify the port that should be used for network connections. The client and all daemons must use the same port in order for the program to work.

4.1.5 Snapshots

Snapshots are used implement the migration of virtual machines functionality. To achieve this goal was necessary to understand how snapshots work in VirtualBox.
A snapshot is a saved virtual machine state at a given time that can be used at any time to go back to that state. A Virtual Box snapshot contains a complete copy of the virtual machine settings including hardware configurations, the state of all the virtual disks of that machine and if the VM was running during the snapshot a memory state is also saved.
When a snapshot is restored all the changes made after the snapshot are undone, including virtual machine and hardware settings and also changes to the machine's disks. Files deleted after the snapshot will be restored and files created will disappear.
The virtual machine settings are saved in an xml file and requiring very little space but the memory state file can be as large as the memory of the virtual machine. The virtual disks are saved through the use of differencing images.
When a snapshot is taken VirtualBox freezes the image file and no longer writes to it. Write operations from the virtual machine are made to a second image file, a differencing image.
A differencing image is a disk image that contains only the differences between him and another disk image. These disks must always point to another image, there are a "child" image and the image that they refer is its "parent". It is a hierarchical structure as shown in figure 4.17. A differencing image it's useless unless it refer's to another image.

When a virtual machine has a differencing image active, that image receives all the write operations instead of its parent. These differencing images contain only the sectors of the virtual disk that have changed since its creation. When using differencing images, the parent image becomes read-only, with write operations using always differencing images. On read operations, VirtualBox checks the differencing images first, if those sectors exists in this image they are returned, if not VirtualBox will read the parent and return from it.

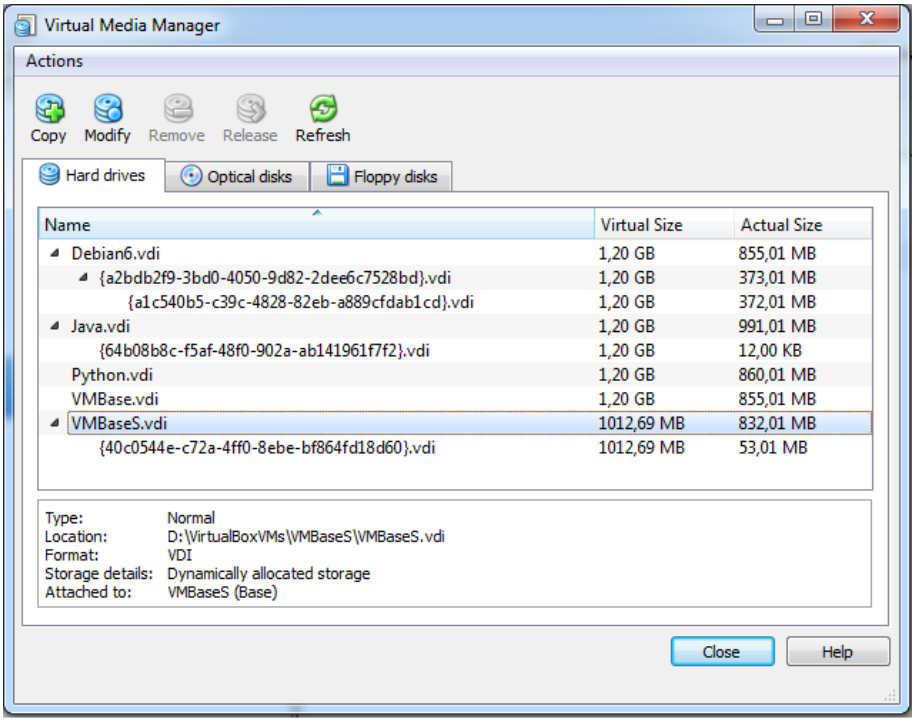

Figure 4.17: Virtual Media Manager

As stated before, differencing images have a hierarchical structure and they can have multiple levels, i.e. , an image can be a child of a child or just a “grandchild” of the original image. In these cases, VirtualBox uses the active image to write operations and for read operations starts with the active image and go up in the hierarchy (reads the active image, then its parent, then its grandparent and so on..).

There are no limit in the number of differencing images that a disk can have, however there is a slight run-time I/O overhead because VirtualBox may have to check the same sector multiples times.

As mention before each differencing image as a parent disk. This information is stored on the image itself in a form of a identifier called uuid.

The Virtual Machine Manager module only sends the snapshots on a virtual machine migration, however a snapshot is a differencing image that is useless on its own. During the migration process and after the host receives the snapshots it creates a new base disk. This base disk needs to be the parent of those snapshot. To accomplish this tasks, VMMD uses the VboxManage tool to change the uuid of the recent created disk to correspond to the uuid of the parent of the received snapshots.

To be more specific, VMMD reads from the snapshots the value of the uuid of its parent by running the following command:

```
# vBoxManage internalcommands dumphdinfo pahtToSnapshotFile
```

After getting the uuid, VMMD changes the value of the new parent with the command:

```
# vBoxManage internalcommands sethduuid pathToNewDisk uuid
```

Note that this process only works because both parents are exactly the same, they are direct clones from the default base disk that should be the same across all hosts. If different hosts have different default base disks, the migration functionality will not work properly.

