A Tool for Environment Deployment in Clusters and light Grids

Yiannis Georgiou, Julien Leduc, Brice Videau, Johann Peyrard and Olivier Richard
*Laboratoire ID-IMAG (UMR5132) Grenoble {Firstname.Lastname}@imag.fr**

Abstract

Focused around the field of the exploitation and the administration of high performance large-scale parallel systems , this article describes the work carried out on the deployment of environment on high computing clusters and grids. We initally present the problems involved in the installation of an environment (OS, middleware, libraries, applications...) on a cluster or grid and how an effective deployment tool, Kadeploy2*, can become a new form of exploitation of this type of infrastructures. We present the tool's design choices, its architecture and we describe the various stages of the deployment method, introduced by* Kadeploy2*. Moreover, we propose methods on the one hand, for the improvement of the deployment time of a new environment; and in addition, for the support of various operating systems. Finally, to validate our approach we present tests and evaluations realized on various clusters of the experimental grid* Grid5000*.*

1 Introduction

The high performance computing clusters (HPCC), are today well known considerably on the scientific scene where they have become an indispensable tool, and they are used for solving the most challenging and rigorous engineering tasks facing the present era. Moreover, in order to accumulate their computing power, there is an increasing need to incorporate the resources of multiple clusters. In this case we speak of Grid Computing.

Under this context, various challenging issues have appeared, mainly in the procedures of exploitation and administration of these two types of infrastructures. In the case of clusters, the need for automated installation and configuration tools has been obvious ever since the first appearance of networked clusters of workstations. In the case of grids the problem the users have to confront, is a software heterogeneity among the interconnected clusters.

*Supported by Grid5000, French Ministry of research, ACI Grid, ACI Data Mass Incentives and RENATER the French National Research and Education Network.

Furthermore, in the context of high performance computing research, scientists seem to need various software environments in order to perform their experiments. A software environment contains all the software layers like the operating system, the libraries, the middlewares and the applications (figure 1). According to their experiments nature and the software layer they are investigating (protocols, OS, ..), they often require specific OS. Hence, a tool with a deep reconfiguration mechanism allowing researchers to deploy, install, boot and run their specific software images, is needed.

Kadeploy2 is a software environment deployment tool which aims to solve the above issues providing automated software installation and reconfiguration mechanisms on all the layers of the software stack. Using kadeploy2, in a typical experiment sequence, a researcher reserves a partition of the cluster or grid, deploys its software image (figure 2), reboots all the machines of the partition, runs the experiment, collects results and finally relieves the machines. This reconfiguration capability allows researchers to run their experiments in the software environment that perfectly matches their needs and provides to users, a software homogeneous grid.

In this article, we present the environment deployment tool Kadeploy2 which introduces a new way of exploitation of clusters and grids. We initially make a research for the need of an environment deployment toolkit and present related works. In Section 3, we describe the architecture of Kadeploy2 and present the deployment process. Section 4 presents optimizations of the default deployment procedure and a global method to support deployment of various operating systems. In section 5, we present deployment scenarios and evaluation results, acquired from tests conducted on different clusters of the Grid5000 experimental platform [13]. Finally, conclusion and perspectives are presented.

2 Motivations and related work

An HPC cluster uses a multiple-computer architecture that features a parallel computing system consisting of one or more master nodes and computing nodes, interconnected in a private network system. The master node acts both as a server and a gateway to the outside world.

1-4244-0054-6/06/$20.00

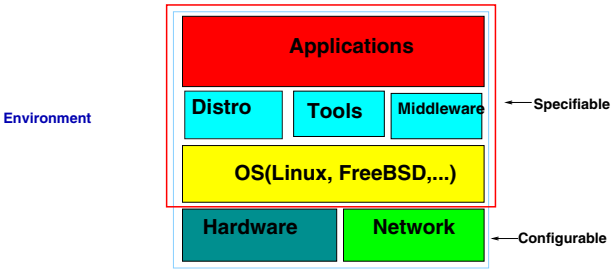

Figure 1. Software environment.

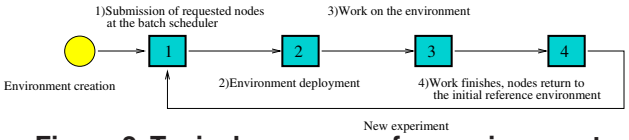

Figure 2. Typical sequence of an environment deployment.

In general, an HPC cluster provides a certain number of central services concentrated on one or more servers: for example a resource management system (also called Batch Scheduler), (ex: OAR [12], PBS [5], SGE [7], ...), a Network File System (NFS) and an authentification service (LDAP, NIS, ...). The computing nodes provide the necessary environment for the execution of the applications submitted by the users.

