Document downloaded from:

http://hdl.handle.net/10251/103725

This paper must be cited as:

López Rodríguez, PJ.; Baydal Cardona, ME. (2017). On a course on computer cluster configuration and administration. Journal of Parallel and Distributed Computing. 105:127-137. doi:10.1016/j.jpdc.2017.01.009

The final publication is available at

https://doi.org/10.1016/j.jpdc.2017.01.009

Copyright Elsevier

Additional Information

On a Course on Computer Cluster Configuration and Administration

Pedro López[a,*], Elvira Baydal[a]

[a]*DISCA Department*
Universitat Politècnica de València
Camino de Vera, 14
46022 Valencia (SPAIN)

Abstract

Computer clusters are today a cost-effective way of providing either high-performance and/or high-availability. The flexibility of their configuration aims to fit the needs of multiple environments, from small servers to SME and large Internet servers. For these reasons, their usage has expanded not only in academia but also in many companies. However, each environment needs a different "cluster flavour". High-performance and high-throughput computing are required in universities and research centres while high-performance service and high-availability are usually reserved to use in companies. Despite this fact, most university cluster computing courses continue to cover only high-performance computing, usually ignoring other possibilities. In this paper, a master-level course which attempts to fill this gap is discussed.

It explores the different types of cluster computing as well as their functional basis, from a very practical point of view. As part of the teaching

*Corresponding author
Email addresses: plopez@disca.upv.es (Pedro López), elvira@disca.upv.es (Elvira Baydal)

methodology, each student builds from scratch a computer cluster based on a virtualization tool. The entire process is designed to be scalable. The goal is to be able to apply it to an actual computer cluster with a larger number of nodes, such as those the students may subsequently encounter in their professional life.
Keywords: computer engineering education, computer cluster configuration and administration, lab project

---

1. Introduction

Computer clusters are nowadays the most cost-effective alternative to build high-performance computer systems. A cluster is a type of parallel or distributed processing system which consists of a collection of interconnected stand-alone computers working together as a single, integrated computing resource [4]. Although the idea of using clusters to improve system reliability dates back to the 60s [23], its usage to improve system performance is more recent. The Beowulf project [3] built a $40,000 cluster with 16 personal computers that was able to reach 1 GFLOPs. Since then, computer clusters have become a low cost alternative to build high-performance systems.

The fact of being composed by several computers can be exploited in two ways. The traditional approach is to obtain an overall high-performance by combining the performance of each individual computer. In addition, the fact of having component redundancy aims to improve system availability. The excellent price-performance ratio of clusters has led to a widespread usage. Their configuration flexibility aims to fit the needs of multiple environments, from small servers to Small and Medium Enterprises (SME) and large Inter-

net servers. Popular Internet services are provided by large scale clusters, usually located in different places to provide resilience against natural disasters. On the other hand, the largest supercomputers are based on computer clusters. Since 2007, more than 80% of the TOP500 list of supercomputers [27] are clusters.

Taking into account their importance, computer engineering curricula may include some subjects about clusters [12]. The topics to cover will be related to system architecture, networking, operating systems, parallel programming and applications. Although the core of these topics will have already been taught previously, a cluster subject should apply them to a particular system: the computer cluster. Instructors have a golden opportunity to integrate and apply the knowledge acquired in previous core subjects.

In particular, in this paper, we describe a course on computer cluster configuration and administration. As cluster computing is an advanced topic [13], the proposed course is targeted to graduate students, although its contents could be applied as well to an advanced undergraduate course in Computer Engineering. Selected topics of the course include cluster configuration, system installation, running sequential and parallel applications, storage, load-balancing and high availability. In addition, a computer cluster course requires hands-on sessions in a realistic environment to readily apply the concepts taught in the classroom. In particular, we propose building a computer cluster from scratch, installing the operating system, configuring the services and installing the required software packages. Taking into account the typical usage "flavours" of clusters [11], two approaches are simultaneously considered. The first one is a high-performance/high-throughput

computing cluster where several users will run their sequential or parallel applications. The second one is a high-availability/load-balanced Internet server. This second usage of clusters is often ignored in cluster related subjects, albeit most cluster installations fit in this second category. Another possible flavour to consider in the next future is a cluster specifically designed for storing and analyzing huge amounts of unstructured data (i.e. a big data cluster).

The straightforward approach to implement the course consists in using a real hardware infrastructure to provide this environment. With this approach, the students have the opportunity to see, touch and interact with actual hardware components of a computer cluster: processing nodes, storage nodes, network switches, PDUs, KVM switches, cluster cabinet, etc. Moreover, they can directly experience the issues related to power consumption and space management.

However, the problem of using real hardware in lab sessions is that each student should use his/her own cluster, composed by several nodes. Sharing a physical cluster is not an option because along the project development the students must have root privileged access to the system to allow them to modify system configuration. Therefore, their work may interfere with each other. Indeed, their activity will result in system reboots or even system crashes with catastrophic consequences to the work of other students. As a part of their learning process, these crashes will appear occasionally along the project. However, the alternative of purchasing a cluster for every student has a prohibitive cost for the University and will result in a very inefficient resource utilization. Indeed, this hardware may remain largely idle during

holidays and weekends and between major assignment due-dates [14].

Fortunately, the availability of free virtualization software and low-cost but high-performance personal and laptop computers makes possible for the students to run a cluster of virtual machines on the available lab desktop computers or even using their own laptops. While the students need administrative privileges to install and manage their virtual machines, they need no privileges on the host machine. In addition, they can update or install cluster software and infrastructure without compromising the work done by other students. However, notice that the use of virtual machines to build a cluster is only for academic purposes and by no means we want to suggest that an actual cluster will be built using virtual machines.

To reduce system requirements, the size of the cluster should be small. As a minimum configuration, the cluster that each student builds is composed of two director nodes (to provide high availability), three server nodes plus a storage node. Despite the reduced number of server nodes, the course is designed as if a large computer cluster were deployed. For instance, installing the system individually on each server node is easy for three servers, but is unfeasible for a large cluster. Thus, a scalable way of installing the system and configuring cluster nodes is proposed. More details can be found in Section 3.