4.2 Interfaces

When running the VMM Client, some arguments must be specified to execute the desired actions. Here is presented the syntax of the VMM Client.

```
# java vmm –new <vm name> <quantity> [-c <path to package>] [<list of packages>]
```

or

```
# java vmm –shutdown <vm name>
```

Arguments:

- -shutdown <vm name>: Name of the virtual machine to be turned off.
- -new <vm name>: Name of the virtual machine to be created.
- <quantity>: the number of virtual machines to be created.
- -c <path to file>: Path to users custom package (optional).
- <list of packages> List of packages to be installed (optional).

Notes:
As the virtual machine OS will be Debian, the selected packages must exist in the Debian software repositories.
The custom software must be in zip format and will be extracted to /opt/ inside the virtual machine.

4.2.1 Examples

Example 1:

```
# java vmm -new PovrayVM 5 povray perl
```

This command tells VMM to create five virtual machines named PovrayVM with the packages povray and perl installed.

Example 2:

```
# java vmm -new PhytonVM 7 -c c:\mysoftware.zip python
```

This example will tell VMM create seven virtual machines named PhytonVM with the content of mysoftware.zip extracted to /opt/ and package python installed.

Example 3:

```
# java vmm -shutdown PovrayVM
```

This example will tell VMM to order the shutdown, in all hostts, of the virtual machines named PovrayVM.

4.3 Condor integration

The implementation of the VMM+Condor integration is intended to be simple and with minor impact to the users or the Condor system. Despite of not being implemented, here is described the necessary changes that must be made in order do implement the two systems.
The impact to the user should small and only visible on the file job description that he must create. With this integration users must specify a job description file similar to the following:

```
executable      = VMM
arguments       = java-1.2 # java jobProgram
transfer_input_files = jobProgram
output          = output.out
```

On a VMM+Condor system the executable should always be *VMM* because (for Condor) it will be the job being executed. The arguments should be: the package list necessary o run the "real job", a "#" character and the user job syntax. In this case we are telling that we want java-1.2

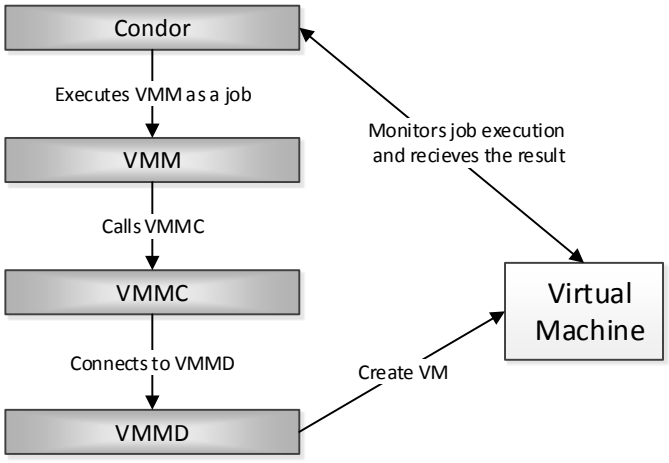

Figure 4.18: Condor+VMM execution

installed and the job that should be executed is "java jobProgram". The output is the name of the file that will contain the result of the execution.

The Virtual Machine Manager client and daemon must be installed on the hosts together with a new VMM program called VMM.

The new VMM program, as stated in the architecture chapter and from a Condor perspective, must have an execution similar to the execution of a normal job. It shouldn't be much more than a interface between Condor and the VMM system. The development of the new program consist in a program that calls the VMMC with the arguments received from Condor but using the VMMC syntax. It will create, configure and execute a virtual machine and after the virtual machine start, it sends the input file to the virtual machine to start the job execution, while monitoring it. When the job execution stops, the new program return the results that will be caught by Condor and returned to the user (figure 4.18). Besides the job description file, this implementation has no need for any changes on the Condor system.

Chapter 5

Evaluation

In order to evaluate the Virtual Machine Manager module performance, it was simulated a number of tasks that representing the tasks that could occur during its usage. Those tasks were executed on the same environment to allow the comparison of results.
Those tasks were:

- Creation of a virtual machine from scratch.
- Creation of a virtual machine from existing base.
- Creation of a virtual machine from a snapshot.
- Migration of a Virtual Machine.

Each task was executed twice, with different packages, the first with a small package and the second with a larger package.

The tests were made on a local network with a maximum speed of 100 Mbps, the host running the virtual machines was a desktop computer with a Intel Core i5-2320 at 3.00GHz, 10 GB DDRIII 1600 MHz of memory RAM, running Windows 7 64bits and the virtual machines disk files were stored on a physical 1 TB SATA II, 7200 RPMs disk. It was also used a 30 Mbps internet connection.
The default base disk used to create the virtual machines was a minimal installation of Debian 6 operating system, occupying 642 MB on a physical disk file of 855 MB and the repository to download the software packages was ftp.pt.debian.org.
Each task is considered completed when the virtual machine is ready to use.