In the case of a computing grid, it is generally necessary to associate an additional middleware, allowing the logical interconnection of the various clusters which do not have necessarily the same services. For that reason, the environment present on the computing nodes is generally different from the one cluster to the other.

The principal solution currently suggested is that of the Globus [14] project which gathers a complete set of services and middlewares making it possible to incorporate various clusters. Initially this approach finds a certain "heaviness" in its installation and maintenance and in addition it does not solve the problem of the environment heterogeneity of the calculation nodes between clusters.

Another solution in the aggregation of cluster, is proposed by the *light grids*. In these infrastructures the step is to simplify the general problem of the grids. This simplification generally goes through a certain homogeneity of services and administration procedures: for example by adopting the same services and configurations on all clusters.

In our research, we have examined a number of configuration technologies, illustrating a range of different approaches dealing with large-scale configuration management. The *Gridweaver* [3] and the *Datagrid* [1] projects reviewed some management tools and techniques, currently available for Large Scale Configuration.

Tools like *Rocks* [24], *LCFG* [10], *Quattor* [21] and *Oscar* [22] are frameworks for managing large scale computing fabrics. They are systems that provide automated installation, configuration and management of large-scale parallel systems. Most of those systems are based on standard node specific installation setup tools like *Kickstart* [25] or *SIS* [8]. *SIS* is a collection of software packages designed to automate the installation and configuration of networked workstations.

The *SIS* tool suite is composed of three packages: *SystemInstaller*, that generates on the frontend the image with the operating system to install into nodes, *SystemImager*, that is responsible to copy the image into nodes and *SystemConfigurator*, that performs a post-install configuration of the nodes once the images have been installed. *SIS* uses a database to store the information about the cluster nodes from which it obtains the cluster configuration information.

Moreover, tools like *Partimage* [4], *g4u* [2] , *Partition Repositories* [20] and *Frisbee* [11] are technologies that focus on disk and partition imaging of single machines on a cluster environment. Among the techniques Frisbee uses are an appropriately-adapted method of filesystem-aware compression, where Frisbee currently handles BSD FFS, Linux ext2fs and Windows NTFS filesystems. If a partition is recognized, a filesystem-specific module is invoked to scan the filesystem free list and build up a list of free blocks in the partition. If a partition is not recognized, it treats all blocks as allocated.

On the other hand, alternative design approaches for large scale configuration based on virtualization and similar techniques are proposed by technologies like VMPlants [19], XenoServer [18], Dynamic Virtual Environments [17] and Condor GliedIn mechanism [15].

Finally, projects like *Stateless Linux* [6] deal with interesting issues on the field. Applications run locally, taking advantage of the processing power that already is well-distributed across most computers. Linux can be configured to work without a local hard drive. In that case the root filesystem is mounted readonly over NFS, shared among multiple computers and the user home directories for this system are also NFS-mounted. This diskless deployment is the simplest example of a stateless Linux configuration. In this prototype approach no state exists on single computers; all state is centralized.

There are currently many tools designed for the exploitation and administration of large-scale parallel systems but all of them use the environment deployment operation only during the startup phases or the programmed phases of maintenance. In addition the solutions provided seem not robust enough.

3 Deployment tool Kadeploy2

3.1 A new way of exploitation based on the deployment operation

To solve the central problem of software environment homogeneity among the computing nodes of a cluster or a

light grid, our proposal is to extend the capacities of the deployment operation. A software environment, contains all the software layers like the operating system, the libraries, the middlewares and the applications. Currently, the deployment operations on a cluster take place only at the time of the startup phase or the programmed phases of a cluster maintenances. Our proposal deals on how any user may use this operation when the environment provided on the cluster, is not the appropriate. Thus, any user accustomed to an environment may consequently use this environment on another cluster as long as it has the same type of processor.

As a matter of fact, this approach introduces a new way of cluster and grid exploitation. The user will not have to make modifications in every computing node of a cluster or a grid. Simply deploys the environment on every single node and thus proceed with the experiment computations.

To allow this approach work, it is necessary to introduce a tool for fast and robust deployment, having an access control to the operations of deployment and a sufficiently simple method of environment creation.

3.2 Architecture and Principles

This tool uses the traditional protocols for network booting: PXE, TFTP, DHCP. As we can see on figure 3, architecture of Kadeploy2 is designed around a database and a set of specialized operating components. The database is used to store all necessary information for the deployment process, the computing nodes and the environments. At the same time, the code is written in Perl, which is perfectly suited for system tasks. In addition uses a fast mechanism of environment diffusion which depends slightly on the number of nodes. This mechanism is based on a pipeline approach (chain of TCP connections between nodes) [23]. This enables operations of deployment on large clusters (1000 nodes).

