HPC (High-Performance the Computing) for Big Data on Cloud: Opportunities and Challenges

Mohamed Riduan Abid

***Abstract*—Big data and Cloud computing are emerging as new promising technologies, gaining noticeable momentum in nowadays IT. Nowadays, and unprecedentedly, the amount of produced data exceeds all what has been generated since the dawn of computing; a fact which is mainly due to the pervasiveness of IT usage and to the ubiquity of Internet access. Nevertheless, this generated big data is only valuable if processed and mined. To process and mine big data, substantial HPC (high-performance computing) power is needed; a faculty which is not that affordable for most, unless we adopt for a convenient venue, e.g., cloud computing. In this paper, we propose a blue print for deploying a real-world HPC testbed. This will help simulating and evaluating HPC relevant concerns with minimum cost.**

Indeed, cloud computing provides the unique opportunity for circumventing the initial cost of owning private HPC platforms for big data processing, and this by providing HPC as a service (HPCaaS). In this paper, we present the subtleties of a synergetic "fitting" between big data and cloud computing. We delineate opportunities and address relevant challenges. To concretize, we advocate using private clouds instead of public ones, and propose using Hadoop along with MapReduce, on top of Openstack, as a promising venue for scientific communities to own research-oriented private clouds meant to provide HPCaaS for Big data mining.

***Index Terms*—High-performance computing, cloud computing, big data, Hadoop.**

I. Introduction

Big data and cloud computing are emerging as new promising IT fields that are substantially changing the way humans dealt with data forever. During the last decade, data generation grew exponentially. IBM estimated data generation rate to 2.5 quintillion bytes per day, and that 90% of the data in the world today has been generated during the last two years [1].

In the last decade, IT witnessed a significant boost in the ubiquity of Internet access especially with the recent advances in mobile networking technologies (e.g., Wi-Fi, Wi-Max, Bluetooth, RFID, 3G, 4G) and smart phones (e.g., Android, iOS). This substantially contributed to the exponential growth of data generation since users can easily access the Internet wherever, whenever, and using whatever device. This increase in data size, variance, and frequency is referred to by "Big Data".

However, this big data is of no value if not processed and mined. To reach this end, significant HPC is needed. Most importantly, the HPC service should be scalable because more and more "users" are becoming interested in big data processing and mining. In fact, to nowadays organization, HPC is becoming an essential part of business success.

Scalability was not the issue with the preceding Grid computing technology where communities (e.g., universities, research labs, organizations) were sharing HPC resources. The issue, instead, was the heterogeneity of the different components of the Grid, and the resulting interfacing complexities. This resulted in the need of a middleware layer masking the heterogeneity. Cloud computing emerges as a potential information technology, inheriting much of the Grid computing paradigm, mainly in terms of resource sharing; And unlike Grid computing, it offers scalability while providing "All" as a service, via the Internet/Cloud.

Cloud computing is a paradigm based on providing computing as a utility (e.g., electricity, water, and phone), whereby usage is metered and the client pays only per-use (when using public clouds). The payment model depends basically on the adopted cloud deployment model: With the public deployment model, users pay per-use, whereas with the private model, users pay none, and the overall cost of maintaining and running the cloud is taken care of by the authority owning the cloud, e.g., research communities. There are plenty of services offered by the Cloud. Still, according to NIST [2] three of them constitute the fundamental ones: SaaS (Software as a Service), PaaS (Platform as a Service), and IaaS (Infrastructure as a Service).

A synergetic fitting rises between cloud computing and data mining. On one hand, big data is in crucial need of "scalable" HPC services; on the other hand cloud computing has been tailored to provide computing services. Regarding big data processing and mining, the cloud offers HPCaaS (High-Performance Computing as a Service). The latter is a variation of the NIST's defined IaaS.

II. HPCaaS

High Performance Computing (HPC) has been around since the dawn of computing, and it was used to solve and analyze complex problems requiring substantial compute power in terms of both processing and storage.

