Towards a next generation of scientific computing in the Cloud

Yassine Tabaa[1] and Abdellatif Medouri[1]

[1] Information and Communication Systems Laboratory, College of Sciences, Abdelmalek Essaadi University Tetouan, Morocco

Abstract

More than ever, designing new types of highly scalable data intensive computing is needed to qualify the new generation of scientific computing and analytics effectively perform complex tasks on massive amounts of data such as clustering, matrix computation, data mining, information extraction … etc. MapReduce, put forward by Google, is a well-known model for programming commodity computer clusters to perform large-scale data processing in a single pass. Hadoop is the most popular open-source implementation of the MapReduce model which provides a simple abstraction for large-scale distributed algorithm; it has become a popular distributed computing and data analysis paradigm in recent years. While, Hadoop MapReduce suits well for embarrassingly parallel problems, it suffers significant troubles when dealing with iterative algorithms; as a consequence, many alternative frameworks that support this class of algorithms were created. In this paper, we propose architecture for such configuration implemented in an SPC (Scientific Private Cloud) prototype, using the Hadoop 2.0 next generation platform to allow the use of alternative programming frameworks respecting a hybrid approach, while retaining the scalability and fault tolerance of Hadoop MapReduce. By adapting scientific problems to execute them in our Scientific Cloud, experiments conducted show the effectiveness of the proposed model and its impact on the ease of frameworks handling.

Keywords: *Scientific Cloud, Hadoop next generation, Hybrid approach.*

1. Introduction

Nowadays, the amount of data generated and stored by scientific instrumentation/simulation (e.g., massive-scale simulations, sensor deployments, high-throughput lab equipment), businesses and industry (e.g., web-data, click-stream, network-monitoring log), and Internet publishing and archiving is growing exponentially. IDC (*International Data Corporation* a market research, analysis and advisory) predicts [1] that the "digital universe" — the total aggregation of information and content created and replicated— will surpass 7ZB (7 trillion gigabytes) by 2015, growing by a factor of 9 in just five years.

The increase in the volume of data also increases the amount of computing power needed to transform raw data into meaningful information. In such situations, the required processing power far exceeds the processing capabilities of individual computers, leading to the use of parallel/distributed computing strategies. Many research works were conducted to design effective frameworks providing the ability to analyze huge amounts of data in a distributed and parallel environment across hundreds or thousands of machines in a reasonable delay.

MapReduce is considered a high productivity alternative to traditional parallel programming paradigms for enterprise computing and scientific computing [2]. It was developed first by Google in 2004 [3] as a parallel computing framework to perform distributed computing on a large number of commodity computers. As the Google implementation is proprietary, several open-source implementations of MapReduce model have emerged, the most famous of which is Hadoop [4]. Hadoop was primarily supported by Yahoo, and currently hosted as an Apache project. Major stakeholders like Facebook, HP and Ebay …etc., are using Hadoop to perform various tasks such as sorting, log analyzing, machine learning and so on.

Scientific computing is usually associated with complex data-intensive computations such as high–dimensional scientific simulations and requires a huge amount of computer resources. Cloud Computing, with its promise of provisioning virtually infinite resources, seems to be a good alternative for solving these scientific computing problems.

In the Cloud, Amazon ElasticMapReduce [5] offers Apache Hadoop as a hosted service on the Amazon AWS (Amazon Web Services) cloud environment that provides resizable compute capacity.

Although Hadoop MapReduce, is widely used, many works [2] [6] [7] [8] show that for specific configurations and applications like processing iterative algorithms, Hadoop MapReduce loses significantly its performance and efficiency especially when the number of iterations is

Copyright (c) 2012 International Journal of Computer Science Issues. All Rights Reserved.

significant. Alternative MapReduce frameworks such as Twister [7], Spark [6], and Haloop [8] are known for their efficient implementations of the MapReduce paradigm that work effectively over iterative algorithms.

