# Preparing and managing an HPC Cluster: Lessons learned

Eduardo José Gómez-Hernández [1] y José Manuel García [2]

*Abstract*— **From the 1990s, building systems able to reach the performance of supercomputers was a priority, mainly by its cost. In the early 2000s, processors started to reach a limit in the power, forcing a change in the paradigm. Our research group is focused on High-Performance Computing (HPC), and having our own cluster allow us to adapt it to our needs at any moment. We present how we built the current research group cluster, reusing most of the parts of the older one, the most important decisions done, and what we have learned during this year. This new configuration is running and has been used to reproduce research scenarios correctly with near 6000 jobs sent to the workload system.**

*Keywords*— **High-Performance Computing, Computer Cluster, Heterogeneous Computing**

## I. Introduction & Motivation

From the early 1990s, there has been motivation for building systems able to reach the performance of traditional supercomputers with lower cost. With this idea, Networks of Workstations (NOW) [1] were developed to replace supercomputers for certain types of applications. The High Performance Computing (HPC) race was starting, and some clusters (such as Beowulf Cluster [2]) became a good rival of supercomputers but costing a tenth of its competitors price.

In the consequent years (near 2000s), processors started to reach a limit in power density, a slowdown in performance improvements was coming. The thread-level parallelism broke this problem allowing the creation of new multi-core processors and increasing the throughput when the workload consists of independent applications [3] but making impossible the creation of supercomputers made from only one machine.

The HPC race continues, every 6 months the TOP500 list is updated with the fastest computers in the world. All of them are heterogeneous machines clustered together, such as the Beowulf Cluster. Currently, the largest supercomputers in the world are competing to break the exascale barrier in floating point operations ($10^{18} FLOPS$).

Besides the HPC race, the main purpose of the computing power given by these powerful machines enables the execution of complex applications such as simulations, mathematical resolutions, prime number calculus (essential for the cybersecurity), cryptocurrency market, machine learning, and so on. These systems are being used anywhere, from weather simulation, gaming servers of thousands of players, models able to classify complex images or predict diseases automatically, among many others.

Our research group is mainly focused on HPC and its applications. Having our own cluster allow us to adapt it to our needs at each moment. With the revolution of machine learning and deep learning, we had to add several machines with special hardware for this task.

This paper presents the main decisions made to build and maintain our research cluster, making it flexible enough to be able to execute jobs of near every type, and providing a good replication environment for other researches.

We improved the previous GNU/Linux computer cluster with a new operating system, modularized the important software, automatized the tedious tasks, and monitored the entire cluster. From the user feedback and the easy daily maintenance, we ended with a success.

The rest of the paper is organized as follows: Section II presents the process of the hardware selection and its installation. Section III describes the system management, from user administration and permissions to monitoring the whole system. Section IV introduces what we have learned during this experience. Section V defines other interesting approaches and useful information. Finally, Section VI draws conclusions.

## II. Installing

### A. Starting Point

Our research group started with clusters in 2006. They had several machines racked in a room, but with the years, they become obsolete. Therefore, in 2015, José Antonio Bernabé and Victoriano Montesinos improved it. The improvement was done by rebuilding it from the start. This time, the machines were organized like a hierarchical cluster, but all of them were connected to the public network, being able to connect any machine at any moment without crossing the head node. But, the login and batch systems were centralized into the head node.

The cluster worked fine for a couple of years, but at the end of 2017, maintenance was too complicated. Powering the cluster after a shutdown (o a power outage) was painful, most of the machines required manual operation. Installing new software, was impossible without breaking any other.

We put up with the state of the cluster until July 2018, when we decided to shut down the cluster to rethink its organization. Then, in September 2018, the project to rebuild the GACOP's Cluster started.

[1]Computer Engineering Department, University of Murcia email: eduardojose.gomez@um.es
[2]Computer Engineering Department, University of Murcia email: jmgarcia@um.es

TABLA I

MACHINES MAIN SPECIFICATION SPLIT BY ITS PURPOSE IN THE CLUSTER.