The first form of HPC used supercomputers and mainframes in the late 60's. These were very expensive and restricted to big companies. To circumvent the expensiveness of owning supercomputers and mainframes, cluster computing emerged at a later stage. In 1990, Donald Becker succeeded in providing HPC by connecting off-the-shelf desktop computers. This gave birth to the well-known

Manuscript received November 6, 2014; revised March 1, 2015.

Mohamed Riduan Abid is with Alakhawayn University, Ifrane, Morocco (e-mail: R.Abid@aui.ma).

Beowulf clusters [3]. Afterwards, Grid computing emerged as the common means of HPC among scientific community.

These former HPC forms faced a set of challenges that consist in peak demand, high capital, and high expertise to acquire and operate the HPC [4]. To deal with these issues, HPC experts have leveraged the benefits of new technology trends including, cloud technologies, parallel processing paradigms and large storage infrastructures. Merging HPC with these new technologies lead to a new HPC model, called HPC as a service (HPCaaS).

HPCaaS is an emerging computing model where end users have on-demand access to pre-existing needed technologies that provide high performance and scalable HPC computing environment [5]. HPCaaS provides unlimited benefits because of the better quality of services provided by the cloud technologies, and the better parallel processing and storage provided by, for example, Hadoop Distributed System [6] and MapReduce [7], [8].

There are plenty of HPCaaS providers in the market. An example of HPCaaS provider is Penguin Computing [9] which has been a leader in designing and implementing high performance environments for over a decade. Nowadays, it provides HPCaaS with different options: On-Demand, HPCaaS as private services and hybrid HPCaaS services. Amazon Web Services is also an active HPCaaS in the market; it provides simplified tools to perform HPC over the cloud. HPCaaS on AWS is currently used for Computer Aided Engineering, molecular modeling, genome analysis, and numerical modeling across many industries including Oil and Gas, Financial Services and Manufacturing. Other leaders of HPCaaS include Windows Azure HPC and Google Compute Engine.

III. Big Data and Cloud Computing: Opportunities

There is a consensus about the potentialities of Cloud computing as a new emerging IT. Topworld-wide IT companies (e.g., Amazon, Google, IBM, Microsoft and Oracle) already enrolled into a tough competitive race towards gaining the best "positions" in the promising global market of cloud computing. Those who have seen Larry Ellison'2009 (Oracle CEO) tirade on cloud computing as nothing other than a hyperbole may be surprised to the see that Oracle now provides pay-per-use services in the cloud and hardly leveraging its Virtual Box. Indeed, Oracle is hardly striving to gain a better position especially that a major portion of the Database business is quickly migrating to the cloud through DaaS (database as a service). Still, the ordinary SQL (structured query language) and RDBMS (Relational Database Management Systems) models prove to be inadequate for big data. The latter mostly falls in the category of K-V (key-value) pairs, where the compute-extensive RDBMS "join" operations are irrelevant. Besides, big data will not frequently use the ordinary "update", "delete", and "add" RDBMS SQL operations. Last, but not least, the semi-structured and unstructured data sets constitute the two fastest growing big data types in the digital universe. The processing and mining of these two data types will not be possible with traditional RDBMS.

In this context, NoSQL (Not Only Structured Query Language) [10] is emerging as the alternative towards transition from RDBMS to non-relational databases.

We classify the major opportunities for big data in cloud computing into four main ones:

1) *Supporting non-Relational Data Processing Models*: Thanks to the abundance of storage and processing power, the cloud can provide the ideal infrastructure for storing big data, and supporting relevant HPC tools.
2) *Initial Cost Circumventing*: To own an HPC capability for big data mining, huge investments are needed; in other words one definitely needs a "Data center" whose size depends on "how much" big data will processed, and "how many" big data jobs are to run in parallel. With cloud computing, we can start storing and mining our big data with an initial cost of zero.
3) *Maintenance and Running Costs*: Unlike with data centers, where qualified personnel are needed (e.g., technicians and engineers) to run and maintain the infrastructure, with cloud computing we do not.
4) *Elasticity*: With cloud computing, a user can ask for the exact amount of HPC processing he needs (usually counted by the number of vCPUs). If more is needed, users can ask for it and get it instantly. Besides, the user can always revert back to the initial settings whenever needed, e.g., as a result of a sharp decrease in demand.

