Methodology for Design and Implementation an Efficient HPC Cluster

L. A. Torres[1,2(✉)] and Carlos J. Barrios[1,2(✉)]

[1] Supercomputación y Cálculo Científico (SC3), Universidad Industrial de Santander, Bucaramanga 680002, Colombia
luis.torres@correo.uis.edu.co, cbarrios@uis.edu.co

[2] Grupo de Investigación Computo Avanzado y a Gran Escala (CAGE), Universidad Industrial de Santander, Bucaramanga 680002, Colombia

Abstract. For years, clusters for HPC have been implemented through the typical process of obtaining the source code, configuring and compiling each of the tools that make up the infrastructure services. Each administrator based on their experience and knowledge assumes a series of considerations to design and implement a cluster that is considered efficient by installing base tools such as NTP, NFS, a task manager (that is, SLURM), LDAP, among others. In order to reduce these times, several open-source initiatives have emerged, such as Rocks, that allow the rapid implementation of an HPC cluster despite its low configuration flexibility. OpenHPC emerges as an alternative that provides the necessary tools in a software repository and that once installed allows the same flexibility of customization and adaptation as if they had been installed in a typical way. It's worth mentioning that OpenHPC provides all of those standardized tools in order to spread best practices in building and managing HPC data centers, but unlike Rocks, OpenHPC requires pre-design of the platform, including network infrastructure, storage services, and the different tools to implement, requiring prior knowledge by the administrator about each of them. The objective of this paper is to present the fundamental basis for implementing an efficient cluster by using OpenHPC without becoming a technical installation guide, but rather a series of steps in a methodology used by the Supercomputación y Cálculo Cienfífico Laboratory SC3.

Keywords: Cluster computing · OpenHPC implementation · HPL metrics and evaluation

1 Introduction

HPC has reached a level where it has become indispensable in the different fields of scientific research. Areas such as artificial intelligence, bioinformatics, climate prediction, among others, are some of these fields that depend on supercomputing centers to carry out their research. However, the design and implementation

S. Nesmachnow et al. (Eds.): CARLA 2020, CCIS 1327, pp. 71–85, 2021.
https://doi.org/10.1007/978-3-030-68035-0_6

of these have always been a task that depends on the experience of the administrators and that despite this can lead to long implementation and commissioning times. To remedy this problem, tools such as Rocks have emerged that have facilitated implementation but their cost-benefit in relation to the administration of the HPC platform makes them little-used [1]. Consequently, OpenHPC appears as an alternative to facilitate administrators with quick implementation and start-up by providing a software repository with different alternatives for administrators [1].

The use of OpenHPC is presented as a set of tools rather than as a definitive solution in the implementation of a cluster. Therefore, this paper presents the methodology used for the implementation of the GUANE[1] cluster of the Supercomputing Center and Scientific Calculation SC3 [13] showing the basic components necessary to take into account in the design and implementation in a supercomputing platform. The first section introduces the basic concepts and tools needed. The second describes the proposed methodology and the different tools implemented in such a way that the order in which they are presented is the order in which they must be installed and configured. The third shows a description of the Linpack benchmark, its configuration, and the results obtained. Finally, the conclusions of the methodology used are presented together with a comparison made at the efficiency level with the first five machines that appear in the Top500 [14].

2 Background

In our experience, the HPC system administrator must know a lot of concepts raging from Linux to the main hardware deployed in any cluster. With this background, we offer an overview of OpenHPC and the main tools implemented in this methodology.

2.1 High Performance Computing

HPC is a field of computing that seeks to improve performance in solving major problems in science, engineering, and business. Generally speaking, these computationally complex problems are mathematically modeled and, through parallel computing techniques, they become code instructions that are executed on machines known as supercomputers. These machines run these intensive programs on specialized CPU or GPU, significantly reducing the time to run on regular hardware [15].

The supercomputers were introduced in the 1960s by Seymour Cray of CDC (Control Data Corporation) [2], and for many years associated companies bearing this surname controlled this market. The vector processor with the ability to operate on large data sets was introduced in the 1970s, and it was not until the 1990s that massively parallel supercomputers began to be used with standard

[1] GpUs Advanced computiNg Environment.

processors, which to date are the norm. It should be noted that the performance of these machines is measured in floating point operations per second (FLOPS) and not in millions of operations per second (MIPS), because the latter is more a measure of the performance of a task in comparison to a reference and not a measure of execution speed. In recent years, higher performance speeds than petaFLOP have been achieved and it is hoped to achieve exaFLOPS in the near future [3].