In this paper, we adopt the next generation Hadoop 2.0 as a principal component in our architecture to design a next generation of scientific computing in the Cloud enabling a hybrid approach that shall allow the utilizations of alternative frameworks than Hadoop MapReduce.

The rest of the paper is structured as follows.

Section 2 discusses the related work. Section 3 introduces the proposed architecture and describes the implemented environment. Section 4 outlines the performances of our system through two experiments. Section 5 concludes the paper and describes the future research directions.

2. Related work

The parallel processing in Spark is based on the MapReduce model with support of iterative applications, Spark utilizes RDDs (resilient distributed datasets) [9] that can be explicitly persisted in memory across the computation nodes. However, Spark framework does not support group reduction operation and only uses one task to collect the reduced result, which can affect the scalability of algorithms.

Twister is an alternative of Hadoop MapReduce, it allows long-lived map tasks to keep static data in memory between jobs in a manner of "configure once, and run many times" [7], using publish/subscribe messaging middleware system for command communication and data transfers. The unique feature of Twister is to support iterative MapReduce programming model. However, Twister does not currently implement fault tolerance.

Saurabh Sehgal et al. [10] introduce SAGA (Simple API for Grid Applications) – an API that support multiple and independent distributed programming – to enable interoperability for distributed scientific applications.

In the Cloud, Amazon ElasticMapReduce [5] offers Apache Hadoop as a hosted service on the Amazon AWS (Amazon Web Services) cloud environment. AppEngine-MapReduce [11] is an open source library for doing MapReduce-style computations on the Google App Engine platform. However, none of them supports iterative MapReduce.

Beside, many researches focus on the performance improvement of aspects in Hadoop MapReduce such as optimizing its job scheduling policy in a cloud environment.

In academia, Open Cirrus [13] is an example of a closed federation between universities and research centers in order to aid research in design, provisioning, and management of services in scale of multi data centers. We present in [12] the beneficial side effect of using Cloud services for education and research purpose. Ali Bagherinia et Al. [14] present a model to execute long computations and other services in cloud computing.

3. Architecture and implementation

The section starts with introducing the architecture of a next generation of Scientific Computing in the Cloud. After providing a description of the small-scale Scientific Private Cloud infrastructure, architectural details of Hadoop 2.0 and Spark, the main components of our model, are given.

3.1 The overall architecture

The overall architecture of our Scientific Private Cloud is described in Fig 1. This architecture aims to help building a next generation of Scientific Cloud infrastructure for research purposes.

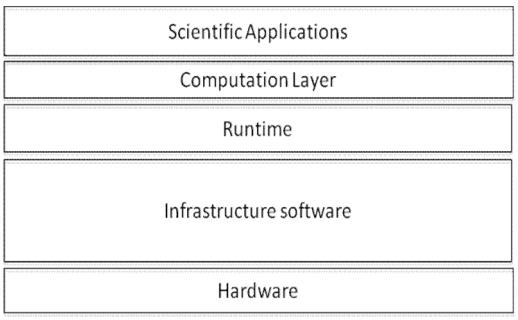

Fig. 1: The overall architecture of our Scientific Private Cloud.

The proposed model has a layered architecture consisting of 5 layers :

- The scientific applications layer aims to provide a set of scientific primitives to the client.
- The computation layer is the most important and innovative component in this architecture. It gives the ability to use other programming frameworks than MapReduce. Practically, the programming framework used to develop a scientific application is no more of big importance.
- The runtime layer is the core of computation. It is responsible for managing aspects like scheduling, jobs, tasks and fault tolerance. This layer is the main container of the data-intensive scientific application.

Copyright (c) 2012 International Journal of Computer Science Issues. All Rights Reserved.