IV. Big Data and Cloud Computing: Challenges

There are plenty of challenges to address regarding the use of cloud computing for big data mining. Still, we delineate three as the major ones:

1) *Dependability*: With Cloud computing, you own nothing. Since "All" is taken care of by the cloud service provider, then "All" is "owned" by the provider.
2) *Privacy*: Since big data needs to be migrated to the cloud provider's servers in order to be processed, it can stay there forever, e.g., by copying it. Furthermore, the algorithms to run on the big data, and the relevant results, can always be "used" by a third part unless we use advanced encryption methods, and send data in an encrypted form.
3) *Mapping Mining Algorithms into MapReduce Jobs*: Most HPC platforms are opting for the MapReduce [7], [8] programming whereby mining algorithms need to be transmuted into a set of independent jobs that run in parallel. The MapReduce programming model is different from the usual programming models, e.g., object-oriented or structured. Thus, the transmutation of these algorithms to fit in the MapReduce model would constitute a major challenge as well.

V. Private HPC Clouds: the Best Venue for Scientific Communities?

To mainly address the challenges of *dependability* and *privacy*, we advocate opting for the private cloud deployment model. In the latter, an authority (e.g., research community, universities consortium) has owns and runs the HPC infrastructure, and each member in the community/consortium would participate by a portion of the required infrastructure(e.g., servers, engineers). By sharing the resources, the community members can reach an optimal

and efficient use of the HPC infrastructure.

In this paper, we propose the use of Hadoop [6] on top of the open-source Openstack cloud computing platform [11]. Besides, we recommend adopting the MapReduce programming model [7], [8] to run big data processing and mining tasks, see Fig. 1.

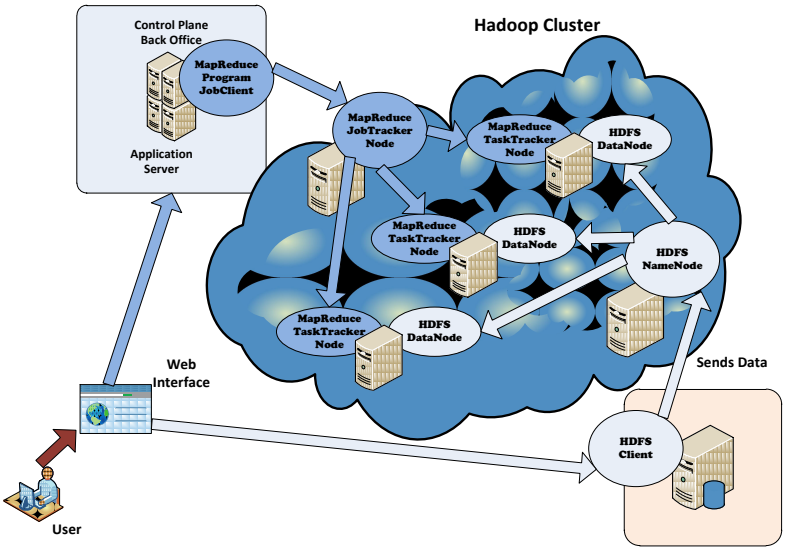

Fig. 1. The general architecture of the HPC service in a private cloud.