Head Nodes

| Machine | CPU | Memory | Accelerators |
|---|---|---|---|
| Main Head | Intel i7-8700 3.2GHz | 8 GiB DDR4 2667 MHz | None |
| Backup Head | N/A | N/A | N/A |

CPU Compute Nodes

| Machine | CPU | Memory | Accelerators |
|---|---|---|---|
| Mendel | 2xIntel Xeon E5-2698 v4 2.2GHz | 4x32 GiB DDR4 2400MHz | None |
| Pacioli | 2xIntel Xeon E5-2650 v2 2.60GHz | 4x8 GiB DDR4 1333MHz | 3xXeon Phi KNC x100 |
| Lejeune | Intel Xeon Phi 7210 1.4GHz | 4x32 GiB DDR4 2133MHz<br>16 GiB MCDRAM | None |
| Pascal | Intel Xeon Phi 7250 1.4GHz | 6x32 GiB DDR4 2400MHz<br>16 GiB MCDRAM | None |

GPU Compute Nodes

| Machine | CPU | Memory | Accelerators |
|---|---|---|---|
| Watt | 2xIntel Xeon E5-2603 v3 1.6GHz | 4x16 GiB DIMM 2133MHz | Nvidia GTX 1080 (GP104) |
| Ampere | Intel Xeon E5602 2.4GHz | 4x4 GiB DIMM 1066MHz | Nvidia GTX 980 (GK110GL) |
| Nikola | Intel Xeon E5602 2.4GHz | 4x4 GiB DIMM 1066MHz | Nvidia Tesla K40c (GK110BGL) |
| Volta | 2xAMD Opteron 6134 | 4x4 GiB DDR3 1333MHz | Nvidia Tesla K20c (GK110GL) |

## *B. Avaliable Hardware*

One of the requirements of the every new cluster is to reuse all the useful parts from the previous ones. Therefore it is necessary to evaluate the existing hardware to find if it meets our requirements. In our case, this hardware is compound by 9 high performance machines, and a pair of network switches and KVMs (Keyboard Video Mouse switch).

In these machines, there are multiple devices such as Nvidia's GPUs, Intel's Scalable Xeon CPUs, AMD's Magny Cours CPUs, and Intel's Xeon Phi CoProcessors, among others. Depending on the needs it is possible to add or remove this hardware from the cluster, therefore, knowing what usage will it have is crucial before building it.

## *C. Purpose*

This cluster is used mainly by professors, researchers, and students during their bachelor's, master's, and Ph.D. thesis. The cluster must execute jobs sent by users with some privileges. Since many of them require time measurement, the mutual exclusion must be guaranteed, but not only for the CPU, also the disk access, RAM, etc. Therefore, each machine will have the needed data in its own disk.

With all of the above in mind, we have physically configured the machines with the characteristics observable in Table I.

## *D. Networking*

Since data is stored at each machine independently, the machines must be interconnected to exchange data, also to be able to work together as a single machine. For isolation and security, all compute nodes will remain connected by an internal network, and only the head node will have access to the outside network.

All the machines have, at least, one gigabit ethernet card, therefore we will use two gigabit network switches for communications. This is not ideal, and an InfiniBand network could improve performance. But, in our case, the machines normally are executed in mutual exclusion, meaning that the communications between machines should not affect performance.

Using this network configuration alongside with each machine specification, the composition is visible at the Figure 1

## *E. Operating System*

In a computer cluster, the main features required to the operating system are performance and security. An operating system has to provide an abstraction to the hardware, manage the resources, provide security, and many other things.

There are several operating systems that can be used: GNU/Linux, Solaris, BSD, Windows Server, etc. But currently, the high performance market is replete with custom GNU/Linux systems. The main reason for being the main operating system used is its security, performance, and code availability, allowing anyone to report and fix any bug or security flaw.