5.1 Creation of virtual machine from scratch

The first task was the creation of a virtual machine from scratch shown in figure 5.19, i.e, a virtual machine created from the default base disk. As explained before this task is achieved by creating a new base disk with the software (cloning of the default base disk and installation of the software packages) and then cloning that base. The results of this execution include: the cloning of the base disk, OS startup, download and installation of the packages, shutdown of the virtual machine and cloning.

On the first execution of this tasks, it was order a virtual machine with the package python (18,2 MB). It took 43 seconds to create the new base, 13 seconds to clone and 25 seconds start it, performing a total o **81 seconds**.
The second execution, with a large package, Java openjdk-6-jdk (172 MB), took 84 seconds to create the new base, 15 seconds to clone and 24 seconds start the virtual machine. A total of **123 seconds**.

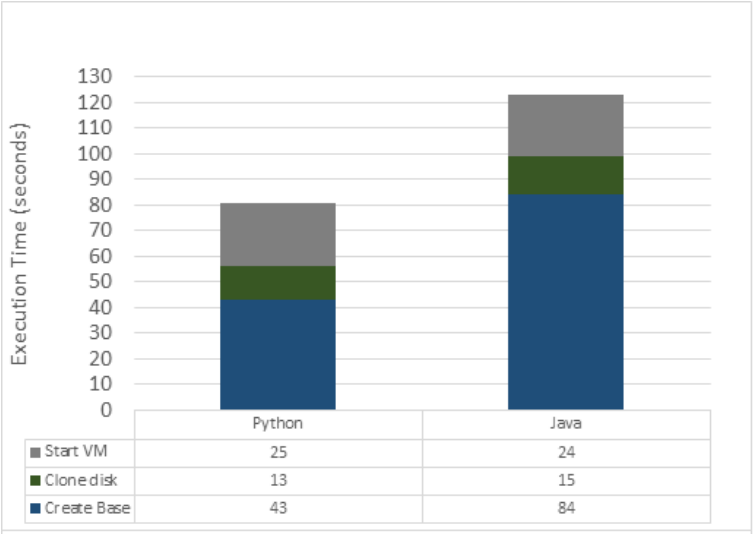

Figure 5.19: Create virtual machine from scratch

As shown above, the major differences between the two executions are in the create base process. This phase includes the cloning of a disk and the download and installation of the software. As will be verified in the next tests, the cloning of a virtual disk is a very fast action (1 or 2 seconds), so the main time consuming action in the creation of a base disk is the download and installation of the software. It depends on the size of the software and the internet connection

speed. When reproducing these tests is necessary to take into account that the internet connection to the repository that can cause considerable changes in the times obtained.

Another important aspect of these tasks is the clone disk time. It was just mention that the cloning process is very fast, however in theses tasks it takes from 13 to 15 seconds to clone the created base. A disk file can only be cloned when it's not in use and because the created base disk is turned off just before the cloning process, it is necessary to wait for the system (operating system and virtual box) to release the lock on the file that represents the disk.

5.2 Creation of virtual machine from existing base

The second task, figure 5.20, consists on the creation of a virtual machine from an existing base. On a normal execution this task would occur if it was requested a virtual machine with a package previously requested. This would mean that the host already has a base with the needed software packages and only has to clone it and then start the virtual machine.

The first execution with requesting a VM with python, had to clone a base disk of 860 MB and took 1 second. The virtual machine started in 25 seconds making a total of **26 seconds** executing the task.
The second execution, with the java package, took 2 seconds to clone a base disk of 991 MB and 27 seconds to start the virtual machine Performing a total of **29 seconds**.