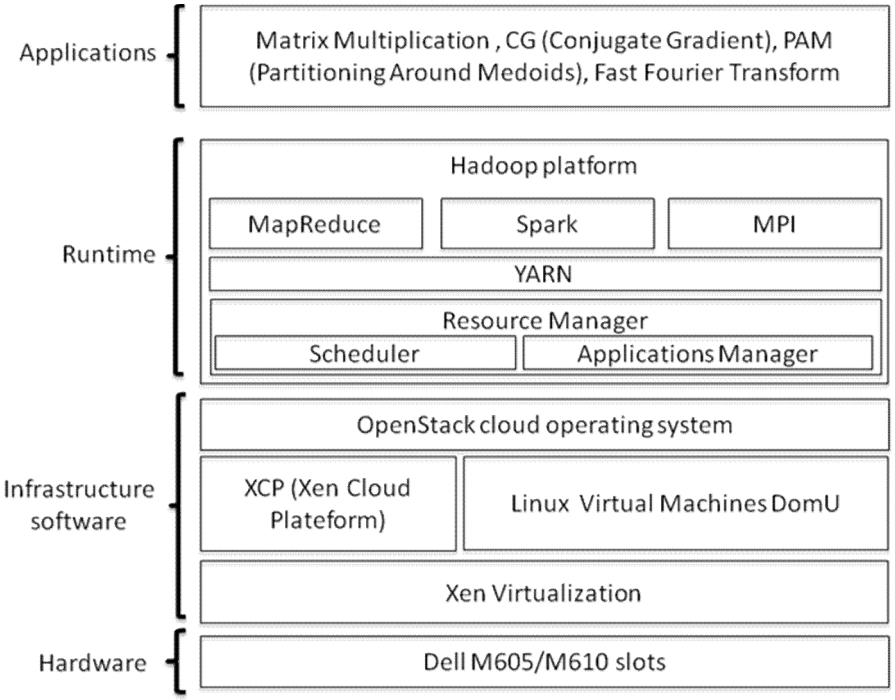

Fig. 2: Infrastructure implementation of our Scientific Private Cloud.

- The infrastructure software layer is a low level layer delivering a platform for server virtualization and cloud computing by combining the isolation and multi-tenancy capabilities required to provide services over the cloud. This layer is orchestrated by a massively scalable cloud operating system which controls provisioning/release of virtual resources.
- The hardware layer represents the hardware infrastructure.

By relying heavily on virtualization technology, the proposed model takes advantage of:

- The best use of distributed physical resources by avoiding hardware re-initialization when starting and stopping runtime environments in a VM (Virtual Machine).
- The minimal effort needed to maintain physical resources.
- The quick deployment of services and applications.
- The use of highly customized environments by enabling the simultaneous coexistence of different runtime environments on the same physical infrastructure.
- The effective isolation of CPU and memory performance between VMs.
- The enhanced flexibility and elasticity of the Cloud Environment by permitting dynamic migration of VMs running on physical servers of the same resource pool.

3.2 Next Generation Hadoop 2.0 (MRv2 or YARN)

Considered as the main component of our architecture, the next generation of Hadoop or MRv2 (MapReduce version 2) (out of Apache) allows building and developing distributed applications using programming frameworks other than MapReduce, through introducing a new aspect of Hadoop called YARN (Yet Another Resource Negotiator) technology. In this case, MapReduce is considered just a *computation layer* on top of the scheduler component and can be swapped out.

Hadoop 2.0 is based on two major functions, a single Resource Manager and a per-application job life-cycle management called Application Master. The Resource Manager has two main services:

Copyright (c) 2012 International Journal of Computer Science Issues. All Rights Reserved.

- A pluggable Scheduler, which manages and applies the resource scheduling policy in the Hadoop cluster.
- An Applications Manager, which manages running Application Masters in the cluster; this component is responsible for starting and stopping application masters and for monitoring and restarting them on different nodes in case of failure.

In addition, The Hadoop open source project of Apache includes the following modules [4]:

- **Hadoop Common**: representing the core of Hadoop, it is the common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System (HDFS)**: The distributed file system that provides high-throughput access to application data.
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets; it is the native implementation of the MapReduce paradigm on Hadoop.