The main contribution of this paper is the proposal of a computer cluster subject that i) considers not only the traditional high-performance/high-throughput computing usage of clusters but also other usage flavours, like their usage as high-availability/load-balanced Internet server; ii) deploys a computer cluster based on virtual machines from scratch, installing and con-

figuring all the required packages. As a consequence, the course contributes to bridging the gap between industry and academia by teaching those contents that are used in today servers from a practical point of view. This approach has been recommended by ACM and IEEE Computer Society in their coming curricula in Computer Engineering [12].

The rest of the paper is organized as follows. Section 2 describes the syllabus. Section 3 presents the hands-on part of the course detailing the steps followed by each student to build his/her cluster. Section 4 discusses the teaching methodology and evaluation criteria. Section 5 discusses some related work and, finally, some conclusions are drawn.

2. Description of the Subject

2.1. Context

One of the key points to consider when instructors have to choose which topics to include in a post-graduated course is the previous background of the students. The proposed course belongs to a Master program in Computer and Network Engineering [18], lectured by the Computer Engineering Department at the Universitat Politècnica de València. It is targeted to bachelors in Computer Science, Computer Engineering or Telecommunications Engineering. Usual topics covered in these bachelors include programming, computer organization and architecture, computer networks and operating systems. Moreover, instructors have to consider the topics included in other subjects of the same Master. In particular, in our Master, interconnection topologies used in high performance networks are already covered. Topics like security in distributed systems or GPU programming are also developed

in other courses of that master. On the other hand, there is also another Master focused on parallel computing and cloud computing [19] at the same university, that offers our course as an optional subject.

2.2. Syllabus

Taking into account the above considerations and the main objective of the course of trying to provide experience about how to design, configure and manage a computer cluster (with the two aforementioned usage flavours), we choose a syllabus that will allow a course with a strong practical component without neglecting the theoretical basis.

2.2.1. Introduction

Clusters represent a good option to get cost-effective systems. They are the dominant system architecture in HPC with 85% of the TOP500 list [28] but they are also present in many companies offering web or database services. This unit reviews the factors that have enabled this success: improvements in personal computer performance and operating systems. In addition, the limitations in the performance of uniprocessor systems and the need for high performance in applications are also discussed. The main goal of the unit is to establish what a cluster is, what kind of applications it can run and which are the main factors that have driven its popularity. Moreover, in order to clarify their study, clusters are classified into four groups: high throughput computing, high performance computing, high availability and high performance service [11]. The main characteristics of each group are presented.

2.2.2. Cluster Configuration

This unit is devoted to the selection of the cluster components, analysing the different factors to consider in that selection. It presents an overview and some of the elements will be developed in detail in later units:

- Main components of the cluster nodes: processor (computing power, relevant benchmarks, power consumption, ...), memory and I/O.
- Interconnection network: Concepts about High Speed Interconnects (HSI). Latency, bandwidth and overhead of the network interfaces. HSI topologies are only mentioned as they are covered in another subject.
- Storage system: Evolution of network storage: Direct Attached Storage (DAS), Network Attached Storage (NAS) and Storage Area Networks (SAN).
- Auxiliary components: racks, KVM switches, PDUs, wiring and UPSs.
- Other aspects to consider: electrical power needs, room, noise, cooling and global cluster cost.

2.2.3. System Infrastructure

This unit summarizes basic computer network concepts applied to clusters. Main useful services in a cluster are reviewed, including DHCP, NTP, DNS, NIS, and SSH. General recommendations about where (on which nodes) execute that services and which services need redundancy are given. On the other hand, how to transport the information (data, management and control networks) as well as IP address assignment and NAT are discussed. Finally, how to improve network performance through the use of jumbo frames, as

well as channel bonding to increase link bandwidth or redundancy are also introduced.

2.2.4. System Installation

In this unit, Linux is presented as the operating system of choice due to performance and cost constraints. The distrowatch site [7] is used to show the available Linux distributions and the top ten most used. A set of criteria is shown as a guide to show the most appropriate distribution. Then, the Linux boot process is revisited, including disk partitioning and boot loader. Both GRUB2 and PXE are discussed, emphasizing their configuration files. Several alternatives for system installation are discussed as well. As stated above, the students must always keep in mind that they must install a large system, composed of a potentially large number of nodes. Therefore, manual or repeated installation processes are not acceptable. The approach that we propose is to prepare a PXE server and then boot the server nodes from the network. Once the server nodes are up, they can be remotely managed to prepare their hard disk and install the system on it. We will perform this kind of installation in the hands-on project. See Section 3 for details.

2.2.5. Storage Systems

Servers often use systems with multiple disks, connected locally or through the network. This set of units studies all the aspects related to storage in clusters. We can distinguish mainly two parts in that study.

First, high performance storage technologies are introduced, beginning with different alternatives to manage disk arrays. Logical Volume Manager (LVM) is used as an example of flexible disk array but without redundancy.

Then, we introduce RAIDs (Redundant Arrays of Inexpensive/Independent Disks) and DRBD (Distributed Replicated Block Devices) [8] to improve fault tolerance. Next, we review how to connect computers and storage. Broadly speaking, you have two different possibilities: direct attached storage through protocols like SCSI or SAS, or alternatively you can use network storage. In that case, protocols such as Fibre Channel, iSCSI, or Fibre Channel over Ethernet (FCoE) are widely used to access the remote disks through the network. The main features of each one are discussed, as well as their suitability depending on the type of cluster considered. Notice that InfiniBand is covered in another subject of the same Master.

The course also covers the file systems used in clusters. Two primary types of storage architecture are available: Network Attached Storage (NAS) and Storage Area Networks (SAN). Although many protocols for network-based systems exist, NFS [20] is the most widely used in the Linux environment. For this reason, we choose it as the NAS example to study. In addition, the limitations and problems of such protocols are analysed and compared with the advantages of SAN storage systems that enable better performance and data consistency for concurrent access. Cluster file systems, also known as shared-disk file systems, can provide not only advantages on scalability but also about system metadata, lock manager or fencing. Different examples of cluster file systems are compared: OCFS2 [21], PVFS2 [24] and GlusterFS [10].

