Annals of the University of Craiova, Mathematics and Computer Science Series
Volume 39(2), 2012, Pages 322–333
ISSN: 1223-6934

Running Hadoop applications in virtualization environment

Mirel Coşulschi, Mihai Gabroveanu, and Adriana Sbîrcea

Abstract. Large amounts of data can be found today in every domain such as data resulted as output from various processes like e-commerce transactions, banking transactions or credit card transactions, web navigation concretized into web logs. Also we have to mention the huge amount of (personal) data stored in social networks like Facebook, Flickr. In this paper we have implemented a map-reduce application under Hadoop framework, and we have performed various tests in order to obtain the best performance in a virtualization environment.

2010 Mathematics Subject Classification. Primary 68W15; Secondary 94C15.
Key words and phrases. MapReduce application, Apache Hadoop, virtualization, big data.

1. Introduction

Mining of massive datasets or *big data analytics* are two frequently encountered hot topics today. For example there are many conferences, presentations, demos etc. all having in the center of their universe mathematical methods and algorithms for solving new problems arisen from analysing big data.

These topics cover the process of analyzing very large amounts of data with emphasis on data mining and machine learning methods. But why massive datasets or big data? Where did this come from?

`Big data` describes datasets that grow so large that they become unpractical to be processed by traditional tools like database management systems, content management systems, advanced statistical analysis software etc.

Large amounts of data can be found today in every domain such as data resulted as output from various processes like e-commerce transactions, banking transactions or credit card transactions, web navigation concretized into web logs. Also we have to mention the huge amount of (personal) data stored in social networks like Facebook[1], Flickr[2]. Google was processing about 20 PB per day in 2008 while Facebook announced in April 2009 that on the servers of the company were stored 2.5 PB.

The reason why they came into the attention of IT community is that the infrastructure to handle these datasets has become more affordable due to cloud computing and *map/reduce* based open-source frameworks. The vision of everyday people inspecting large quantities of data, interacting easily with graphical interfaces of software applications that handles data mining tasks has excited the practitioners.

Received November 30, 2012.
The presented work was supported by the grant CNCSIS 55/2008 funded by Romanian National Council of Academic Research (CNCSIS).
This work is the extended version of the paper [4] presented at the 12th International Conference on Artificial Intelligence and Digital Communications - AIDC 2012.

[1] `http://www.facebook.com/`

[2] `http://www.flickr.com/`

Those tools have to leverage people's ability for answering big questions by inference through large amounts of data. Data centers have emerged as distributed information systems for massive data storage and processing providing many online applications and infrastructure services through its large number of servers [6].

The problem with this data is not the complexity of the processes involved rather than the quantity of data and the rate of collecting new data. Two examples are rather illustrative of this phenomenon:

- *EBay* on April 2009 was using two data warehouses, Terradata and Greenplum, the former having 6.5 PB, 17 trillion records already stored, and collecting more than 150 billion new records/day, leading to an ingest rate well over 50 terabytes /day[3].
- *Facebook* on May 2009 estimates an amount of 2.5 PB of user data, with more than 15 TB of new data per day[4].

2. Cloud computing

Cloud computing is, according to a special document published under the auspices of NIST[5], "*a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*" [17].

Another definition can be encountered in an official document emerged from *EU Comission*[9]:

> a 'cloud' is an elastic execution environment of resources involving multiple stakeholders and providing a metered service at multiple granularity for a specified level of quality (of service).

As a conclusion, cloud computing can be seen as a new approach for using IT services with the advantage of accessing them with less costs and usage complexity, maintaining in the same time a high level of scalability and reliability.

The best service behavior from a client point of view is a cloud computing service offering the access to resources (hardware and software) with the appearance to him as being infinite. This requires an automated allocation and management of resources in order to handle the requests. Virtualization is a solution to this problem becoming common that scales well. Also, for security reasons virtualization is a strong mechanism.

