Thinking BigData: Motivation, Results and a Few Recipes for a Balanced Growth of High Performance Computing in Academia

Thinking BigData: Motivation, Results and a Few Recipes for a Balanced Growth of High Performance Computing in Academia

P. Margara, N. Nepote, E. Piccolo, C. G. Demartini and P. Montuschi
Department of Control and Computer Engineering - Politecnico di Torino
C.so Duca Degli Abruzzi 24, 10129 Torino (TO)
[paolo.margara, nicolo.nepote, elio.piccolo, claudio.demartini, paolo.montuschi]@polito.it

Big Data is today both an emerging research area and a real present and future demand. High Performance Computing (HPC) Centers cannot neglect this fact and have to be reshaped to fulfill this need. In this paper we share our experience of upgrading a HPC Center at Politecnico di Torino, originally designed and deployed in 2010. We believe that this issue could be common to some other existing "general purpose" HPC centers where, at least in the short term, the possibility to start from scratch a new Big Data HPC center cannot be afforded but a balanced upgrade of the existing system has to be preferred.

Keywords: Big Data, High Performance Computing, HPC, MapReduce, Hadoop, Parallel Systems, Academia.

1. Introduction

A possible solution to survive the economic crisis and keep the global competition in all fields, is to develop large-scale projects. From crash simulation to disaster and climate modeling, from ethical clinical trials to drug discovery, from Graphene to The Brain Project, from nano-technologies to smart cities. All these studies share commonalities: the demand for fast, complex computation and the need to analyze huge datasets.

So far, these two aspects have been treated separately, leading both to the development of parallel algorithms running on supercomputers, and to the birth of data mining which exploits large DBMS. Today we need instead a greater integration between the tools for processing massive amounts of data and the supporting underlying computation architecture. The key to this breakthrough has a name: Big Data. Differences are more and more emerging between large amounts of data and Big Data, both in terms of correlation with physical or natural events that generate them, and also for the semantics that are assigned to them: it is a complete reversal of perspective. In other words, the first has a background model, while the second is assumed to be able to derive different models, even in many different fields [Kindratenko & Trancoso, 2011].

Dealing with Big Data has therefore become a stimulating challenge for many small-to-medium sized supercomputing centers, also in Academia. For three years our High Performance Computing (HPC) center at Politecnico di Torino has provided support to research groups by means of a standard cluster widely described in [Della Croce et al. 2011] [Nepote et al. 2013]. So far, the demands focused on rough computing power rather than Big Data analysis, but now the time has come to take up the challenge.

We have taken this new demand both as a challenge and as an opportunity, and 6+ months ago have started a project of upgrading our HPC center to efficiently run also tasks massively using Big Data.

At the beginning we figured out that this request could be addressed by installing some new packages followed by proper tuning of some parameters. The results of this first experimental phase have been very negative but have stimulated us to explore further and think at more structural solutions, by entering each specific problem with the aim to find feasible solutions besides the trivial one to start from scratch a completely new Big Data HPC center.

Today we can proudly say that, given the small resources available, we have not only met our goals, but also obtained a set of specifications for future balanced upgrades of our HPC. As we believe that this issue could be common especially nowadays as Big Data is emerging as a new and real requirement, we have decided to share our choices and experience with other Readers both to get their feedback and to transmit what we have learned during this enhancement of our HPC system.

In the following sections we outline the steps taken to make our HPC system able to run MapReduce [Dean & Ghemawat, 2008] tasks using Apache Hadoop, the results obtained, and the considerations that have followed.

2. The CASPER cluster at PoliTO

With HPC@POLITO we refer to the initiative boosted in 2010 to start a "supercomputing" center at Politecnico di Torino, and aimed at providing computational resources and technical support to both academic research and university teaching. Over the years, the computing center has set up a general-purpose and campus-wide available and dynamically evolving cluster called CASPER (Cluster Appliance for Scientific Parallel Execution and Rendering), now serving several research groups operating in different areas [Della Croce et al. 2011] [Nepote et al. 2013].