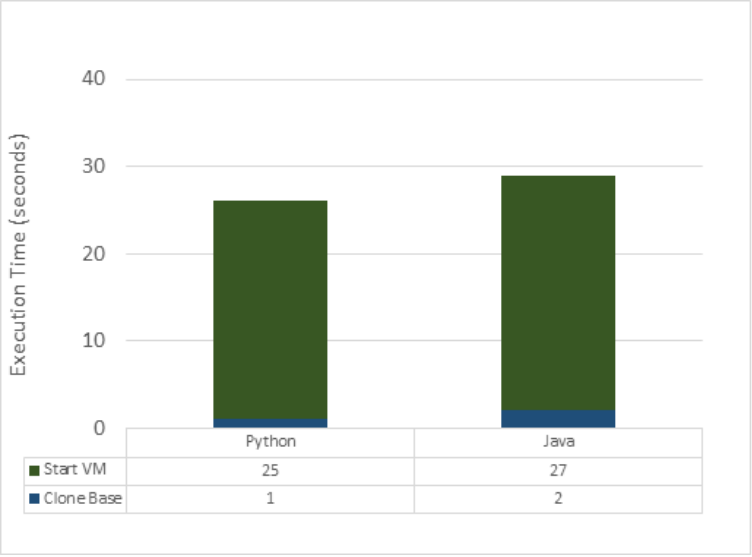

Figure 5.20: Create virtual machine from existing base

As observed by the obtained results, this is the simplest and ideal task in the creation of virtual machines through virtual machine manager module. If a host already had a virtual machine with the same software packages as the request, it will take very little time to create the virtual machines.
The virtual disks cloning process is a very fast process that takes 1 or 2 seconds to complete. The use of this feature allows a faster creation of virtual machines, there is no need to install the operating system (as all host already have a default base disk pre-installed) and may not even be necessary to install software if there is already a disk with that software, however there is a downside. When cloning a disk to create base disk, there is a trade off between virtual machine creation speed and the host storage usage.

Despite the fact that the creation of base disk will use more storage space it will pay off when we want to create a virtual machine with software that already exists in a saved base disk.

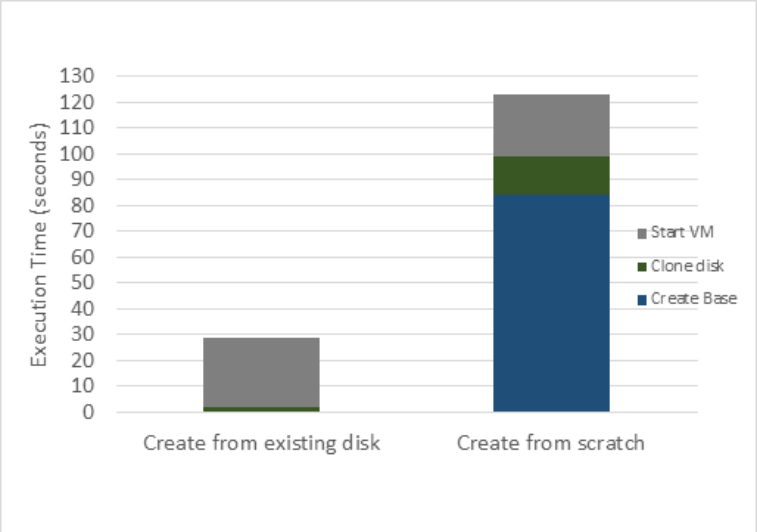

Figure 5.21: Create virtual machine from existing base vs from scratch

Figure 5.21 shows the difference between creating a virtual machine with Java from scratch and the creation of a virtual machine with Java when there is already a base disk with it. As mention previously, the creation of a virtual machine with Java takes a total of **123 seconds**, however if there is a base disk containing Java, creating a virtual machine it will only take **29 seconds**. Here can be seen one of the main advantages of the Virtual Manager Machine. When executing multiple jobs, the requirements repeat themselves, making this scenario occur frequently.

5.3 Creation virtual machine from a snapshot

The creation virtual machine from a snapshot task was done in order to answer one question. What is better, create a new base (clonning and installing the software packages) in multiple hosts or do it in one host and send a differencing snapshot to the remaining hosts?
This task includes the times to transfer the snapshot (first from the host to the VMM Client and then to the new host), cloning the default base, applying the snapshot and boot the virtual machine, times are shown in figure 5.22.

On the execution containing the python package, was transferred a snapshot with 132 MB in 42 seconds. The cloning process took 1 second, applying the snapshot 25 seconds and starting the virtual machine 26 seconds, making a total of **94 seconds** executing the task.
In second execution, it was transferred a snapshot of 375 MB in 128 seconds, with the remaining process similar to the times above, clone 1 second, applying snapshot 29 seconds and starting of the virtual machine 26 seconds. A total of **184 seconds**.

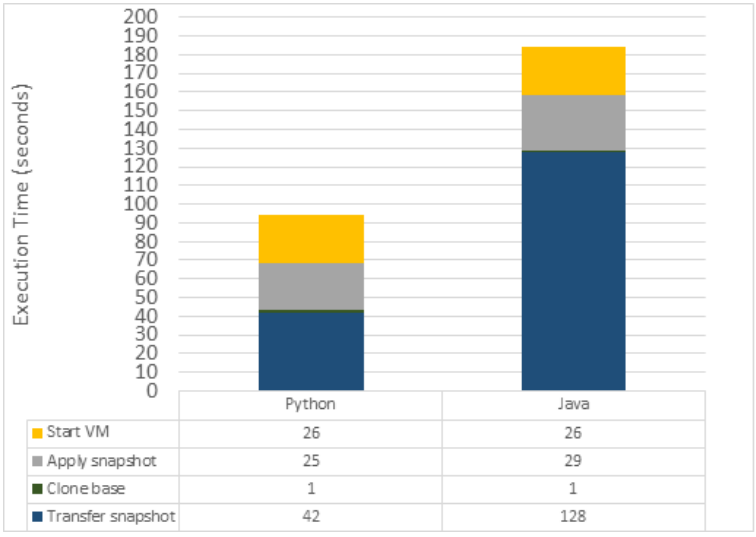

Figure 5.22: Create a virtual machine from snapshot

The action of applying a snapshot consist, as explained in the implementation chapter, in the copy of the received files (virtual machine settings and snapshot) to the virtual machine location on the host file system and on the change of the uuid of that VM to the parent uuid of the snapshot. Even with generally slow task of moving files on the host disk, the majority of time used in these tasks is on the sending and receiving the snapshots files.
The quantity of time that process of downloading and installing software in a virtual machine depends on the internet connection speed that may not be very fast or even not very reliable, this

lead to motive being this 3rd test. It would be faster to install the software in all host or just in one and then sending the snapshot to the other hosts?