Figure 4, shows a concurrent deployment of two computing environments on a cluster. The deployment process will write a complete environment, on a partition of the disk of each computing node, which will be followed by a reboot on this partition. The process ensures that the partition of the disk where the reference environment of the node is installed, remains intact during diffusion. To guarantee a greater function reliability, Kadeploy2 tool directs clusters to be coupled with remote mechanisms of hardware reboot. Thus, if a particular problem occurs on one or more nodes during a deployment, a restarting on the reference partition is ordered automatically, on defected nodes.

An environment is created very simply by making an archive of the root partition in compressed `tar` format. To ensure a high level of portability and to permit that an environment is usable on various clusters of similar processor architectures , the environment should not contain information corresponding to the initial cluster. That is possible because the majority of the services have autoconfiguration mechanism (ex: protocol DHCP for the network) and the majority of the operating systems have hardware autodetection mechanisms making it possible to adapt to the minor differences (network cards, disks...). For the services that lack autoconfiguration procedure during the deployment, a procedure known as post-installation process supplements the parameter setting.

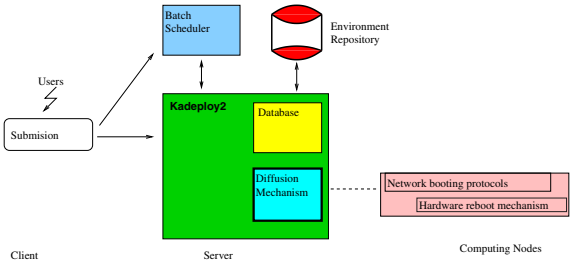

Figure 3. Kadeploy2 Architecture

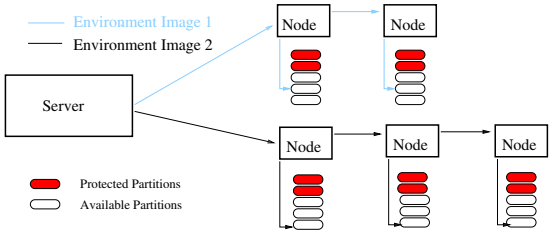

Figure 4. Concurrent deployment of two Computing environments on a cluster using the kadeploy2 toolkit

Finally, a permissions control authorizes the user to use the deployment operations only on the nodes which were allocated to the user by the resource management system (batch scheduler). This control is achieved by the interconnection of the deployment tool and the batch scheduler.

3.3 Deployment Process

The steps of the deployment procedure along with the different states of a computing node, during the process, are featured on figure 5.

The first step of the deployment procedure, is the user submission of the request to the batch scheduler for a specific number of computing nodes with the option of deployment. This is followed by writing appropriate information on the database. The third step, consists of all the permission controls for the computing nodes and their partitions just before the issue of the first reboot on the nodes. On this point, it has to be noted that the user has permission to deploy on certain partitions excluding the reference partition where the reference environment is installed.

Minimal system. On the following step, the deployment process uses the traditional protocols for network booting: PXE, TFTP, DHCP and issues a boot of a previously pre-

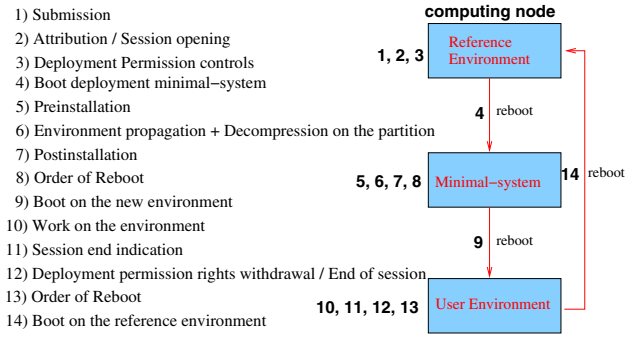

Figure 5. Default deployment procedure using kadeploy2 toolkit

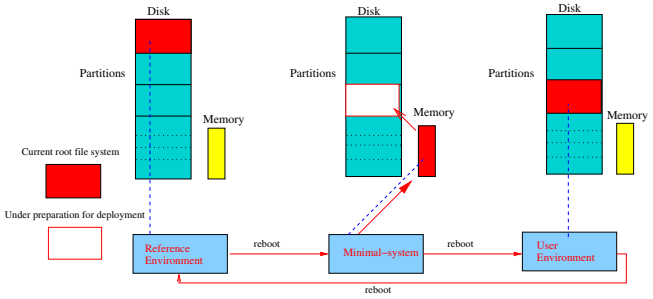

Figure 6. Different states of a computing node of a cluster during the default deployment procedure with kadeploy2 toolkit.

pared Linux minimal-system (initrd/mini kernel) mounted on RAM, on each computing node under deployment. This minimal-system is appropriately equipped with all that is needed (diffusion mechanism software, pipes, ...) in order to make all the necessary disk modifications, copy the new environment on a partition and configure it for deployment.

