A Personal Platform for Parallel Computing in the Cloud

Bruno Macedo
bruno.macedo@ist.utl.pt

Instituto Superior Técnico
Universidade Técnica de Lisboa
Av. Prof. Dr. Anbal Cavaco Silva
2744-016 Porto Salvo, Portugal
www.ist.utl.pt

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 users personal computers is practically impossible due to the time it would take to complete. There are job managers 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. 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 arent an option without virtualization technology.

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

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.
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 or 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.
Another possibility is 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*[12] or *HTCondor*[13]. 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 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. Or they lack the knowledge to use parallel computation systems or they lack the access to all available resources (from personal computers to dedicated infrastructures).
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.
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. 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.

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. 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 physical 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. Some of the main execution environments are:

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

Job managers allow users to do a parallel execution of their jobs across a pool of resources. Theses resources can be 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.

2.1 Condor

HTCondor[3] 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 **??**, 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.
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.

2.2 SPADE

SPADE 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 remote execution

of jobs to all users, including those less knowledgeable.
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.

2.3 Apache Hadoop

Apache Hadoop 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.

3 Architecture

The main objective of this work is to allow users to execute their jobs in a parallel computing environment. 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.
The 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 1.

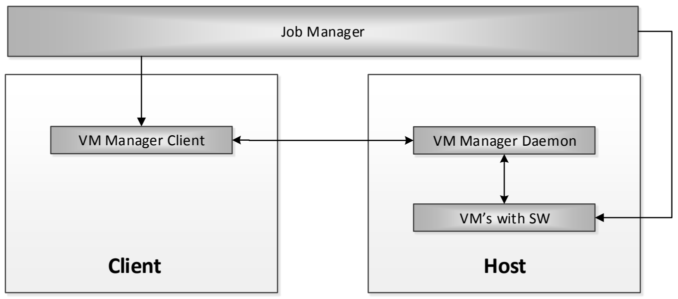

Fig. 1. Architecture

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 1, 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.
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.
Hosts will save previously virtual machines installations as templates. It's com-

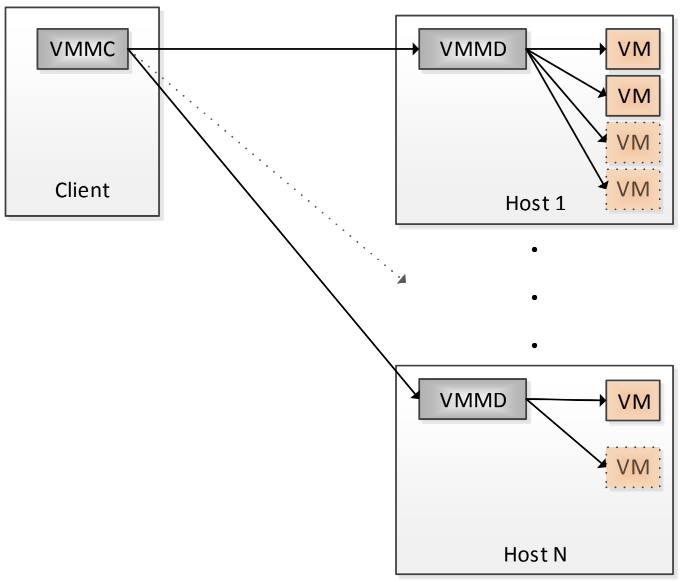

Fig. 2. Virtual machine deployment

mon 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.
For each job, VMM can launch multiple virtual machine in different computa-

tional 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.
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).

4 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.
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 very different.
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.

References

[1] David P Anderson. Boinc: A system for public-resource computing and storage. In *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*, pages 4–10. IEEE, 2004.
[2] M. Baker and R. Buyya. Cluster computing: the commodity supercomputer. *Software-Practice and Experience*, 29(6):551–76, 1999.
[3] 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.
[4] I Foster. What is the grid? a three point checklist, gridtoday, vol. 1, no. 6, 2002.

[5] Ian Foster and Carl Kesselman. *The Grid 2: Blueprint for a new computing infrastructure*. Access Online via Elsevier, 2003.

[6] A. Fox, R. Griffith, et al. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Tech. Rep. UCB/EECS*, 28, 2009.

[7] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. kvm: the linux virtual machine monitor. In *Proceedings of the Linux Symposium*, volume 1, pages 225–230, 2007.

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

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

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

[11] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10. IEEE, 2010.

[12] J.N Silva. *New Environments for Parallel Execution and Simulations*. PhD thesis, Universidade Tecnica de Lisboa - Instituto Superior Tecnico, 2011.

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

[14] TOP500 project. TOP500 project. http://www.top500.org, 2013.

[15] PC Zikopoulos, Chris Eaton, Dirk DeRoos, Thomas Deutsch, and G Lapis. Understanding big data. *New York et al: McGraw-Hill*, 2012.