2.2.6. Compute Cluster (I): Running Sequential Applications

One of the main goals of a cluster is running applications. This unit introduces job management systems, analysing two different approaches: MOSIX

[16] and HTCondor [6]. Both of them allow users to run sequential and parallel applications, doing dynamic load balancing between the cluster nodes.

With MOSIX, users have the illusion that applications run locally on the home-node where they were launched. Actually, MOSIX checks the cluster resources and migrates processes to the least loaded nodes. As load changes in the cluster or if a node is disconnected from the cluster, workload is dynamically adjusted, migrating processes to other nodes. Moreover, as MOSIX works transparently to the applications, users do not need to modify them. It allows users to use local commands on the home-node as if the processes were running locally. Probably, its main disadvantage relies on I/O, which is done via the home-node of the process, reducing performance for processes with significantly amount of I/O and/or file system access.

Alternatively, users may want to execute some applications in a non-interactive mode (jobs). Batch queuing systems help users to manage their jobs, allocating resources to jobs, providing a framework for submitting, executing and monitoring jobs and arbitrating contention for resources. Popular batch execution system for clusters are HTCondor or the SLURM workload manager [25]. The main features of job schedulers for clusters are explained. In particular, HTCondor defines different *running universes* for sequential and parallel applications, with distinct requirements and limitations, e.g. the "standard universe" requires to link applications with HTCondor I/O library but as a counterpart it allows checkpoints and remote I/O without needing a shared file system; on the contrary, the "vanilla universe" works transparently to the applications but it does not include that facilities. In addition, a summary of the HTCondor commands to submit and control

user jobs and the DAGMan mechanism to describe job dependencies are also introduced.

2.2.7. Compute Cluster (II): Running Parallel Programs

Although it is not the aim of this course to teach parallel computing, the students still need some knowledge about parallel applications. In this way, they will understand better what are the cluster resources that the execution of this kind of applications require. As an alternative point of view, we want the student to know the capabilities a cluster provide for running parallel applications. We devote two units to give a basic introduction to parallel programming.

The first unit deals with the message passing programming paradigm using MPI (Message Passing Interface) [17]. The structure of a MPI program is explained, the basic communication primitives (MPI_Send, MPI_Recv, MPI_Bcast and MPI_Reduce) are introduced and simple programs based on the concept of manager–worker are shown. A lab session completes the unit. A simple parallel program to compute the dot vector is provided and the students have to insert some send/receive primitives and develop a new version that exchanges blocks of data instead of scalars and makes use of collective communications.

The second unit is devoted to shared-memory parallel programming. In particular, the OpenMP programming model is presented [22]. Basic OpenMP directives to parallelize loops and sections are introduced. The shared and private scopes of variables and how to deal with critical sections are shown to the students. A lab session that computes the π constant by numerical integration is used as a working example to parallelize. To complete

the part devoted to parallel programming, a hybrid MPI–OpenMP version of the dot product is developed. This example illustrates the fact that current clusters are composed of several nodes and every node is composed of several processors and/or cores. A way of exploiting this structure is using message passing among nodes and shared variables between the cores inside a node.

2.2.8. Internet Server Cluster (I): Load Balancing

As stated above, clusters are often used to build high-availability/load-balanced Internet clusters. This unit is devoted to deal with load balancing. The Load Balancer (LB) or director is the set of hardware and software tools that allows a system administrator to implement usage policies and allocation of resources in the cluster. It distributes client requests among cluster members transparently to the clients. Therefore, it is one of the key elements in the cluster operation.

Depending on the packet information considered, load balancing can work at different layers of the OSI Architecture, offering different criteria for load distribution. Load balancing at level 4 uses only information of the IP protocol (layer 3) and transport protocols (layer 4), TCP and UDP, while working at application layer (layer 7) allows analysing the content of application messages, thus enabling more complex decisions. For example, for HTTP protocol, decisions based in URLs or based in cookies can be made. The main scheduling algorithms for both layers are explained. In spite of all the packets of a TCP connection will be sent to the same internal node, problems can arise. For example, with HTTP sessions spread between different TCP connections. This drawback can be solved using source IP affinity or persistence based on application layer information like HTTP cookies.

While when operating at layer 7 load balancer behaves as a reverse proxy, working at level 4 allows more possibilities for packet routing. Although NAT is the most frequently used one, Direct Routing and Tunneling are also introduced.

At the end of the unit, several examples of open source Linux LBs such as LVS and HAProxy are compared.

2.2.9. Internet Server Cluster (II): High Availability

The load balancer needs to know which nodes are available in order to submit them the jobs. Some load balancers as HAProxy make this function for themselves but others as LVS rely on external tools, like ldirectord. In particular, Cluster Resource Managers (CRMs) can be used. CRM is additional software that checks the node state and the node resource configuration, stops the node services when there is a problem and informs the load balancer of the available nodes. To meet those goals, the node state is usually verified through heartbeats or connections to the offered services. Moreover, the LB is a single point of failure that should be replicated and monitored to get a high availability system. Tools like keepalived or corosync can do this job.

In addition, concepts about how to solve problems with nodes as split-brain situations, fencing or Stonith are also introduced in this unit. Finally, pacemaker structure and configuration are shown as a CRM example.

3. Lab Project

This section presents the hands-on part of the course. As stated above, an academic cluster project based on the use of virtual machines will be de-

ployed. At the beginning of the course, the specifications of the computer cluster to be built are given to the students. Care must be taken to reduce the size of the cluster as well as the specifications of each node to be representative of actual requirements but also keeping complexity under reasonable limits. The virtual nodes of the cluster will be run either on the desktop computers of the lab or on the students' own laptops. At the time of writing, machines with a 4-core processor, 4 or 8 GB of RAM and more than 500 GB of disk storage are common. We propose a cluster composed of 6 nodes: two director nodes, three server nodes and one storage node. The two director nodes provide high availability. The server nodes do the computational tasks or provide the required service. The number of server nodes could be increased if resources are available on the host machine. Anyway, the course encourages a cluster deployment keeping in mind that the number of server nodes could be potentially very high. Finally, the storage node provides a common storage for user and system data.