Preinstallation. The fifth step, consists of the preinstallation phase. In more detail, the deployment process with the help of the minimal-system of each node sends the preinstallation scripts that have to be executed on each node, by using the diffusion mechanism described. The execution of each script is then launched, by using the parallel launcher [23]. The scripts executed, make some necessary controls and partition the disk.

Environment Transfer+Installation. On the next step, the deployment process prepares the partition that is going to be deployed, propagates the image of the environment on every node by using the diffusion mechanism and uncompresses it on the already prepared partition.

Postinstallation. For the services that do not have a procedure of autoconfiguration (ssh keys, ldap, ...), the phase of postinstallation is issued on the seventh step. In more detail, just like the preinstallation phase, the deployment process with the help of the minimal-system of each node and using the diffusion mechanism, sends the postinstallation scripts and files that have to be executed or copied on each node. The execution of each script is then launched by using the parallel launcher.

On the eighth step, the process prepares the reboot on the partition where the new environment is installed by generating the relevant *grub* image, which is responsible for loading and transferring control to the operating system kernel software. On the next step, by using the network booting protocols (PXE, TFTP, DHCP) the process reboots on the modified partition where the new desired environment is installed.

After working on the environment, the user indicates the end of the session where the relevant information is written on the database, the deployment permissions on the nodes are dropped and the order of reboot on the reference partition is ordered. For those last steps, the deployment process cooperates with the batch scheduler. Finally, all the computing nodes reboot on the reference partition where the default computing environment of the cluster remains intact. Figure 6 shows an analytical view of the different states of a computing node during the default deployment procedure using the kadeploy2 toolkit.

Robustness. The deployment procedure mechanism introduced by Kadeploy2 toolkit is robust and concrete. To achieve this, from the hardware side, Kadeploy2 directs clusters to be coupled with remote mechanisms of hardware reboot. By this, it is guaranteed that no matter what software failure might arise the process will always be capable of rebooting the node and return the machine on its initial condition.

On the software implementation side, Kadeploy2 is equipped by timeout loops that permit the process to control the node's transitions of states. In more detail, every parallel launch of a command on a cluster is equipped with a timeout beyond which the node is excluded of the deployment and the procedure continues with the rest of the nodes.

3.4 Deployment process optimizations

A fast deployment execution time is very important for the viability of this approach. The study conducted on [16] proposes optimizations for the default deployment procedure method. The proposal is to minimize the deployment process execution time by eliminating the reboot phases of the procedure.

Based on results of experiments, we realized that steps 4,9 and 14, of the deployment procedure steps, spend a disproportionally execution time. Those are the 3 reboots that take place in the deployment procedure. Another step that seems slow on execution time, but not as slow as the reboot stages, is the 6th step of image environment propagation and decompression. As was already mentioned on the previous section, the deployment process uses the diffusion mechanism to propagate the image among the cluster nodes. The diffusion mechanism that is used is very fast and it slightly

depends on the number of nodes. Hence, this step needs no further optimization for execution time improvement.

We have considered to add 2 different methods as an option to the default method of the Kadeploy2 deployment procedure. *Nomini* method eliminates the 1st reboot and *pivot* eliminates all 3 reboots. The user has the opportunity to choose the preferable method according the experiment needs. Nevertheless, the optimizations are appropriately equipped to guarantee robustness.

Concepts and Implementation. Studying the design concepts of Kadeploy2 tool and its default deployment procedure as presented on previous section we understand the reasons of using a deployment minimal system which is mounted on memory: To prepare the disk and the partitions for the new environment to be installed. Hence, it can be possible to generate a deployment without booting a deployment minimal system. For this, we have to sufficiently equip the reference environment to play the role of minimal system. This means that the reference environment must be able to prepare the new partition, transfer and decompress the image of the new environment on this partition and configure the new environment for deployment. We can then eliminate the 1st reboot and proceed directly to the 2nd reboot on the newly installed environment. Figure 7, enumerates the steps and shows the different states of a computing node, during the nomini optimized deployment procedure.

The second optimization method *pivot* (figure 8), goes further and tries to "gain" all 3 reboots of the deployment process, by changing the root file system. Initially, it depends on the first deployment optimization.

In more detail, *pivot* method starts just after the execution of the postinstallation phase on the nomini optimization method. At this exact point the 2nd optimization method continues by changing the root file system without making a reboot. For this the system command `pivot_root` [9] is used, which is a UNIX system command that moves the current root file system to the directory for the old file system and makes the new root file system the current root. Moreover since the procedure is reversible we can easily return to the initial reference environment without making a reboot, hence eliminating the 3d reboot of the default deployment procedure.