This evolution of supercomputers and high performance computing is due to the increasing demand for computing speeds to solve problems in areas such as quantum mechanics, weather forecasting, oil and gas exploration, among others. In the last decade, this demand has reached very high levels due to the use of different artificial intelligence algorithms, particularly machine learning and deep learning algorithms.

Lastly, the HPC hardware falls into three main categories: Symmetric multiprocessor (SMP), vector processors, and clusters [15] and with the latter being the subject of study in this methodology.

Cluster. It is the most widely used supercomputer and is a collection of many servers (nodes) which are connected through a high speed and high bandwidth network. These clustered servers can behave as a single server and a combination of the following services must be provided: high performance, high availability, load balancing, and scalable. Clusters can be classified according to their characteristics into:

- Fail-over clusters
- Load-balancing clusters
- High-performance clusters

These differ depending on the type of applications and their purpose. The Fail-over clusters and the Load-balancing clusters are used in mission-critical applications where consistent, throughput availability of services is required through many instances of one or more applications on different nodes. Finally, the High-performance clusters are designed to increase performance and decrease computing times when running work on multiple nodes at the same time [15].

2.2 OpenHPC

OpenHPC was launched in 2015 and formalized as a collaborative project of the Linux Foundation in 2016 [1]. This project is comprised of 25 organizations with representation in academia, research laboratories and industry. It has a large number of software components that include provisioning tools, resource management and scientific libraries. The main objective of this project is to make best practices available to administrators and to provide a software repository for HPC clusters.

For administrators, manually installing and configuring an HPC cluster can be tedious and complicated. For this reason, several open-source solutions

emerged, among which Rocks and OSCAR stand out. Rocks [4] is a CentOS-based Linux distribution that contains additional software components for cluster deployment and administration without the need for other external packages. On the other hand, OSCAR (Open Source Cluster Application Resources) [5] is a fully integrated software package that, unlike Rocks, you must first install the frontend and then download and install the cluster configuration and administration tools. The project is no longer maintained and the latest version was released in 2011.

A common issue with these tools is the lack of balance between customization and ease of use [6], which is why OpenHPC takes a more basic approach when providing a software repository. This approach requires the administrator to be experienced but offers a variety of software components to promote flexibility in different environments and scales. OpenHPC includes two end-user projects that seek to reduce the complexity of installing and configuring scientific and HPC software: Easybuild [7] and Spack [8].

2.3 Lightweight Directory Access Protocol

LDAP is a set of open protocols that are used to access information that is centralized through the network. It is based on the X.500 standard but is less complex and uses fewer resources. The information is organized in a hierarchical and categorized model through the use of directories that can contain a large amount of information. LDAP is a client/server system where the server uses a database to store directories and is optimized for fast, high-volume readings. When connecting to the server, the LDAP client can make queries or modify a directory. In the latter case, the server verifies that the user has the permissions to carry out this operation before making the change and updating the information [16].

OpenLDAP is the free and open-source implementation of LDAP, supports LDIFv1 and LDAP versions 2 and 3. In relation to supercomputing clusters, OpenLDAP provides HPC infrastructures with a way to manage platform users. One of the great advantages of using LDAP v3 is the possibility of using dynamic groups, which allow the system administrator to create a tree with different access privileges to the directories of the HPC system storage.

2.4 Simple Linux Utility for Resource Management

Linux clusters require a resource management system that performs tasks such as scheduling user jobs, monitoring machine and job status, and managing machine settings as such. This system should be simple to use, fault-tolerant, efficient, scalable, and portable. With this in mind, Lawrence Livermore National Laboratory, SchedMD, Linux NetworX, Hewlett-Packard, and Groupe Bull produced the first slurm design [9].

Slurm enables efficient management of clusters regardless of size or architecture, is highly scalable, requires no kernel modification, and is relatively self-contained. The basic components are shown in Fig. 1.