Fig. 1 depicts the general architecture of the proposed infrastructure. By connecting to the private cloud using the Internet (http), users can submit their jobs and data via a web interface. The latter implements the relevant business logic, and decides on the appropriate action to undertake. In fact, the web server/interface plays the role of a proxy server which forwards users "requests" using two main flows:

1) Towards the Hadoop HDFS (Hadoop Distributed File System) *Client* which is responsible of establishing communication with the underlying Hadoop HDFS file system. The HDFS client forwards the data to the appropriate HDFS *Namenode*. The latter is responsible for dividing the big data into chunks, replicating them, and keeping track of their location. The selection of the appropriate locations (i.e., HDFS *Datanodes*) depends mainly on load-balancing, i.e., keeping balanced loads (in terms of free storage capacity) between the different HDFS *Data nodes*.
2) Towards the Application server which runs the MapReduce *Jobclient* entity. The latter is responsible for establishing communication with the Hadoop MapReduce platform, and contacting the appropriate *Jobtracker* node which monitors the tracking of submitted jobswhile on the run. The *Jobtracker* divides jobs into relevant sub-tasks and assigns them to specific individual *Tasktracker* nodes depending on load-balancing and data locality. The latter stipulates that a job's sub-tasks should run on the HDFS *Datanode* containing the data in order to avoid the expensive cost of moving big data over the network.

VI. Real-World Hadoop Testbed Installation

This section presents a real-world deployment of a Hadoop cluster that has been deployed to simulate the environment of a web search engine.

This Hadoop cluster is constructed using:

1) Ubuntu Linux 12.04 LTS ,Ubuntu Linux 11.10 and Ubuntu Linux 13.04
2) Hadoop 1.2.1

A. Cluster Structure

We built a multi-node cluster using four slave machines having (Hostname, IP Address) as follows:

- *(node1, 10.50.1.9)-(node2, 10.50.1.199)-(node3, 10.50.1.139)-(node4, 10.50.1.242)*

One master node which has as (Hostname, IP Address):

- *(master, 10.50.1.126)*

B. Run Single Node Hadoop Cluster

Before running the Multi-Node Hadoop cluster, we started by setting up five single-node Hadoop ones. We describe here the detailed steps for setting up this pseudo-distributed single-node Hadoop cluster, running on Ubuntu Linux machines.

As a prerequisite, Hadoop requires a Java installation. We installed Java 1.6.

1) Creating a Hadoop system user

We used a special Hadoop account to run our cluster. The purpose is the separation of the Hadoop installation from other software applications running on the same machine.

1) Create a special group called hadoop
- *sudoaddgrouphadoop*

2) Create user hduser and add him to the group hadoop
- *Sudoadduser—ingrouphadoophduser*

2) Configuring SSH

Hadoop requires SSH access to access and manage the nodes. For our single-node setup of Hadoop, we need to configure SSH access to the *localhost* for our Hadoop user called *hduser*.

We started by setting up *SSH* and we ran it on the five machines and configured it to allow *SSH* public key authentication:

1) Install SSH client

- *sudo apt-get install openssh-client*

2) Install SSH server

- *sudo apt-get install openssh-server*

Second, we generated an *SSH* key for the Hadoop user (*hduser*)

- *su-hduser*
- *hduser@ubuntu: ssh-keygen -t rsa -P ""*

Third, we enabled SSH access to the local machine with this newly created key.

- *hduser@ubuntu:~catHOME/.ssh/id_rsa.pub>>$HOME/.ssh/authorized_keys*

Afterwards, we tested the *SSH* access by connecting to the *local* machine with the Hadoop user (hduser).

- *hduser@ubuntu:~$ sshlocalhost*

This command also saves the local machine's host key to *hduser* user's *known_host* file for future access.

3) Hadoop installation

We downloaded Hadoop from (*www.apache.org/dyn/closer.cgi/hadoop/core*) and extracted the contents of the Hadoop package to */usr/local*.

1) Create a new directory (aka. *Hadoop)* and move the untared files to it

- *sudo mv hadoop-1.0.3 hadoop*

2) Change the ownership of the new created directory to the Hadoop user (*hduser*)

- *sudochown-R hduser:hadoophadoop*

4) Update $HOME/.bashrc

This file consists of a set of shell commands. It is typically used to change prompts, set environment variables, and define shell procedures. In this step, we appended the following lines to the end of the *$HOME/.bashrc* file of user *hduser*. Open *HOME/.bashrc*file using:

- Sudo apt-get gedit HOME/.bashrc

5) HDFS and MapReduce configuration

a) Hadoop-env.sh

This file contains the environment variables settings used by Hadoop, and it is used to affect some aspects of Hadoop *daemon* behavior, such as where log files are stored and the maximum amount of heap memory used.