When running multiple virtual machines in cloud computing, the CPUs, main memory, network devices and hard-disks have to be shared by the VMs launched on the same physical server. Although the CPUs and main memory can be shared well, the bottlenecks occur at the network and disk I/O level. Here the engineers of the hardware architecture and operating systems developers still have to work in order to improve the virtualizations of interrupts and I/O channels.

The main *deployment models* for the clouds are:

(1) *Public cloud* - in this deployment model, cloud's resources are made available for general public;

[3]source `http://www.dbms2.com/2009/04/30/ebays-two-enormous-data-warehouses/`

[4]`http://www.dbms2.com/2009/05/11/facebook-hadoop-and-hive/`

[5]National Institute of Science and Technology - `http://www.nist.gov/index.html`

(2) *Community cloud* - community cloud is operated by a community allowing the member organisations to share the cloud's infrastructure. The member organisations have in common the same interests, objectives, goals.
(3) *Hybrid cloud* - is an inbetween model from *public cloud* to *private cloud*;
(4) *Private cloud* - the service is available only for a single entity, being developed and maintained by the same actor.

The cloud infrastructure lies on three *service models*:

(1) *Infrastructure as a Service* (`IaaS`) - cloud providers offer the infrastructure for cloud users, usually as virtual machine [9] (see *Amazon S3*[6], *SQL Azure*[7], *Amazon EC2*[8]);
(2) *Platform as a Service* (`PaaS`) - cloud providers offers to cloud users an entire development platform and beyond composed from operating system, programming language environment, database, web server (see *Google App Engine*[9], *Windows Azure*[10]);
(3) *Software as a Service* (`SaaS`) - cloud providers offer implementations of various software applications to cloud users (see *Google Docs*[11], *Microsoft Office 365*[12]).

Some researchers argue [1] that even the cost of pay-as-you-go service is more expensive than the amortization cost of a computer server over a certain period, the benefits is overwhelmed by the transfer of risk (overprovisioning or underprovisioning) and the property of service elasticity.

Overprovisioning appears when the service operators estimate a certain workload (resource usage) for a period of time and, from some reasons, like seasonal demand, the resources remain underutilized with certain amount of money loss - the amount of money spent for renting resources which were partial used.

Underprovisioning is characterized by a saturation of the resources as a result of demand burst due to some external events. In such situation resources are not able to fulfill users requests, concretized in users rejected which with a high probability will not come back again. It is difficult to assess losses produced by the excess users turned down, because we cannot know with certain precision the number of users that will not abandon the service.

The elasticity is the property of adding or removing a large quantity of hardware or software resources, even at a fine grain level.

In conclusion, the characteristics that make cloud computing so attractive are short-term usage, no upfront cost and infinite capacity on demand [1].

2.1. Virtualization. Among various kind of virtualization, a hosted Virtual Machine is installed as an application program, the virtualizing software being executed on top of the existing operating system. This operating system is responsible for providing virtualizing software with device drivers and low-level services. *VMware GSX server* is an implementation of a hosted VM [19], [20].

The architecture of *VMware ESX server* is different from the architecture of *VMware GSX server*. The former provides a virtual intermediate layer between the hardware and the virtual machines while the later provides a virtual layer between the

[6] `http://aws.amazon.com/s3/`
[7] `http://www.windowsazure.com/en-us/home/features/data-management/`
[8] `http://aws.amazon.com/ec2/`
[9] `https://developers.google.com/appengine/`
[10] `http://www.windowsazure.com`
[11] `http://docs.google.com`
[12] `http://www.microsoft.com/office365/`

host operating system and the virtual machines. This can be achieved by having its own kernel, called *vmkernel*, with microkernel design, and as a consequence the ESX server run directly on the host server without any third-party operating system. The virtual layer introduced by ESX server abstracts the system hardware into a pool of logical resources that can be allocated to the VMs. The computational performances of VMs running under ESX server are superior to the VMs running under GSX server.