Concerning the specifications of each node, it depends on the requirements of the operating system and applications. As the cluster is only used for academic purposes, we only consider the minimal requirements of the operating system. At the time of writing, for Ubuntu 14.04 Server LTS (see below) a minimum 192 MB of RAM and 1.4 GB of storage are required. We have successfully worked with uniprocessor configurations with 512 MB of RAM at the director nodes and 256 MB of RAM at the server and storage nodes. An 8 GB virtual disk is used in all nodes but the storage one, where 10 GB are provided.

The lab project consists of the following steps (see Figure 1). First, the

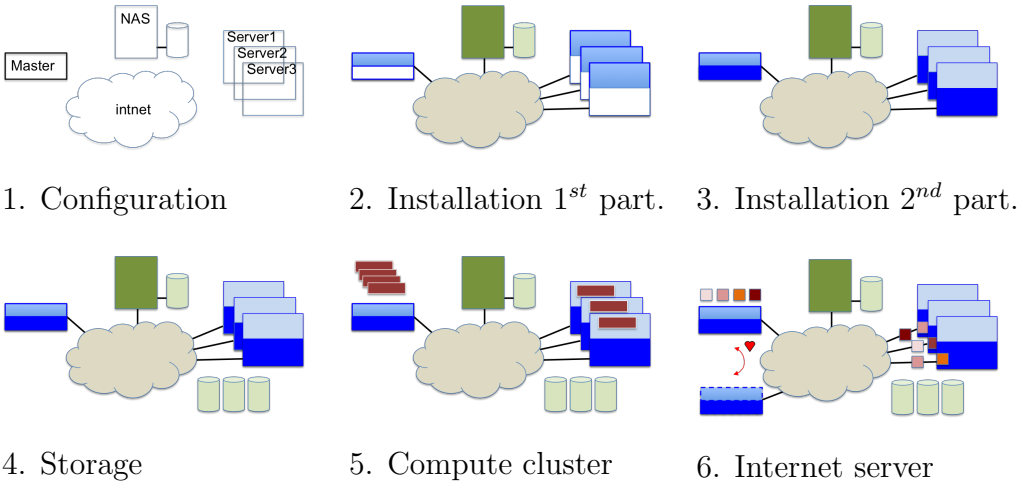

Figure 1: Lab project steps

virtual machines that will become the nodes of the cluster will be created and configured. Next, the operating system will be installed on the cluster. As stated above, we will install two different systems to allow the cluster to run in two different flavours. Once the cluster is configured and working, several storage alternatives are also configured and tested. As the first flavour corresponds to a compute cluster, we install the required packages to manage sequential jobs and run parallel programs in the cluster. Finally, we install and configure the corresponding software tools to provide load-balancing and high-availability that allow the cluster to work as an Internet server.

3.1. Cluster Configuration

As the very first step for building a cluster, its nodes must be carefully selected and configured. Therefore, in the lab project this step is resembled by creating and configuring the virtual machines. Any virtualization platform can be used. We have successfully used VirtualBox [29] since the first edition

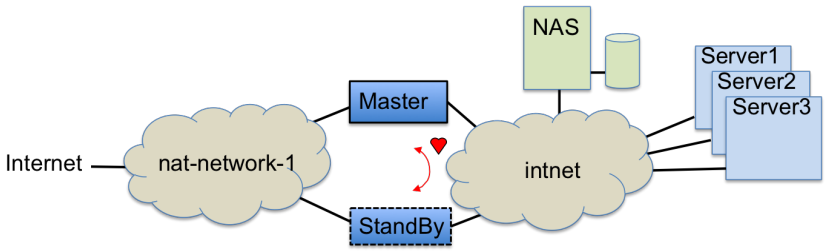

Figure 2: Block diagram of the cluster

of the course. VirtualBox is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2 and is available for Windows, Mac OS X and Linux hosts, thereby allowing the project to be developed on any desktop or laptop computer.

Using the visual interface of VirtualBox, the six nodes that compose the cluster are created (see Figure 2):

Master, StandBy: Director nodes. 512 MB of RAM. Boot order: DVD→HD, 8 GB of SATA storage. Two network interfaces (adapter 1 connected to NAT Network; adapter 2 connected to internal network).

Server1, Server2, . . . : Server nodes. 256 MB of RAM. Boot order: Network→HD, 8 GB of SATA storage. One network interface (adapter 1 connected to internal network).

NAS: Storage node. 256 MB of RAM. Boot order: DVD→HD, 10 GB of SATA storage. One network interface (adapter 1 connected to internal network).

NAT Network: "nat-network-1". DHCP enabled.

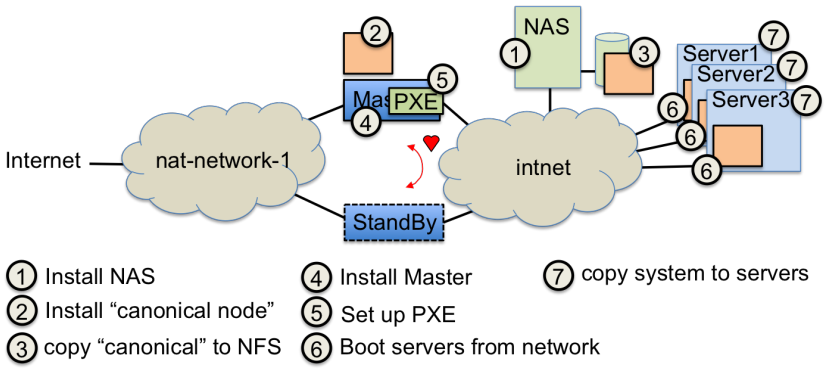

Figure 3: Installing the system on the cluster

Internal network: "intnet".

3.2. System Installation (I)

Once the virtual nodes are configured, the student is ready to install the system. Notice that taking advantage of using a virtualization platform, we could use the trick of installing one node and cloning it using the available tools. However, as this could not be feasible in a real cluster, we avoid it.

As operating system, we have successfully used Ubuntu Linux in several course editions. In particular, we choose the Long Term Support (LTS) server editions (14.04 LTS in the last course edition), which should be a proper choice in a real cluster.