The only variable we should change in this file is *JAVA_HOME*, which specifies the path to the Java 1.6 installation used by Hadoop

Open the file using:

- *Sudo apt-get gedit /usr/local/hadoop/conf/hadoop-env.sh*

Add this line to the file

- *export JAVA_HOME=/usr/lib/jvm/java-6-sun*

b) Create the tmpdirectory

In this section we configure the directory where Hadoop stores its data files and the network ports it listens to.

The first step consists on creating the directory and setting up the required ownerships and permissions:

Create the directory:

- *sudomkdir -p/app/hadoop/tmp*

Change the ownership of the directory to the hduser

- sudochownhduser:hadoop /app/hadoop/tmp

Make the file accessible by any user (This is not recommended but for the purpose of testing,we made it accessible by every user to not having any accessibility issue later)

- *sudochmod 777/app/hadoop/tmp*

c) conf/core-site.xml

This file contains configuration information that overrides the default values for core Hadoop properties.

We added the following snippets between the *<configuration></configuration>* tags in the respective configuration XML file:

```
<property>
    <name>hadoop.tmp.dir</name>
    <value>/app/hadoop/tmp</value>
    <description>A base for other temporary
    directories.
    </description>
</property>
<property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:54310</value>
    <description>The name of the default file system.
    A URI whose scheme and authority determine the
    FileSystem implementation.The uri's scheme
    determines the config property (fs.SCHEME.impl)
    naming the FileSystem implementation class. The
    uri's authority is used to determine the host, port,
    etc. for a filesystem.
    </description>
</property>
```

d) conf/mapred-site.xml

This file contains configuration information that overrides the default values for MapReduce configuration properties.

We added the following snippets between the *<configuration></configuration>* tags in the respective configuration XML file:

```
<property>
    <name>mapred.job.tracker</name>
    <value>localhost:54311</value>
    <description>The host and port that the
    MapReduce job tracker runs at. If "local", then
    jobs are run in-process as a single map and reduce
    task.
    </description>
</property>
```

e) conf/hdfs-site.xml

This file contains Configuration settings for *HDFS daemons*: the *namenode*, the *secondary namenode*, and the *datanodes*.

We added the following snippets between the *<configuration></configuration>* tags in the respective configuration XML file:

```
<property>
    <name>dfs.replication</name>
    <value>1</value>
    <description>Default block replication.
    The actual number of replications can be specified
    when the file is created.
    The default is used if replication is not specified in
    create time.
```

</description>
</property>

6) Starting Hadoop installation

The first step to start the Hadoop cluster is to format the Hadoop file system which is implemented on top of the local file system. We do this using the following command:

- *hduser@ubuntu:~$* /usr/local/hadoop/bin/hadoopnamenode-format

The services that must be started in the single node cluster are:

- *Namenode*
- *Datanode*
- *Jps*
- *JobTracker*
- *TaskTracker*

Normally, not all the services have to be started in one machine. However, in a single node cluster all the services are started on the same station.

To start all the services, we use the following command:

- *hduser@ubuntu:~$ /usr/local/hadoop/bin/start-all.sh*

We checked if all the services are running:

- *hduser@ubuntu:/usr/local/hadoop$ jps*

The output looked as:

- *2287 TaskTracker*
- *2149 JobTracker*
- *1938 DataNode*
- *2085 SecondaryNameNode*
- *2349 Jps*
- *1788 NameNode*

7) Testing Hadoop installation

To test the Hadoop installation, we followed the following steps:

1) Creating a directory in the local file system:
- *hduser@ubuntu:~$ mkdir /home/hdusertestfile*
2) Downloading file1.txt and file2.txt to the directory (testfiles)
3) Copying the files from the local HDFS file system:
- *hduser@ubuntu:/usr/local/hadoop$ bin/hadoopdfs-copyFromLocal /home/hduser/testfile /home/hduser/testfiledfs*
4) Running the WordCount example job
- *hduser@ubuntu:/usr/local/hadoop$ bin/hadoop jar hadoop*examples*.jar wordcount /home/hduser/testfiledfs /home/hduser/testfiledfsoutpute*
5) Retrieving the output:
- *hduser@ubuntu:/usr/local/hadoop$ bin/hadoopdfs-cat testfiledfs /home/hduser/testfiledfsoutpute/part-r-00000*

VII. Running Multi-node Hadoop Cluster

After deploying five single node Hadoop clusters, we moved on to deploy the required steps to set up a multi-node Apache Hadoop cluster, using the five nodes configured in the former part.

A. Networking Configuration

The five machines must be able to reach each other over the network. The easiest solution is to put all machines in the same network with regard to hardware and software configuration. In our case, the five machines are connected to a local network using Ethernet.

To make the machines reachable from each other, we updated */etc/hosts* on all machines with the following lines:

- 10.50.1.126 master
- 10.50.1.9 node1
- 10.50.1.199 node2
- 10.50.1.139 node3
- 10.50.1.242 node4

B. Establish SSH Access to Slaves

In this step, we are making the *hduser* on the *master* able to connect to the *hduser* on the *slaves* machines , this is done by adding the hduser@master's public SSH key (which should be in $HOME/.ssh/id_rsa.pub) to the authorized_keys file of hduser@slave (in this user's $HOME/.ssh/authorized_keys). This is done using this command on the master:

- hduser@master:~$ ssh-copy-id-i $HOME/.ssh/id_rsa.pub hduser@node1
- hduser@master:~$ ssh-copy-id-i $HOME/.ssh/id_rsa.pub hduser@node2
- hduser@master:~$ ssh-copy-id-i $HOME/.ssh/id_rsa.pub hduser@node3
- hduser@master:~$ ssh-copy-id-i $HOME/.ssh/id_rsa.pub hduser@node4

C. Multi-node Configuration

1) conf/masters

The file *conf/masters* file identifies the machines on which Hadoop will start the secondary *NameNodes.*

In the master node, we updated the *file /conf/masters* by adding the line:

Master(which is the hostname of the master node)

a) conf/slaves

The *conf/slaves* file lists the hosts, one per line, where the Hadoop slaves' daemons (*DataNodes* and T*askTrackers*) will be run.

In the master node, we updated the file /conf/slaves by adding the lines:

- node1
- node2
- node3
- node4

b) conf/core-site.xml

We must change the configuration file *conf/core-site.xml* in all machines, in this case we changed the *fs.default.name* parameter (in *conf/core-site.xml*), which specifies the *NameNode* (the HDFS master) host and port.

<property>
<name>fs.default.name</name>
<value>hdfs://master:54310</value>
<description>The name of the default file system. A URI whose scheme and authority determine the FileSystemimplementation. Theuri's scheme determines the config property (fs.SCHEME.impl) naming the FileSystem implementation class. The uri's authority is used to determine the host, port, etc. for a filesystem.

</description>
</property>

c) conf/mapred-site.xml

Finally, we changed the *HDFS replication factor* parameter, which specifies the default block replication. It defines how many machines a single file should be replicated to before it becomes available. We set it to 3.

<property>
<name>dfs.replication</name>
<value>3</value>
<description>Default block replication. The actual number of replications can be specified when the file is created. The default is used if replication is not specified in create time.
</description>
</property>

d) Formatting the HDFS Filesystem via the Namenode

To format filesystem we used the command:

- hduser@master:/usr/local/hadoop$ bin/hadoopnamenode–format

e) Starting the multi-node cluster

To start the multi-node cluster we need to:

1) To start the HDFS daemons: the *NameNodedaemon*is started on master, and *DataNode daemons* are started on all slaves (node1-node2-node3-node4)
2) To start the *MapReduce daemons*: the *JobTracker* is started on master, and *TaskTracker* daemons are started on all slaves.