3. Hadoop

Hadoop is an open-source project from *Apache Software Foundation*, its core consisting of *MapReduce* programming model implementation. This platform was created for solving the problem of very large amounts of data processing, a mixture of complex and structured data. It is used with predilection in the situation when data analytics applications requiring a large number of computations have to be executed.

The key characteristics of Hadoop is the property of enabling the execution of applications on thousands of nodes and pentabytes of data. A typical enterprise configuration is composed from tens or thousands of physical or virtual machines interconnected through a fast network. The machines run a POSIX compliant operating systems and a Java virtual machine with a version newer than 6 has to be installed and functional.

Doug Cutting was the creator of Hadoop framework, two papers published by Google's researchers having decisive influence on his work: *The Google File System* [10] and *MapReduce: simplified data processing on large clusters* [5]. In 2005 he started to use an implementation of MapReduce in *Nutch*[13], an application software for web searching, and very soon the project became independent from *Nutch* under the codename *Hadoop*, aiming to offer a framework for distributed running many jobs within a cluster [27].

The language used for the Hadoop framework development is Java. The framework is providing the users a Java API for developing MapReduce applications. In this way Hadoop becomes very popular in the community of scientists whose name is linked to the big data processing. Yahoo! is an important player in this domain, this company having for a longer period a major role in the design and implementation of the framework. Another big player is Amazon[14], company currently offering a web service[15] based on Hadoop that is using the infrastructure provided by the *Elastic Compute Cloud - EC2*[16]. Meanwhile, Hadoop becomes very popular, and for sustaining this allegation it is sufficient just to mention the presence of other important players like IBM, Facebook, The New York Times, Netflix[17], Hulu[18], in the list of companies deploying Hadoop based applications [11].

There is a quite large application domain for Hadoop. Some examples includes online e-commerce applications for providing better recommendations for clients looking after certain products or finance applications for risk analysis and evaluation with sophisticated computational models whose results cannot be stored in a database [6].

Basically, Hadoop can be executed on each major OS: Linux, Unix, Windows, Mac OS. The only platform officially supported for production is Linux.

---

[13] `http://nutch.apache.org/`
[14] `http://www.amazon.com`
[15] `http://aws.amazon.com/elasticmapreduce/`
[16] `http://wiki.apache.org/hadoop/AmazonEC2`
[17] `https://signup.netflix.com/global`
[18] `http://www.hulu.com/`

Hadoop framework can be used in one of the following modes:

- *Standalone* - there are no daemons launched, all processes being executed on a single virtual machine. It is used for development, testing and debug;
- *Pseudo-distributed* - Hadoop daemons are running on the local machine, but on separate virtual machines, simulating a real cluster. The inter-machine communications represented by network traffic are absent;
- *Distributed* - Hadoop daemons are executed on a cluster.

The framework is implementing a master-slave architecture used both for distributed data storage and for distributed computations.

Hadoop framework running on a completely configured cluster is composed of a set of daemons or internal modules executed on various machines from the cluster. Those daemons have specific tasks, some of them running only on a single machine, others having instances running on several machines. The Hadoop daemons are [13], [22]:

- *NameNode* - the most important daemon. NameNode daemon is the HDFS principal controller, coordinating the slave DataNode daemons for performing I/O operations. This is the only critical failure point from the cluster;
- *DataNode* - each slave machine from the cluster will host a DataNode daemon reading and writing HDFS blocks from the files stored in the local file system;
- *Secondary NameNode - SNN* - has the role to monitor the health status of HDFS cluster. Each cluster has one NameNode and a Secondary NameNode running on separate machines, the main difference between the two daemons is that the last one does not receive realtime changes inside the HDFS;
- *JobTracker* - ensures the synchronization between the application and the Hadoop framework. JobTracker establishes the execution plan in concordance with the files which will be processed, assigns nodes to each task and monitors all running tasks. Each Hadoop cluster has a single JobTracker daemon, running on a server as a master node;
- *TaskTracker* - is responsable of performing the tasks assigned by JobTracker. Each slave node has a single TaskTracker that can start several JVMs for running in parallel more tasks.

