Analysis and Evaluation of MapReduce Solutions on an HPC Cluster

Jorge Veiga*, Roberto R. Expósito, Guillermo L. Taboada, Juan Touriño

Computer Architecture Group, University of A Coruña
Campus de Elviña, s/n, 15071 A Coruña, Spain

Abstract

The ever growing needs of Big Data applications are demanding challenging capabilities which cannot be handled easily by traditional systems, and thus more and more organizations are adopting High Performance Computing (HPC) to improve scalability and efficiency. Moreover, Big Data frameworks like Hadoop need to be adapted to leverage the available resources in HPC environments. This situation has caused the emergence of several HPC-oriented MapReduce frameworks, which benefit from different technologies traditionally oriented to supercomputing, such as high-performance interconnects or the message-passing interface. This work aims to establish a taxonomy of these frameworks together with a thorough evaluation, which has been carried out in terms of performance and energy efficiency metrics. Furthermore, the adaptability to emerging disks technologies, such as solid state drives, has been assessed. The results have shown that new frameworks like DataMPI can outperform Hadoop, although using IP over InfiniBand also provides significant benefits without code modifications.

Keywords: MapReduce, High Performance Computing (HPC), Big Data, Energy Efficiency, InfiniBand, Solid State Drive (SSD)

1. Introduction

In the past several years, organizations have been adopting the Big Data paradigm to obtain valuable information from large volumes of data. The required capabilities to do so is increasing over and over, since the amount of data managed by organizations grows every year. This situation demands more efficient, scalable and capable systems, while maintaining some of the axioms that unify the way that data are managed and processed in most environments.

In order to overcome the limitations of current systems, some organizations have put an eye on High Performance Computing (HPC), which could provide with the technology to achieve Big Data goals. At the same time, HPC environments could benefit from Big Data programming models, such as MapReduce [1]. MapReduce is a programming model and an associated implementation for generating and processing large data sets, which is widely used to solve many analytic problems in several application fields, from biology to finances. The Apache Hadoop project [2] has

*Corresponding author: Tel: +34 981 167 000 ext. 1212, Fax: +34 981 167 160
Email addresses: jorge.veiga@udc.es (Jorge Veiga), rreye@udc.es (Roberto R. Expósito), taboada@udc.es (Guillermo L. Taboada), juan@udc.es (Juan Touriño)

gained significant attention in the last years as a popular open-source Java-based implementation of the MapReduce paradigm derived from the Google's proprietary implementation.

The use of Hadoop in HPC systems is expected to increase its performance, which is mainly limited by disk and network bandwidths. In this respect, network bandwidth can be improved by using high-performance interconnects (e.g. InfiniBand [3]), which are usually available in HPC environments. Hadoop relies on the ubiquitous TCP/IP protocol to implement its communications support through Java sockets, and thus it is not able to leverage high-performance interconnects in an optimal way. This issue has caused the appearance of many HPC-oriented MapReduce frameworks like Mellanox UDA [4], RDMA-Hadoop [5], DataMPI [6] and others. These solutions attempt to overcome the limitations of standard Hadoop in HPC environments.

This paper presents an in-depth analysis and evaluation of representative HPC-oriented MapReduce solutions, indicating their similarities and differences. The proposed taxonomy is intended to be used to determine the suitability of each solution in specific use cases. The evaluation of these frameworks has been performed using several representative benchmarks. Furthermore, the trend to increase the number of cores in current systems has turned their power consumption into one of the most relevant issues. Therefore, the evaluation of the frameworks has been assessed not only in terms of performance but also taking into account their energy efficiency. The obtained results can be useful to identify the strengths and weaknesses of each solution in order to get valuable characteristics for future implementations.

The rest of this paper is organized as follows: Section 2 presents background information and related work. Section 3 describes the experiments performed with the different frameworks and analyses the results in detail. Finally, Section 4 extracts the main conclusions of the paper and proposes ongoing work.

2. Background and Related Work

This section addresses the MapReduce model and its de-facto standard implementation, Hadoop. It also includes some details about InfiniBand, which has become the most widely adopted interconnect in the TOP500 list [7]. Next, it classifies the different HPC-oriented MapReduce solutions. Finally, related evaluation works are discussed.

2.1. MapReduce and Hadoop

The MapReduce paradigm performs the data processing in two main phases: Map and Reduce, which name the model. In the Map phase, the cluster nodes read the data from their own local filesystem and extract the significant features of each data element. Every feature is represented by a $\langle key, value \rangle$ pair in which every *value* of the significant feature is associated with a *key*. The *key* is used to index the *value* and add it to a group. Elements with the same key belong to the same group. The Reduce phase receives the $\langle key, value \rangle$ pairs after being grouped and sorted. Every *value* is operated with the rest of the elements of its group, leading to a final result. This result is also represented by a $\langle key, value \rangle$ pair, and it can be seen as the valuable information of the group identified by the *key*.

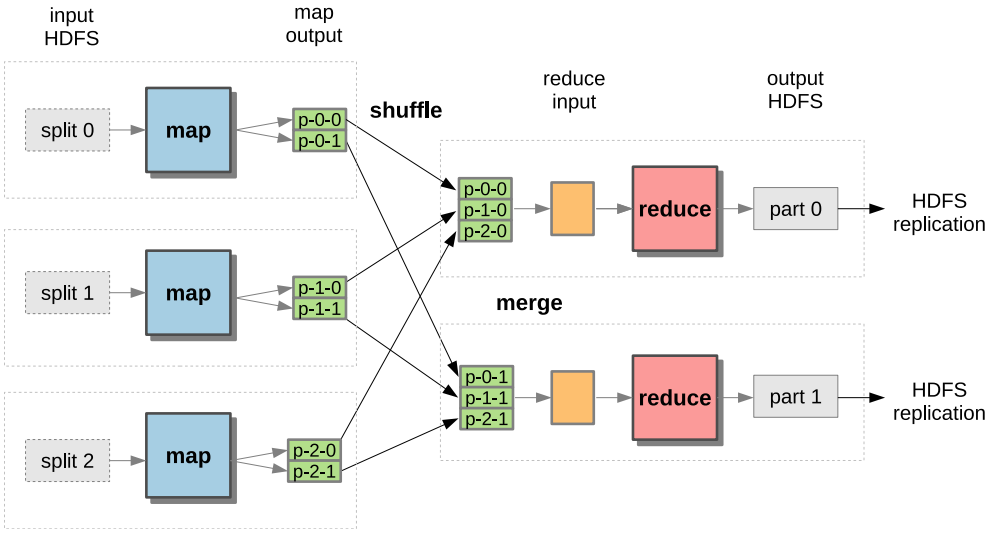


Figure 1: Hadoop data flow with multiple map and reduce tasks

Currently, the most popular MapReduce implementation is Apache Hadoop [2], which is widely used by large corporations like Yahoo!, Facebook, Twitter and many others. Hadoop is an open-source Java-based framework which enables to store and process Big Data workloads. It mainly consists of two components: (1) the Hadoop Distributed File System (HDFS), which distributes the storage of data among the nodes of a cluster; and (2) the Hadoop MapReduce engine, which allocates data processing to the node where it resides.

Figure 1 shows an overview of the overall MapReduce process performed by Hadoop. At the beginning of the process, the input data set is divided into splits and processed by the map tasks. Each time a map task finishes, the map output is partitioned into different fragments (*p-x-x* in the figure) which are sent to their corresponding reducers and merged to form the reduce input. Each reduce task processes its input to generate the final results and write them to HDFS.

The first version of Hadoop (Hadoop 1) suffered from certain problems, like lack of scalability in some components, which caused the appearance of Yet Another Resource Negotiator (YARN) [8] in Hadoop 2. In YARN, the resource management has been decoupled from the MapReduce engine. Due to this division, several kinds of applications other than MapReduce ones can run over the Hadoop cluster, such as Message-Passing Interface (MPI) [9] codes. Moreover, most of the previous Hadoop applications can run on YARN with no changes.

2.2. InfiniBand

InfiniBand [3] is a networking technology which is widely used by scientific computing centers world-wide. In fact, 51.8% of the supercomputers in the June 2015 TOP500 list [7] use this technology, being the most used interconnect family. Its popularity is caused by the high-performance features that InfiniBand provides, such as Remote Direct Memory Access (RDMA). RDMA allows to directly access the memory of the remote nodes without involving CPU usage at the remote side. This feature enables to implement high-performance communication protocols using

zero-copy and kernel-bypass mechanisms. The InfiniBand specification declares the different tasks of software and hardware elements. To interact between them, the network elements make use of the Verbs layer, a low-level interface for RDMA communication. This communication is performed completely in user space, avoiding the use of kernel space to buffer the data as it occurs with the TCP/IP protocol. The reference implementation of the Verbs layer is provided by the OpenFabrics Alliance (OFA) [10].

InfiniBand can also be used as an IP network. The IP layer exposes the InfiniBand device to the system as an available Ethernet card, which is commonly known as IP over InfiniBand (IPoIB) [11]. The communication with IPoIB is not kernel-bypassed, so the data must be buffered at the kernel space before sending it through the network, preventing the implementation of zero-copy protocols.

2.3. HPC-oriented MapReduce Solutions

The taxonomy proposed in this paper classifies the HPC-oriented MapReduce solutions by the way each one addresses the problem of leveraging high-performance interconnects. The degree of modification of Hadoop can be useful to identify the expected performance improvement of a certain solution, as a deeper modification would potentially involve a better adaptation to HPC systems. The main drawback of replacing Hadoop with a new framework is that applications may need to be rewritten.

Some solutions are transparent or semi-transparent to the user, and do not require an overall modification of the Hadoop core implementation. Other proposals modify specific parts of Hadoop which are responsible for performing the communication phase. Finally, there are some new frameworks that implement the MapReduce model from scratch using a different design and/or approach. More details of this taxonomy are given next.

2.3.1. Transparent Solutions

The solutions which do not require changes in the Hadoop implementation (or these changes are minimal) are Hadoop with IPoIB and Mellanox UDA. Generally, only changes in the Hadoop configuration are needed.

Hadoop with IPoIB.