CASPER is a Linux-based MIMD distributed shared memory InfiniBand heterogeneous cluster, reaching 1.3 TFLOPS with its 136 cores and 632 GB of overall main memory. As the majority of clusters, in 2010 CASPER was designed to focus more on the need to perform several computations on relatively small amount of data possibly stored together with the Operating System, inside a small local hard disk of each computational node of the architecture. The role of shared resource to store all experimental data, especially when it was too much large for the local disks, was played by a Network Attached Storage (NAS).

Over the years, this initial configuration (Figure 1) has mostly evolved towards the increasing of the local node computational power, as the applications used were more computation-intensive than data-intensive, i.e. they were very rarely were using big amount of data.

In Figure 1 it is depicted the current structure of CASPER when observed from the viewpoint of the general job scheduler. It can be noted that the frontend node provides all the storage, as local disks are only used to maintain cache and temporary files. In CASPER communications occur under the umbrella of InfiniBand connection, which is fast and with low latency linking all the nodes in the cluster as well as coexisting with the Ethernet network. To run custom code or third-party software, including also the widely used Message Passing Interface (MPI) libraries, CASPER is normally operated through the industry standard job scheduler and resource manager Oracle GridEngine (formerly known as Sun GridEngine).

Several research activities have been supported by HPC@POLITO, where, during the past three years 30+ research projects and 20+ scientific papers have benefited of CASPER computational resources.

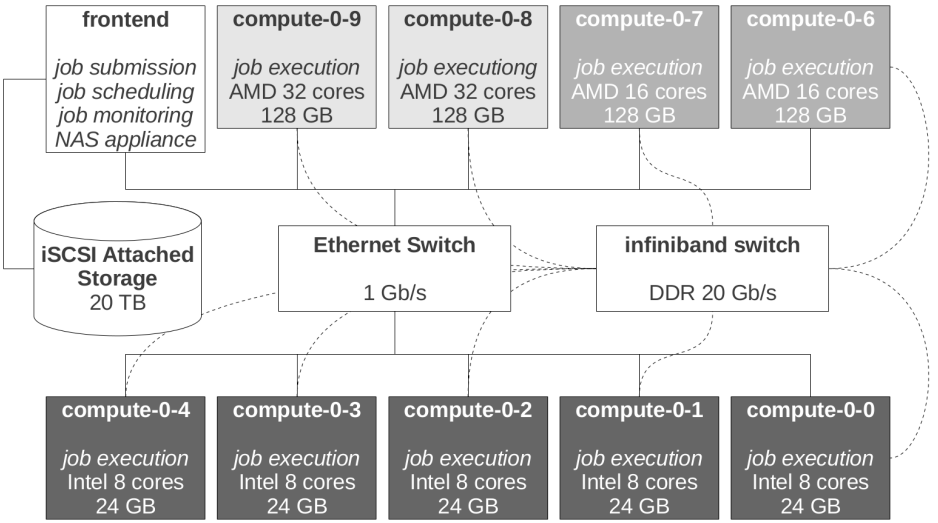

Figure 1: The CASPER cluster configuration (mid 2013) from SGE's point of view.

3. Addressing the BigData problem

3.1 Motivation and our choice

During the last academic year, we have experienced the starting of a new need: the capability to perform analysis of large amounts of data has been more and more often required by CASPER users. Due to budget restrictions, we have immediately discarded the option of a complete rejuvenation and/or replacement of our cluster with another one specifically designed for Big Data. Therefore, we have decided to study the feasibility of "upgrading" the current configuration to tackle also Big Data. We have started by verifying if an implementation of a framework for Big Data processing would perform well inside a general purpose HPC system like CASPER and, if so, what changes to the general architecture would have been necessary to achieve better performances. We believe that this issue could be common and therefore we are here to share our choices and experience with other Readers both to get their feedback and to transmit what we have learned.

After long discussions, some pre-tests and a deep analysis of advantages/drawbacks/tradeoffs/costs, we have converged to a first-experimental solution adopting the Apache Hadoop open source framework, which exploits the well known *MapReduce* [Dean & Ghemawat, 2008] paradigm for the efficient implementation of data-intensive distributed applications. Besides these characteristics, Hadoop is also well known to: run very well on large clusters of commodity hardware, provide a framework possibly minimizing the impact of hardware failures and exploit local computation and storage.