If we choose a scenario with two hosts as shown in figure 5.23, where the host 1 created a virtual machine from scratch like in the first test and host 2 receives a snapshot, we can see that creating a VM from scratch is considerable faster than applying a snapshot. These results happen because snapshots are always bigger than the installed packages, in this case from 172 MB to 375 MB, and the fact that the snapshot has to be transferred two times (2x 375 MB), first to the client and then to the new host. It will be transferred 750 MB of data instead of 172 MB, even in the cases where the hosts are on the local network (like the scenarios in these tests), it will not pay off.

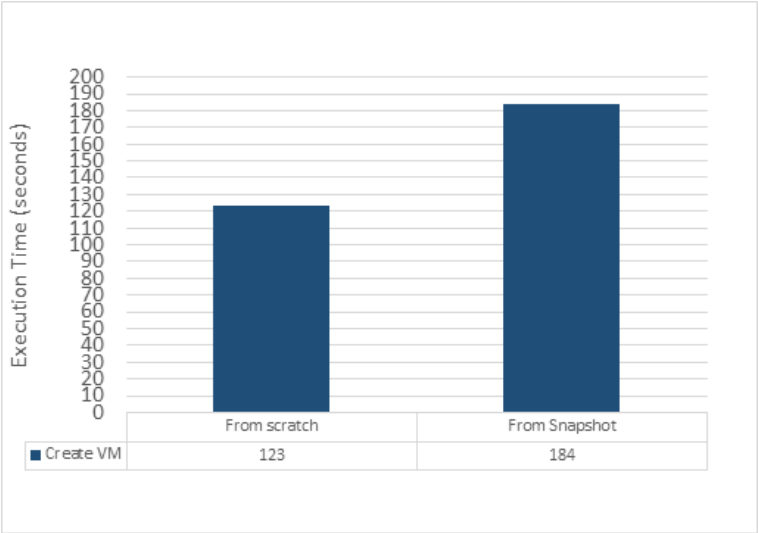

Figure 5.23: Install software VS Applying Snapshot

With more hosts the scenario may look better for the snapshots because the transfer to the client only happens once but even if we remove that time from the problem, the other hosts can only began to create their virtual machines after the first one has complete its base disk. The figure 5.24 shows a a scenario between a creation and installation from scratch and the time to apply a snapshot *assuming that the snapshot is already on the client*, meaning that the transfer time is 64 seconds instead 128 seconds (which would be the case for every host beyond the second). When installing a virtual machine from scratch in multiple hosts, they will began the process almost simultaneous and thus terminating within the same time window.

If analysed only the data needed to be transferred in both cases, we see that when sending snapshots the amount of data greatly increases. For example, the creation of five virtual machines with Java in five different hosts, from scratch will require the transfer of 860 MB of data (five times 172 MB), the transfer of snapshots will require 2047 MB (172 MB from the first installation, 375

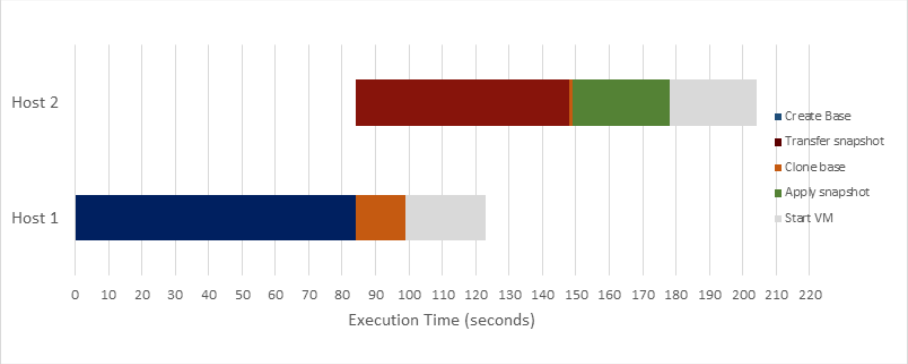

Figure 5.24: Applying snapshots

MB to send snapshot to client and 4x375 MB to send to the other hosts).
More hosts mean a bigger difference in the data needed to be transferred.