To enable the utilization of IPoIB it is only necessary to change the Hadoop network configuration, so that the IPoIB interface is used instead of the Ethernet one. Once the configuration is modified, Hadoop will use the new IP addresses without changing its behaviour regarding the interconnection network. Therefore, this solution is the one which minimizes the changes required to use InfiniBand, and it can be applied to any version of Hadoop.

Mellanox UDA.

Mellanox UDA [4] is basically a plug-in intended for improving the communications between mappers and reducers. It has been developed by the Parallel Architecture and System Laboratory of the Auburn University together with Mellanox. Currently, it supports both Hadoop 1 and 2 versions.

UDA combines an RDMA-based communication protocol along with an efficient merge-sort algorithm, which is based on the network levitated merge algorithm [12]. This algorithm modifies the original overlapping of Hadoop phases, involving a single disk access to write the reduce input data. Additionally, it directly implements its RDMA protocol on top of the Verbs layer to natively support InfiniBand.

2.3.2. Modifications over Hadoop

Currently, the most popular solutions that modify some of the subsystems of Hadoop to take advantage of high-performance interconnects are RDMA-Hadoop and HOMR, described next.

RDMA-Hadoop.

RDMA-Hadoop adapts Hadoop communications to take advantage of RDMA features. This framework has been extended to the High Performance Big Data (HiBD) project [5], which encloses RDMA-based implementations of Hadoop 1, Hadoop 2 (YARN) and Memcached [13], as well as a set of benchmarks for evaluating Big Data middleware.

Previous works have revealed the benefits of using high-performance interconnects to increase the performance of HDFS [14]. This has led developers to adapt different components of Hadoop to use RDMA communications: HDFS [15], MapReduce [16] and Hadoop RPC [17]. The specific solution for MapReduce (i.e. RDMA-Hadoop) redesigns the communications between mappers and reducers to take full advantage of the RDMA interconnect. It also improves the transmission of the map output by introducing a pipeline between mappers and reducers, while also performing data prefetching and caching mechanisms. As in the case of Mellanox UDA, RDMA-Hadoop modifies the overlapping of Hadoop phases for faster data processing.

HOMR.

HOMR [18] is a hybrid approach to exploit maximum overlapping in MapReduce over high-performance interconnects. Using both standard Hadoop and RDMA-Hadoop implementations, it can communicate by using sockets and RDMA. The main aim of HOMR is to obtain the maximum overlapping among the different MapReduce phases by leveraging the behaviour characteristics of both frameworks, adjusting the used implementation to a specific moment of the MapReduce process. This mechanism improves the use of the interconnection network along the whole process and reduces disk accesses in the reduce phase. In order to enhance the transmission between mappers and reducers, HOMR also determines which map outputs should be sent sooner than others. HOMR is a relatively recent project that is not publicly available yet, and thus its evaluation has not been included in this work.

2.3.3. Implementations from Scratch

Currently, the most representative framework developed from scratch is DataMPI, described next.

DataMPI.

The feasibility of using the MPI model for MapReduce-based computing has been studied since 2009 [19], which has led to several projects such as DataMPI [6, 20], BDMPI [21] and MR-MPI [22]. However, DataMPI has moved

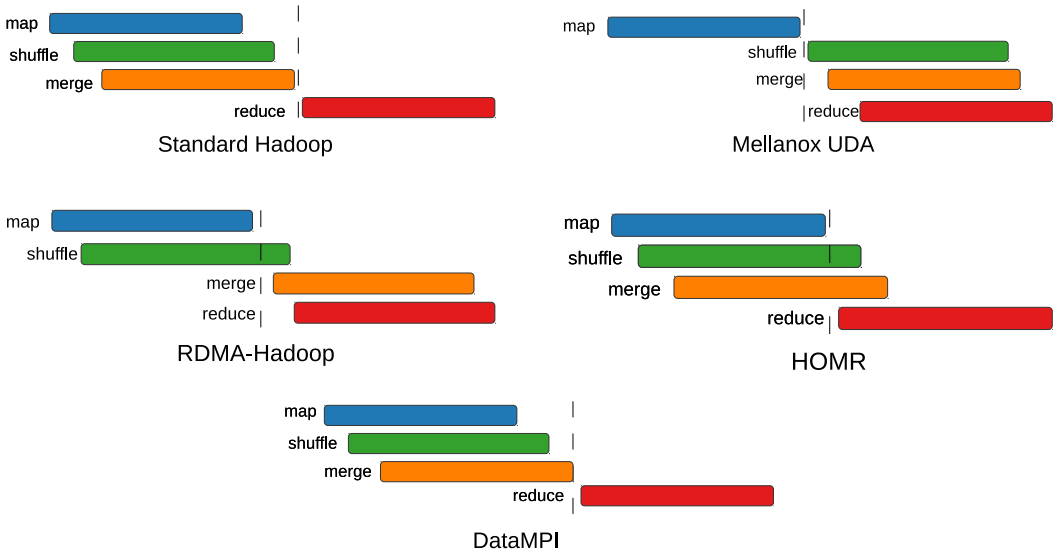


Figure 2: Overlapping of the MapReduce phases in each solution

forward by providing simple implementation examples and a full benchmarking suite, BigDataBench [23], which also includes benchmarks for Hadoop, Spark [24] and MPI [25].

DataMPI implements the MPI-D specification [26], a proposed extension of the MPI standard which uses the $\langle key, value \rangle$ semantics taken from the MapReduce model. Based on Hadoop, the data processing architecture of DataMPI includes additional features like reduce-side data locality, improved shuffle pipeline and optimized buffer management. It considers four processing modes: Common, MapReduce, Streaming and Iteration. The proposal of DataMPI aims to cover a wide range of applications which use different communication models by unifying them in a single architecture. The current version implements the functionalities and features of the Common mode, which is intended to support Single Program Multiple Data (SPMD) applications. Although the rest of the processing modes are yet to be implemented, some representative MapReduce workloads are already available, such as Wordcount, Sort, TeraSort and Grep.

2.3.4. MapReduce Phases Overlapping

One of the key factors in the MapReduce model is the overlapping of the different phases involved in the overall data processing. Each solution schedules these phases in a specific way to obtain a certain benefit, such as the reduction in disk accesses by using a network levitated merge (Mellanox UDA) or an in-memory merge algorithm (RDMA-Hadoop), maximum overlapping of the phases (HOMR), and an increase in data locality during the reduce phase (DataMPI). Figure 2 shows at a glance how each of these frameworks addresses this issue.

2.3.5. Summary

Table 1 shows a brief summary of the MapReduce frameworks that have been discussed in this section, including some of their differential characteristics. The first and second columns indicate whether the data sets can be

Table 1: Characteristics of the MapReduce solutions

	HDFS	YARN	Available	Transparent	Application compatible	RDMA
Hadoop IPoIB	✓	✓	✓	✓	✓	
Mellanox UDA	✓	✓	✓	✓[a]	✓	✓
RDMA-Hadoop	✓	✓[b]	✓		✓	✓
HOMR	✓				✓	✓
DataMPI	✓[c]		✓			✓[d]

[a] Some versions of Mellanox UDA require to modify Hadoop in a minimal way to enable the use of the shuffle library
[b] The HiBD project includes adapted versions of Hadoop 1 and Hadoop 2 (YARN)
[c] DataMPI can work on data stored in HDFS. It currently supports Hadoop 1 but not Hadoop 2 communication schemes
[d] The RDMA support depends on the underlying MPI implementation

read/written from/to HDFS and the compatibility with Hadoop 2 (YARN), respectively. The table also shows if the framework is publicly available, whether the solution is transparent or it replaces the Hadoop implementation, if it is compatible with standard Hadoop applications or they must be modified, and if it supports RDMA communications.

2.4. Related Work

The different proposals previously discussed usually include experimental evaluations that show the achieved performance improvement. Generally, developers tend to compare their frameworks with standard Hadoop over Gigabit Ethernet (GbE), Hadoop over IPoIB and previous proposals (if any). That is the case of Mellanox UDA [12], RDMA-Hadoop [16], HOMR [18] and DataMPI [20]. The DataMPI team has also evaluated comparatively their framework with Hadoop and Spark in terms of several performance and efficiency metrics.

Previous works have assessed different MapReduce implementations, usually comparing Hadoop with other frameworks like Twister [27] and LEMO-MR [28]. Other works focused on evaluating the performance of Hadoop when executing data-intensive scientific applications, identifying some trade-offs due to design decisions in Hadoop components [29]. In fact, the benchmarking of Hadoop has caused the emergence of multiple benchmarking and application suites other than BigDataBench, such as HiBench [30] and MRBS [31].

As energy efficiency has been growing as a big concern, several studies have discussed how to measure it in Hadoop clusters. In [32], the ED^2P metric is adapted to determine the energy efficiency of a specific Hadoop cluster. This metric, proposed in [33], was originally used in the context of pipeline applications. Other studies on the energy efficiency of Hadoop that perform extensive evaluations using several workloads can be found in [34] and [35]. However, these works do not compare their results with those of other MapReduce frameworks.

Our previous work [36] analysed the main issues of evaluating MapReduce frameworks in HPC systems. One of its outcomes, the MapReduce Evaluator (MREv) tool, has been used in this work for easing the configuration of the solutions and collecting the results.

To the best of our knowledge, this work presents the first independent evaluation of HPC-oriented MapReduce solutions, covering the most representative frameworks and considering both performance and energy efficiency metrics.

3. Performance Evaluation

This section presents the main results of the evaluation performed in this work using the most representative HPC-oriented MapReduce solutions introduced in Section 2.3. First, Section 3.1 describes the evaluation methodology and Section 3.2 the experimental configuration. Next, evaluation results are shown in terms of performance and energy efficiency (Section 3.3), power consumption over time (Section 3.4) and power variability between nodes (Section 3.5).

3.1. Evaluation Methodology

In order to evaluate the different frameworks, several experiments have been conducted using representative workloads. These are divided into two groups, micro-benchmarks and application benchmarks. On the one hand, micro-benchmarks are useful to carry out specific performance measurements on a cluster, as they perform simple workloads that focus on the utilization of a certain resource, such as CPU or network. On the other hand, application benchmarks based on more complex workloads can better approximate the overall performance of a solution in a real environment. The selected benchmarks are described in Table 2.

Micro-benchmarks.