3.2 General Remarks

Before starting the deployment, we spent some more time to analyze the current configuration of the cluster to identify and possibly quickly address critical issues of a straight implementation of Hadoop on CASPER. We focused our attention on the main components, i.e. storage, CPU & Memory, and Network. In the following we have listed our main findings and issues.

- **Storage**

  Fact: Hadoop accesses memory by using its own distributed file system, called HDFS. To prevent storage from becoming the bottleneck, it is necessary to properly harmonize the configuration of the physical storage and the way it is accessed by the software.

  Problem: We observed that, being a distributed file system, HDFS uses the local storage resources of the individual computational nodes. However, usually HPC general purpose clusters (like CASPER), are based on centralized storage systems (such as a NAS). In such cases, the computational nodes of the cluster are equipped with low performance very basic local hard disks, very poorly suitable to efficiently store and handle any big data. In Figure 2 we observe that in CASPER the amount of local disk space available for each CPU has even decreased during the years. Other HPC cluster configurations have even diskless computational nodes bootstrapping from the network and using a small ramdisk for local operations.

  Our Solution: An easy way to workaround the problem before installing Hadoop, is to add more local hard disks on each computational node for the exclusive use of HDFS.

- **CPU and memory**

  Fact: Hadoop exploits CPU and memory especially during the Map phase of the MapReduce process. At this stage, many tasks are executed in parallel by all the Mappers processes running on each computational node. For correct operation, all CPUs in the cluster should have similar performances. If not, the slower ones would represent a bottleneck. In other words the cluster should be as much symmetric (or homogeneous) as possible.

  Problem: The last nodes (from compute-0-6 to compute-0-9 in Figure 1) added to the initial configuration of CASPER were acquired as part of "fellowship" agreements [Della Croce et al. 2011] with some research groups. The needs of these groups have therefore driven the growth of CASPER, basically towards three different types of computational nodes (Figures 1 and 3), while at the same time aiming the RAM/CPU ratio to the initially target value of 4 GB (Figure 2). The outcome of this growth process is that the four most recent nodes are characterized by more CPUs operating at lower frequencies than these of the first five nodes, thus resulting into a current configuration of CASPER which is highly heterogeneous.

  Our Solution: This problem is difficult to solve without radically changing the current configuration of the cluster. A quick, but limiting, workaround could be to exclude the slower computational nodes from the pool used by Hadoop. We will discuss more effective solutions in our “Recipe list” in section 3.5.

- **Network**

- *Fact:* HDFS commonly relies on IP for node-to-node communication. Much attention should be paid while choosing the type of Layer 2 network on which IP traffic flows, in order to ensure high performances of HDFS when transferring blocks of data between nodes.