Also, inside the GNU/Linux operating system, there are a lot of distributions. A distribution is a specific version of GNU/Linux with a lot of preconfigured settings and built-in software, and the most known one in the HPC environment is CentOS Linux, based in Red Hat Enterprise Linux. Another great choice is Scientific Linux (supported by Fermilab), but it is recently getting into its end of life. THeir authors will continue to support the latest versions, but they are migrating to CentOS.

Our selection was CentOS Linux, because it is a very stable distribution with big community support, and most of the scientific applications and libraries support it. Also, one of its objectives is to make a reproducible platform, which nowadays, is necessary for research.

## *F. Nvidia GPUs*

The GPGPU (General Purpose GPU) programming has become a requirement in some applications to make them run at reasonable times, like Deep

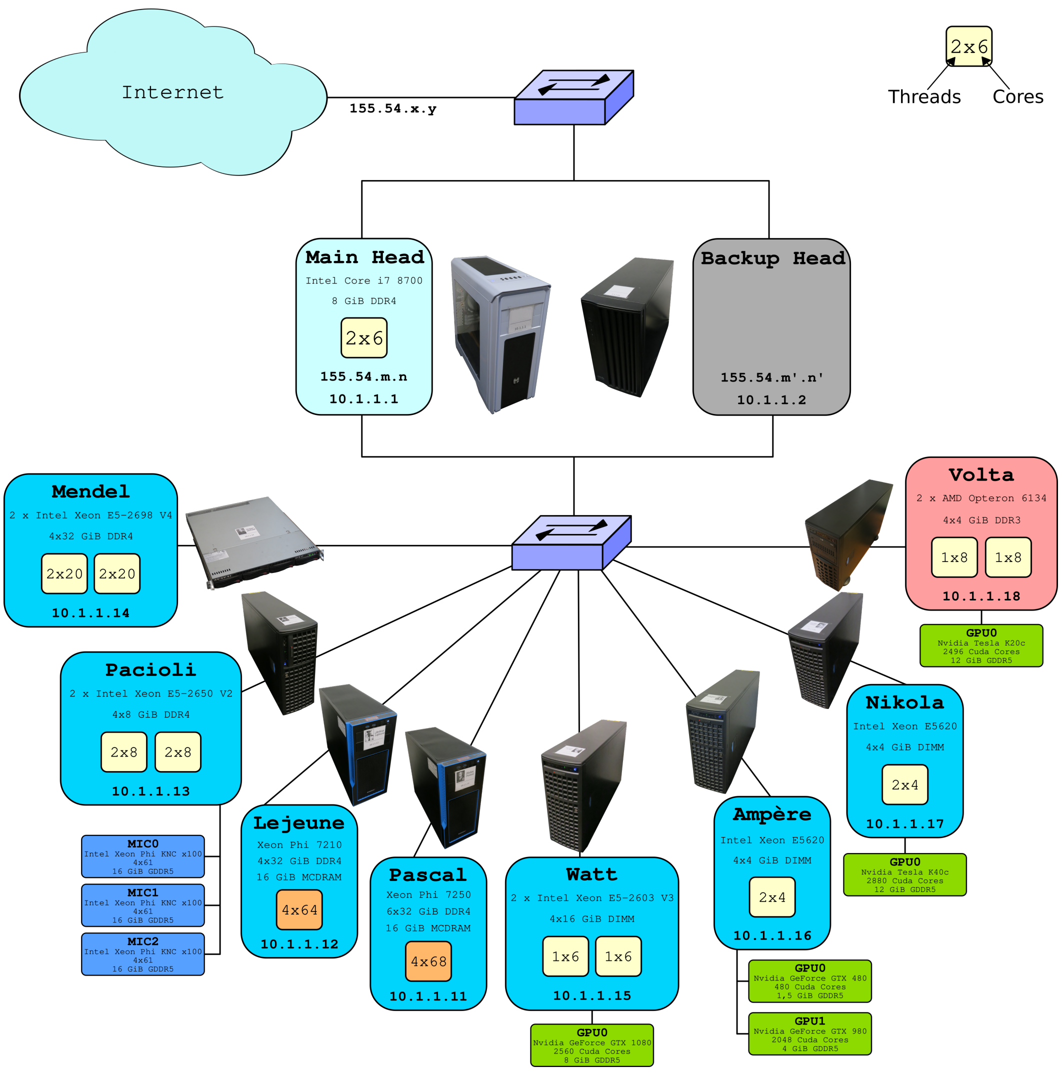

Fig. 1. Overview of the cluster architecture.

Learning. It is a powerful accelerator for SIMD (Single Instruction Multiple Data) paradigms. Nowadays there are three main GPU designers: Nvidia, AMD/ATI and Intel. Each designer has its own installation process, even could differ between different GPUs from the same designer, and each one has its own language or programming methodology. For example, the main programming option for Nvidia is CUDA, but OpenCL is available for all of them (with less performance).

Normally, installing a GPU is not very complicated. To get all the computing power it can offer, it is necessary to download the proprietary driver from the designer's webpage, and install it. Sometimes, it may conflict with the open source driver, if it was previously installed. Lastly, if the device requires a special programming environment, such as CUDA or OpenCL, it needs to be installed aside.

### G. Xeon Phi KNC

Intel Xeon Phi KNC (or Xeon Phi x100) is a coprocessor made by Intel using the MIC (Many Integrated Core) architecture [4]. It was recently deprecated [5], but it continues to be very useful for executing some jobs and must not be disregarded.

For GPUs, normally you install a driver and an SDK and it is almost ready, but Xeon Phi Coprocessors are a bit different. The Intel MPSS (Intel Manycore Platform Software Stack) includes the driver and most of the tools needed to manage a Xeon Phi Coprocessor. It is currently under long term support, but not compatible with the latest versions of RHEL or Centos (7.5 and 7.6). Fortunately, Jan Just Keijser continues patching it from the Intel sources [6].