Robustness and Constraints. On the issue of robustness, we want to assure that the new optimization methods are reliable and concrete. To guarantee that, we follow the guidelines of the default deployment procedure. Thus, the method is coupled with timeouts on every parallel command launch.

On the other hand, there are many constraints, especially in the 2nd optimization method *pivot*. The most important constraint that both optimization methods have, is the fact that the reference environment has to be equipped with the diffusion mechanism for the image propagation on the

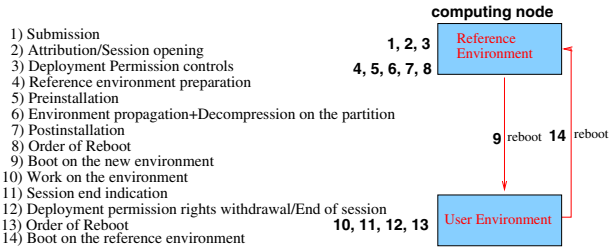

Figure 7. Nomini optimized deployment procedure using kadeploy2 toolkit.

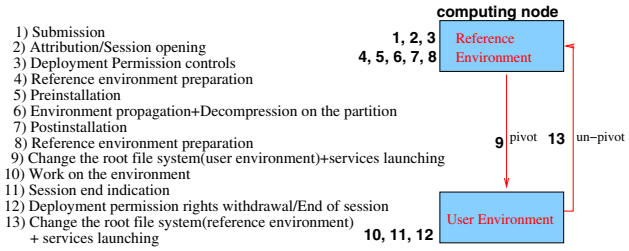

Figure 8. Pivot optimized deployment procedure using kadeploy2 toolkit.

nodes. Another constraint arises because the optimized deployment methods are unable to make a deployment on the same partition, where the environment is installed. Moreover, the 2nd optimization method introduces an additional important constraint, that the desired environment has to function with the same kernel and the same kernel parameters as the reference environment. This issue is very important and crucial and limits the method to be used on limited occasions only.

4 Performance Evaluations

The experiments and the performance measures which are presented in this section have been effectuated upon the Grid5000 experimental grid [13]. In more detail we have used two different clusters of the infrastructure: the IBM cluster (AMD Opteron biprocessor 2GHz, 2G de RAM, Myrinet/Gigabit Ethernet) and the GDX cluster (AMD Opteron biprocessor 2GHz, 2G de RAM, Gigabit Ethernet).

We conducted two different experiments to evaluate kadeploy2 toolkit. The first experiment measures the deployment procedure using only one cluster of the Grid5000 platform and the second measures the deployment procedure on the grid level. For that reason we used 2 different clusters of the Grid5000 platform.

4.1 Measures of the deployment procedure on cluster level

The objective of this experiment is to measure the deployment procedure execution time considering the steps from the beginning of the deployment session until the boot on the new desired environment. The deployment procedure execution time depends not only on the performance

of Kadeploy but also on the environment to be booted (environments can have different configurations and run different set of services). In this case we used a simple environment without complex services.

In more detail, the experiment measures the 5 most ”time-consuming” steps of the deployment procedure and presents results for the default method and the two optimization methods proposed. The figures that follow present the deployment procedure execution time according to the number of nodes, for a simple kernel without service and a cluster of 180 nodes (GDX).

Initially, in the experiment we extracted the average execution time of 10 continuous repetitions of a deployment, for the same number of nodes. As it was expected, the results were figures that had great ”imbalances” in their curves. This is a result of the sensitivity, the complexity and the number of the different protocols used in the deployment procedure along with the security precautions taken by kadeploy2 tool, which makes the process sensitive to alterations.

Hence, in order to demonstrate our approach validation, we present the figures acquired by extracting the medium value of the execution time of 10 continuous repetitions of a deployment, for the same number of nodes.

Figures 9, 10 and 11 represent the experiment effectuated on GDX cluster of 180 nodes for each deployment method respectively: default, 1st optimization *nomini* and 2nd optimization *pivot*. The figures decompose the deployment procedure execution time in 5 steps:

1. the time to boot the minimal system for the environment preparation or time for the first check for ready nodes (vertical line),
2. the time to prepare the disk partitions before the installation of the user environment (inclined cross),
3. the time to propagate and copy the user environment archive (star),
4. the time for the postinstallation procedure (empty square),
5. the time to boot the new user environment or the time change the root file system on the new user environment and launch the services (filled black square).

The figures show valuable results. First of all, the boot time depends on the number of nodes. This is because the boot time is different for all machines and we consider only the slowest one. In contrast, the disk preparation, the environment transfer and the postinstallation procedure increase negligibly with the number of nodes. On the other hand, the curves show that the mechanisms of the optimization methods are working according to our expectations.