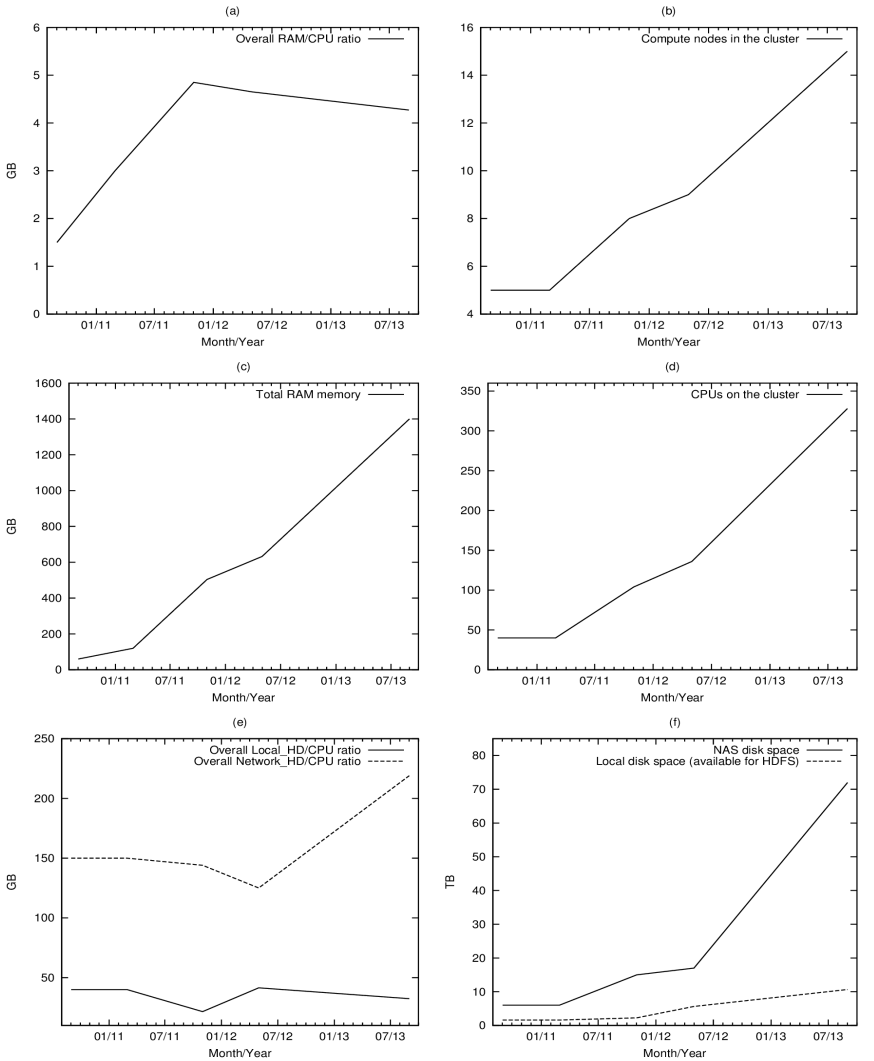

Figure 2: Temporal evolution of some hardware features of the CASPER cluster

Problem: CASPER has a total of three Layer 2 networks carrying different information. A InfiniBand DDR 20 Gb/s network carries all the MPI node-to-node data transfer during standard jobs (i.e. not using Hadoop) execution. A dedicated 4x1 Gigabit Ethernet aggregate link connects the central NAS storage to the nodes via NFS. Finally, a Gigabit Ethernet (Gbe) network still remains as a supporting layer 2 for carrying the remaining IP traffic, usually related to system management communications or data generated by monitoring tools. Unfortunately HDFS traffic falls under the slower Gbe network.

Our Solution: The only possible solution that we have found to be effective in our environment without doing a complete redesign of the cluster, is to implement an aggregated link also on each slave node.

3.3 Hadoop deployment and tuning

The installation of Apache Hadoop [Garza et al. 2013] was made by trying to harmonize its needs to those of our specific HPC system. From the operating system viewpoint, CASPER is an installation of ROCKS Cluster Distribution 5.4.31. Therefore, to install Hadoop 1.0.4 we used the packages from the EPEL repository for CentOS, as derived by ROCKS. As it can be observed from Figure 3 , one of our main choices has been to use the frontend node of the cluster as master node of Hadoop and the computation nodes as slave nodes.

Targeting to performances improvements HDFS was configured with permissions disabled, data block size set to 256MB and number of replicas per data block set to 2. On the first five nodes of the cluster (from *Slave 0* to *Slave 4* in Figure 3), we added one single 1TB 7200 rpm Sata-3 local hard disks for the exclusive use of HDFS. On the remaining four nodes (the most recent) this was not possible because of lack of physical space in the case and therefore the available storage for HDFS is only 160GB.