The steps followed to install the cluster are shown in Figure 3 and summarized below. First, we will completely install the **NAS** node. Next, to install the system in the server nodes we will perform an installation of one server node (the "canonical node"). Afterwards, a copy of the "canonical node" system will be stored at the **NAS** node. Then, a PXE server will be set up at the **Master** node, to boot the server nodes mounting their root

filesystem on the **NAS** node. As the network is running, all the server nodes are accessible through ssh and the **Master** node will issue the proper commands to complete the installation. Once the installation of the server nodes is done, the PXE server is stopped and the machines are rebooted from their own hard disk, and the cluster is installed.

Let us analyze the process step by step. First, **NAS** node is individually installed by booting from an ISO image inserted into the virtual DVD drive. To fully take control of the installation process, the students are encouraged to manually define disk drive partitions (two partitions plus swap). After rebooting, the root user is activated, the network is statically configured and some additional packages are installed and configured (ssh and nfs server). At every step of the process, the students are given the command to issue and/or the files to configure, with a brief explanation of its purpose.

Next, the **Master** node is installed to generate the "canonical node" that will be used to clone the server nodes. Again, to fully take control of the installation, the boot loader (grub2) is manually configured. The SSH keys are generated and configured to allow public key validation and the /etc/hosts file is created. The system is now ready to be copied to the **NAS** node. Care must be taken to edit the network configuration in the just copied system to avoid bringing the network interface up automatically, as it will have already been brought up by PXE booting. In addition, the /etc/fstab file must be edited to mount in the server nodes the root filesystem on the **NAS** node through NFS.

Now, it is time to finish the **Master** node installation to become the cluster director node. The system variables and the iptables firewall are

configured to provide NAT to the internal network. A lightweight window system might also be installed to easy system interaction. Finally, the PXE support files are installed (syslinux, dnsmasq) and properly configured to mount the root filesystem on the **NAS** node. The dnsmasq utility will be also in charge of DHCP and DNS services. Finally, the PXE server is started.

Then, the server nodes will be started and booted up from PXE, mounting their root file system on the **NAS** node. The students can check the boot process from the screen messages and on the dnsmasq log file at the **Master** node. As soon as all the server nodes are started, we can proceed to install the system in all of them. We use two valuable and easy scripts referred to as psh and pscp ("parallel shell" and "parallel secure copy", see Figure 4) to launch a given command to all the server nodes and broadcast a file to all nodes, respectively. Although there are some packages (for instance, pdsh) that provide this functionality, the students can check by themselves how easy can be to install and admin a cluster by using simple tools. As an example, Figure 5 shows the command sequence issued from the **Master** node to partition and format the disks of the server nodes. Then, all the server nodes copy the root filesystem (mounted on the **NAS**) to their own disk. Once the copy is done, filesystems, network configuration, boot loader and hostname are set up. All these stuff is performed from the **Master** node, by using the psh script or some simple variations. Notice that the installation process is the same regardless of the number of server nodes. At this time, the PXE server is stopped on the **Master** node and the server nodes are ready to reboot. The system is installed.

When the cluster is booted from the 1st partition, it should work as an

HPC/HTC oriented cluster. Therefore, some support is required to ease user management. To do so and for the sake of simplicity, NIS service is installed in the system. In an actual cluster, NIS service will be provided by some administrative nodes. In our cluster, it is provided by the **NAS** node. In previous course editions it was installed on the director nodes, which can seem more logical. However, as the director nodes could be booted in two working modes and NIS is only required in the HPC/HTC mode, it can create some situations where some node hung for long waiting for the NIS server. Thus, in our project, the NIS server package is installed on the **NAS** node, and the NIS client package on the first partition of the director and the server nodes of the cluster. The configuration files are updated and some tests adding and deleting users are performed.

Alternatively, LDAP service could be used for user management. However, administration of LDAP server from command line is quite difficult. For this reason, it is advisable to install a GUI (Graphical Unit Interface) administration tool like phpldapadmin.

3.3. System installation (II)

At this step, another system is installed in the 2nd partition of the node hard disk (all nodes but the **NAS** one). The fact of having two different installed systems can be justified from different points of view. A first reason is fault tolerance. If there is a system corruption due to faults or even user errors, another partition comes to the rescue. However, there are more interesting reasons. For example, both a minimal and a production system could be installed. The minimal system helps to install and upgrade the production one. On the other hand, different systems or applications could

```
#!/bin/bash
#SERVER_NR   --> nr of servers
#SERVER_NAME --> server prefix
echo $0 $*

i=1
while [ $i -le $SERVER_NR ]
  do
    echo "=================="
    echo $SERVER_NAME$i
    echo "------------------"
    ssh $SERVER_NAME$i $@
    let i=i+1
  done
```

```
#!/bin/bash
echo $0 $*

i=1
while [ $i -le $SERVER_NR ]
do
  scp $1 $SERVER_NAME$i:$2
  let i=i+1
done
```

Figure 4: psh and pscp scripts

be installed in each partition. This latter approach is what we follow in the course. As already stated, the computer cluster could be used in two working modes. The first one corresponds to a compute cluster, where users submit and execute their jobs. The second one is an Internet server and will be installed in the 2nd partition. A web server package is required at the server nodes and load-balancing and high-availability packages will be installed at the director nodes.

The system will be installed in the 2nd partition following an easy approach. Once the cluster is started and operational from the first partition,

```
...
# HD Partitioning
# Master
echo "Partitioning, formatting servers ..."
echo "Take a copy of master partition table"
sfdisk -d /dev/sda > $NFS/root/sda.out

# Servers
echo "Partition from a file"
psh "sfdisk -f /dev/sda < sda.out"
echo "Format partition 1"
psh "mkfs -t ext4 /dev/sda1"
echo "Preparing swap"
psh "mkswap /dev/sda3"
psh "swapon /dev/sda3"
...
```

Figure 5: Example of psh script usage to clone partition table and prepare storage at servers

the second partition is formatted and the system is copied on it. The filesystem and grub boot loader configuration files must be updated. It is important to notice that all the commands will be issued on the **Master** node. We try to instill in the students that the cluster management should be centralized to minimize the administration effort.