3.3 Spark framework

Spark is an alternative open source MapReduce framework that supports applications based on iterative algorithms which reuses a working set of data across multiple parallel operations. The most important enhancement of Spark is that it allows some data to be shared between the mappers while they are computing, a fact that significantly reduces required data over the network. Indeed, Spark framework distinguishes between static data that not change during the iterations and dynamic data which may change in the course of each iteration. Spark overcomes this problem of data reuse across iterations by providing a new storage primitive called RDDs (resilient distributed datasets) [9], which allows clients to store data in memory through queries. This can be useful, for example, if each of the mappers is looking for an optimal solution, and they all want to share the current best solution and to eliminate bad solutions early.

Spark framework is built on top of Mesos project [15] [16]: a "cluster operating system" that permits multiple parallel applications to entirely share a cluster and provides an API for applications to launch tasks on a cluster. To develop Spark applications faster, Spark integrates Scala [17], a popular functional language for the JVM; it allows developers to manipulate distributed datasets (RDDs) like local collections.

3.4 Experimental setup

To analyze the performance of our model for scientific computing, we set up a small-scale private Cloud using open source components on Hadoop 2.0 (YARN) composed of one master acting as the Resource Manager and 18 nodes, each node is a virtual machine with 2.4 GHz, 2 GB of RAM memory and 20 GB disk space allocated for HDFS (Hadoop Distributed File System), making the total size of 360 GB, these nodes are monitored through a Node Manager service. In this work, we use XCP (Xen Cloud Platform) [18] and OpenStack [19] to deliver Cloud resources and services to the client, XCP is an open source enterprise-ready server virtualization and cloud computing platform, based on the Xen Hypervisor and support a wide range of guest operating systems. While OpenStack is an open source software for building private and public Clouds, in our case it will orchestrate our Scientific Private Cloud.

Fig. 2 depicts the physical infrastructure employed in the implementation of our SPC, following a hybrid approach. The infrastructure enables the building and deploying distributed applications based on alternative frameworks other than MapReduce, which is natively configured in Hadoop, while using the next generation of Hadoop 2.0 platform. Indeed, clients can take full advantage of all functionalities of the Hadoop 2.0 platform such as scheduling, managing jobs and fault tolerance.

We decided to test the Spark MapReduce framework for iterative algorithms, using a YARN implementation of Spark [20] over the Hadoop 2.0 platform as a hybrid approach. First, we assessed the easiness of switching from a MapReduce framework to another (in this case the Spark framework) in just few seconds while maintaining the same virtual resources and retaining the scalability and fault tolerance of Hadoop 2.0 platform. Then, we compared the performances of our implementation through the proposed Scientific Private Cloud respectively with three different configurations, running applications into:

- The default framework MapReduce of Hadoop 2.0.
- An implementation of a YARN compatible version of Spark over Hadoop 2.0.
- A simple Spark cluster.

4. Evaluation and performance

In this section, we present the evaluation of our Scientific Private Cloud prototype. Actually, the two experiments conducted on our next generation Scientific Cloud prototype in a small-scale private Cloud, show the efficiency of the proposed approach.

Copyright (c) 2012 International Journal of Computer Science Issues. All Rights Reserved.

Usually used for solving linear algebra required in complex scientific applications, MM (Matrix Multiplication) and the CG (Conjugate Gradient) are the two basic primitives implemented in these experiments.
The two experiments aim at demonstrating that our prototype is suitable for this class of iterative algorithms using the Spark framework as a computation engine over the Hadoop 2.0 platform (Hadoop 0.23.1 version).
In this evaluation, we compare three different implementations of Matrix Multiplication and Conjugate Gradient algorithms, varying the matrix dimension from 100 to 8000. The first, *MapReduce-Hadoop*, uses the native MapReduce framework as a computation layer of the Hadoop 2.0 platform. The second configuration, *Spark-Hadoop,* is an alternative implementation of the Hadoop MapReduce computation layer; it uses a YARN based version of Spark [20] over the Hadoop 2.0 platform as a hybrid approach. And the third configuration uses a basic Spark Cluster, we simply named it *Spark*.