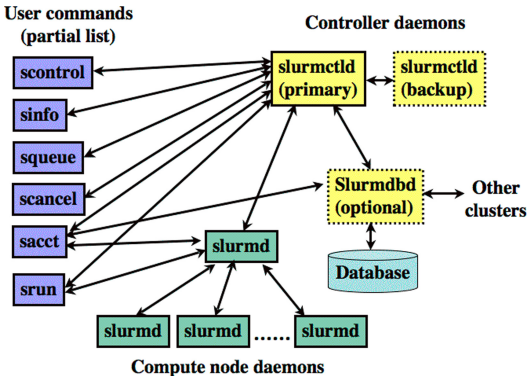

Fig. 1. Slurm components [17]

In essence, slurm works using two daemons, one on the frontend called slurmctld and the other on nodes called slurmd. The slurmd daemon provides fault-tolerant hierarchical communications and is responsible for initiating and managing user jobs. On the other hand, the slurmctld daemon sometimes referred to as the "controller", is in charge of orchestrating slurm activities, including job queuing, monitoring the status of jobs, and allocating resources to jobs. As shown in the Fig. 1, a backup to this daemon can be included which will automatically take over in case of failure of the primary controller, which will regain control when service is restored [9]. There is another optional daemon in slurm called slurmdbd, which allows storing the accounting records of the jobs in a database, allowing to generate reports about the platform.

Lastly, we find the user commands that allow them to run and monitor each of the jobs that are sent to the HPC platform. The sbatch and srun commands are used to run jobs, scancel command to cancel them, scontrol is used to view or modify Slurm configuration and state and, sacct command displays accounting data for all jobs and job steps in the Slurm job accounting log or Slurm database [9].

2.5 System Security Services Daemon (SSSD)

This daemon has the primary function of providing remote access to different authentication mechanisms through a common framework. These mechanisms are known as identity providers and SSSD allows them to connect to it as backends [18].

SSSD provides caching and offline support for applications that require authentication using standard PAM and NSS interfaces. With this feature, applications do not need to connect directly to identity providers (e.g. LDAP, NIS, Samba, etc.) and even if they are not available, the SSSD cache allows the applications to authenticate. Another important feature of SSSD is its ability to use multiple providers of the same type, such as two different LDAP servers [18].

2.6 High-Performance Linpack (HPL)

HPL is an implementation of the Linpack benchmark [11] for computers with distributed memory. This benchmark solves a dense linear random system in double-precision arithmetic. Basically it only requires a configuration file where the main parameters for creating the problem to be solved are specified.

The main parameters are:

- N: Order of the coefficient matrix A
- NB: Block size
- P: Number of processes - row
- Q: Number of processes - column

In general, the product of PxQ should be the number of MPI processes and the value of Q should be greater than or equal to P. The value of N should be chosen as close as possible to the total physical memory. For choosing N, the following formula is usually used:

$$N \approx \sqrt{\frac{Total\_Memory\_Size\_in\_bytes}{sizeof(double)}} \tag{1}$$

Where N must be an integer and must be a multiple of the selected NB block size. The size in bytes for the double-precision floating-point is 8.

The results obtained by HPL are the effective performance measures Rmax finds for each of the configurations. Another important value to calculate is the theoretical peak of Rpeak performance using the following equation:

$$R_{peak} = Num_{CPU} * Num_{Core} * Frequency * Num_{FLOPs/cycle} \tag{2}$$

Finally, the efficiency of the cluster is obtained by [10]:

$$Efficiency = R_{max}/R_{peak} \tag{3}$$

These topics cover the main components in the HPC cluster and in our implementation but exist others that didn't name in this section because we consider tools that any Linux administrator knows. The integration of these topics and other tools will be shown in the next section where will be described our methodology.

3 Methodology

By referring to the word "efficiency", the aim is to adequately fulfill a certain function. There are several ways to design and implement an HPC cluster but the knowledge and experience of the administrators is what leads to what can be considered to be really efficient. Many variables can be evaluated to determine the real efficiency of the cluster such as: user experience, performance, security, among others. However, the performance obtained from the HPL test will only

be evaluated in order to show that it is possible to deploy an HPC cluster using this methodology, obtaining acceptable performance values. Finally, the objective of this methodology is to provide a guide that allows rapid deployment, both to experienced administrators and those new to the HPC world, and that is adaptable to the knowledge and experience of administrators.