3.4. Storage Systems

Once the operating system is installed in both cluster partitions, it is time to play with storage alternatives. A first enhancement to the NFS storage configuration consists of using link bonding. A second network interface is added to the **NAS** node and the system configuration is accordingly updated. The network bandwidth of the NFS node is doubled.

On the other hand, by adding some (virtual) disks to the **NAS** node, the flexibility of Logical Volume Manager can be explored. In addition, to provide fault-tolerance, the students build a software RAID on the storage node of the cluster. For instance a RAID-5 can be configured. After setting up the configuration files, some tests can be run to force an error and launch the recovery process. Other alternatives are also feasible. For instance, a DRBD can be also easily configured and tested. The students can also play with iSCSI protocol by configuring a target device at the **NAS** node that is mounted on some nodes of the cluster that work as initiators. The iSCSI target device can be simultaneously mounted on several nodes if a cluster filesystem is used. Therefore, as a next step, the OCFS2 filesystem is configured on the initiators and some proof tests are run.

To finish this part, a small SAN with GlusterFS is developed. Several disks are added to the server nodes of the cluster and several GlusterFS configurations are tested. In particular, both replicated and distributed GlusterFS volumes are configured and mounted from the **Master** node. Then, some files are copied onto the filesystem, checking whether there are replicated in all bricks or distributed among them, respectively. The replicated volume also allows us to perform some fault-tolerance tests.

Next, some packages are installed in the 1st partition to allow the cluster to work as an HPC/HTC platform. We think about three possible scenarios. The first scenario corresponds to a batch oriented system where the users submit their jobs, possibly composed of multiple executions of a given application (for instance a simulator). In the second scenario, the users log in to the system and interactively run their programs. The third scenario corresponds to a parallel machine, where the users want to run their parallel applications.

The system is configured to support the three scenarios. The first one is achieved with the HTCondor package. We chose this package since we have used it for managing simulations in our research group for several years and it is useful for the master students that also do research in our group. The **Master** node is set up as central manager as well as being able of submitting and executing jobs. The server nodes can submit and execute jobs. As in other parts of the project, all the steps are performed thinking about a large cluster composed of not only three nodes but a relatively high number of nodes. Once installed, a simple job consisting of a simulator with different input data sets is submitted to Condor, examining its behavior.

To support the second scenario, the MOSIX package is installed in all the nodes of the cluster. To demonstrate how it works, some test programs are run from one node and it is observed how the load is balanced between all nodes.

Finally, to run parallel programs, both an MPI programming environment (for instance the OpenMPI package) and the OpenMP development library

are installed in all the nodes of the cluster. Some simple parallel programs are written and run to check that everything is working properly.

3.6. Internet Server Cluster

In this part of the project, a high-availability load-balanced web server is configured in the 2nd partition. As a first step, the apache Web server is installed in all the server nodes, including the PHP module. A simple php page is written to return the current date, time and the IP address of the server. Then, the load-balancing tool is installed on the **Master** node. Several options are available. In the project, we have successfully configured for several editions a level-4 load balancer based on the IPVS kernel module. The HAProxy package could be another widely used alternative. After preparing the configuration file, the load-balancer is ready to run. By repeatedly issuing requests to the URL that contains the aforementioned php page, the students verify how the load is balanced between the server nodes by checking the IP address of the machine answering the request.

Once the load-balancing part is done, it is time to prepare the **StandBy** machine. In an actual scenario, a complete installation would be performed from scratch. In the project, taking into account the scarce time available, the **StandBy** node is obtained by cloning the current image of the **Master** node. This is the only time where virtual machine cloning is allowed. After the cloning procedure, the IP address and the hostname are updated and the machine is rebooted. At this moment, the two director nodes are running. It is time to install the high-availability required packages. We have successfully used for this purpose the corosync, pacemaker and ldirectord packages. Alternatively, the keepalived package could be also used. The corosync package

provides the infrastructure to check the health of the director nodes. The pacemaker package is a CRM manager that allows defining resources that will be shared between both director nodes, starting/stopping and migrating that resources. In the project, two virtual IP addresses (external and internal) are defined. These VIPs will float from one director to another in the event of a node failure. Finally, the ldirectord resource checks the server health, thereby avoiding sending requests to the faulty server nodes.

Once the high-availability load-balanced web server is configured, it is time to check its behaviour. In particular, the students issue several http requests to the server and verify that: (i) when all the components are up, requests are balanced among all the server nodes; (ii) when a server fails, requests are balanced among the remaining servers; and (iii) when a load-balancer node fails, the other one assumes the service. In the configured cluster, it is very easy to simulate a faulty node by disconnecting on-the-fly the (virtual) network interface link.

If there is still time available, some tests can be also performed by using simple benchmark tools like apache benchmark.

4. Teaching Methodology

This section presents the teaching methodology and the assessment criteria of the course.

The subject has 40 assigned hours in the syllabus, spread over 16 sessions of 2.5 hours each. The course is taught in an intensive way, at 2 lectures per week, lasting for about two months. The sessions are organized so that approximately 40-50% of the available time is devoted to the lectures and the

rest to complete small laboratory exercises related to the contents explained that day and to work in the hands-on project. In addition, we reserve four complete sessions throughout the course to install and configure the computer cluster using virtual machines. A total of 15 classroom hours is devoted to deploy the computer cluster by the students. Although most of the students have enough with the reserved time slot, some of them must advance some work at home and/or attend meetings with the teachers to solve some issues.

In order to help the students to debug problems when his/her cluster does not behave as expected, we give them a FAQ (Frequently Asked Questions) document that includes information about the most frequent problems and how to solve them. Typical issues are why internet access fails on the nodes (usually because the network, the NAT or the DNS are not properly configured); in addition, the server nodes may have problems to boot from the PXE server while they are being installed due to bugs in their configuration files. Other mistakes are related to NFS service, usually due to typos in the configuration files. Finally, typical problems with high availability tools, such as pacemaker or load balancers as IPVS, are also described in the FAQ document.