TestDFSIO, Wordcount and TeraSort are classic MapReduce workloads that enable to evaluate the performance in terms of distinct resources. The aim of TestDFSIO is to measure the throughput of HDFS, and so it is mainly I/O bound with heavy disk I/O. Although this work is focused on benchmarking MapReduce workloads, TestDFSIO results are useful to extract information about the performance of basic HDFS operations (read/write) that can be taken into account when analysing the rest of benchmarks. Wordcount is mainly CPU bound due to its low communication overhead: it extracts a very small amount of information in the map phase (the number of times each word appears) to be sent to the reducers. TeraSort, which sends the entire data set from mappers to reducers when compression is not used, is both communication and I/O bound with heavy network traffic and disk I/O. The implementation of these workloads can be found in any distribution of Hadoop. In the case of DataMPI, equivalent implementations of Wordcount and TeraSort are provided with its bundle distribution. However, DataMPI currently does not implement TestDFSIO, so this benchmark has only been evaluated with Hadoop-based solutions.

These benchmarks have been selected to ensure the reproducibility of the experiments, as they are provided with the frameworks. For the same reason, the input data sets have been generated using standard tools. In the case of Wordcount and TeraSort, these tools are RandomTextWriter and TeraGen, respectively. Both of them generate a set of random text which is uniformly distributed among the map tasks to be processed. In the case of TestDFSIO, the benchmark consists of two phases, the *write* phase, in which the data set is generated and written to HDFS to measure

Table 2: Benchmarks used in the experimental evaluation

Micro-benchmarks	
TestDFSIO	Tests the throughput of HDFS by generating a large number of tasks performing reads and writes simultaneously
Wordcount	Counts the number of times each word appears in the input text data set, which is set up using the RandomTextWriter data generator
TeraSort	Sorts 100B-sized $\langle key, value \rangle$ tuples. Each key is 10B-sized and each value is 90B-sized. The input data set is set up using the TeraGen data generator
Application benchmarks	
PageRank	Ranks websites by counting the number and quality of the links to each one. Developed by Google, it is used to obtain Google search results

Table 3: TestDFSIO data distribution

Slaves	Data per slave	Data per map
2	84 files (5040 MB)	12 files (720 MB)
4	42 files (2520 MB)	6 files (360 MB)
8	21 files (1260 MB)	3 files (180 MB)
12	14 files (840 MB)	2 files (120 MB)

the observed throughput, and the *read* phase, which reads the generated data set from HDFS. The size of the input data set for Wordcount and TeraSort has been set to 50 GB, as it was the largest volume of data with which RDMA-Hadoop has not shown memory issues in the shuffle phase. With larger data sets, the MapReduce job failed because of out of memory errors in some of the reduce tasks. TestDFSIO has also been affected by this problem, and thus the input data size for this benchmark has been set to 10 GB using 168 files of 60 MB, following a regular task distribution pattern as shown in Table 3.

To test the scalability of each solution, micro-benchmarks have been executed using 3, 5, 9 and 13 nodes. Each cluster of n nodes can be understood as (1 master, $n-1$ slaves). The graphs presented for the Wordcount and TeraSort benchmarks show the average values from a minimum of 10 measurements for each experiment using a logarithmic scale. In the case of TestDFSIO, the scale of the graph is linear and also shows maximum and minimum values.

Application benchmarks.

The PageRank algorithm represents a workload with higher complexity than the micro-benchmarks. This benchmark enables to analyse each solution using a workload much more similar to a real-world application. The implementation of PageRank has been obtained from Pegasus 2.0 [37], a graph mining system built on top of Hadoop. This benchmark is not included in the DataMPI distribution yet, so it has only been evaluated with Hadoop-based solutions.

The input data set of PageRank is generated by a Kronecker graph generator included in the BigDataBench

suite [23]. In these experiments, the input graph has around 17,000 vertices (1.5 GB) and takes 24 iterations to be computed. The goal of this evaluation is to obtain overall performance measurements for each framework using the maximum number of available nodes. Hence, PageRank has been executed using 9 and 13 nodes when using Solid State Drives (SSD) and magnetic disks, respectively[1]. The graphs show the average, maximum and minimum values from a minimum of 10 measurements for each experiment using a logarithmic scale.

3.1.1. Energy Efficiency Metrics

Nowadays, energy efficiency is one of the most important factors to be considered, as the growth of current systems is causing their power consumption to reach challenging values. This affects not only hardware providers but also software developers, who have to take into account power consumption when designing and evaluating new software and applications.

Consequently, power measurements have also been obtained along all the experiments by using the HP Integrated Lights-Out (iLO) technology [38] provided by the cluster[2]. This tool has allowed to estimate energy-performance ratios in order to study the power consumption over time and analyse the power variability among the cluster nodes.

The metric used to assess the energy-performance ratio is ED^2P, mentioned in Section 2.4. ED^2P was first proposed in [33] to relieve the problems of previous efficiency metrics, such as EDP. Later, it was also recommended for measuring the energy-performance efficiency of variable voltage/frequency processors [39]. Although it was originally used in the context of pipeline applications, it can also be applied to our evaluation as done in [32], since a MapReduce computation is similar to a pipeline application consisting of map and reduce stages running on a cluster of power-aware nodes. The ED^2P metric uses the elapsed time (ET) and the energy consumed (EC) by an application, calculating the efficiency ratio as shown in Equation 1. The lower the value of the metric, the higher the energy efficiency of the application.

$$ED^2P = EC \times ET^2 \tag{1}$$

The first set of ED^2P graphs show the metric values for the Wordcount, TeraSort and PageRank benchmarks when using each cluster size, in a similar way to the time graphs. Additionally, a second set of graphs present the correlation between time and the ED^2P metric. Both kinds of graphs use a logarithmic scale.

3.2. Experimental Configuration

Table 4 shows the frameworks evaluated in the experiments and their respective versions. The chosen version of standard Hadoop is 2.7.1. Experiments with standard Hadoop have been conducted using Gigabit Ethernet (GbE) and IPoIB communications. Moreover, Hadoop 2.7.1 has also been configured together with Mellanox UDA version 3.3.2,

[1] In addition to the 13-node case, PageRank has been executed using 9 nodes with magnetic disks to enable the comparison with SSD disks
[2] HP iLO consists of a Baseboard Management Controller (BMC) accessible through a REST interface

Table 4: Evaluated frameworks

Name	Version	Release Date	Interconnect
Hadoop-2-GbE	2.7.1	06/07/2015	Gigabit Ethernet
Hadoop-2-IPoIB	2.7.1	06/07/2015	InfiniBand (IPoIB)
Hadoop-2-UDA	3.3.2	24/11/2013	InfiniBand (RDMA & IPoIB)[a]
RDMA-Hadoop-2	0.9.7	26/05/2015	InfiniBand (RDMA & IPoIB)
DataMPI	0.6.0[b]	16/04/2014	InfiniBand (RDMA & IPoIB)

[a] Mellanox UDA uses RDMA communications in the shuffle phase and IPoIB for HDFS
[b] DataMPI uses HDFS (Hadoop 1.2.1) as distributed filesystem configured with IPoIB

Table 5: Configuration of the cluster

Hardware configuration	
Cluster size	13 nodes
CPU	2 × Intel Xeon E5-2660 Sandy Bridge-EP
CPU Speed/Turbo	2.20 GHz/3 GHz
#Cores per node	16
RAM Memory	64 GB DDR3 1600 Mhz
Disk technologies	1 × HDD 1TB & 1 × SSD 480 GB (8 nodes)
Networks	InfiniBand FDR & Gigabit Ethernet
Software configuration	
OS version	CentOS release 6.4
Kernel	2.6.32-358.6.2.el6.x86_64
Java version	Oracle JDK 1.8.0_20
MPI implementation	MVAPICH2 1.9 (for DataMPI)
Pegasus version	2.0 (for PageRank)

which enables RDMA communications in MapReduce and IPoIB as the underlying network protocol for HDFS. Even though newer versions are available, Mellanox UDA version 3.3.2 has shown the best performance and stability. In the case of RDMA-Hadoop-2 version 0.9.7, it adapts Hadoop 2.6.0 to use RDMA communications on both MapReduce and HDFS components, but it has also been configured with IPoIB for ensuring the best possible configuration. Finally, DataMPI uses HDFS as distributed filesystem, also configured with IPoIB. Additional experiments have been conducted to test the performance of DataMPI using HDFS over GbE. As these results were worse than those obtained using IPoIB, they have been omitted for clarity purposes.

The experiments have been carried out in Pluton, a 13-node multi-core cluster interconnected by both GbE and InfiniBand networks. Each node is equipped with a single magnetic disk of 1 TB. Furthermore, 8 nodes also have a single SSD disk of 480 GB. Table 5 shows more details about the hardware and software configuration of Pluton. The evaluated solutions have been configured to leverage the maximum capabilities of Pluton by following their

Table 6: Framework common configuration

Mappers per node	7
Reducers per node	7
Java heap size	3 GB
HDFS block size	128 MB
Replication factor	3

corresponding user guides and adjusting the main parameters to the specific characteristics of the hardware (e.g. SSD disks). The common configuration settings that have been applied to all the solutions are briefly described in Table 6, including the number of Map/Reduce tasks per node and the Java heap size of each task.