3.1 Buildind a Efficient Cluster

Basic Architecture. In this first part, a general proposal of the organization of the elements that are considered basic in the deployment of an HPC cluster is presented. Two main points have been taken into account: ease of administration and speed of communication. The design scheme is presented in Fig. 2.

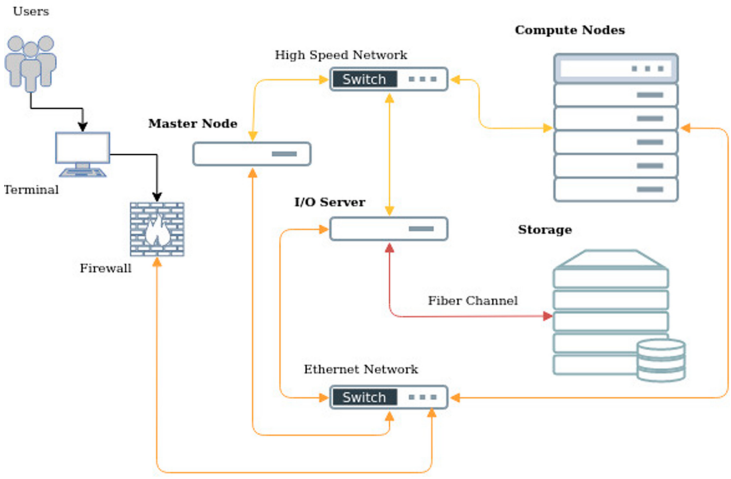

Fig. 2. General system architecture

In the age of artificial intelligence and big data, large volumes of data are common in HPC clusters, making the storage system one of the main elements to consider. The characteristics of these systems should allow for high transfer rates along with low latency, redundancy, and above all high storage capacity. Some HPC system designs typically mount the volumes created on the storage system directly on the master node, which does not incur functionality issues, but can affect performance by assigning cluster management tasks and traffic between the storage system and compute nodes. For this reason, we recommend that you have a unique I/O server that handles the transactional load on the cluster.

In relation to the above, another important element to consider is high-speed networks that are not included in many HPC clusters due to high acquisition costs but can consequently lead to network bottlenecks when there is a high

demand for the network. In the implementation of this methodology, two high-speed networks have been used: Infiniband for communication and synchronization between nodes and a fiber-optic channel for the storage system and I/O node communication. It should be noted that new computing requirements and the high scalability of emerging applications make this type of high-speed network an essential part of any HPC cluster.

Finally, in the Fig. 2 shows other basic elements in any HPC cluster deployment such as ethernet switches, the master node, compute nodes, and an edge protection system for accessing the cluster from external networks.

Network Configuration. This methodology bases your network configuration on the recommendations that OpenHPC provides in your installation guide, however, we talk about general recommendations because each infrastructure differs according to the components at your disposal. Standards such as the Uptime Institute[2] recommend that all Datacenters have redundancy across all systems, including the network, but for this methodology, we assume that only the essential switches are available for each of the networks implemented in Fig. 2.

It is proposed to implement two base networks for the cluster that will be separated as two different LANs within the configuration. The first will be responsible for the communication between the master node and the compute nodes and will be for the exclusive use of the resource manager. The second network will be responsible for data traffic between the master node, compute nodes, and the I/O server. To provide Internet access to compute nodes, the gateway can be redirected to the IP of the master node, keeping in mind that the master node must have access to the Internet. This solution is not optimal, but it can simplify the administrator's work. It should be noted that this can cause network bottlenecks dedicated to resource management especially if applications running in the cluster require access to large databases available on remote servers. Therefore, the most recommended solution is the implementation of a third network that handles this external traffic, although this configuration makes it necessary to count an extra server that serves as a gateway for the output to the Internet.

Base System. In what has been described so far, a basic organization of the essential components in an HPC cluster has been shown together with the minimum network configuration required to obtain an acceptable performance and that can be considered efficient or, in other words, that performs its function in the most appropriate way.