To start creating binaries that can used by the coprocessor, Intel Parallel Studio is required, specifically the last version from 2017. But, if another Intel Parallel Studio or Intel Compiler is installed, a conflict could appear. An easy way to change between one compiler to other must exists for the cluster's users. In our case, we have two versions (2017 and 2019) packed inside modules, therefore the user can switch from one to other like any other existing module. In addition, we have created wrappers for most used applications from the Intel Parallel Studio to change automatically to the 2017 version without noticing the user.

Lastly, to launch jobs to the Xeon Phi Coprocessor, we are using different queues, one for each coprocessor, and another one for the host machine, allowing the native execution or the offloading one.

## III. Management

### A. Users and Directories

Users have to log in using the head node, but all machines need to know which user is. To afford it there are various user management services. From the most complex one, LDAP, to the simples, NIS. The cluster is not particularly big, therefore something too complex like LDAP is not needed. Keras is also a valuable candidate, but similar to LDAP, it is to complex to handle medium-size cluster. NIS makes exactly what we need, share users and groups between machines.

Then, when launching a job, due to the resources mutual exclusion, the program, and the data must be at the desired machine to be able to execute it. This can be achieved with the help of some programs. First, NFS allows mounting a directory from another machine easily. Then AutoFS can manage this for us, unmounting the directory when is not needed. Lastly, using a custom script, the machines' directories are linked to the home directory of each user. Since not all users have access to all machines, it is necessary to restrict access to only the ones he needs, a simple way to do it is using groups. If a user has the group "machine1", he will have a directory with name "machine1" linked to the NFS directory of that machine.

### B. Batch System

One way to maintain mutual exclusion between users is to force them to use a batch system. A batch system (or workload system) manages access to the resources thought jobs using queues. There are many different policies to establish these queues. They can represent from a specific unit of the system to an entire system. Also, they can overlap themselves and using any kind of priority system order the execution of the jobs.