The course assessment includes the following aspects. First, the students are encouraged to attend and participate in all the lectures (10%). A successful completion of all the laboratory exercises (15%) is also assessed. In order to cover the part of the course concerning cluster architecture and configuration, the students have to select the components of a hypothetical cluster given some specifications. After selecting the components, they also have to estimate the peak computing power in MFLOPS, their cost and the energy

consumption. Three configurations are considered: high-performance, low-cost and energy-efficient. A final report is written and presented (10%). The most important part of the course assessment corresponds to the hands-on project regarding the deployment of a virtual-machine-based computer cluster. The project has a weight of 35%. A written exam with a 30% weight completes the assessment. The topics of the exam comprise all the contents taught throughout the course but with special emphasis on the configured cluster. Figure 4 shows some question examples we have asked to the students. It must be noticed that the exam is done in the lab so that every student may have his/her own cluster up and running. Therefore, they can access the configuration files and/or issue commands to their cluster if they want to do so.

Concerning the achieved results, in general, the students like very much the subject. They are interested in the contents and very motivated in the hands-on project. Each time a project step is committed, they are proud of their work. In particular, there are three milestones in the computer cluster development. The first one corresponds to the basic installation of the system, once the cluster is able to boot on his own and the students are able to log in to the master node and issue commands to the server nodes. The second one corresponds to the installation of the load management tools (i.e., HTCondor and MOSIX). At that time, the cluster can be used to submit user jobs. The final and most important milestone is achieved when the high-availability load-balanced web server is set up. The students are very excited when they check that the http requests are being balanced or when they simulate faults on the server or director nodes and the web server continues

- What file would you modify to configure network interfaces in Ubuntu? Write its contents to automatically assign an IP address to the eth0 interface.
- Which commands must issue a normal user to configure ssh access from the master node to the servers without password (public key validation)?
- Which services provide the dnsmasq utility? Which ones of them are required at installation time and which ones once the system is installed? Which nodes of the cluster must run dnsmasq?
- Explain what is the purpose of the /etc/ethers file. Which service is this file related to?
- After attaching a new physical disk to a node, which command can you issue to know the assigned device name (for instance, /dev/sdc)?

Figure 6: Sample questions of the written exam

working properly. Overall, student opinion is that they have learned practical and useful issues of today servers. The results obtained in the yearly survey of the University confirm student satisfaction. A five-point Likert scale was used in the questionnaire, with the typical format (Strongly Agree, Agree, Undecided, Disagree, Strongly Disagree). In the last available results, our course obtained 80% of strong agree, 20% of agree in the questions related to teaching methodology, used resources and learning activities.

From a quantitative point of view, the assessment results are overall very good. Usually, all the students pass the subject with very good grades. Typically, only 10% of the students obtains a Fair grade. This latter group corresponds to the students that had not devoted enough time to the hands-on project and either they did not finished it or they did not fully understand all the steps involved in the installation, so they were not able to fully respond the written exam questions.

5. Related Work

Some previous papers have described experiences teaching advanced undergraduate or graduate cluster computing courses. For instance, [1] presents a selection of possible topics for cluster computing courses, based on the experience of the authors teaching this subject in different universities in USA and Australia. The proposed material is pretty wide and covers many different aspects of cluster computing. The goal was to make a proposal that allowed the instructors to select the contents best suited to their course objectives. In the actual courses shown as examples in the paper, we find that frequently an important part of the course is focused on parallel program-

ming. Moreover, depending on the universities, more or less contents on system architecture for parallel and distributed systems are also developed. Only in one case, part of the course is also devoted to build a cluster, shared between the students to evaluate performance of different network technologies, network topologies and file systems. Most of these courses have still been available in the 2015-16 catalogue.

Other works are focused on building clusters to use them in different courses. In [2], [5] and [15], they implement real hardware clusters while [26] and [30] choose virtualization. In [2] several options to implement clusters based on cluster building kits are compared. However, the paper highlights that usage and cluster management at an advanced level are more flexible and allow for better possibilities when manual configuration is employed. A single cluster, consisting of one master and 10 slave nodes, is built. The cluster is shared between all the students. [5] proposes the construction and management of reduced high performance clusters as part of a course aimed at engineering and science students. The goal is to train them since they may need to manage and/or build these systems for their research labs. Each student group develops in the laboratory sessions a 2-node Linux cluster. Most of the services are configured only in one of the nodes and loaded by the other through DRBL package [9]. Other services, such as SSH and NTP are manually installed using the YasT tool, thus making the process not scalable for a big number of nodes.

The work in [15] presents the development of a cluster computing project for business undergraduate students. A small scale Linux cluster was implemented taking advantage of used equipment donated by universities and

major corporations. The course focuses on the benefits of the project for the business students as well as faculty members. The students did not only implement the cluster but also managed it, providing them with a practical understanding of the studied technologies. Regarding academic staff, the cluster was a valuable classroom resource for teaching several courses related with cluster computing. Some general suggestions about how to teach a cluster computing course are also given.

Finally, [26] and [30] draw on virtualization to develop "training" clusters. [26] proposes a very simple Linux cluster made of only two nodes that could be used for developing and debugging parallel software (e.g. using MPI). It is not intended to be a complete cluster solution at all since it only offers very basic services. Moreover, as the second node is added cloning the first one, the solution does not scale to hardware clusters (not virtual ones). The paper is mainly focused on describing the configuration of the VirtualBox machine. [30] describes the development of a virtual cluster through the VCNet tool. VCNet is a virtual technology cluster solution based on Windows 2008 HPC Server. It provides a reliable environment where users can test and debug applications before installing them on the hardware university cluster. Performance of VCNet under different conditions is also evaluated.

All the clusters developed in these courses are based on Linux operating system apart from [30]. Overall, they use quite basic configurations that do not represent real life installations. Regarding to the cluster usage, they are intended for the high performance computing flavour [11] usually ignoring the high performance service one. However, this latter usage represents a very high percentage of cluster installations [11]. Internet servers that han-

dle a high demand on a given service (web, mail, file transfer, database, videos, etc.) are everywhere and are usually based on clusters. Finally, almost no course addresses high availability or advanced cluster storage. On the contrary, the course proposed in this paper considers both high performance/high throughput computing and high availability/high performance service flavours of computer cluster usage. In addition, the course is taught using a hands-on approach, taking advantage of virtualization to allow each student to build his/her own cluster. Despite the reduced size of the cluster deployed, the teaching methodology guides the students to think as if they were building a large cluster.