Within this framework, the most important and essential software components for administrators have been considered, without addressing their installation and configuration. In Sect. 2.2. It was mentioned that OpenHPC has been designed to provide administrators with best practices and a software repository that allows the system base to be easily installed, configured, and updated.

[2] https://uptimeinstitute.com/.

However, it was not mentioned that within this repository there are several tools that perform similar tasks and that it is the administrator who must make the selection of which ones were used, taking into account the experience of this and their knowledge about them. These tools range from choosing the task handler between Slurm or PBS, to a system file system such as BeeGFS or Lustre.

The following describes the tools that were considered in the cluster deployment process using the OpenHPC repository and that serve as the basis for the implementation methodology suggested in this job:

- **NTP (Network Time Protocol):** It is an Internet protocol that is used to synchronize clocks from different computer systems on local or global networks. For the suggested configuration, the cluster master node will be used as the NTP server and will be responsible for keeping the system clock synchronized on all compute nodes and the I/O server in the cluster. Examples of the need for cluster synchronization can be seen in co-scheduling techniques in parallel applications with sensitive bulk synchronous workloads, (ii) performance analysis tools and (iii) autotuning strategies that want to exploit State-of-the-Art (SoA) high-resolution monitoring systems [12].
- **NFS (Network File System):** It is a client/server file system that allows users to access files and folders over the network and treat them as if they were local. It will be used primarily for each user's /home directory and for /opt where the platform software will be installed. These two directories belong to two logical volumes of the cluster storage system and are exported by the I/O server to the different system nodes. [19].
- **Support for Infiniband:** Infiniband is a network communication standard that provides high throughput and low latency. This type of high-speed network is not required for cluster operation, but as mentioned earlier in the network configuration section, the bottlenecks generated by high file transfer and the communication required between nodes by using inadequate networks make it an essential part of deploying an efficient cluster. In fact, OpenHPC also comes with included support for Omni-Path but Infiniband has been selected for the current cluster configuration where this methodology has been developed, but it does not mean that better results cannot be presented with Intel technology[3].
- **Memory usage limits:** Linux systems have the ability to limit the system resources that are available to user processes, and one of these limitations is the use of memory by the different components of a process that is running. Good practice in HPC is to establish new rules that allow the execution of demanding tasks by parts of users, which include new rules for memory limits and the maximum number of open files.
- **HPC modules - LMOD:** In short, a module is the setting of environment variables within a script. Each module is defined for a specified application where their respective environment variables, license files if required, are defined among the other requirements required for its successful execution.

[3] https://www.intel.com/content/www/us/en/high-performance-computing-fabrics/omni-path-architecture-performance-overview.html.

It should be noted that there are two types of scripts when defining modules. The first one is using TCL[4] while the second one is using LUA[5], which is distinguished from the first by using the .lua extension. Both types work for the creation of a basic application module that only requires defining the typical environment variables such as PATH, LD_LIBRARY_PATH, among others, but LUA has a number of functions[6] that allow the administrator to create more optimized modules[7]. The use of Spack[8] is also recommended, which provides the cluster with a tool for managing multiple versions and software configurations through environment modules. In the proposed implementation Spack was used instead of EasyBuild[9] due to the experience of the administrators in this tool, but it can be used both software repositories if so required and if deemed necessary by the administrator.

- **PowerShell:** It is a tool developed in Python that allows you to execute commands in parallel on all the nodes of the cluster and is highly scalable[10]. These types of tools are essential and allow simplifying routine tasks such as updates, maintenance, making copies of files and folders in directories not mounted via NFS. This tool was chosen over Clusterssh[11] motivated by the search for simplification in administrative tasks and the reduction of implementation times of a new cluster.
- **NHC (node health check):** One of the main tasks of administrators is to ensure the correct operation of each node of the HPC platform. NHC allows SLURM to monitor that each node is working properly, preventing user jobs from running on nodes marked as unhealthy. If SLURM finds any hardware failure or misconfiguration reported by NHC, the node is marked drained.