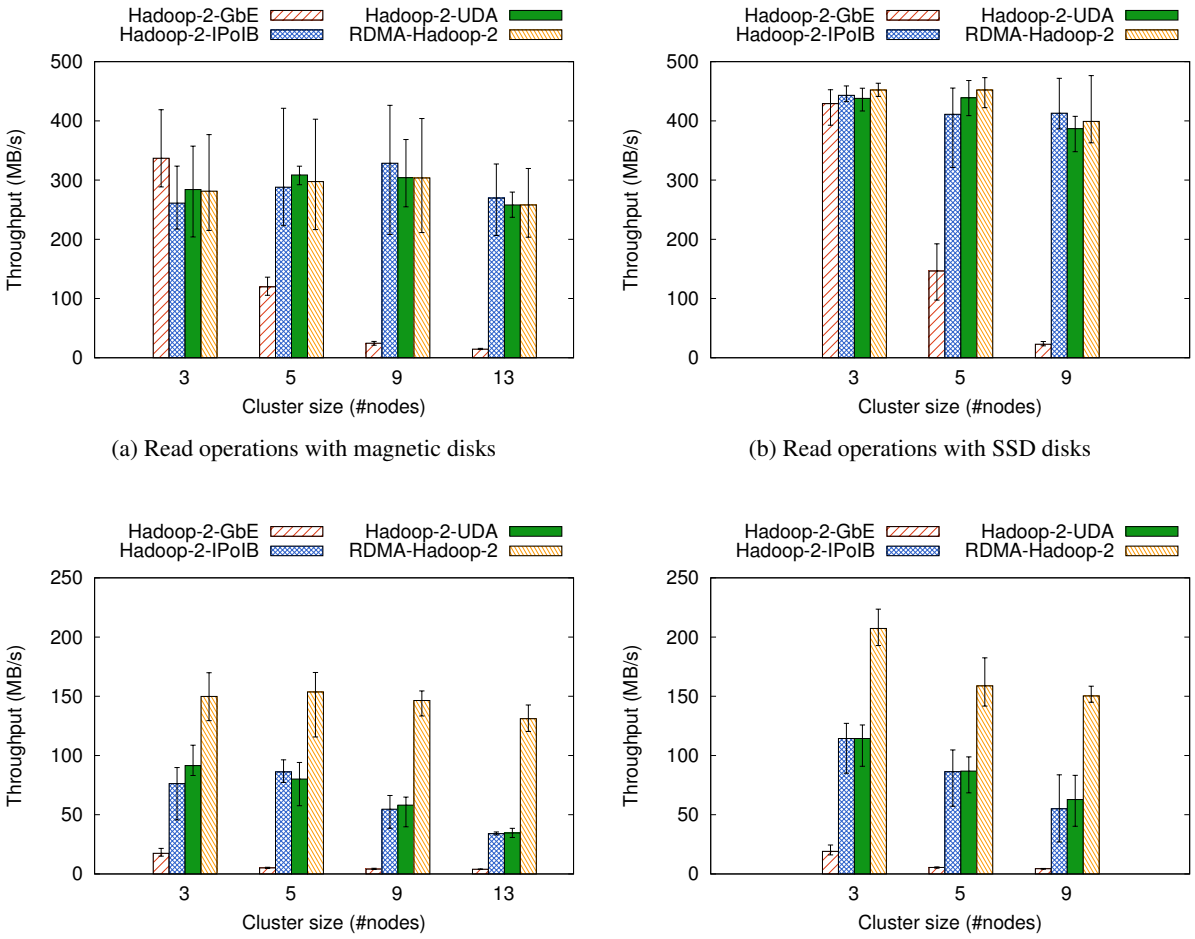
As mentioned in Section 2.4, the execution of the experiments has been eased by using the MREv tool [36], which is available to download at `http://mrev.des.udc.es`. MREv automates the configuration of the frameworks, the generation of the input data sets, the execution of the experiments, the collection of the results and the generation of the graphs for each benchmark.

3.3. Performance and Energy Efficiency Results

This section presents the results in terms of performance and energy efficiency. This includes throughput values for the TestDFSIO benchmark (Section 3.3.1) and execution times for Wordcount (Section 3.3.2), TeraSort (Section 3.3.3) and PageRank (Section 3.3.4).

3.3.1. TestDFSIO

Figure 3 shows TestDFSIO throughputs for read and write operations using magnetic (left graphs) and SSD disks (right graphs). Standard Hadoop is strongly limited by the network bandwidth, as the throughput is much lower when using GbE as interconnect. For read operations (see Figures 3a and 3b), it can be seen that the 3-node case does not show much difference between Hadoop-2-GbE and Hadoop-2-IPoIB. This is caused by the default replication factor of HDFS, which is 3 (see Table 6). In this case, the 2 slave nodes (i.e. DataNodes) have every data block locally available for reading, without using the network. As the number of nodes in the cluster increases, the difference between Hadoop-2-GbE and Hadoop-2-IPoIB becomes significantly wider. Moreover, write results (Figures 3c and 3d) show a great difference between Hadoop-2-GbE and Hadoop-2-IPoIB with all cluster sizes, since the blocks must be replicated over the network even when using 3 nodes. These results confirm that IPoIB significantly enhances the performance and scalability of HDFS operations. The throughput values for Hadoop-2-UDA are close to those of Hadoop-2-IPoIB, as it also uses IPoIB as the underlying network protocol for HDFS. That is also the case of the read throughput of RDMA-Hadoop-2, because it only modifies the behaviour of HDFS for write operations [15]. In fact, the write throughput of RDMA-Hadoop-2 is clearly the highest among the evaluated frameworks.



(a) Read operations with magnetic disks

(b) Read operations with SSD disks

(c) Write operations with magnetic disks

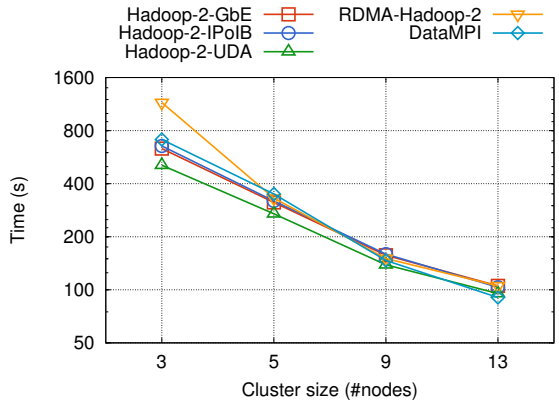
(d) Write operations with SSD disks

Figure 3: Throughput results for the TestDFSIO benchmark (higher is better)

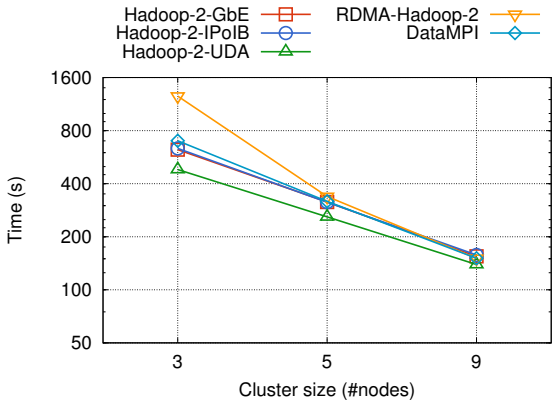
Comparing the throughput of read/write operations when using magnetic and SSD disks, it can be observed that the SSD technology improves the results for all solutions and cluster sizes in general, although the network bandwidth limits this improvement when using large cluster sizes. The comparison among the different solutions remains roughly the same for magnetic and SSD disks.

3.3.2. Wordcount

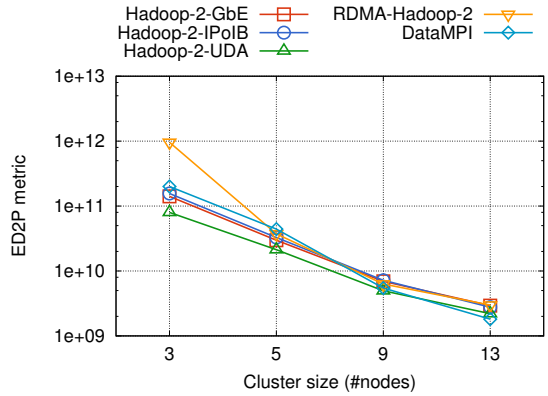
Figures 4a and 4b show time results for the Wordcount benchmark using magnetic and SSD disks, respectively. Wordcount is mostly CPU bound, involving light disk/network activity. As a result, the use of SSD disks barely improves the results of magnetic disks. As expected, there are not significant differences between Hadoop-2-GbE and Hadoop-2-IPoIB. Regarding Hadoop-2-UDA, it shows a constant improvement of the execution time with all cluster sizes and both disk technologies. As the network and disk load of this benchmark is light, this improvement may be due to the different overlapping of the MapReduce phases. RDMA-Hadoop-2 shows the same performance of



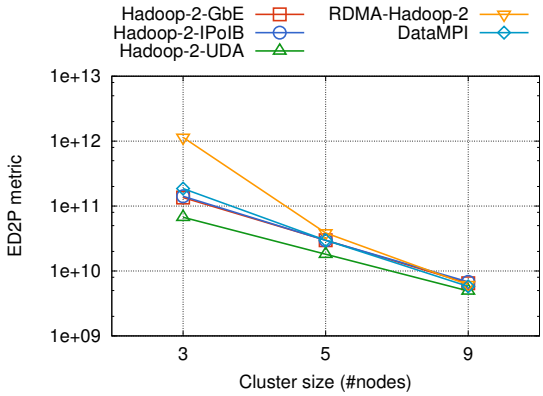
(a) Time with magnetic disks



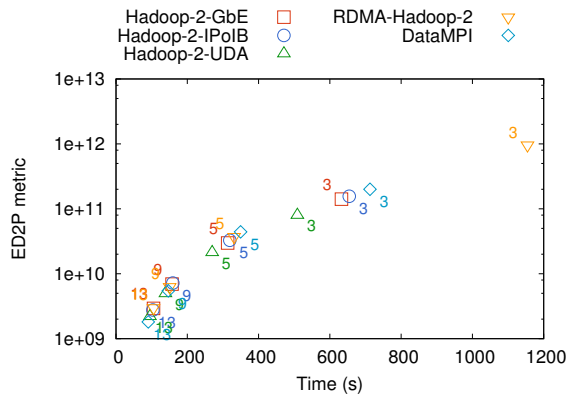
(b) Time with SSD disks



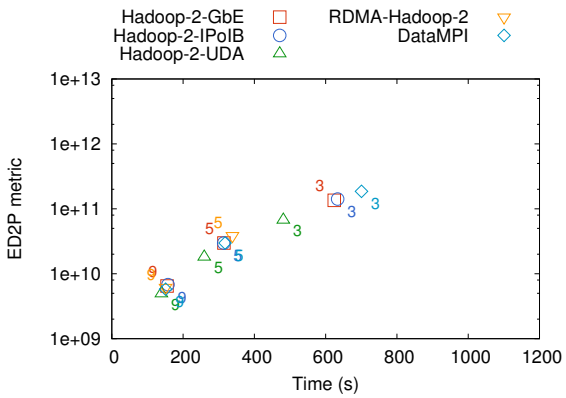
(c) ED^2P metric with magnetic disks



(d) ED^2P metric with SSD disks



(e) ED^2P metric vs time with magnetic disks



(f) ED^2P metric vs time with SSD disks

Figure 4: Time and ED^2P results for the Wordcount benchmark (lower is better)

standard Hadoop except when using 3 nodes, in which case its performance is lower. Finally, DataMPI obtains nearly the same performance and scalability than standard Hadoop.