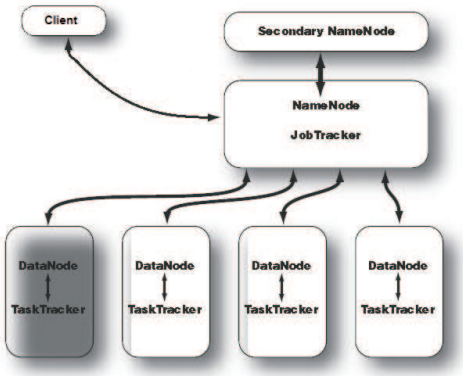

FIGURE 1. Hadoop cluster architecture

A sample Hadoop cluster architecture is depicted in Figure 1. For a small cluster, *Secondary NameNode* daemon can be hosted on a slave node, meanwhile, on a large

cluster, it is customary to separate the *NameNode* and *JobTracker* daemons on different machines. In Figure 1, we have a master node running *NameNode* and *JobTracker* daemons and an independent node hosting the *Secondary NameNode* daemon, just in case of failure for master node.

Each slave machine hosts two daemons, a *DataNode* and a *TaskTracker*, in order to run computational tasks on the same node where their input data are stored.

4. Hadoop cluster

We have decided to use a preconfigured *VMware* image[19] from a Yahoo's tutorial [23], [26].

There are three supported modes for running a Hadoop cluster[20]:

- local (standalone) mode;
- pseudo-distributed mode;
- fully-distributed mode.

We have tested our application in all three modes in order to dose the grade of the configuration difficulty from easy to complex. For the first mode, the local (standalone) one, Hadoop is configured by default to run in a non-distributed mode. We have used this mode for debugging the application.

The second mode, the pseudo-distributed mode, is more close to the *real*-like behavior, Hadoop running also as a single node. The major difference between those two initial modes relates to the observation that in the former mode there is only a Java process running in the Java Virtual Machine while in the latter all daemons run in separate processes, each one for a distinct Java Virtual Machine. The pseudo-distributed name comes from the fact that although all processes have separate space for memory, they are running on the same machine.

The fully-distributed mode is the one recommended for processing large amounts of data in a properly tuned cluster. In order to perform computation on various quantities of big data analytics in reasonable time we have to proper adjust the size and the resources' allocation. This can be concretized by choosing a proper number of machines with specified RAM memory and space on hard-disk.

4.1. Design. All machines in the cluster have been prepared in the same way in order to reduce the maintenance. They have identic hardware and software configuration with exception of a few characteristics related to the role and identity details for each node, which must be different.

We have allocated 6 *virtual machines* as follows: the name of each machine is `hadoopx` where `x` is a digit from the set $\{1, \ldots, 6\}$, while the associated IP address is $10.100.20.X$, $X \in \{1, \ldots, 6\}$.

Table 1. Allocation of daemons inside the specified cluster

Component	Machines host names					
	hadoop1	hadoop2	hadoop3	hadoop4	hadoop5	hadoop6
MapReduce monitor	X					
MapReduce worker		X	X	X	X	X
HDFS monitor	X					
HDFS worker		X	X	X	X	X

[19] `http://ydn.zenfs.com/site/hadoop/hadoop-vm-appliance-0-18-0_v1.zip`

[20] `http://hadoop.apache.org/common/docs/r0.20.2/quickstart.html`

The reason that stands up for deciding where will be hosted each type of daemon (*worker* or *monitor*) relates to the facts that while a *monitor* is using more RAM for monitoring and coordination activities of the *workers*, a *worker* is more computationally oriented and due to this it is using more CPU and I/O bandwidth, with a reduced demand for RAM due to the distributed architecture of the cluster.