6. Conclusions

In this paper we described a master-level course on computer clusters. The course has been developed around four keystones. Firstly, a hands-on but in deep design, trying to bring the course contents closer to the professional practice as it is widely recommended (for instance by the IEEE Computer Society and the Association for Computing Machinery in their last Computing Engineering Curricula). Secondly, it deals with the main concepts of all types of computer cluster usage, including the ones that are not usually covered in most cluster-related courses in other universities but widely used in companies, i.e., High-Availability/High-Performance Service clusters. Thirdly, the course relies on virtualization (and Linux operating system) to allow every student to develop his/her own cluster without interfering with the work of other students. Fourthly, despite the small size of the implemented cluster, every step of the process is designed to be scalable, by

thinking about a large cluster.

Both authors have taught the course for several years. The student assessment results are overall very good. Indeed, the student satisfaction at the end of the course is very high, as shown by the results of surveys conducted by the university.

Finally, all the contents of the course have been exposed in detail so that it can provide guidance to other teachers interested in using this approach.

Acknowledgements

This work was supported in part by the Spanish Ministerio de Economía y Competitividad (MINECO) and by FEDER funds under Grant TIN2015-66972-C5-1-R.

[1] A. Apon R Buyya, H Jin, J Mache. Cluster computing in the classroom: topics, guidelines, and experiences. In: IEEE/ACM Int. Symposium on Cluster Computing and the Grid, pp. 476-483, (2001).

[2] S. Aydin and O.F. Bay. Building a high performance computing cluster to use in computing course applications. *Procedia - Social and Behavioral Sciences*, 1(1): pp. 2396-2401, (2009).

[3] D.J. Becker, J. Salmon, T. Sterling and D.F. Savarese, *How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters*, (MIT Press, 1999).

[4] R. Buyya, *High Performance Cluster Computing: Architectures and Systems, Vol. 1*, (Prentice Hall, 1999).

[5] M-H. Chen and T-L. Li. Construction of a High-Performance Computing Cluster: A curriculum for Engineering and Science Students. *Computer Applications in Engineering Education* 19(4), pp. 678-684. (2011).

[6] HTCondor: High Throughput Computing, https://research.cs.wisc.edu/htcondor/ (Accessed 10-06-2016).

[7] Top Ten Linux Distributions, http://distrowatch.com/dwres.php?resource=major (Accessed 10-05-2016).

[8] DRBD - User's guide 9.0, http://www.drbd.org/en/doc/users-guide-90 (Accessed 12-06-2016).

[9] National Center for High-Performance Computing: DRBL: Diskless Remote Boot in Linux. http://drbl.sourceforge.net/ (Accessed 18-05-2016)

[10] Storage for your cloud: GlusterFS, https://www.gluster.org/ (Accessed 12-06-2016)

[11] D.C. Hyde, M. Baker, editor. *Cluster Computing White Paper* pp. 110-119, (2000). Available from http://arxiv.org/pdf/cs/0004014.pdf.

[12] Computer Engineering Curricula 2016 (draft). Association for Computing Machinery (ACM)/IEEE Computer Society. (2015) https://www.computer.org/cms/Computer.org/professional-education/curricula/ComputerEngineeringCurricula2016.pdf

[13] S. K. Prasad, A. Chtchelkanova, F. Dehne, M. Gouda, A. Gupta, J. Jaja, K. Kant, A. La Salle, R. LeBlanc, A. Lumsdaine, D. Padua, M. Parashar,

V. Prasanna, Y. Robert, A. Rosenberg, S. Sahni, B. Shirazi, A. Sussman, C. Weems, J. Wu (2012) NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing - Core Topics for Undergraduates, Version I. http://www.cs.gsu.edu/ tcpp/curriculum/index.php, 55 pages.

[14] E. Johnson, P. Garrity, T. Yates, R. A. Brown. Performance of a Virtual Cluster in a General-Purpose Teaching Laboratory. In: IEEE Int. Conf. on Cluster Computing, poster 35 (2011).

[15] Integrating IS Curriculum Knowledge through a Cluster-Computing Project - A successful Experiment. *Journal of Information Technology Education*, 3, pp. 263-278, (2004).

[16] MOSIX Cluster Management System, http://www.mosix.cs.huji.ac.il/ (Accessed 18-05-2016)

[17] Message Passing Interface Forum, https://www.mpi-forum.org/ (Accessed 25-05-2016)

[18] Master's Degree in Computer and Network Engineering, https://www.upv.es/titulaciones/MUIC/index-en.html (Accessed 15-05-2016).

[19] Master's Degree in Parallel and Distributed Computing, https://www.upv.es/titulaciones/MUCPD/indexi.html (Accessed 15-05-2016).

[20] T. Haynes, D. Noveck, Network File System (NFS) Version 4 Protocol, https://tools.ietf.org/html/rfc7530, March 2015. (Accessed 12-05-2016)

[21] Project: OCFS2, https://oss.oracle.com/projects/ocfs2/ (Accessed 12-05-2016)

[22] The OpenMP API specification for parallel programming, http://http://openmp.org/ (Accessed 18-05-2016)

[23] G.J. Pfister, *In Search of Clusters*, (Prentice Hall, 1998).

[24] Welcome to the PVFS Project, http://www.pvfs.org/ (Accessed 12-05-2016)

[25] Slurm workload manager, http://slurm.schedmd.com/ (Accessed 18-05-2016)

[26] G.K. Thiruvathukal et al. Virtualization for Computational Scientists, *Computing in Science & Engineering*, 12(4), pp. 52-61, (2010).

[27] TOP500 Supercomputer Lists, http://www.top500.org/lists (Accessed 12-06-2016)

[28] TOP500 Overview, http://www.mellanox.com/page/top_500 (Accessed 6-05-2016)

[29] Oracle VM VirtualBox, https://www.virtualbox.org/ (Accessed 12-06-2016)

[30] R.L. Warrender, J. Tindle and D. Nelson, Development of a Virtual Cluster, In: Int. Conf. on High Performance Computing and Simulation (HPCS), pp. 545-551, (2013).