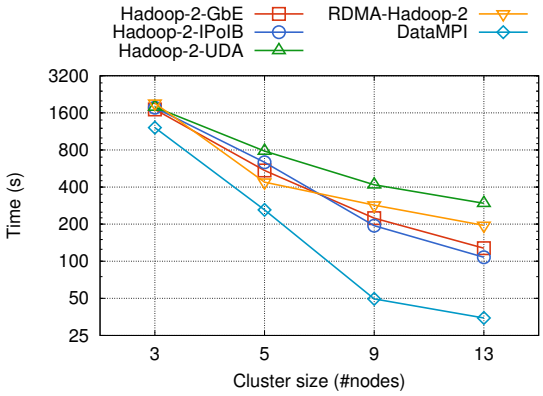
Figures 4c and 4d present ED^2P results for the Wordcount benchmark using magnetic and SSD disks, respectively. These values show a high dependence of the corresponding execution times, as shown in Figures 4e and 4f, which depict the correlation of the ED^2P metric and the execution time. This is because the difference in power consumption among the solutions is not significant enough to offset the differences in the time taken to perform the benchmark. Consequently, in most cases a higher execution time will involve a worse energy-performance ratio. This behaviour can also be observed when comparing results over different cluster sizes. The use of a higher number of nodes generally improves the energy-performance ratio since the reduction in the execution time is more relevant than the increase in power consumption produced by the additional nodes. Again, the time to perform the workload prevails over the difference in power consumption.

Figures 4e and 4f also show the differences between solutions when using each cluster size (3, 5, 9 or 13 nodes). For both disk technologies it can be seen that, as the number of nodes increases, the points become closer to each other. In Wordcount, the performance differences of the frameworks become less significant for large cluster sizes due to their good overall scalability.

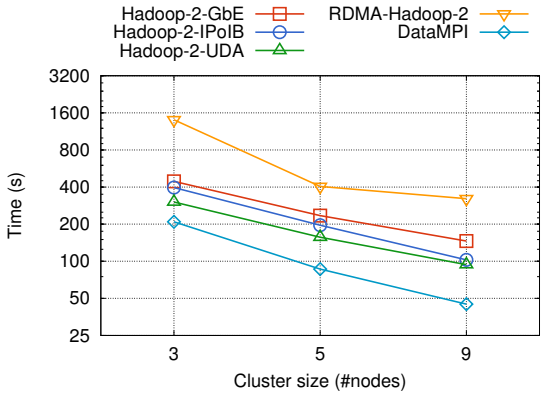
3.3.3. TeraSort

Figures 5a and 5b show execution times for the TeraSort benchmark using magnetic and SSD disks. The TeraSort benchmark has a higher disk/network activity than Wordcount and, as can be seen in the graphs, it is highly limited by the bandwidths of disk and network. Hence, the use of SSD disks (see right graph) reduces significantly the execution times, especially for small cluster sizes. The difference between Hadoop-2-GbE and Hadoop-2-IPoIB becomes slightly wider when using large cluster sizes, from 9 nodes on with magnetic disks and from 5 nodes on with SSD disks. Using magnetic disks, Hadoop-2-UDA shows lower performance than standard Hadoop and RDMA-Hadoop-2. However, Hadoop-2-UDA outperforms them with SSD disks, being the difference so high that it achieves nearly the same performance on 3 nodes with SSD disks than on 13 nodes with magnetic disks. This points out that some design decisions on Mellanox UDA, like the reduction in disk accesses, behave differently when applied to one or another disk technology. This issue can also be observed in RDMA-Hadoop-2, as the use of SSD disks does not improve its performance like in the rest of the frameworks. DataMPI is clearly the best performer using all cluster sizes and both disk configurations.

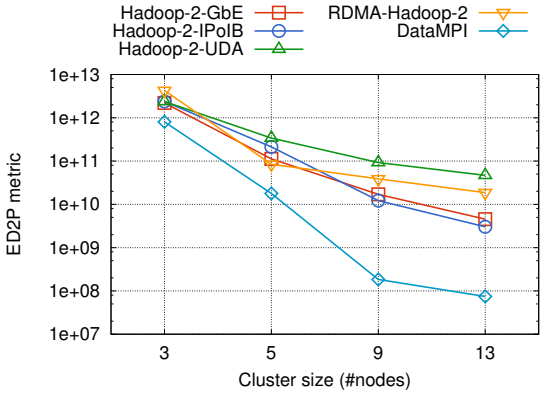
Figures 5c and 5d present ED^2P results for TeraSort. As mentioned for Wordcount, these values are strongly dependent on the execution times. However, there are some specific cases in which a reduction of the execution time does not imply a better energy-performance ratio. That is the case of RDMA-Hadoop-2 with SSD disks when comparing the use of 5 and 9 nodes. The power measurements with 9 nodes are high enough to cause nearly the same ED^2P value than with 5 nodes, even though the execution time is lower with 9 nodes. In this case, the user would have to prioritize either the performance or the energy efficiency of the system.



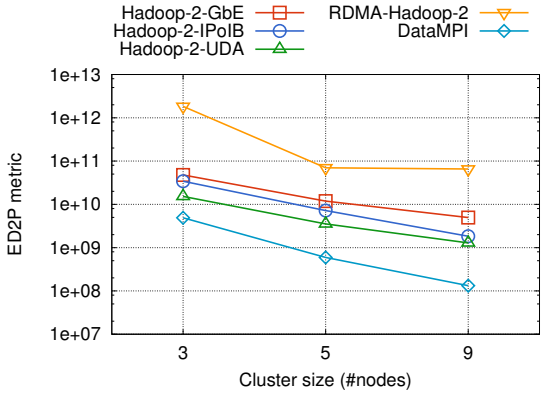
(a) Time with magnetic disks



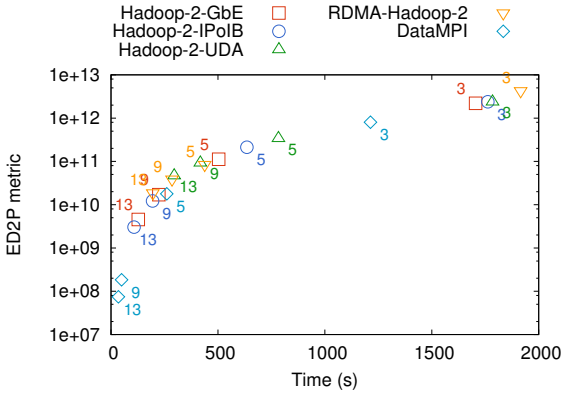
(b) Time with SSD disks



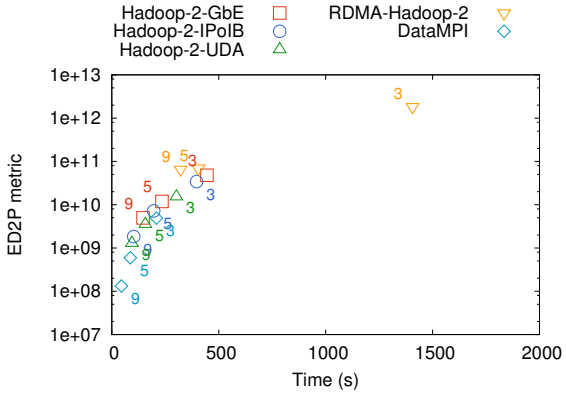
(c) ED^2P metric with magnetic disks



(d) ED^2P metric with SSD disks



(e) ED^2P metric vs time with magnetic disks



(f) ED^2P metric vs time with SSD disks

Figure 5: Time and ED^2P results for the TeraSort benchmark (lower is better)

The use of SSD disks generally improves the energy-performance ratio as a consequence of reducing significantly the average execution times (up to 78% with the 9-node cluster). Depending on the framework, changing the disk technology could improve the energy efficiency more than increasing the number of nodes. In fact, Hadoop-2-UDA results are far better when using SSD disks for all cluster sizes, while RMDA-Hadoop-2 gets better performance with 13 nodes and magnetic disks than with 9 nodes and SSD disks.

The correlation of the ED^2P metric and the execution time is shown in Figures 5e and 5f. The values of the metric keep showing the dependence of ED^2P over time, but now there are more cases of points with separated ED^2P values and close execution times. It seems that the differences in energy consumption over time become wider for this benchmark, which is more I/O and network bound. The points of the graph do not become closer to each other as the cluster size increases, and thus they can not be grouped by cluster size as clearly as happened in Wordcount (see Figures 4e and 4f). In other words, the groups of points belonging to each cluster size overlap between themselves. In this benchmark, the performance differences between the solutions become more significant as the number of nodes increases, to the point that it may be more worthy to change the solution and/or disk technology than increasing the number of nodes. It is to be noted the good energy-performance ratio achieved by DataMPI in this benchmark, especially with large cluster sizes.

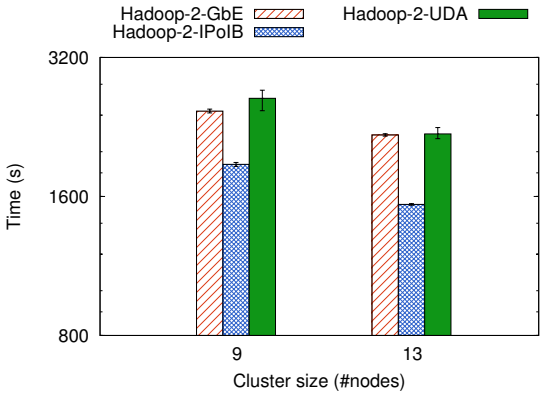
3.3.4. PageRank

Figures 6a and 6b show time results for the PageRank workload using magnetic and SSD disks. This code is not available for DataMPI, and thus its result is not included. The results for RDMA-Hadoop-2 are also not shown, as it did not succeed to finish the algorithm. It can be seen that Hadoop-2-IPoIB improves the execution times of Hadoop-2-GbE, being the best performer for both disk technologies. However, Hadoop-2-UDA outperforms Hadoop-2-GbE only with SSD disks.