4.2. Installation and configuration.

Remark 4.1. *In order to proper configure the cluster used for application testing, it is better to stop first the daemons of Hadoop framework, daemons that were automatically launched at startup by the default configuration of the local (standalone) mode.*

This task can be performed with the help of the following commands `stop-mapred.sh` *that stops MapReduce daemons and* `stop-dfs.sh` *that stops HDFS daemons.*

Remark 4.2. *A first problem encountered during the configuration process of the machines was related to the operating system limitation. We recall that we have used* `Ubuntu Linux 8.04.1`[21]*. It was necessary to increase the limit of the maximum number of simultaneous open files and maximum concurrent processes.*

In order to see maximum number of simultaneous open files log as `hadoop-user` *and execute the following command:*

```
hadoop-user$ ulimit -n
```

and to see maximum concurrent processes execute:

```
hadoop-user$ ulimit -u
```

In order to change these limits log as `root` *and edit* `/etc/sysctl` *file and add (or update) the following config options:*

```
#increase the system-wide maximum number of open files
fs.file-max = 32768
```

and the `/etc/security/limits.conf` *file and add the following:*

```
#Increase open file limits
hadoop-user     -    nofile  32768
#Increase number of concurrent processes
hadoop-user     -    nproc 32000
```

Also, check if `/etc/pam.d/common-session` *contains the following line:*

```
session required pam_limits.so
```

Reboot system to apply the changes. Now, log as `hadoop-user` *and check the setting values again:*

```
hadoop-user$ ulimit -n
32768
hadoop-user$ ulimit -u
32000
```

The machines are configured to obtain their IP address automatically through **DHCP**[22] in concordance with their **MAC address**[23]. As a direct consequence of this configuration, each time a machine is booting/rebooting it gets the same IP address. Also, the assigned addresses are automatically enlisted in the local DNS server[24].

---

[21] `http://cdimage.ubuntu.com/ports/releases/hardy/release/`

[22] Dynamic Host Configuration Protocol - `http://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol`

[23] Media Access Control - `http://en.wikipedia.org/wiki/MAC_address`

[24] Domain Name System - `http://en.wikipedia.org/wiki/Domain_Name_System`

As stated before, we have chosen to work with 6 virtual machines running under the *VMware ESX Server*. VMware ESX Server is a thin software layer designed to multiplex hardware resources efficiently among virtual machines [21]. A machine was configured as *master*, the others running as *slaves*.

The machine named `hadoop1` was configured as *master*, while `hadoopX` with $X \in \{2, 3, 4, 5, 6\}$ as *slaves*. On each of this machines the files `/etc/hostname` and `/etc/hosts` were modified accordingly.

For example, for the `hadoop1` machine the above mentioned files have the following configurations:

- the file `/etc/hostname`:
  ```
  hadoop1
  ```
- the file `/etc/hosts`:
  ```
  127.0.0.1       hadoop1.localdomain localhost

  10.100.20.1     hadoop1
  10.100.20.2     hadoop2     hadoop2.localdomain
  10.100.20.3     hadoop3     hadoop3.localdomain
  10.100.20.4     hadoop4     hadoop4.localdomain
  10.100.20.5     hadoop5     hadoop5.localdomain
  10.100.20.6     hadoop6     hadoop6.localdomain
  ```
  where `10.100.20.X` (with $X \in \{2, 3, 4, 5, 6\}$) are IP addresses for the slaves machines members of the cluster.

The steps required for configuring the Hadoop framework in order to work in fully-distributed mode inside a cluster are:

(1) The first step involves rewriting the files `conf/masters`, `conf/slaves` and `conf/hadoop-site.xml` located in the `conf` subfolder from the Hadoop installation folder.
   - `conf/masters` file:
     ```
     10.100.20.1
     ```
   - `conf/slaves` file:
     ```
     10.100.20.2
     10.100.20.3
     10.100.20.4
     10.100.20.5
     10.100.20.6
     ```
     The last two files were changed only on *master* machine.
   - The contents of the `conf/hadoop-site.xml` file is:
     ```
     <?xml version="1.0"?> <?xml-stylesheet type="text/xsl"
     href="configuration.xsl"?>
     <configuration>
       <property>
         <name>fs.default.name</name>
         <value>hdfs://10.100.20.1:9000</value>
       </property>
       <property>
         <name>mapred.job.tracker</name>
         <value>10.100.20.1:9001</value>
       </property>
       <property>
         <name>dfs.replication</name>
         <value>3</value>
       </property>
     ```

```
  <property>
    <name>mapred.child.java.opts</name>
    <value>-Xmx256m</value>
  </property>
  <property>
    <name>mapred.system.dir</name>
    <value>/hadoop/mapred/system</value>
  </property>
</configuration>
```