In the previous generalizations, the entire base system required for a basic implementation of a cluster is shown, keeping in mind the fundamental idea of providing those tools that improve the efficiency of the cluster. The next task to carry out is the choice of the task manager and its configuration. It was mentioned earlier that OpenHPC has two options, Slurm or PBS. The proposed implementation methodology will discuss Slurm and some configuration options that are considered necessary to improve performance and efficiency, but it does not mean that the use of PBS will incur design deficiencies or any decrease inefficiency.

SLURM. One of the most time-consuming tasks in the implementation of an HPC cluster is the installation and configuration of the task manager, especially

[4] https://en.wikipedia.org/wiki/Tcl.
[5] https://en.wikipedia.org/wiki/Lua_(programming_language).
[6] https://lmod.readthedocs.io/en/latest/050_lua_modulefiles.html#lua-modulefile-functions-label.
[7] https://lmod.readthedocs.io/en/latest/015_writing_modules.html.
[8] https://spack.readthedocs.io/en/latest/.
[9] https://easybuild.readthedocs.io/en/latest/.
[10] https://clustershell.readthedocs.io/en/latest/.
[11] https://github.com/duncs/clusterssh.

when you want to include certain features such as Infiniband within its configuration. Advanced administrators have the experience to easily deal with these settings, but it can still take some time to get it working properly. Therefore, it is proposed to use the SLURM and MUNGE[12] packages that come in the OpenHPC repository. Finally, it only remains to dedicate time to configuring Slurm according to the characteristics of the cluster and the policies for the task manager that are defined by the administrator.

There are a large number of configuration options in the slurm.conf file[13] that will not be detailed as they are outside the scope of this paper. However, the configuration parameters related to the SCHEDULING section will be discussed and some recommendations for their configuration will be given that will help the administrator to improve the efficiency of the use of resources.

- **SchedulerType:** Specifies the scheduler plugin to use. This parameter has two options: **schedbackfill** and **schedbuiltin**. The default option is schedbackfill and we recommended use it but the following parameters should be established to improve to the scheduler: **DefaultTime** (default job time limit), **MaxTime** (Maximum job time limit) and **OverTimeLimit** (Amount by which a job can exceed its time limit before it is killed). The optimal values of these parameters must be set according to the infrastructure and it is a task to trial and error.
- **SelectType:** Establishes how the resources of each node are used. The default option allocates nodes to jobs in exclusive mode, in other words, another job can not use the node even if resources are available. The best form to use the total resources is to set this parameter in cons_res (consumable resource) allowing manage them on a much more fine-grained basis.
- **SelectTypeParameters:** Consumable resources in our cluster. There are several values but the main consumables are the memory and the cores of the nodes, so, we set this value **CR_Core_Memory**. The rest of the configurations depend on the cluster and must be established by the systems administrator.
- **PriorityType**: By default, SLURM use FIFO (First In, First Out) to assigns the run priorities to each job. The best option in an efficient cluster is to set this value to priority/multifactor. This value depends on another series of parameters to calculate the priority of each of the jobs and they no will show in this paper but the values of these parameters are a task to trial and error [20].

It should be noted again that the detail of the SLURM configuration is not entered into because of the differentiation that must be made in the configuration depending on the hardware resources and the policies defined by the administrators. Likewise, the configuration of generic resources (GRES) such as GPU cards is not mentioned and the task/cgroup plugin is not included, but the

[12] https://github.com/dun/munge.
[13] https://slurm.schedmd.com/SLUG19/Priority_and_Fair_Trees.pdf.

reader is recommended to delve into this topic if its configuration requires the use of these resources[14].

Finally, it is worth remembering that within the SLURM configuration file you must specify the use of NHC after all the configuration of the nodes has been performed and their characteristics added to the slurm.conf file.

Lightweight Directory Access Protocol. Despite the fact that OpenLDAP is not an essential tool in terms of performance if it is in terms of efficiency and is mentioned here as one of the main components of a functional cluster.

LDAP is typically configured with its basic schema and is not usually modified to its tree because it is sufficient for system user management. However, we have noted that collaborative work has increased among users, leading to storage spaces shared among members of a research group or involved in the same project. Therefore, good practice in implementing LDAP is the use of dynamic groups [21] that allows you to assign different levels of access to different storage spaces within the HPC platform.