5.4 Migration of a Virtual Machine

Migration of a virtual machine will occur when a user closes a running virtual machine. When a virtual machine is closed by a user, VMM will send the snapshot and saved state of that machine to another host.
This task is almost the same as the previous task but adding the need of also sending the saved state of the virtual machine. The saved state size of the virtual machine depends of its execution at the time of the shutdown and can have a maximum size equal to the total memory of the virtual machine.
In the first execution with the package python, the virtual machine was idle at the time of the shutdown and it was sent a snapshot of 132 MB and a saved state of 38 MB. The transfer process took 50 seconds, the cloning 1 second and applying snapshot plus saved state 29 seconds. With the 26 seconds for starting the virtual machine, this task took a total of **102 seconds**.
The second execution took 132 seconds to send the snapshot (375 MB) and the saved state (170 MB), 1 second to clone, 29 seconds to apply the snapshot and saved state and 26 seconds to start the machine in a total of **188 seconds**.

The migration process includes the transfer of snapshots and has shown before, snapshots have a considerable size. Also in the migration process it is necessary to send the saved state that can have a size equal to the memory of that virtual machine.
We mus now evaluate if the machine migration is viable. It depends on how long the virtual machine has been running the task, how many time would take to migrate the virtual machine and how many time would take to create a virtual machine with the same software. If the time

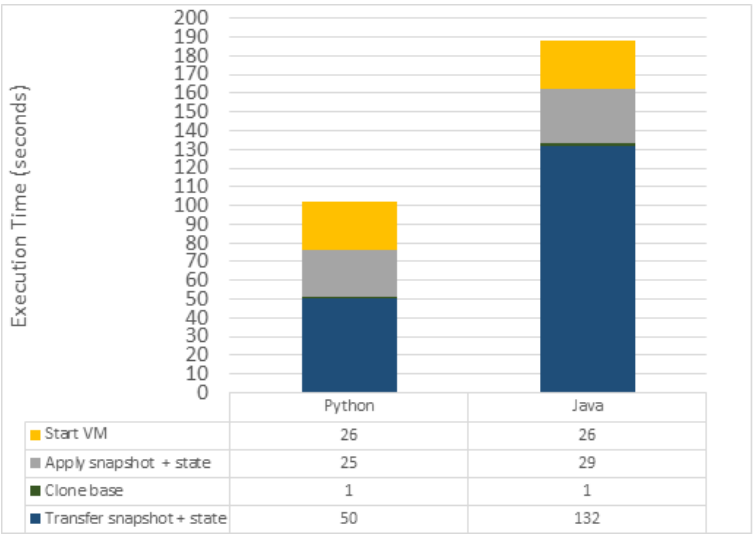

Figure 5.25: Migrate a virtual machine

that the virtual machine is running the task plus the time to create a similar virtual machine is greater than the migration time then yes. The migration process would add a gain of time in the job execution. Otherwise, the migration process would take more time than create a new virtual machine in another host and restart the task.

Lets compare the second execution of the tests 1 and 4. In the first one, it's created a virtual machine with Java and on the fourth test is migrated a machine with the same environment.

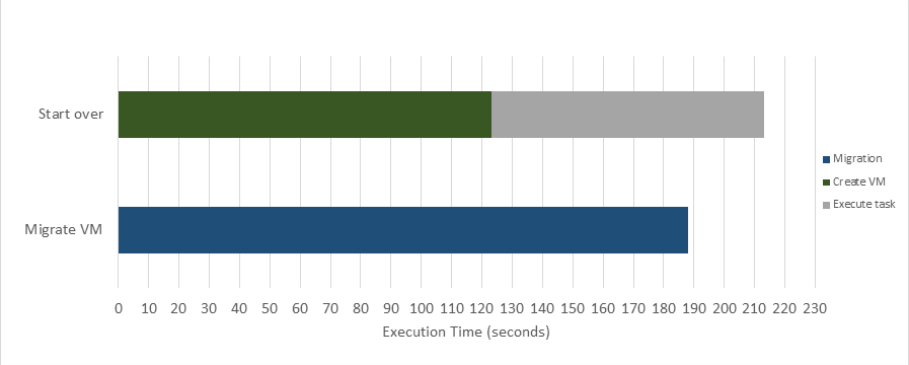

Figure 5.26: Migration vs Start over example 1

For example, if the original virtual machine was executing a task for 90 seconds at the time of the migration, should this virtual machine be migrated or should we start over? The answer can be viewed in the figure 5.26. It will that more time to create a new virtual machine and execute a task for 90 seconds (to reach the same state) than to migrate the virtual machine. The previous example shows that the migration process pays off, however if we assume that the task was only

running for 60 seconds on the original machine we observe just the opposite, figure 5.27. In this new example the time to create a new virtual machine plus the time to execute the task till the current state is less than the time that would take to migrate the virtual machine making the migration a bad option.