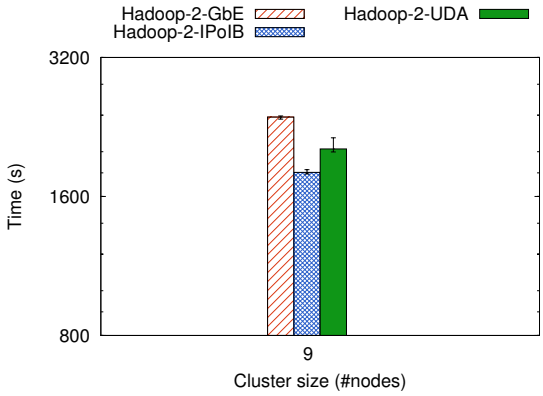
These results can be used to estimate the benefit that a similar workload could obtain in a real environment by using InfiniBand and SSD disks. With magnetic disks, Hadoop-2-IPoIB reduces the execution time of Hadoop-2-GbE by 23.3% using 9 nodes, and by 29.3% using 13 nodes. With SSD disks, Hadoop-2-IPoIB reduces the execution time of Hadoop-2-GbE by 24.1% using 9 nodes. With this number of nodes, the use of SSD vs magnetic disks reduces the execution time of Hadoop-2-UDA, Hadoop-2-IPoIB and Hadoop-2-GbE by 22.3%, 3.9% and 2.9%, respectively.

As PageRank iterates several times over the data, it involves more than one MapReduce process in order to complete the whole algorithm. Each one of these processes generates an intermediate result which is stored in HDFS. The storage of these intermediate data and the final output is accelerated in the case of IPoIB, as it takes advantage of the InfiniBand network. IPoIB improves not only the communication between mappers and reducers but also the communication between DataNodes to replicate blocks in HDFS.

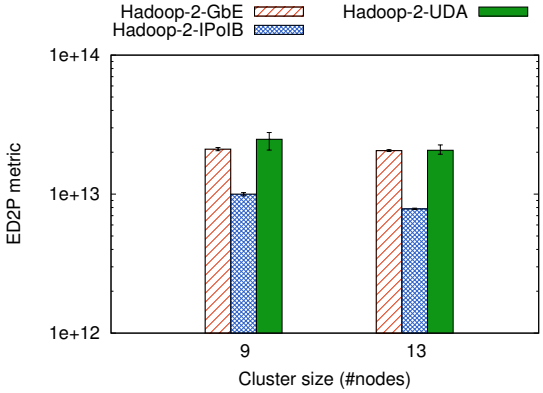
Figures 6c and 6d present ED^2P results for PageRank, which are again strongly dependent on the execution times. With 9 nodes, the use of SSD disks improves the energy-performance ratio for all frameworks. In fact, Hadoop-2-GbE and Hadoop-2-UDA obtain better ratios using 9 nodes and SSD disks than using 13 nodes and magnetic disks. The



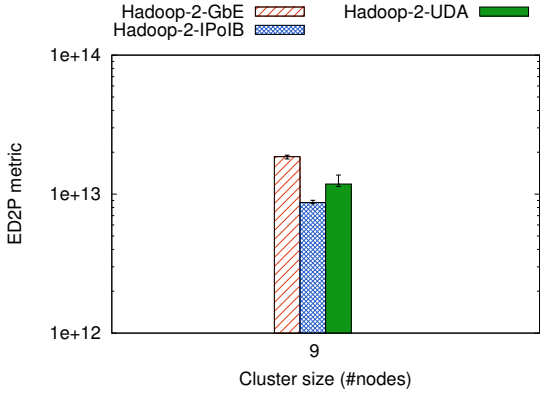
(a) Time with magnetic disks



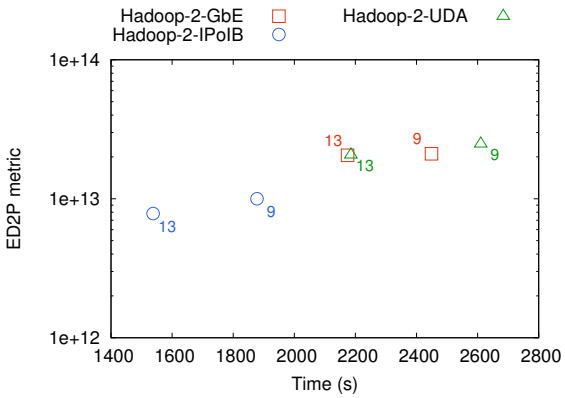
(b) Time with SSD disks



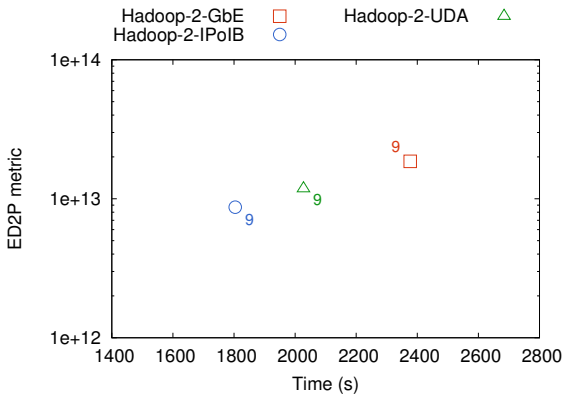
(c) ED^2P metric with magnetic disks



(d) ED^2P metric with SSD disks



(e) ED^2P metric vs time with magnetic disks



(f) ED^2P metric vs time with SSD disks

Figure 6: Time and ED^2P results for the PageRank benchmark (lower is better)

case of Hadoop-2-IPoIB is different, as its ratios are better when using 13 nodes and magnetic disks. Again, it can be seen that the framework used ultimately determines which resource would be better to improve.

Figures 6e and 6f show the correlation of the ED^2P metric and the execution time. In this benchmark and for these cluster sizes, the points are clearly grouped by solution. Hence, choosing the suitable framework is more significant than increasing the cluster size or changing the disk/network technology for achieving good performance and energy efficiency.

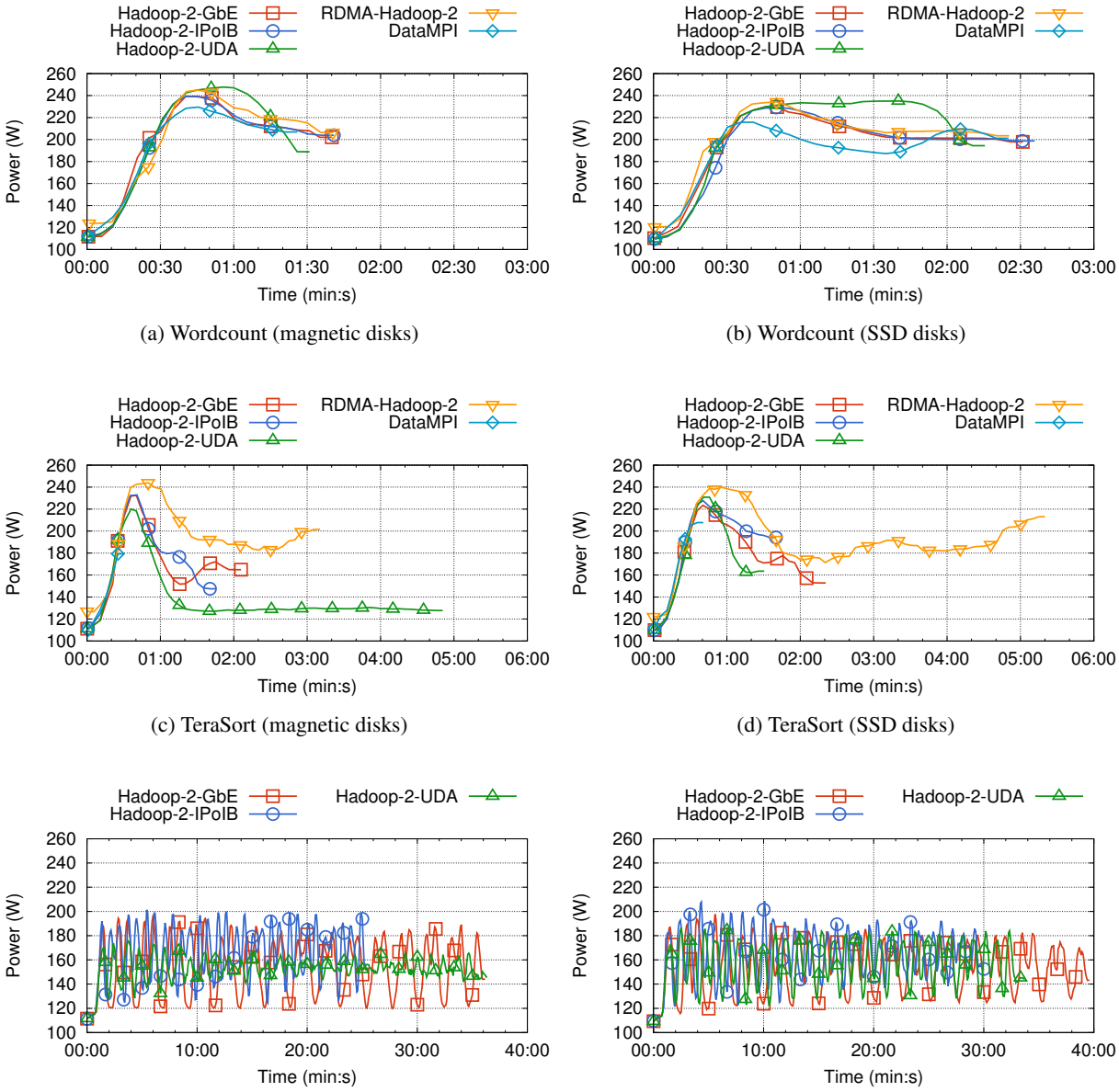
3.4. Power over Time

The analysis of the evolution of the power consumption over time is useful to analyse the phases of the MapReduce process that are more power demanding. This information can be used to identify the main load peaks that would be targeted to improve energy efficiency. Figure 7 shows the evolution of the power consumption over time for Wordcount, TeraSort and PageRank workloads using magnetic (left graphs) and SSD disks (right graphs) on 13 and 9 nodes, respectively. The lines depict the average power values (in watts) among the slave nodes at each moment of the execution of the benchmark, also averaged among the different executions.