Figure 12, shows the total execution time of the kadeploy2 deployment procedure according to the number of nodes, for the default method and the 2 optimization methods, on the GDX cluster. The featured chart validates our expectations for both optimization methods proposed, and show us that all deployment procedure methods introduce a very good scalability according to the number of nodes. As expected the *pivot* optimization method is the fastest with a difference of about 160 seconds from the default method. The *nomini* optimization method has a difference that ranges from 70 to 100 seconds from the default deployment procedure method which is a very good result, considering the simplicity of its constraints according to the default method. On figure 12 we can also observe that the optimization benefits does not grow proportionally to the system scale. This is a result of the ”first check” delay as we can see on figures 10 and 11. This problem is going to be addressed on a following version of the tool.

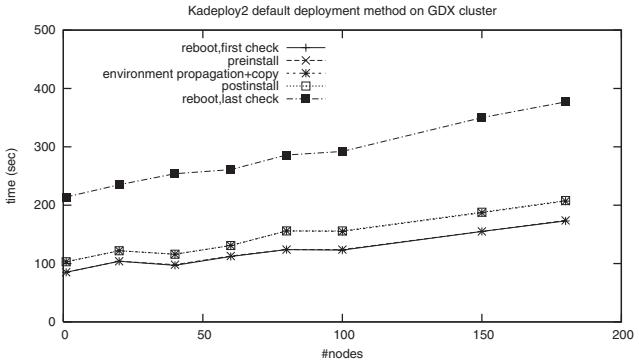

Figure 9. Kadeploy2 default deployment method

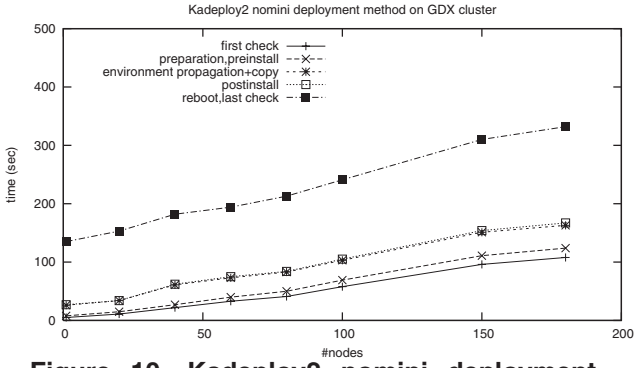

Figure 10. Kadeploy2 nomini deployment method

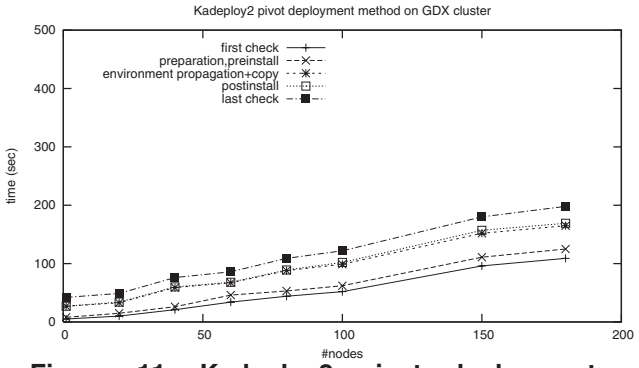

Figure 11. Kadeploy2 pivot deployment method

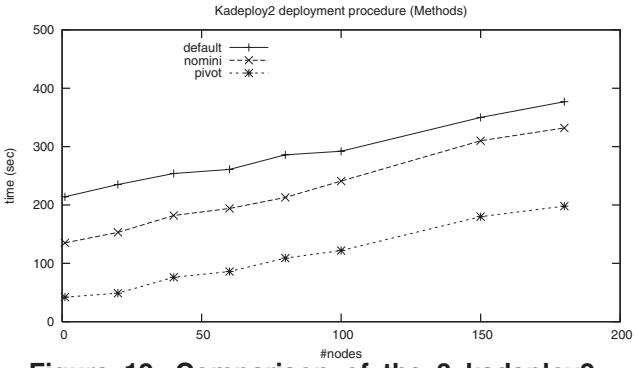

Figure 12. Comparison of the 3 kadeploy2 tool's deployment methods

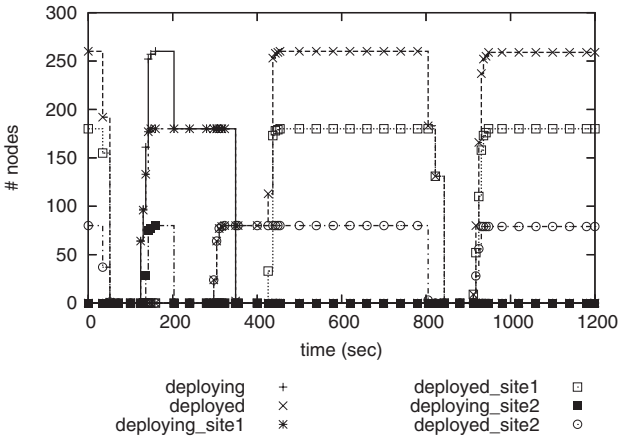

Figure 13. Time diagram for the deployment and reboot of a user environment on 2 sites.