The most popular batch systems are Torque and SLURM. Our users are used to SLURM, but there are new users who come from another cluster in the university which uses Torque. To accomplish this, SLURM has a compatibility layer with Torque, allowing users to use the system they prefer without knowing which one is behind it.

### C. Software

Maintain software in a cluster is a very tedious task. They can have security flaws, incompatibilities, or simply a user needs a specific version for any reason. When using operating systems of the Red Hat Server family, you end with a very old compiler (at least until RHEL 7), and all libraries depend on it. But, to support new features like different accelerators, or new platforms, it is necessary to use the new versions. Some time ago, the C++ ABI changed, and all software and library for Red Hat Server family use the old ABI, making a headache every time a user wants new software.

Unfortunately, there is not an easy system to do this. There are some approximations like easybuild, but they are not a swiss knife. There are always situations where you need to manually install some specific version of specific software.

There is another approximation called Environment Modules, a simple package that allows change user's environment variables through rules in a modulefile. In this case, a module is not restricted to anything specific, if it is possible to use a software modifying only environment variables, then is possible to use Environment Modules.

In our case, we have a special directory for modules, synced between all machines. We have several versions of GCC and Intel compilers, and we are planning to add LLVM. Also, we have a lot of deep learning modules for python, and some specific libraries and applications. To generate the modulefile for Intel compiler, we have used env2, a tool to convert environment variables from one scripting language to another.

### D. UPS

When aiming to have a cluster powered during months, a power outage is a catastrophic problem. A power outage can corrupt information, stop jobs that could be running for months, or in the worst case, cause physical damage. Having some UPSs is a must when building a cluster.

Our most power hungry machine has a UPS only for it, and it depletes in some seconds, sometimes in 1 or 2 minutes. Therefore, we prefer saving data and prevent any kind of damage to the users' data shutting down the machines as fast as we can when detecting a power outage.

The current UPS take around 10-15 seconds to detect it and report to the head node. Then, the head node sends a signal to shut down each machine in parallel and it shutdowns itself. In these kinds of situations, we prefer integrity over availability.

### E. Maintenance & Monitoring

Maintaining a cluster requires maintaining two different things, software, and hardware. The hardware must be secured, avoiding high temperatures and any kind of disaster, and the software must be secure and fail-safe. We have our cluster located in a specific room only for it, with a constant ambient temperature about 19-21 °C. Also, only authorized people have access to it, even some administrators can only access remotely.

On the other hand, we have blocked any petition not attending port 22, and only active users can log in. We recommend an RSA key authentication, but password login is allowed with some length restrictions. If an unauthorized person enters into a user account, he only has access to that user data and permissions. Any system file can only be modified by its own user. And the root account is password protected and cannot be logged from outside.

There are some monitoring services to check that all is running fine and there is nothing strange. A monitoring service will check all the inserted rules at certain timed moments, and if something is not working as expected, it will set an alert taking actions or informing the administrators. We have selected Nagios, a well-known monitoring system, highly configurable and easy to use, adding rules as we see it necessary. At this moment, we are checking availability, disk space, and users. Since machines are used through the workload system, only the admins and the workload system can be logged in.

### *F. Documentation*

In a cluster environment, there are at least two different documentation reports. The first one is the administrator document, it contains all the installation process, emergency situations, hardware information, and any other relevant information necessary to make the cluster work from the start, or after an emergency situation. This document can be passed to one administrator to another to improve or update it over the years. It has to track all the different changes made to the cluster. The second document is for the users, it must contain anything the user has to know to use the cluster from the very beginning (connecting to the cluster for multiple operating systems) to launch jobs, load software, upload files, etc. Also, a brief overview of the physical architecture of the cluster usually helps.

There are many ways of doing these documents, but we have chosen a wiki for the administration document, and a manual for the second one. The wiki has been developed similar to a diary, specifying each day the work done, the issues found, steps to solve those issues, etc. The manual includes a lot of step by step sections to help the user to get into the usage of the cluster.