4.1 experiment 1: Matrix Multiplication (MM)

For many data-intensive scientific applications such as large-scale numerical analysis, computational physics, and graph rendering, massive matrix computation is used as the primary means. In this experiment we implement a simple iterative approach of the matrix multiplication algorithm. We assume that square matrix A and B are used for multiplication in the following algorithm:
Adapting matrix multiplication to the MapReduce Model
The Mapper:

- Map tasks will need the following inputs :
  - The input files for A and B are streams of (key,value) pairs in matrix format.
  - Each key is a row index of B.
  - Each value is the corresponding matrix column vector of the row.
- Then, the map task multiplies all columns of i-th row of A with the received column vector.

The reducer:

- The reduce task collects the i-th product into the result matrix C=A*B.

After running the algorithm according to the three configurations, the results are depicted in Fig. 3, Fig. 4, and Fig. 5. The Fig. 3 shows the run times of the MM and CG algorithms through MapReduce-Hadoop, while Fig. 4 and Fig. 5 represent respectively the run times of the MM and CG algorithms through Spark-Hadoop and Spark implementations.
The fact of using Spark over Hadoop permits to mitigate the data locality problem of Hadoop, which explains the comparable performance, as depicted in Fig. 4, of Spark-Hadoop and Spark implementations.

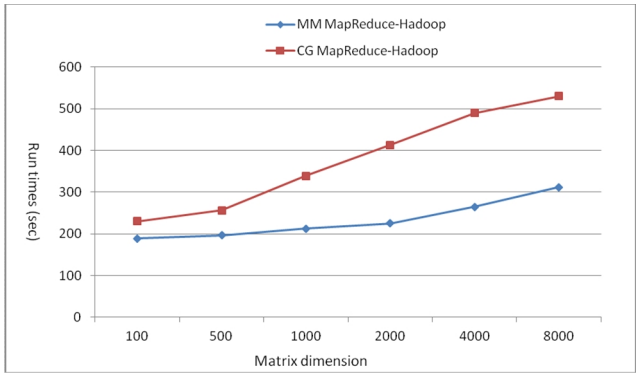

Fig. 3 Run times for MM and CG over MapReduce-Hadoop.

4.2 experiment 2: CG (Conjugate Gradient)

The Conjugate Gradient algorithm (CG) is an iterative algorithm that is commonly used to solve systems of linear equations in matrix form:

$$Ax = b$$

Where the A is a Known, square, symmetric, and positive-definite matrix, b is a known vector and x is the solution vector of the linear system.
CG is a relatively complex algorithm, insofar as, it is not possible to directly adapt the whole algorithm to MapReduce model. To deploy CG algorithm, first we have to reduce matrix and vector operations to the MapReduce model separately, indeed, operations needed to be adapted are:

- Matrix – vector multiplication
- Dot product
- Two vectors addition
- Vector and scalar multiplication

Although, some minor computation is done outside of these methods, the majority of the time is spent in these operations above.
Fig. 3 shows the run times of the CG algorithm through MapReduce-Hadoop implementation, and Fig. 5 depicts the run times of the CG algorithm through Spark-Hadoop and Spark implementations.
In both MM and CG implementations, comparing the Spark-Hadoop and MapReduce-Hadoop run times clearly shows that Spark-Hadoop is more efficient for iterative algorithms. Spark-Hadoop implementation can solve larger problem size in less time and for the same size problem it is 3-7 times faster than MapReduce-Hadoop implementation, when running on 18 nodes. The significant advantages in using Spark over Hadoop come from:

- Its ability to keep static input data in memory across iterations.

Copyright (c) 2012 International Journal of Computer Science Issues. All Rights Reserved.