4.2 Measures of the deployment procedure on grid level

Figure 13, presents the time diagram of a deployment and reboot phase involving 2 Grid'5000 sites for a total of 260 nodes (180 nodes in site 1 (GDX) and 80 nodes in site 2 (IBM)). The vertical axis corresponds to the number of nodes in deployment and reboot state. At t=0s, all the nodes are running an environment. At t=30s, a deployment sequence is issued. At t=50s, all nodes are rebooting the deployment kernel. At t=160s all nodes have rebooted and are preparing the user partition. The clusters start the second reboot at t=200s for site 2 and t=340s for site 1. Site 2 nodes are rebooted with the user's environment at t=320s. All nodes are rebooted with the user's environment (including Site 1) at t=450s. At t=800s, the user experiment is completed and a reboot order is issued making all nodes rebooting default environment. This figure demonstrates that the current "boot-to-boot" time at the Grid level (450 seconds) is well below a 10 minutes mark.

5 Work in Progress

5.1 Support of various operating systems

Currently, Kadeploy2 is implemented to support the deployment of environments that use a GNU/Linux operating system and ext2 file system. The toolkit's aim is to introduce a new way of exploitation of clusters and grids by making possible to the users to deploy and use the environment of their choice. Thus, Kadeploy2 should be implemented to support various operating systems (FreeBSD, Windows, Solaris, Mac OS X, etc) and different filesystems (UFS, HFS, ZFS, etc).

Any operating system which Linux supports creation of, read/write access to and cohabitation on the same disk(different partitions), their filesystem should be able to be supported by Kadeploy2 in some manner.

The first fact that we consider is the partition scheme of each node, where the reference partition has a Linux operating system and an ext2 filesystem environment installed. This reference Linux environment has to be kept untouched.

Thus to integrate non-Linux operating systems (and non-ext2 filesystems) upon Kadeploy2 toolkit, the deployment procedure has to appropriately construct a partition scheme that will keep the reference partition without changes and can bear a new operating system (and filesystem) on a different partition.

Partition imaging procedure has to be unaware of the filesystem that it compresses, thus making the method widely applicable and scalable. For this, the Unix `dd` command can be used which works on disk-device level where there is no restriction on the data layout of a disk and everything can be replicated.

Furthermore, this new extended deployment procedure, has to propose a method of environment construction for this operating system that is going to support. Finally, since Kadeploy2 uses *grub*, as the network bootloader, for the deployment procedure, the filesystem of the operating system to be installed, must be supported by *grub*.

Hence, we concluded to a flexible global method approach that can be applied for FreeBSD and Solaris, which are two widely used operating systems on high performance parallel systems. The deployment procedure of this global method can be summarized by the following stages:

1. Boot on the deployment minimal system.
2. Write in the start of a large (20G) primary partition a minimal system of the OS that it is going to be deployed, using the `dd` command .
3. Boot on the new OS minimal deployment system.
4. Do the appropriate changes on the disk partitions according the needs of the new filesystem and write the normal deployment environment.
5. Reboot on the new OS environment.

This method will be the base for various operating systems integration on kadeploy2 toolkit.

6 Conclusion

Nowadays, Cluster and Grid Computing have become widely used infrastructures, that gain an ever increasing world recognition. As computer hardware has become more powerful, cheaper and smaller, the number of individual nodes that comprise such infrastructures, is increasing rapidly.

This new era of computing technologies has generated numerous challenging issues, specially in the procedures of exploitation and administration. In the automated installation and configuration of those infrastructures, a number of important tools solve critical problems and propose innovative ideas.

In this article, we presented Kadeploy2 environment deployment toolkit which provides automated software installation and reconfiguration mechanisms. The main contribution of kadeploy2 toolkit is the introduction of a prototype idea, aiming to be a new way of cluster and grid exploitation. That is to let the users concurrently deploy computing environments exactly fitted to their experiment needs, on different sets of nodes.