In this sense, it is understood that the LDAP service is essential and must have a mirror server that provides high availability of access to the platform. In other words, if the main access server experiences a service outage, the backup server is expected to offer the services while the main server recovers. This type of design requires a more dense infrastructure that the vast majority of small HPC labs do not have. To avoid access problems to the platform in the event of a total crash of the LDAP servers or if there is no redundant server, SSSD is used, which was explained in Sect. 2.5.

Finally, not only user access to the platform should be regulated. Storage spaces such as the user's home and project and research group folders must have restrictions that are implemented through the use of disk quotas in conjunction with LDAP.

This series of cluster implementation steps are proposed as an agile and efficient methodology that can be replicated in other HPC laboratories and that can optimize the task manager and the administration of the cluster and users by the administrator. The following section will show the results obtained from the HPL benchmark in order to show the correct operation of the cluster using the proposed methodology. However, these results are not directly related to various administrative tools that have been presented in this work and that cannot be quantitatively evaluated and are recommendations of the authors based on their experience in the implementation and administration of HPC laboratories.

4 Benchmarks and Results

HPC labs usually evaluate the performance of their clusters by running different benchmarks where the most common is Linpack and the most used implementation is HPL[15] explained in Sect. 2.6. This in turn is the tool used to position

[14] https://slurm.schedmd.com/SLUG19/cgroups_and_pam_slurm_adopt.pdf.
[15] https://www.netlib.org/benchmark/hpl.

the most powerful supercomputers in the world listed in the TOP500. It should be noted that these performance measures obtained and the comparison made with the five most powerful supercomputers in the world do not really show the efficiency of the cluster itself, but they do show its correct operation by using the proposed methodology.

4.1 Results and Evaluation

The cluster for which the design and implementation was carried out following the proposed methodology is made up of 14 servers

- 11 Servers:
  - ProLiant SL390s G7
  - 2 Intel Xeon E2.40 GHz processors
  - 102 GB of RAM
- 3 Servers:
  - ProLiant SL390s G7
  - 2 Intel Xeon E2.67 GHz processors
  - 102 GB of RAM
- Infiniband Mellanox Technologies MT26438 IB QDR/10GigE of Mellanox

Table 1 shows the results obtained by carrying out three tests on the cluster. Two of them were made for the Intel Xeon E5645 processor model using 6 and 11 nodes, the third test was performed on the three nodes with Intel Xeon E5640 processor. The parameter values for HPL were found using the equations presented in Sect. 2.6. The NB value was selected from the values 96, 112, 128, and 144 where the best Rpeak was for the value 112.

The $R_{peak\_cluster}$ of the cluster is obtained by multiplying the value of R_{peak} by the number of processors in a server by the number of servers used in the measurement.

Table 1 shows the results obtained by carrying out three tests on the cluster. Two of them were made for the Intel Xeon E5645 processor model using 6 and 11 nodes, the third test was performed on the three nodes with Intel Xeon E5640 processor. The parameter values for HPL were found using the equations presented in Sect. 2.6. The NB value was selected from the values 96, 112, 128, and 144 where the best Rpeak was for the value 112.

Table 1. HPL best results

Processor	Nodes	MPI Proc	NB	PxQ	N	R_{peak}	$R_{peak\_cluster}$	R_{max}	Efficiency
Xeon E5645	6	144	112	12×12	256256	57,6	691,2	556,9	0,804
	11	264	112	12×22	372512	57,6	1267,2	1035	0,8168
Xeon E5640	3	48	112	8×6	162064	42,72	256,32	215,84	0,842

Table 2 shows the performances of the first 5 supercomputing machines presented in the TOP500 in November 2019[16]. These results obtained by the implementation of the methodology described in this document show an acceptable operation by simplifying the tasks of deploying an HPC cluster.

Table 2. First HPC supercomputers - TOP500

Rank	System	R_{max}	R_{peak}	Effiency
1	Summit	148600	200749,9	0,7402
2	Sierra	94640	125712	0,7589
3	Sunway TaihuLight	93014,6	125435,9	0,7415
4	Tianhe-2A	61445,5	100378,7	0,6121
5	Frontera	23516,4	38745,9	0,6069

5 Conclusions

In conclusion, the use of the proposed methodology in the design and implementation of the cluster relying on the software repository and the best practice recommendations provided by OpenHPC simplified the tasks for the HPC laboratory start-up. In fact, the decrease in platform update times was also observed along with the complexity of installation and configuration of certain scientific applications through the use of Spack. Finally, according to the results obtained from the HPL tests, it was observed that the implementation of the cluster using the described methodology presents good performance results when executing tasks that require intensive computation.