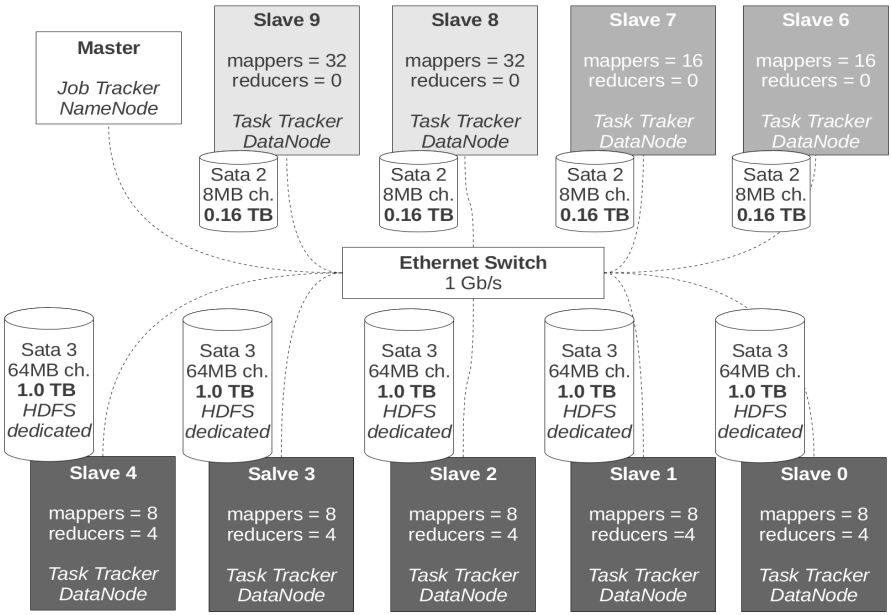

Figure 3: The CASPER cluster configuration (mid 2013) from Hadoop's point of view.

We then configured Hadoop so as to meet the heterogeneous nature requirements of CASPER. Specifically, the maximum number of *mappers* for each node was set equal to the number of cores, while the maximum number of *reducers* was set to 4 for the five nodes with a large and efficient 1TB local disk, and to 0 for the other four nodes, in order to comply with potential decreases of the overall file system performances due to the slow

efficiency and small size and their hard disks.

3.4 Benchmarking the system

We run a set of experiments based on Hadoop to evaluate its performances and scalability on CASPER, in order to possibly identify which upgrades could be useful for CASPER to become an efficient system for Big Data. In particular, we run one of the most widely used and available benchmarking tests to analyze efficiency and scalability on Hadoop, i.e. the Hadoop-based implementation of *TeraSort*. For the purpose of running TeraSort in different boundary conditions, we used two datasets of sizes 100GB and 200GB respectively.

As it is well known that for Hadoop, asymmetry equals inefficiency, we decided to evaluate TeraSort this two different configurations of CASPER:

- The **SomeNodes** test configuration is based solely on the 5 first nodes of the cluster, labeled in Figure 3 as *Slave 0* to *Slave 4*. It is the only possible homogeneous configuration that we can build in CASPER using more than two nodes. It relies on 5 Intel 3.2GHz nodes with 1TB of secondary memory per node, accounting a total of 40 CPUs. The 1TB disk was added before deploying Hadoop to possibly enhance HDFS performances.

- The **AllCluster** test configuration is basically an extension (or an enhanced version) of *SomeNodes* exploiting all the resources of the cluster. It is designed to squeeze every drop of power from CASPER, but it is also extremely heterogeneous. It is based on 136 CPUs with different frequencies (3.2 GHz and 2.4 GHz) and local disks with size ranging from 160GB to 1TB. This configuration has +240% more CPUs than *SomeNodes*, therefore the expected execution time decrease when running TeraSort should be -58%.

The results reported in Table 2 show that TeraSort can process in less than 4 hours a 200GB dataset and that, even on a cluster that was not specifically designed for Hadoop, it is possible to process files larger than those manageable by traditional parallel frameworks like MPI [Garza et al. 2013].

However, the time scored by the *SomeNodes* configuration is only slightly worser than the one scored by the *AllCluster* configuration. The *AllCluster* configuration decreases the execution time by only -13% when the file size is 200GB and -31% when the file size is 100GB, thus showing that the extra 4 non-homogeneous nodes appear not to contribute in a relevant way to the overall computations.

Name	CPUs	Tot. Memory	Tot. HDFS size	Mappers	Reducers
SomeNodes	40	120 GB	1.6 TB	40	20
AllCluster	40+96	632 GB	5.64 TB	136	20

Table 1: Test configurations chosen for TeraSort

Dataset	Test Configuration	Execution Time
100 GB	SomeNodes	1h23m2s
	AllCluster	57m26s
200 GB	SomeNodes	3h52m57s
	AllCluster	3h21m49s

Table 2: TeraSort execution time on CASPER

We have identified some possible reasons for this result on the *AllCluster* configuration:

- Having set to zero the number of *reducers* on the most recent slaves without additional 1TB disk, has allowed us to not degrade the performance of HDFS, but at the same time has forced the same slaves to transfer all the data to the first 5 slaves having at least 4 *reducers* (see Figure 3) . This generates a quantity of traffic on the Gigabit Ethernet network so high that the advantage given by having more than double the mappers is almost completely zeroed.