On the master Node, We run the command:

- hduser@master:/usr/local/hadoop$ bin/start-dfs.sh

This command starts the *NameNode daemon* on the master node and the *DataNode daemon* on the slaves.

On the *master* node, we run the command:

- hduser@master:/usr/local/hadoop$ bin/start-mapred.sh

This command starts the *JobTracker daemon* on the *master* node and the *TaskTracker daemon* on the *slaves*.

The steps to run a *MapReduce job* in a Multi-Node Hadoop cluster are the same as a Single-Node one.

The *following*s napshot shows in which node the different tasks are run (Fig. 2).

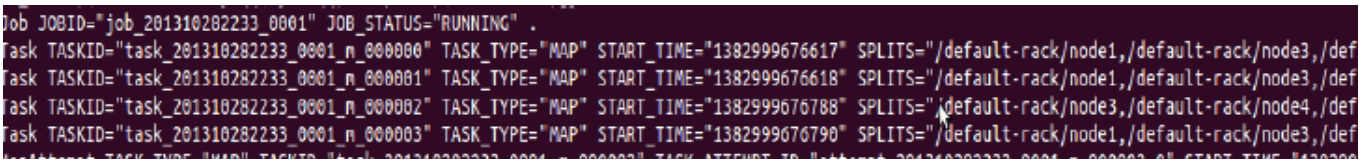

Fig. 2. Jobs status and corresponding tasks and nodes.

VIII. Conclusion and Future Work

In this paper, we presented an overview of a natural fitting between big data and cloud computing. We delineated relevant opportunities and challenges. We delineated an architecture for deploying an HPC private cloud, as a testbed meant basically to simulate and study HPC relevant issues, and highlighted its main components, e.g., Hadoop, Openstack, and MapReduce.

Furthermore, we delineated a step-by-step blueprint for deploying Hadoopon real-world cluster. The latter can be used to extensively simulate MapReduce Jobs.

As a future work, we are planning to deploy a real-world implementation of the proposed architecture. The latter can be used by most scientific communities in need of big data processing and mining.

References

[1] IBM. Big data at the speed of businesses. [Online]. Available: http://www-01.ibm.com/software/data/bigdata/
[2] P. Mell and T. Grance, "The NIST definition of cloud computing," *NIST Special Publication 800-145 (Draft).*
[3] T. Sterling, *Beowulf Cluster Computing with Linux*, Publisher: MIT Press, 2001.
[4] J. Ernstein and K. McMahon, "Computing on demand — HPC as a service: High performance computing for high performance business," White Paper, Penguin Computing & McMahon Consulting.
[5] Y. Xiaotao, L. Aili, and Z. Lin, "Research of high performance computing with clouds," in *Proc. International Symposium Computer Science and Computational Technology*, 2010, pp. 289-293.
[6] ApacheHadoop. [Online]. Available: http://hadoop.apache.org/
[7] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. the 6th USENIX OSDI*, 2004, pp. 137-150.
[8] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[9] Penguincomputing. [Online]. Available: http://www.penguincomputing.com/services/hpc-cloud/pod
[10] K. Grolinger, W. A. Higashino, A. Tiwari, and M. A. M. Capretz, "Data management in cloud environments: NoSQL and NewSQLdata stores," *Journal of Cloud Computing: Advances, Systems and Application*, Springer Open, vol. 2, 2013.
[11] The Openstack Cloud Software. Open source software for building private and public clouds. [Online]. Available: http://www.openstack.org/

M. R. Abid received a PhD degree in computer science in 2010 from Auburn University (USA). He received the Excellence Fulbright Scholarship in 2006. He is currently an assistant professor of computer science at Alakhawayn University, Morocco. His recent research interests include cloud computing, IoT (internet of things), and wireless mesh networking.