This file has to be copied on all machines that belongs to the current cluster.

(2) Next we have to clear the content of the folder `/tmp/hadoop-hadoop-user/`
(3) Optionally it is deleted the contents of the folder `$HADOOP_HOME/logs/`
(4) We have to format the *namenode* (the *namenode* is located on *master* machine):
```
$HADOOP_HOME/bin/hadoop namenode -format
```
(5) The daemons *namenode*, *datanode(s)* and *secondarynamenode* are started with the help of the following command executed on *master* machine:
```
$HADOOP_HOME/bin/start-dfs.sh
```
(6) Also, the *jobtracker* and *tasktracker* daemons can be started with a command executed on the same *master* machine:
```
$HADOOP_HOME/bin/start-mapred.sh
```

Remark 4.3. *In order to check if all daemons were launched without any problems it is recommended to consult the log files:* `datanode.log`, `tasktracker.log` *etc.*

At the URL address `http://hadoop-master-host-name:50070/dfshealth.jsp` one can obtain specific details related to the running of the *namenode* and *datanodes* processes, while at the address `http://hadoop-master-host-name:50030/jobtracker.jsp` can be seen information about the *MapReduce* process evolution.

After the configuration process was finished and the current state was tested with default examples provided together with Hadoop framework, the *master* machine and a *slave* machine were snapshotted in order to preserve them for the situations arisen from a hardware failure or a wrong software installation.

For fine tuning the Hadoop configuration the interested user can read other essential documentation like [25], [24]. We have described with details the above steps, our approach being motivated by the large number of difficulties encountered during the process, even if we have guided our attempt with details from many available tutorials describing the configuration of heterogeneous Hadoop cluster.

5. Experimental results

The application developed obeys *MapReduce* general structure [16] and is composed of two parts whose goal is to compute the Jaccard index [18] and its frequency (implementations details are available in [3]).

The application was tested on small size files, but for unleashing the power of distributed data processing under Hadoop we searched for some real big data test files.

A good choice for us were the data from `WEBSPAM-UK2007` collection[25]. The data set[26] contains a large number of hosts annotated as spam/nonspam, together with

[25]The data was downloaded in May 2007 by the Laboratory of Web Algorithmics, Università degli Studi di Milano, with the support of the DELIS EU - FET research project.

[26]`http://barcelona.research.yahoo.net/webspam/datasets/uk2007/`

the links between pages. Also if there is a webpage A on $host_1$ with a link to a webpage B on $host_2$ then there is a directed link from $host_1$ to $host_2$. So we have a directed graph $G = (V, E)$ with V = the set of hosts while the URL links between two pages from distinct hosts forms an arc (oriented edge) from the first host to the second one ($E = \{(u, v) | \exists$ a webpage $wp_1 \in host_u, \exists$ a webpage $wp_2 \in host_v$ *and* $\exists$ a URL link inside wp_1 *to* $wp_2\}$).

The size of V is 114.529, the vertices being numbered from 0 to 114.528. The input file from the data set has n lines with the following structure, where $|V| = n$:

```
dest1:nlinks1 dest2:nlinks2  ... destk:nlinksk
```

where the $dest1$, $dest2$, ..., $destk$ denote the hosts, and $nlinks1$, $nlinks2$, ..., $nlinksk$ represent the number of links between the current host designated by the number of the current line and the host $destX$. If a host does not have any links to the other hosts, then the corresponding line from the file is empty.