- Although *AllCluster* involves several additional CPUs, their operating frequency is too low: 2.4 Ghz versus 3.2 Ghz of the nodes of *SomeNodes*. This has the effect of slowering the overall *map()* phase of the MapReduce paradigm, thus showing that this configuration is probably too much heterogeneous.

3.5 Some recipes for a better Hadoop implementation on general purpose HPC

Based on the results that we have obtained, it can be observed that a slightly "upgraded" CASPER, and in general also possibly any similar general-purpose architecture, can potentially be used to run also the new Hadoop-based applications. The key is what and how to implement the slightly upgrading specially targeted to improve performances on large datasets.

While integrating Hadoop within a general purpose HPC cluster as CASPER we identified and set up some possible recommendations to achieve integration with good performance at the same avoiding the need to change the architecture of the cluster itself. In particular, we have observed that the main ingredients are, as expected, **Network**, **Distributed Storage** and **Computational Power**. Clearly, their combination, interoperability and configuration can really make the difference. Here below our "recipes", i.e. how to weight, size and combine the different ingredients.

- **Network:** Communication in Hadoop suggest a Gigabit Ethernet connection for the slave nodes, and a faster connection for the master node, such as:

  - An aggregate of several 1 Gigabit Ethernet links. Incidentally, this has been proven to be working well with the current configuration of CASPER.

  - The combination of a faster 10 Gigabit Ethernet link on the master node and some aggregate links on the slaves.

  - An InfiniBand link, if already available in the cluster, implementing the IP-over-InfiniBand stack, which unfortunately implies a complete redesign of the cluster.

- **Distributed storage:** The distributed storage in a Hadoop based cluster is managed through HDFS, which has a master/slave architecture. In HDFS, a cluster consists of: at least one *NameNode*, a master server that manages the file system namespace and controls access to files by clients, and several *DataNodes,* usually one per node in the cluster, each one managing their local node storage. Under this premise, our thoughts are:

  - We strongly advise to setup a RAID on the *NameNode*, since it contains all the metadata needed by HDFS to ensure data consistency. On the contrary, no RAID configuration is required on *DataNodes*, as HDFS already operates

with intrinsic internal data redundancy.

- Regarding the number of disks on the slave nodes, we would like to recommend a configuration with 2 disks, one for the exclusive use of HDFS and another one for storing temporary files needed for processing the tasks that will be assigned to the computational nodes. As in a general purpose HPC cluster the computational nodes could also be diskless, it could be likely that often it is not possible to add more than 1 or 2 additional disks per node due to physical limitations of the individual nodes. In general, in a cluster specifically designed for Hadoop the number of disks recommended for node usually ranges from 2 to a number depending by the number of cores available on the node and the physical space needed/available. In our case we have tested CASPER to exhibit good performances with one additional 1 TeraByte disk, as we did not have more physical space to install more disks.

- As the frequency of communication between *NameNode* and *DataNodes* increases together with the fragmentation of the data (i.e. when several small files are often exchanged), in order to reduce the negative impact of a possibly slow *NameNode*, it is highly recommended to equip the master node with both enough RAM memory and computing power. A fine tuning on “how much”, depends on number and size of files that will be likely stored into HDFS, and can be done by running some practical experiments.

- **Computational power:** Into a Hadoop cluster the jobs are submitted to the MapReduce framework consisting of a single master *JobTracker* and one slave *TaskTracker* per slave node. The master is responsible for: scheduling the tasks of each job on the slaves, monitoring them and re-executing the failed tasks.

  - For the master node we have not identified additional relevant constraints besides these outlined in the previous points.

  - On the other hand, slave nodes require a sufficiently large memory size to keep the processor busy without frequently swapping. Although this is highly dependent on the type of job that will run on the cluster, we have found that satisfactory performances for basic configuration are obtained with slave nodes equipped with 8 cores and 24GB of RAM.