- Using the Resource Manager of next generation Hadoop which was revamped into an enhanced scheduler and applications manager.
- The strong fault tolerance of Hadoop

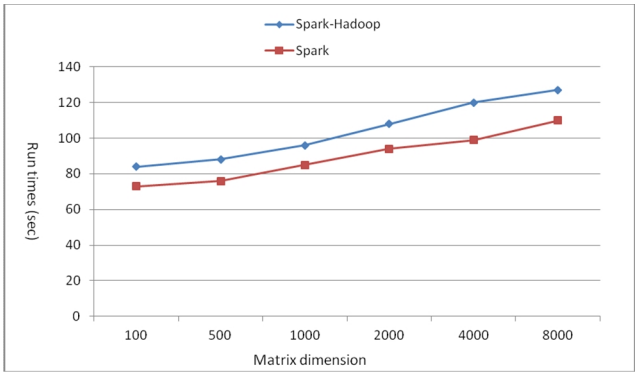

Fig. 4 Run times for MM over Spark-Hadoop and Spark.

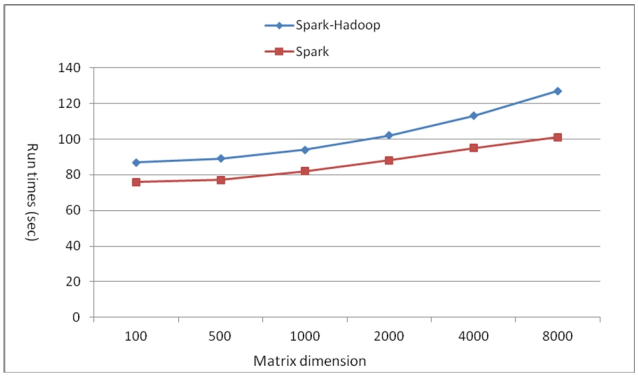

Fig. 5 Run times for CG over Spark-Hadoop and Spark.

Although, Spark implementation, as depicted in Fig. 4 and Fig. 5, performs much better for the iterative MM and CG algorithms. It has several limitations for distributed applications, first, when it comes to treating data intensive tasks such as Terabytes of data (2 TB for example) Spark would require more than 236 nodes with 8 GB of memory each to only store the data into the memory, not to mention the memory needed for the rest of the application, the framework and the operating system.

4.3 Common performances

As described earlier, the Spark framework distinguishes between static data that not change during the iterations and dynamic data which may change in the course of each iteration. This is the main characteristic which distinguishes Spark from Hadoop MapReduce and explains the efficient support for iterative algorithms by Spark implementations in both experiments.

In addition, we note the flexibility achieved when swapping between the frameworks. Indeed, switching from the first configuration where we used the default Hadoop MapReduce framework, to the configuration implementing Spark framework as a YARN based component while using the same Hadoop 2.0 environment takes only 13 seconds in average (for all the eighteen-18 SPC nodes) to deploy and configure all prerequisites to run *scala* applications over Spark environment. However, to reset and configure the overall environment (releasing virtual resources, create new ones, initializing the runtime) to set up a Spark Cluster takes approximately 22 minutes.

5. Conclusions and Perspectives

This work presents a next generation of Scientific Computing in the Cloud using Hadoop 2.0 platform, By adopting such architecture, it has been shown that Spark can be a powerful complement to Hadoop for iterative applications while retaining, on one hand, the same runtime environment namely Hadoop 2.0 for scientific applications, and on the other hand, the full performances of Hadoop MapReduce known for working effectively over embarrassingly parallel algorithms.
In future Work, we envision to deploy more frameworks, such as MPI, Twister… and so on, over the next generation of Hadoop 2.0 platform, and examine their efficiency through the implementation of complex algorithms using greater numbers of nodes. Also, we plan to extend the proposed architecture to use Public Cloud resources from providers like AWS or Google App Engine.
Furthermore, we believe that the use of Hadoop platform in scientific computation is even more promising than it is narrated in the current distributed systems literature. Therefore we will focus our work on this platform by elaborating an abstract computation engine component based on the YARN technology, enabling Hadoop users to deploy straightforwardly multiple scientific applications developed using multiple frameworks over the same environment.

Acknowledgments

We are indebted to the staff of the Computer Resources Center of Abdelmalek Essaadi University; our thanks go also to S.Ibn Elahrach for supporting us in this work.