In Wordcount (Figures 7a and 7b) and TeraSort (Figures 7c and 7d), the map phase progresses until reaching the maximum power values (around 240 watts for standard Hadoop), and then the power consumption decreases slowly. The Wordcount benchmark, which is CPU bound, presents a similar pattern for all the solutions, with a more stable consumption from the maximum power value until the end of the process. As mentioned in Section 3.3.2, there are not significant differences between the solutions.

The TeraSort benchmark, which has a heavy disk/network activity, decreases its power consumption once it has reached the maximum value, while it performs the shuffle and merge phases. These phases involve high network and I/O traffic to send the map outputs, write the final results to disk and perform the block replication mechanism of HDFS. Therefore, the CPU can run in a lower frequency/voltage, thus decreasing power values. Furthermore, IPoIB improves the shuffle phase and the block replication, while SSD disks also improve the writing of the final output to HDFS. This is more significant in the case of Hadoop-2-UDA, which takes a long time to complete the reduce phase with magnetic disks, but decreases significantly this time with SSD disks. Moreover, RDMA-Hadoop-2 obtains the highest peak values (up to 240 watts); with SSD disks, the increase of the power values at the end of the MapReduce process (see Figure 7d) is related to the higher energy consumption of RDMA-Hadoop-2 (see Figure 5d). Finally, DataMPI with both disk technologies finishes before reaching the peak power values, due to its good performance.

The PageRank workload (Figures 7e and 7f), which executes several iterations over the input data set, shows a cyclic behaviour. Each cycle of high power consumption corresponds to a MapReduce process, which executes an iteration of the overall algorithm. Generally, the frameworks only show differences in the execution time, although Hadoop-2-IPoIB presents higher power values on average. Moreover, the power consumption of Hadoop-2-UDA is more stable with magnetic disks, which means that the disk bandwidth is constraining the CPU frequency and thus power variation.



(a) Wordcount (magnetic disks)

(b) Wordcount (SSD disks)

(c) TeraSort (magnetic disks)

(d) TeraSort (SSD disks)

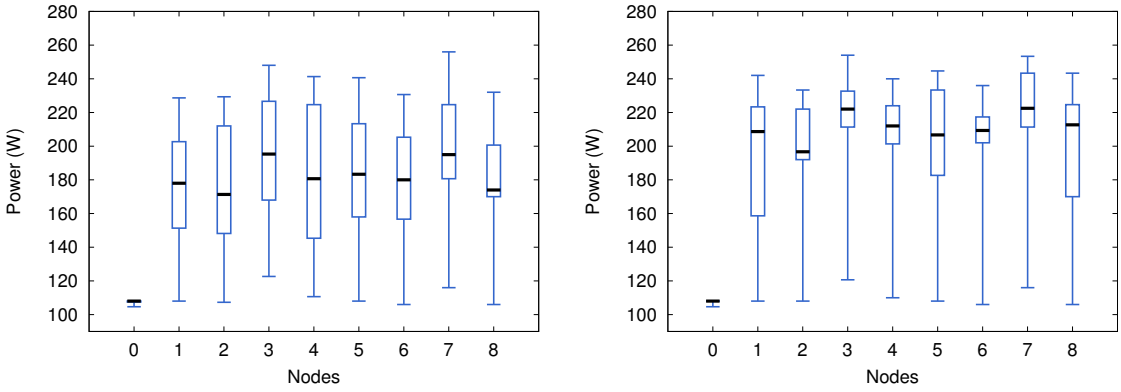
(e) PageRank (magnetic disks)

(f) PageRank (SSD disks)

Figure 7: Power consumption over time (lower is better) using 13 (left graphs) and 9 nodes (right graphs)

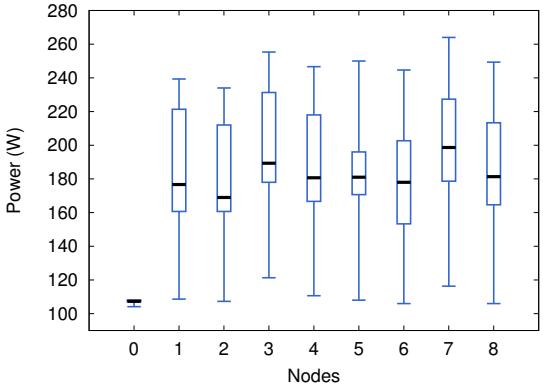
3.5. Power Variability between Nodes

Power variability can be studied to analyse how the workload is balanced between nodes, as a uniform distribution of power values means a proper load balancing. In this work, power variability has been analysed for the TeraSort benchmark using 9 nodes with SSD disks, as this configuration has obtained the most energy-efficient results on average (see Figure 5d). Figure 8 shows the variability of the power consumption between nodes using box and whisker diagrams. These graphs report the measure of the minimum sample, the first quartile (Q1), the median (Q2),

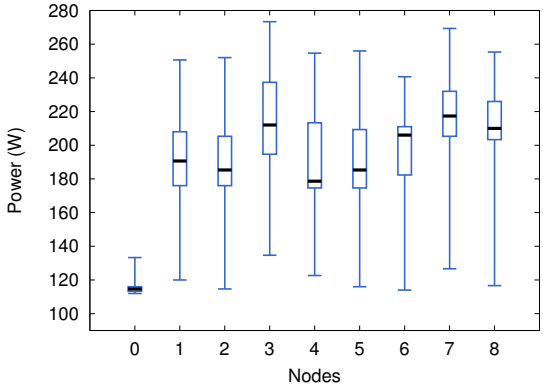


(a) Hadoop-2-GbE

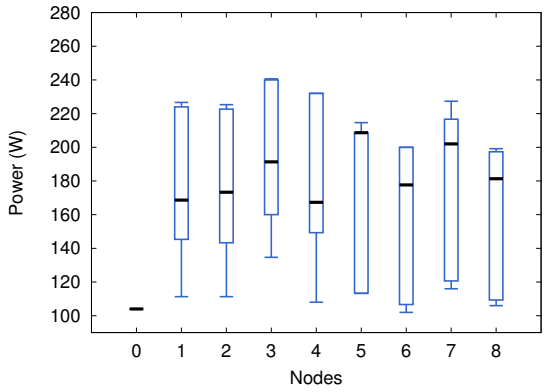
(b) Hadoop-2-IPoIB



(c) Hadoop-2-UDA



(d) RDMA-Hadoop-2



(e) DataMPI

Figure 8: Power variability between nodes for the TeraSort benchmark using 9 nodes and SSD disks

the third quartile (Q3) and the maximum sample. For every framework, it can be observed that the node 0, which runs the master process (i.e. JobTracker/ResourceManager), presents low energy consumption as it does not perform any computation-intensive task.

These results reveal that power consumption, and hence workload, is uniformly distributed for every solution. Moreover, there are not significant differences in the distribution that each framework obtains. However, there are some characteristics that can be observed. For instance, Hadoop-2-IPoIB (see Figure 8b) shows similar maximum and minimum power values to Hadoop-2-GbE (see Figure 8a), but its median values are higher. This means that Hadoop-2-IPoIB has higher power values on average, although its maximum values are similar to those of Hadoop-2-GbE. Hadoop-2-UDA (Figure 8c) also shows similar maximum and minimum values to Hadoop-2-GbE. Its range between Q1 and the median is shorter than between the median and Q3, which means higher variability in the latter range. RDMA-Hadoop-2 (Figure 8d) shows higher maximum results than the other frameworks, which means that higher peak values are reached during the execution. Finally, DataMPI (Figure 8e) differs slightly from the other frameworks, as it shows the lowest maximum power values. It also presents a large interquartile range (between Q1 and Q3) that implies a higher variability for the most common power values.

4. Conclusions

The use of high-performance interconnects for Big Data computing can improve significantly the performance of applications. In particular, popular MapReduce frameworks like Apache Hadoop can greatly benefit from using InfiniBand, which allows to achieve higher scalability in order to address challenging real-world workloads. This work has analysed in detail several HPC-oriented MapReduce frameworks that provide different approaches to the same issues. Future implementations need to take into account the behaviour of the existing solutions, taking advantage of those specific characteristics that provided the best performance. Moreover, the most appropriate framework may vary from problem to problem. The main conclusion that can be drawn from the analysis of the results is that DataMPI obtains the best performance and energy efficiency on average, although existing Hadoop applications can also be transparently boosted by using IPoIB without source code modifications.

Mellanox UDA has definitely contributed with important ideas to optimize the MapReduce process, such as the network levitated merge algorithm. However, its overall performance is highly dependent on the underlying disk technology. Hence, the utilization of magnetic disks can cause congestions in the shuffle/merge phases, delaying the whole process. This issue is solved by using SSD disks, but there is still the constraint to adapt this solution to overcome this limitation. Moreover, Mellanox UDA does not improve the performance of application benchmarks. Regarding RDMA-Hadoop, it constitutes a strong attempt to adapt Hadoop to high-performance interconnects. In fact, it achieves good performance when using magnetic disks, but it has shown severe memory problems when managing large data sets or iterative workloads, hindering its adoption in real environments. Moreover, RDMA-Hadoop does not improve the results of standard Hadoop when using a high number of nodes. The approach proposed by DataMPI is totally different from the other solutions, as it tries to cover a wider range of applications (e.g. MapReduce, Iterative, Stream). Its performance results are better than those of Hadoop in HPC systems, even though applications must be rewritten for using it.

Table 7 presents an overall review of the evaluated MapReduce solutions in terms of performance, energy efficiency, scalability and stability. It also shows how easy it is to configure each framework, the quality of the available documentation and how easy the user can retrieve information about its specific behaviour. Note that this table assesses these features for the particular versions evaluated in this paper (see Table 4), using the specific infrastructure and configuration shown in Tables 5 and 6, respectively. Therefore, these values are subject to change as the projects keep on evolving.