  - As we have seen in the previous sections, good performances require a homogeneous system with at least the same choice of hardware for the slave nodes both in the number of cores and in the frequencies of the processors. In CASPER we have experienced that any asymmetry between the individual nodes can cause re-schedule of tasks and long waiting times between the operations of *map()* and *reduce()*, due to the different execution times of similar tasks scheduled on differently equipped nodes. Unfortunately, in our case these issues should have been addressed both in the design phase of the initial configuration (i.e. when CASPER was designed in 2010) and in its following upgrades, as later (i.e. today) there is no way to make such changes without full replacing at least some of the computational nodes.

Additional points that, based on our experience, we recommend to observe are:

- Future upgrades of the system should comply with scalability and keep the system as most homogeneous as possible.

- Based on our tests, Hadoop seems to achieve better results when the nodes have sufficiently large and efficient local disks. In other words, the choice of the disk's

size and type will be in a trade-off between performances and total size of the distributed file system, based on the footprint that experimental data will have at runtime (i.e. considering intermediate writes).

- One way to avoid a problem of cluster-heterogeneity, like in CASPER, is to exploit the features provided by the general scheduler (see Figure 1). All popular schedulers for HPC systems, in fact, provide the ability to create different execution queues including homogeneous subsets of nodes, therefore creating many “smaller”, more efficient Hadoop clusters.

4. Conclusions and Future Developments

Big Data is an emerging issue and the need to rethink at High Performance Computing to be able also to manage tasks with massive Big Data is not just a research topic, but a real need. At Politecnico di Torino we have taken this new issue both as a challenge and as an opportunity and have started a project of identifying the requirements for a balanced upgrading of our HPC towards Big Data. In this paper we have shared our experience gained while designing and deploying this growth. In particular, we have briefly outlined the steps that we have taken to make our HPC system able to run MapReduce tasks using Hadoop, the results obtained, and shared a set of possibly useful general recommendations.

The plans for the close future of our HPC center include the expansion of CASPER up to 296 CPUs, 1.3 TB of memory and 60 TB of central storage by September 2013. Following the recipes in this paper, we have adopted some measures:

- The system will be kept as homogeneous as possible by adding computational nodes similar to the current ones.
- Each new node will have 2 additional disks for the exclusive use of Hadoop.
- The master node will be connected with a 10 Gigabit Ethernet link.

Finally, although the primary goal is to have a stable and reasonably efficient system in every field of HPC, it is our intention to follow the current trend of Big Data: not only "large data-bases", but also new algorithms, computational models and applications, starting from NoSQL DBMS like Apache HBase. This will bring the computing center to face new stimulating challenges such as: strengthening the Ethernet network with multiple switches, optimizing the sharing of local resources between HBase and SGE, redounding the master node to make the whole system more fault tolerant.

References

[Nepote et al. 2013] N. Nepote, E. Piccolo, C. G. Demartini, P. Montuschi, “Why and How Using HPC in University Teaching? A Case Study at PoliTo”. Proc. of the DIDAMATICA 2013 conference, Pisa, Italy, 2013.

[Garza et al. 2013] P. Garza, P. Margara, N. Nepote, L. Grimaudo, E. Piccolo, **“**Hadoop on a Low-Budget General Purpose HPC Cluster in Academia”. Proc. of the ADBIS 2013 Conference - Special Session on Big Data (in press), Geneva, Italy, 2013.

[Kindratenko & Trancoso, 2011] V. Kindratenko, P. Trancoso, “Trends in High-Performance Computing”. In: IEEE Computing in Science & Engineering, vol. 13, n. 3, pp. 92-95, Sept. 2011.

[Della Croce et al. 2011] F. Della Croce, N. Nepote, E. Piccolo, “A Terascale Cost-Effective Open Solution for Academic Computing: Early Experience of the DAUIN HPC Initiative”. Proc. of the 49th AICA national conference on smart tech and smart innovation, Turin, Italy, 2011.

[Dean & Ghemawat, 2008] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters”. In: Commun. ACM, vol. 51, n. 1, pp. 107–113, 2008.