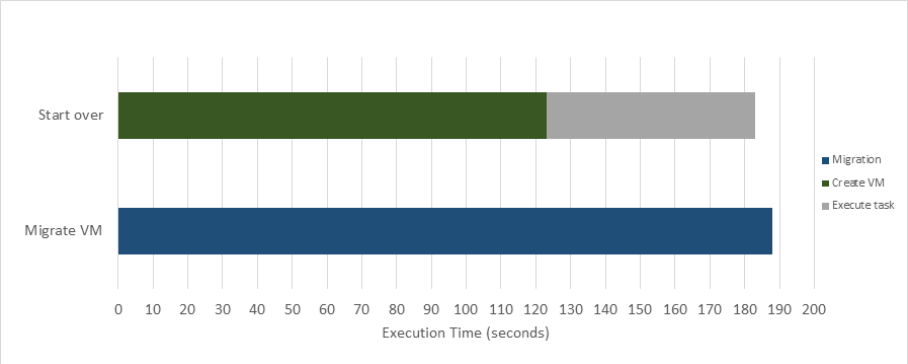

Figure 5.27: Migration vs Start over example 2

In this environment, if the task has been running for more than 65 seconds, the migration will be a better solution otherwise it will be better to start over.
The migration feature it is important and must exists but that doesn't mean that it should be always applied. In this case it would be better not to use the migration feature.
When integrating the Virtual Machine Manager with a job manager a run-time mechanism must evaluate the past task execution time and decide on the migration or restart of the job.

5.5 VMM+Condor

The evaluation of the VMM+Condor system can be used to observe the impact that such integration would have on the systems and users.
The implementation of the VMM+Condor integration is simple and with few impact on its usage:

- No modifications to Condor are required.
- Minimal impact to the user as they have to use some specific options in the job description file. The execution of jobs is done has the same way users do when using only Condor, through the command *condor_submit* and the job description file.
- The Virtual Machine Manager must be installed on the existing nodes that will run both client and daemon programs.

The usage of virtualization also brings some security advantages. The default base disk used by VMM is created by the system administrators and can be configured to be a more restrictive environment decreasing the execution of malicious software. For example, software can be executed at a user-level so it as no access to the network card and prevent packet sniffing and similar techniques.

When using Condor VM universes, users provide a virtual machine disk. This disk is installed and configured by the users that can use it to run harmful software as it can run as root user inside of that virtual machine. When using VMM, users don't have access to root-level when executing software (unless the default base disk has been made to allow it).

Chapter 6

Conclusion

There are a lot of options for users to do a parallel execution of jobs, however users have to gain access to an infrastructure that allows parallel execution of jobs. In these infrastructure, users don't always have the execution environments needed to execute their jobs. They must use the execution environments available in those infrastructure and even if they are using their own infrastructure it is necessary to manually prepare the environment in each computational node.
The use of virtualization technology in these infrastructures increases flexibility in the creation of execution environments. This work had as objective to use the benefits from the virtualization technology to allow an on-demand creation of a execution environment for jobs.
Providing an easy creation of virtual machines and the automated installation of software packages in those machines, users aren't limited to the existing execution environment in a infrastructure. It also enables the use of resource to execute certain jobs that weren't possible to execute before. For example, using virtual machines, a node can execute Linux dependent jobs independently of their operating system.
It was developed a system, called Virtual Machine Manager, that provides the ability to create a number of virtual machines across a pool of hosts. The application consists in two programs, the client and a daemon that must be running on the host computers. Besides the creation of virtual machines it has two other main features: the automatic installation of software packages in those machines and an optimization mechanism that allows a faster provision of virtual machines. When creating virtual machines, VMM receives a list of software packages that will be automatic downloaded and installed on those machines. In order to optimize this process, Virtual Machine Manager, saves as templates the past requests of virtual machines environments to in future similar request provide a faster deployment. If there is a template with the same software

that was requested, the Virtual Machine Manager will use it to the provisioning of the virtual machine saving the downloading and installation time.
Although Virtual Machine Manager can be used as a standalone application to provide an automated deployment of virtual machines, it was develop with the objective of being integrate on a distributed execution of jobs environment. This environments are manager by a job manager like Condor, SPADE or Hadoop, due to its simple command line interface, VMM can be easly integrated into an external system. It was proposed a possible solution for a Condor+VMM solution that working together both system provide a better user experience in the remote execution of jobs and also solves some security issues.
When evaluating the Virtual Machine Manager it is necessary to take into account that the obtained results greatly depend on the performance of the virtualization software used, in this case VirtualBox. If it was used another virtualization software, the results could be different, nonetheless the time to start/restart and migrate tasks are solved in a 2 minutes window, and those tasks shouldn't take much longer. Also for repetitive tasks (tasks that use the same software) the deployment of the environment takes less than 30 seconds.
One of the main concerns about the performance of this work was the cloning process of the virtual machines disks. This process is critical in the creation of virtual machines whether we are creating a virtual machine from the default disk or from a temple there is always the process of cloning a disk image file. Despite the considerable size of these files the process proved to be extremely fast. Using existing template we can save more than 50% of total time of provision and software installation of a virtual machine with the requested software.
The virtual machine migration proved to be a good solution to in some situations however it may not always pay off to execute this process and with the integration with a job manager that already migrates jobs it is necessary to analyse which method is better.
The obtained results allowed to observe that a tool such this can improve the users experience when executing jobs remotely. It should allow the usage of more resources and remove some environment limitations that may existing in current systems.