References

1. Schulz, K.W., et al.: Cluster computing with OpenHPC. In: HPC Systems Professionals Workshop (2016)
2. Thornton, J.E.: The CDC 6600 Project. Ann. Hist. Comput. **2**(4), 338–348 (1980). https://doi.org/10.1109/MAHC.1980.10044
3. Sen, S.K., Agarwal, R.P.: Computing: birth, growth, exaflops computation and beyond. In: Flaut, D., Hošková-Mayerová, Š., Ispas, C., Maturo, F., Flaut, C. (eds.) Decision Making in Social Sciences: Between Traditions and Innovations. SSDC, vol. 247, pp. 3–47. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-30659-5_1
4. Papadopoulos, P.M., Katz, M.J., Bruno, G.: NPACI rocks: tools and techniques for easily deploying manageable Linux clusters. Concurr. Comput.: Pract. Exp. **15**(7–8), 707–725 (2003)

[16] https://www.top500.org/lists/2019/11/.

5. Scott, S.L.: OSCAR and the Beowulf arms race for the "cluster standard". In: 2001 IEEE International Conference on Cluster Computing (CLUSTER 2001), 8–11 October 2001, p. 137, Newport Beach (2001)
6. Aydin, S., Bay, O.F.: Building a high performance computing clusters to use in computing course applications. Procedia - Soc. Behav. Sci. **1**(1), 2396–2401 (2009)
7. Hoste, K., Timmerman, J., Georges, A., Weirdt, S.D.: EasyBuild: building software with ease. In: 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, Salt Lake City, UT, USA, 10–16 November 2012, pp. 572–582 (2012)
8. Gamblin, T., et al.: The spack package manager: bringing order to HPC software chaos. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2015, Austin, TX, USA, 15–20 November 2015, pp. 40:1–40:12 (2015)
9. Yoo, A.B., Jette, M.A., Grondona, M.: SLURM: simple Linux utility for resource management. In: Feitelson, D., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2003. LNCS, vol. 2862, pp. 44–60. Springer, Heidelberg (2003). https://doi.org/10.1007/10968987_3
10. Wang, L., et al.: BOPS, Not FLOPS! a new metric and roofline performance model for datacenter computing (2018). http://arxiv.org/abs/1801.09212
11. Dongarra, J., Luszczek, P., Petitet, A.: The LINPACK benchmark: past, present and future. Concurr. Comput.: Pract. Exper. **15**, 803–820 (2003). https://doi.org/10.1002/cpe.728
12. Libri, A., Bartolini, A., Cesarini, D., Benini, L.: Evaluation of NTP/PTP fine-grain synchronization performance in HPC clusters. In: ACM International Conference Proceeding Series (2018)
13. Supercomputación y Cálculo Científico (SC3). https://www.sc3.uis.edu.co. Accessed 20 May 2020
14. Top500. https://www.top500.org/. Accessed 20 May 2020
15. Clustering fundamentals. https://developer.ibm.com/articles/l-cluster1/. Accessed 12 May 2020
16. Lightweight Directory Access Protocol (LDAP). http://web.mit.edu/rhel-doc/5/RHEL-5-manual/Deployment_Guide-en-US/ch-ldap.html. Accessed 5 May 2020
17. SLURM Overview. https://slurm.schedmd.com/overview.html. Accessed 8 May 2020
18. SSSD. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/migration_planning_guide/sect-migration_guide-security_authentication-sssd. Accessed 15 May 2020
19. Network File System (NFS). https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/storage_administration_guide/ch-nfs. Accessed 20 May 2020
20. SLURM Priority Multifactor. https://slurm.schedmd.com/priority_multifactor.html. Accessed 15 May 2020
21. ZYTRAX - Configuring Dynamic Groups. https://www.zytrax.com/books/ldap/ch11/dynamic.html. Accessed 2 May 2020