This article presented the architecture, the mechanisms and the deployment procedure steps of kadeploy2. Since the deployment execution time is a very important aspect for the viability of this approach, the main work effectuated was to propose optimization methods for the deployment procedure and measure the performance of the toolkit. Multiple performance measurements were conducted. They validated our approach, achieved our expectations and generated ideas for deeper optimizations.

In addition, a first preliminary study upon the aspect of multiple OS integration on kadeploy2 toolkit, indicates to consider further research of that issue. We also found that there are areas that need to be more thoroughly examined. One such area is the ”quality of service” of the approach. Since the tools' objective is to provide flexibility and ease of use, a more delicate way of environment creation and update method is needed.

References

[1] *DataGrid Report on Current Technology*. https://edms.cern.ch/file/332371/1/datagrid-04-d4.1-0101-3_0.pdf.

[2] *g4u - Harddisk Image Cloning for PCs*. http://www.feyrer.de/g4u/.

[3] *GridWeaver Project*. http://www.gridweaver.org/.

[4] *Partition Image for Linux*. http://www.partimage.org/.

[5] *Portable Batch System*. http://www.openpbs.org/.

[6] *Stateless Linux*. http://people.redhat.com/ hp/stateless/ StatelessLinux.pdf.

[7] *Sun Grid Engine*. http://gridengine.sunsource.net/.

[8] *System Installation Suite*. http://sisuite.org.

[9] W. Almesberger. Booting linux: The history and the future. Ottawa Linux Symposium 2000, July 2000.

[10] P. Anderson and A. Scobie. LCFG - the Next Generation. In *UKUUG Winter Conference*. UKUUG, 2002.

[11] C. Barb, J. Lepreau, L. Stoller, M. Hibler, and R. Ricci. Fast, scalable disk imaging with frisbee, June 14 2003.

[12] N. Capit, G. D. Costa, Y. Georgiou, G. Huard, C. Martin, G. Mouni, P. Neyron, and O. Richard. A batch scheduler with high level components. In *Cluster computing and Grid 2005 (CCGrid05)*, 2005.

[13] F. Cappello, F. Desprez, M. Dayde, E. Jeannot, Y. Jegou, S. Lanteri, N. Melab, R. Namyst, P. Primet, O. Richard, E. Caron, J. Leduc, and G. Mornet. Grid'5000: A large scale, reconfigurable, controlable and monitorable grid platform. In *Grid2005 6th IEEE/ACM International Workshop on Grid Computing*, 2005.

[14] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, 1997. ftp://ftp.globus.org/pub/globus/papers/globus.pdf.

[15] J. Frey, T. Tannenbaum, M. Livny, I. T. Foster, and S. Tuecke. Condor-G: A computation management agent for multi-institutional grids. In *HPDC*, page 55, 2001.

[16] Y. Georgiou. Environment deployment for high performance cluster computing. Report of master in ”mathematics, computer science”, University Joseph Fourier, September 2005.

[17] K. Keahey, K. Doering, and I. T. Foster. From sandbox to playground: Dynamic virtual environments in the grid. In *GRID*, pages 34–42, 2004.

[18] E. Kotsovinos, I. Pratt, S. H, and T. Harris. Controlling the xenoserver open platform, Apr. 20 2003.

[19] I. V. Krsul, A. Ganguly, J. Zhang, J. A. B. Fortes, and R. J. Figueiredo. VMPlants: Providing and managing virtual machine execution environments for grid computing. In *SC'2004 Conference CD*, Pittsburgh, PA, Nov. 2004. IEEE/ACM SIGARCH. (ACIS) Lab, University of Florida.

[20] C. Kurmann, F. Rauch, T. Hochschule, and T. M. Stricker. Partition repositories for partition cloning - OS independent software maintenance in large clusters of PCs. Technical report, Dec. 08 2000.

[21] R. G. Leiva, M. B. López, G. C. Meliá, B. C. Marco, L. Cons, P. Poznański, A. Washbrook, E. Ferro, and A. Holt. Quattor: Tools and techniques for the configuration, installation and management of large-scale Grid computing fabrics. *Journal of Grid Computing*, 2(4):313–322, Dec. 2004.

[22] D. Ligneris, N. Gorsuch, S. Scott, and T. Naughton. Open source cluster application resources (OSCAR) : design, implementation, June 28 2003.

[23] C. Martin and O. Richard. Parallel launcher for clusters of pc, parallel compting. In *Parco'01 (Parallel Computing), Naples*, 2001.

[24] P. M. Papadopoulos, M. J. Katz, and G. Bruno. NPACI Rocks: tools and techniques for easily deploying manageable Linux clusters. *Concurrency and Computation: Practice and Experience*, 15(7–8):707–725, June/July 2003.

[25] B. Schwarz. Hacking Red Hat Kickstart. *Linux Journal*, 108:??–??, Apr. 2003.