Table 7: Review of the evaluated MapReduce solutions
VH:Very High H:High M:Medium L:Low

	Performance	Energy efficiency	Scalability	Stability	Ease of configuration	Documentation quality	Usage information
Hadoop-2-GbE	L	L	H	H	M	H	H
Hadoop-2-IPoIB	H	H	H	H	M	H	H
Hadoop-2-UDA	M	M	H	H	M	M	M
RDMA-Hadoop-2	H	M	H	L	H	H	L
DataMPI	VH	VH	VH	M	H	M	M

Future work includes further evaluations using larger cluster sizes and analysing the impact of storage systems in frameworks developed from scratch like DataMPI. The evaluation of additional MapReduce frameworks and high-performance interconnects (e.g. RoCE) would also be of great interest.

Acknowledgements

This work was supported by the Ministry of Economy and Competitiveness of Spain and FEDER funds of the EU (Project TIN2013-42148-P).

References

[1] J. Dean, S. Ghemawat, MapReduce: Simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113.

[2] Apache Hadoop, `http://hadoop.apache.org/`, [Last visited: October 2015].

[3] IBTA, InfiniBand Trade Association, `http://www.infinibandta.org`, [Last visited: October 2015].

[4] Mellanox Technologies: Hadoop, `http://www.mellanox.com/page/hadoop`, [Last visited: October 2015].

[5] High-Performance Big Data, `http://hibd.cse.ohio-state.edu/`, [Last visited: October 2015].

[6] DataMPI, `http://datampi.org/`, [Last visited: October 2015].

[7] TOP 500 List (June 2015), `http://top500.org/lists/2015/06/`, [Last visited: October 2015].

[8] Hadoop YARN, `http://hadoop.apache.org/docs/r2.7.1/hadoop-yarn/hadoop-yarn-site/YARN.html`, [Last visited: October 2015].

[9] W. Gropp, E. Lusk, N. Doss, A. Skjellum, A high-performance, portable implementation of the MPI message passing interface standard, Parallel Computing 22 (6) (1996) 789–828.

[10] The OpenFabrics Alliance (OFA), `https://www.openfabrics.org/`, [Last visited: October 2015].

[11] IP over InfiniBand (IPoIB) Architecture, http://www.ietf.org/rfc/rfc4392.txt.pdf, [Last visited: October 2015].

[12] Y. Wang, X. Que, W. Yu, D. Goldenberg, D. Sehgal, Hadoop acceleration through network levitated merge, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11), Seattle, WA, USA, 2011, pp. 57:1–57:10.

[13] Memcached, http://memcached.org/, [Last visited: October 2015].

[14] S. Sur, H. Wang, J. Huang, X. Ouyang, D. K. Panda, Can high-performance interconnects benefit Hadoop Distributed File System?, in: Proceedings of the Workshop on Micro Architectural Support for Virtualization, Data Center Computing, and Clouds (MASVDC'10), Atlanta, GA, USA, 2010.

[15] N. S. Islam, M. Wasi-Ur-Rahman, J. Jose, R. Rajachandrasekar, H. Wang, H. Subramoni, C. Murthy, D. K. Panda, High performance RDMA-based design of HDFS over InfiniBand, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'12), Salt Lake City, UT, USA, 2012, pp. 35:1–35:12.

[16] M. Wasi-Ur-Rahman, N. S. Islam, X. Lu, J. Jose, H. Subramoni, H. Wang, D. K. Panda, High-Performance RDMA-based design of Hadoop MapReduce over InfiniBand, in: Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium Workshops and PhD Forum (IPDPSW'13), Boston, MA, USA, 2013, pp. 1908–1917.

[17] X. Lu, N. S. Islam, M. Wasi-Ur-Rahman, J. Jose, H. Subramoni, H. Wang, D. K. Panda, High-Performance design of Hadoop RPC with RDMA over InfiniBand, in: Proceedings of the 42nd International Conference on Parallel Processing (ICPP'13), Lyon, France, 2013, pp. 641–650.

[18] M. Wasi-Ur-Rahman, X. Lu, N. S. Islam, D. K. Panda, HOMR: a Hybrid approach to exploit maximum Overlapping in MapReduce over high performance interconnects, in: Proceedings of the 28th ACM International Conference on Supercomputing (ICS'14), Munich, Germany, 2014, pp. 33–42.

[19] T. Hoefler, A. Lumsdaine, J. Dongarra, Towards efficient MapReduce using MPI, in: Proceedings of the 16th European PVM/MPI Users' Group Meeting (EuroPVM/MPI'09), Espoo, Finland, 2009, pp. 240–249.

[20] X. Lu, F. Liang, B. Wang, L. Zha, Z. Xu, DataMPI: Extending MPI to Hadoop-like Big Data computing, in: Proceedings of the 28th IEEE International Parallel and Distributed Processing Symposium (IPDPS'14), Phoenix, AZ, USA, 2014, pp. 829–838.

[21] D. LaSalle, G. Karypis, MPI for Big Data: New tricks for an old dog, Tech. Rep. UMN-CSE TR 13-032, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA (2013).

[22] MapReduce-MPI Library, http://mapreduce.sandia.gov, [Last visited: October 2015].

[23] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang, C. Zheng, G. Lu, K. Zhan, X. Li, B. Qiu, BigDataBench: A Big Data benchmark suite from Internet services, in: Proceedings of the 20th IEEE International Symposium on High-Performance Computer Architecture (HPCA'14), Orlando, FL, USA, 2014, pp. 488–499.

[24] Apache Spark, https://spark.apache.org/, [Last visited: October 2015].

[25] F. Liang, C. Feng, X. Lu, Z. Xu, Performance benefits of DataMPI: A case study with BigDataBench, in: Proceedings of the 4th Workshop on Big Data Benchmarks, Performance Optimization and Emerging Hardware (BPOE'14), Salt Lake City, UT, USA, 2014.

[26] X. Lu, B. Wang, L. Zha, Z. Xu, Can MPI benefit Hadoop and MapReduce applications?, in: Proceedings of the 7th International Workshop on Scheduling and Resource Management for Parallel and Distributed Systems (SRMPDS'11), Taipei, Taiwan, 2011, pp. 371–379.

[27] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, G. Fox, Twister: A runtime for iterative MapReduce, in: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC'2010), Chicago, IL, USA, 2010, pp. 810–818.

[28] Z. Fadika, E. Dede, M. Govindaraju, L. Ramakrishnan, Benchmarking MapReduce implementations for application usage scenarios, in: Proceedings of the 12th IEEE/ACM International Conference on Grid Computing (GRID'11), Lyon, France, 2011, pp. 90–97.

[29] Z. Fadika, M. Govindaraju, R. Canon, L. Ramakrishnan, Evaluating Hadoop for data-intensive scientific operations, in: Proceedings of the 5th IEEE International Conference on Cloud Computing (CLOUD'12), Honolulu, HI, USA, 2012, pp. 67–74.

[30] S. Huang, J. Huang, J. Dai, T. Xie, B. Huang, The HiBench benchmark suite: Characterization of the MapReduce-based data analysis, in: Proceedings of the 26th IEEE International Conference on Data Engineering Workshops (ICDEW'10), Long Beach, CA, USA, 2010, pp. 41–51.

[31] A. Sangroya, D. Serrano, S. Bouchenak, MRBS: Towards dependability benchmarking for Hadoop MapReduce, in: Proceedings of the 18th International Euro-Par Conference on Parallel Processing Workshops (Euro-Par'12), Rhodes Island, Greece, 2012, pp. 3–12.

[32] N. Tiwari, S. Sarkar, U. Bellur, M. Indrawan, An empirical study of Hadoop's energy efficiency on a HPC cluster, in: Proceedings of the 14th International Conference on Computational Science (ICCS'14), Cairns, Australia, 2014, pp. 62–72.

[33] A. J. Martin, Towards an energy complexity of computation, Information Processing Letters 77 (2–4) (2001) 181–187.

[34] T. Wirtz, R. Ge, Improving MapReduce energy efficiency for computation intensive workloads, in: Proceedings of the 2nd International Green Computing Conference and Workshops (IGCC'11), Orlando, FL, USA, 2011, pp. 1–8.

[35] E. Feller, L. Ramakrishnan, C. Morin, On the performance and energy efficiency of Hadoop deployment models, in: Proceedings of the 2013 IEEE International Conference on Big Data (IEEE BigData 2013), Santa Clara, CA, USA, 2013, pp. 131–136.

[36] J. Veiga, R. R. Expósito, G. L. Taboada, J. Touriño, MREv: An Automatic MapReduce Evaluation Tool for Big Data Workloads, in: Proceedings of the International Conference on Computational Science (ICCS'15), Vol. 51, Reykjavík, Iceland, 2015, pp. 80–89.

[37] U. Kang, C. E. Tsourakakis, C. Faloutsos, PEGASUS: A peta-scale graph mining system implementation and observations, in: Proceedings of the 9th IEEE International Conference on Data Mining (ICOM'09), Miami, FL, USA, 2009, pp. 229–238.

[38] HP Integrated Lights-Out (iLO) - HP Servers, `http://www.hp.com/go/ilo`, [Last visited: October 2015].

[39] P. Bose, M. Martonosi, D. Brooks, Modeling and analyzing CPU power and performance: Metrics, methods, and abstractions, in: Tutorials at the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2001), Cambridge, MA, USA, 2001.

Jorge Veiga is a Ph.D. student in the Department of Electronics and Systems at the University of A Coruña. His research interests are focused on the performance evaluation and optimization of Big Data models and frameworks in High Performance Computing environments.

Roberto R. Expósito is a postdoctoral researcher in the Department of Electronics and Systems. His research interests are in the area of High Performance Computing, focused on message-passing communications on high-speed cluster networks, and network support for cloud and Big Data computing.

Guillermo L. Taboada is an Associate Professor of computer engineering in the Department of Electronics and Systems at the University of A Coruña. His main research interest is in the area of High Performance Computing, focused on high-speed networks, programming languages, cloud and Big Data computing. His homepage is `http://gac.udc.es/~gltaboada`.

Juan Touriño is a Full Professor of computer engineering in the Department of Electronics and Systems at the University of A Coruña. He has extensively published in the area of parallel and distributed computing, currently focusing on the intersection of High Performance Computing and Big Data. He is coauthor of more than 150 technical papers in this area. His homepage is `http://gac.udc.es/~juan`.