The first set of tests was performed on the entire `WEBSPAM-UK2007` data set.

The application was executed first in pseudo-distributed mode on a single virtual machine running under *VMware ESX* 5.0.

The hardware configuration of the virtual machine consists of 1GB of RAM, a dual core CPU with 1.99 Ghz frequency on each core and 10GB of free space on local hard-disk. The version of the Hadoop framework available on the virtual machine was 0.18.0.

The size of the input file is 16 MB while the size of the output file is 2 GB. The running time for this configuration was 45 minutes and 2 seconds.

The next test was performed on a cluster configured on the same physic machine, the cluster being composed of 2 virtual machines identic with the one from the previous test. Three of the daemons, *NameNode*, *secondary NameNode* and *JobTracker*, are running on one of the machines, the *master*, while the rest, *DataNode* and *TaskTracker*, are running on the second machine, the *slave*.

The time obtained for the same input was 36 minutes and 8 seconds.

Further on, we augmented the cluster with another 2 *slave* machines, each one running two daemons, *DataNode* and *TaskTracker*. The running time on this configuration was 31 minutes and 52 seconds.

From the last two configurations it can be observed that the processing time does not decrease as it was expected - an inverse linear dependency on the number of slaves from the cluster and the running time of the application. We continued to add other 2 *slave* machines and have changed the number of mappers from 2 to 10 and setted 5 reducers for the last job. These changes in the configuration led to a significantly reduction on the running time up to 14 minutes and 13 seconds (see figure 2).

6. Related work

There are many resources available that describe the process of installing and/or configuring a cluster for running Hadoop on multi-node like *Running Hadoop On Ubuntu Linux* [27], *Yahoo tutorial - Managing a Hadoop Cluster*[28], *Cloudera Hadoop Tutorial*[29], *Hadoop on Windows with Eclipse*[30], *Distributed data processing with Hadoop*[31].

[27] `http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/`

[28] `http://developer.yahoo.com/hadoop/tutorial/module7.html`

[29] `https://ccp.cloudera.com/display/DOC/Hadoop+Tutorial`

[30] `http://v-lad.org/Tutorials/Hadoop/00-Intro.html`

[31] `http://www.ibm.com/developerworks/linux/library/l-hadoop-1/index.html?ca=dgr-lnxw57Hadoop-Clusters`

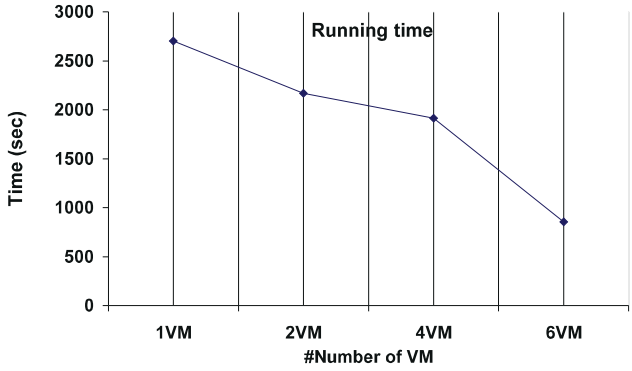

FIGURE 2. The graph with the running time vs number of virtual machines processing input test files obtained from WEBSPAM-UK2007.

In this work we gather, configured, tested and put together all the peculiarities and features needed to run a Hadoop applications in a virtualized cluster. In fact, each node of the configured cluster is running under a virtual environment.

Many similarity measures were defined and studied, the researchers knowing their advantages and disadvantages [18]. One of this measures, the Jaccard similarity index has various applications [2], [15], [14], [8], [12]. Our implementation of the application that computes the Jaccard similarity index is inspired from the work of Bank and Cole[28].

7. Conclusion

Big data did not appear out of nowhere in recent years, companies have collecting somehow their strategic data for a while: e.g. pharmaceutical companies have collected huge amounts of information during their research that are available through their private intranets in local datawarehouse. Surveillance cameras, RFID, all kind of sensors used in physical security systems, all have produced data that could and are stored.