## IV. Lessons Learned

We have deployed a CentOS 7.5 HPC cluster for researching and educational purposes. Getting all the experiences throughout its building and maintenance this year. We would like to show the main lessons we have learned during this period.

Building a computer cluster is not a trivial task, after working with a couple of machines, scaling it may appear to be very simple, but in fact, it is not. There are always troubles with machine names, collisions between services, etc. In spite of the normal way to install a cluster is to prepare the head node and later a compute node and replicate it, it is necessary to never forget that there are more machines and each machine could be different.

Parodying the KISS principle (Keep It Simple, Stupid), for the users, the cluster must be simple to use and keep running, KISR (Keep It Simple and Running). Users normally do not matter how the cluster works, or the time needed to install new software or adapt the existing one to their necessities. Automatization is the key to making this view.

The security is very important, not only the allowed actions of a user, but also the outside attacks. First, to prevent the users to use unauthorized resources, we used a permission system using UNIX groups. Also, executing binaries from home directories (at the head node) is not allowed, this is not only for security, but it is also to prevent mistakes when running applications in local. Otherwise, the outside attacks are different, we have two main types of them: username and password cracking, and services attack. We receive around 4,000 invalid user logins each day, therefore to prevent an intrusion we force the users to have passwords with at least some strength. Service attacks are different, they try to find a security flaw in a running service. In our case, only SSH is visible from outside (using iptable/firewalld), making it the only service available.

Good documentation is a time- and life-saver. At any moment, anything can happen and require manual intervention, having any previous issue documented can save a lot of time. Also, when adding a new machine it is possible to repeat exactly the same steps as others, and only change the specific settings for that new machine, or hardware. Additionally, these documents can evolve, and get better with the time becoming a valuable piece of information in the future. Lastly, if a new administrator takes the cluster, he can review all the tasks done to the cluster, and improve any of them.

## V. Related Work

From the creation of the Beowulf Cluster, a 16 Intel DX4 machines cluster [2], a lot of systems emerged. A lot of them with the same principle of low cost high performance, but also clusters based on low power consumption. This movement allowed the creation of many systems with extravagant machines, and it is worth to mention some of them.

"Iridis-Pi" is a computer cluster built from 64 Raspberry Pi B computers [7]. Its main objective was to work as a single machine for education. Also, videogames consoles have been converted into a cluster. One of the most recent videogame consoles to be converted into a cluster is the PlayStation 3 [8]. Nowadays, smartphones are everywhere, therefore there are clusters made with them, like the "Droid-Cluster" [9].

We are in a heterogeneous era, where accelerators are essential in computing, hence many heterogeneous clusters with accelerators (such as GPUs and FPGAs) were developed, like Axel [10] and QP [11].

There are a lot of cluster administrators with their own clusters and sharing their experiences through the internet, and developing gold valuable documents for others. "Server World" [12] is a web page dedicated to making manuals configuring servers. Its main operating system is CentOS, but they have information for other systems. It is not limited to one configuration, there are manuals for a lot of different services, and also alternatives to the main ones. Also, the "ADMIN Magazine" has two goods references when building a cluster [13], [14], they are quite old, but the fundamentals are the same.

Lastly, is worth to mention that the University of California (Irvine) has two large clusters running, and they are planning to deploy a new one. To know the requirements, they have a large report studying the utilization of the actual ones [15]. It is very interesting to understand how to evaluate a cluster and the main factors in deploying a new one.

## VI. Conclusions

Building an HPC cluster is not an easy job, the system administrator has to teach the users how to use it, maintain the system up and running, install the necessary software and hardware, keep the system secure without allowing unauthorized people to use the cluster, or even preventing some users from using some parts of it, among many other tasks.