References

[1] The Value of SSD Integration in Evolving Storage Architectures, Adapted from "Worldwide Solid State Drive 2011–2015 Forecast and Analysis" by Jeff Janukowicz, IDC #228894

Copyright (c) 2012 International Journal of Computer Science Issues. All Rights Reserved.

[2] Srirama, S. N., Jakovits, P., & Vainikko, E. (2012). Adapting scientific computing problems to clouds using MapReduce. Future Generation Computer Systems, 28(1), 184-192. Elsevier B.V. doi:10.1016/j.future.2011.05.025
[3] J. Dean, S. Ghemawat, MapReduce: simplified data processing on large clusters,in: OSDI'04: Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation, USENIX Association, Berkeley, CA, USA,2004, pp. 107–113.
[4] Apache Hadoop Project, http://hadoop.apache.org/
[5] Amazon ElasticMapReduce Web Services, http://aws.amazon.com/elasticmapreduce/
[6] M. Zaharia, M. Chowdhury, M.J. Franklin, S. Shenker, I. Stoica, Spark: cluster computing with working sets, in: 2nd USENIX Conf. on Hot Topics in Cloud, Computing, HotCloud'10, p. 10.
[7] Jaliya Ekanayake, Hui Li, Bingjing Zhang, Thilina Gunarathne, Seung-Hee Bae, Judy Qiu, Geoffrey Fox, Twister: A Runtime for Iterative MapReduce," The First International Workshop on MapReduce and its Applications (MAPREDUCE'10) - HPDC2010
[8] Y. Bu, B. Howe, M. Balazinska, M.D. Ernst, HaLoop: efficient iterative data processing on large clusters, in: 36th International Conference on Very Large Data Bases, Singapore.
[9] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M.J. Franklin, S. Shenker, I. Stoica. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing, NSDI 2012,
[10] S. Sehgal, M. Erdelyi, A. Merzky, S. Jha, Understanding application-level interoperability: Scaling-out MapReduce over high-performancee Grids and clouds, Future Generation Computer Systems 27 (2011) 590–599.
[11] AppEngine-MapReduce open-source library http://code.google.com/p/appengine-mapreduce/
[12] Y.Tabaa, A.Medouri, Cloud Computing for education, an opportunity for Universities, EMSCE2011 conference, May19_20 2011 Algeziras, Spain.
[13] A.I. Avetisyan, et al, Open Cirrus : a global cloud computing testbed, Computer 43 (4) (2010) 35-43
[14] A.Bagherinia, et al, A Cloud Computing Collaborative Architecture Model, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 3, No 1, May 2012
[15] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica. Mesos: A platform for fine-grained resource sharing in the data center. Technical Report UCB/EECS-2010-87, EECS Department, University of California, Berkeley, May 2010.
[16] B. Hindman, A. Konwinski, M. Zaharia, and I. Stoica. A common substrate for cluster computing. In Workshop on Hot Topics in Cloud Computing (HotCloud) 2009, 2009.
[17] Scala programming language, http://www.scala-lang.org/
[18] Xen Cloud Platform Project, http://www.xen.org/products/cloudxen.html
[19] OpenStack Open source software http://www.openstack.org/
[20] YARN branch of the Spark repository, https://github.com/mesos/spark/tree/yarn

Yassine Tabaa received in 2005 the computer engineer degree from the National School of Applied Sciences in Tangier, Morocco. Currently, he is a Ph.D Student in the LaSIT Laboratory (Information and Communication Systems Laboratory) at Abdelmalek Essaadi University. His interests focus on Distributed Computing, Cloud Computing and eLearning.

Abdellatif Medouri studied applied physics at the University of Granada in Spain, he got his Ph.D. in signal processing from the University of Granada, Spain (1995). Currently, he is the vice-head of polydisciplinary faculty and lead of Modeling and Information Theory team. His main research interests include cloud computing and new technologies for education.

Copyright (c) 2012 International Journal of Computer Science Issues. All Rights Reserved.