6.1 Future Work

As all projects, there is always space for improvements. Here are presented the work that may improve the Virtual Machine Manager and offer a better experience to users:

- Implementation of Condor and VMM integration. Although it was though the integration with Condor, it wasn't implemented. This should allow a better perception of the impact

of the Virtual Machine Manager system and on a large environment should allow a better evaluation analysis.

- Heuristic for the migration process. As was explained before, the migration process may or may not be, depending on the situation, a good choice. The development of a heuristic to, in runtime, decide if the migration process is worthy should eliminate this problem.

Currently the Virtual Machine Manager only supports VirtualBox as its software virtualization. Increasing the support for others virtualization software will increase the number of environments that VMM can be used.
VMM can also be integrated to Amazon EC2 for the easy creation, deployment and management of VM's on the cloud and an architecture similar to the Condor+VMM can be applied to BOINC or allowing a more secure execution environment.

Bibliography

[1] Silva, J.: New Environments for Parallel Execution and Simulations. PhD thesis, Universidade Tecnica de Lisboa - Instituto Superior Tecnico (2011)

[2] Thain, D., Tannenbaum, T., Livny, M.: Distributed computing in practice: The condor experience. Concurrency and Computation: Practice and Experience **17**(2-4) (2005) 323–356

[3] TOP500 project: TOP500 project. `http://www.top500.org` (2013)

[4] Sterling, T.: Beowulf cluster computing with Linux. (2001)

[5] Foster, I., Kesselman, C.: The Grid 2: Blueprint for a new computing infrastructure. Access Online via Elsevier (2003)

[6] Foster, I.: What is the grid? a three point checklist, gridtoday, vol. 1, no. 6, 2002

[7] Anderson, D., Cobb, J., Korpela, E., Lebofsky, M., Werthimer, D.: Seti@ home: an experiment in public-resource computing. Communications of the ACM **45**(11) (2002) 56–61

[8] Mersenne Research, Inc.: Great Internet Mersenne Prime Search - GIMPS. `http://www.mersenne.org/` (2013)

[9] Robinson, R.: Mersenne and fermat numbers. Proceedings of the American Mathematical Society **5**(5) (1954) 842–846

[10] Mell, P., Grance, T.: The nist definition of cloud computing (draft). NIST special publication **800** (2011) 145

[11] Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications **1**(1) (2010) 7–18

[12] Amazon.com: Amazon elastic compute cloud. `http://aws.amazon.com/ec2/` (2013)

[13] Microsoft Corporation: Windows azure virtual machines. `http://www.windowsazure.com/en-us/home/scenarios/virtual-machines/` (2013)

[14] Amazon.com: Amazon elastic beanstalk. http://aws.amazon.com/elasticbeanstalk/ (2013)

[15] Microsoft Corporation: Windows azure compute. http://www.windowsazure.com/en-us/develop/net/compute/ (2013)

[16] Google inc.: Google app engine. https://developers.google.com/appengine/ (2013)

[17] Google inc.: Google apps. www.google.com/Apps (2013)

[18] Microsoft Corporation: Microsoft office 365. www.microsoft.com/office365 (Setptember 2013)

[19] University Of California, Berkeley: Berkeley open infrastructure for network computing. http://boinc.berkeley.edu/ (2013)

[20] Anderson, D.P.: Boinc: A system for public-resource computing and storage. In: Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on, IEEE (2004) 4–10

[21] The Apache Software Foundation: Apache hadoop. http://hadoop.apache.org/ (2013)

[22] Zikopoulos, P., Eaton, C., DeRoos, D., Deutsch, T., Lapis, G.: Understanding big data. New York et al: McGraw-Hill (2012)

[23] Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on, IEEE (2010) 1–10

[24] Center for High Throughput Computing at the University of Wisconsin-Madison: Htcondor manual. http://research.cs.wisc.edu/htcondor/manual/v7.8/index.html (2013)

[25] Marinescu, D., Kröger, R.: State of the art in autonomic computing and virtualization. Distributed Systems Lab, Wiesbaden University of Applied Sciences (2007)

[26] Oracle: Virtualbox. https://www.virtualbox.org (2013)

[27] The Xen Project: Xen. http://www.xenproject.org/ (2013)

[28] Popek, G.J., Goldberg, R.P.: Formal requirements for virtualizable third generation architectures. Communications of the ACM **17**(7) (1974) 412–421

[29] Kivity, A., Kamay, Y., Laor, D., Lublin, U., Liguori, A.: kvm: the linux virtual machine monitor. In: Proceedings of the Linux Symposium. Volume 1. (2007) 225–230

[30] Russell, R.: virtio: towards a de-facto standard for virtual i/o devices. ACM SIGOPS Operating Systems Review **42**(5) (2008) 95–103

[31] VMware, Inc,: Vmware. `http://www.vmware.com/` (2013)