During this year, this new architecture of the cluster has been tested with the users, they are running several jobs from different environments. And some research scenarios have been reproduced correctly, getting, in most of the cases, the same results than the original authors. At the moment of this writing, near 6000 jobs have been run in the batch system.

We have found some things that can be improved to get better user experience and more flexibility. We will implement them the next time we have to rebuild the cluster for a major update. For example, we found that a shared directory is necessary for jobs with a lot of data, or when time is not a restriction.

## Acknowledgements

This work was partially supported by the Spanish MCIU and AEI, as well as European Commission FEDER funds, under grant RTI2018-098156-B-C53. We acknowledge the excellent work done by José Antonio Bernabé and Victor Montesinos who developed the old cluster architecture and documents while they were doing a research internship supported by the University of Murcia. Also, Gonzalo Caparrós Laiz who is helping us in some administration tasks.

## Referencias

[1] William Kramer, “Clustered workstations and their potential role as high speed compute processors,” 1994.

[2] Thomas L. Sterling, Daniel Savarese, Donald J. Becker, John E. Dorband, Udaya A. Ranawake, and Charles V. Packer, “BEOWULF: A parallel workstation for scientific computation,” in *Proceedings of the 1995 International Conference on Parallel Processing, Urbana-Champain, Illinois, USA, August 14-18, 1995. Volume I: Architecture.*, 1995, pp. 11–14.

[3] Antonio Gonzalez, “Trends in processor architecture,” *CoRR*, vol. abs/1801.05215, 2018.

[4] Andrey Vladimirov, Ryo Asai, and Vladim Karpusenko, *Parallel Programming and Optimization with Intel Xeon Phi Coprocessors*, Colfax International, 2 edition, 2015.

[5] Intel, “Intel xeon phi x100 product family,” `https://ark.intel.com/content/www/us/en/ark/products/series/92649/intel-xeon-phi-x100-product-family.html`.

[6] Jan Just Keijser, “Intel xeon phi (knc) mpss modules for centos 7.5 and 7.6,” `https://www.nikhef.nl/~janjust/mpss/`.

[7] Simon J. Cox, James T. Cox, Richard P. Boardman, Steven J. Johnston, Mark Scott, and Neil S. OB́rien, “Iridis-pi: a low-cost, compact demonstration cluster,” *Cluster Computing*, vol. 17, no. 2, pp. 349–358, 2013.

[8] Fernando Pardo and Jose A. Boluda, “Laboratorio de arquitecturas avanzadas con cell y playstation 3,” *XXI Jornadas de Paralelismo*, 10 2010.

[9] Felix Büsching, Sebastian Schildt, and Lars C. Wolf, “Droidcluster: Towards smartphone cluster computing - the streets are paved with potential computer clusters.,” in *ICDCS Workshops*. 2012, pp. 114–117, IEEE Computer Society.

[10] Kuen Hung Tsoi and Wayne Luk, “Axel: a heterogeneous cluster with fpgas and gpus,” 01 2010, pp. 115–124.

[11] Michael Showerman, Jeremy Enos, Avneesh Pant, Volodymyr Kindratenko, Craig Steffen, Robert Pennington, and Wen mei Hwu, “Qp: A heterogeneous multi-accelerator cluster,” in *In Proc. 10th LCI International Conference on High-Performance Clustered Computing - LCI'09, 2009. Memory reliability*, 03 2009.

[12] Server World, “Build network server,” `https://www.server-world.info/en/`.

[13] Gavin W. Burris, “Setting up an hpc cluster,” `http://www.admin-magazine.com/HPC/Articles/real_world_hpc_setting_up_an_hpc_cluster`.

[14] Jeff Layton, “The fundamentals of building an hpc cluster,” `http://www.admin-magazine.com/HPC/Articles/Building-an-HPC-Cluster`.

[15] Harry Mangalam, “Hpc3 cluster planning stats,” `http://moo.nac.uci.edu/~hjm/hpc/hpc3/HPC-Cluster-Stats.html`.