The goal of this paper was to illustrate how Hadoop and MapReduce can be used to calculate Jaccard index for big graphs. We have tried to benefit from the properties of Hadoop framework and performed experiments with all supported modes for running a Hadoop application. Moreover we have installed and configured a cluster in a virtualization environment.

8. Acknowledgements

The presented work was supported by the grant CNCSIS 55/2008 funded by Romanian National Council of Academic Research (CNCSIS).

References

[1] M. Armbrust, A. Fox, R. Griffith, R. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and D. Zaharia, A view of cloud computing, *Communication of the ACM* **53** (2010), no. 4, 50-58.

[2] C. Blundo, E. De Cristofaro and P. Gasti, EsPRESSo: Efficient Privacy-Preserving Evaluation of Sample Set Similarity, *7th ESORICS Workshop on Data Privacy Management (DPM 2012)*, 2012.
[3] M. Coşulschi, M. Gabroveanu and A. Sbîrcea, Implementing MapReduce in virtualization environment: Hadoop Case Study, *TR–2012–1*, 2012.
[4] M Cosulschi, M. Gabroveanu and A. Sbircea, Running Hadoop applications in virtualization environment, in *Proceedings of 12th International Conference on Artificial Intelligence and Digital Communications*, 51-61, 2012.
[5] J. Dean and S. Ghemawat, MapReduce: simplified data processing on large clusters, *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation (OSDI04)* **6** (2004), 137–150.
[6] Z. Ding, D. Guo, X. Chen and X. Luo, Performing MapReduce on Data Centers with Hierarchical Structures, *Int. J. Comput. Commun.* **7** (2012), no. 3, 432–449.
[7] J. Dittrich and J.-A. Quian-Ruiz, Efficient big data processing in Hadoop MapReduce, *Journal Proceedings of the VLDB Endowment* **5** (2012), no. 12, 2014–2015.
[8] S. Engen, V. Grtan and B.-E. Sther, *Estimating similarity of communities: a parametric approach to spatio-temporal analysis of species diversity*, Ecography, 34 (2011), 220–231.
[9] European Commission Expert Group Report, *The Future of Cloud Computing*, January, 2010. `http://cordis.europa.eu/fp7/ict/ssai/docs/cloud-report-final.pdf`
[10] S. Ghemawat, H. Gobioff and S.-T. Leung, The Google File System, *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP03)*, ACM (2003), 29–43.
[11] B. Irving, Big data and the power of Hadoop, *Yahoo! Hadoop Summit*, June 2010.
[12] D. A. Jackson, K. M. Somers and H. H. Harvey, Similarity Coefficients: Measures of Co-Occurrence and Association or Simply Measures of Occurrence?, *The American Naturalist* **133** (1989), no. 3, 436–453.
[13] C. Lam, *Hadoop in Action*, Manning Publications, 2010.
[14] J. Machaj, R. Pich and P. Brida, Rank Based Fingerprinting Algorithm for Indoor Positioning, *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2011, 1–6.
[15] C. M. Mulqueen, T. A. Stetz, J. M. Beaubien and B. J. O'Connell, Developing Dynamic Work Roles Using Jaccard Similarity Indices of Employee Competency Data, *Ergometrika* **2** (2001), 26–37.
[16] J. Lin and C. Dyer, *Data-Intensive Text Processing with MapReduce*, Morgan & Claypool Publishers, 2010.
[17] P. Mell and T. Grance, *The NIST Definition of Cloud Computing*, National Institute of Science and Technology, 2011.
[18] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*, Cambridge University Press, 2012. `http://i.stanford.edu/~ullman/mmds.html` retrieved at
[19] J. E. Smith and R. Nair, The Architecture of Virtual Machines, *Computer* **38** (2005), no. 5, 32–38.
[20] J. Sugerman, G. Venkitachalam and B. H. Lim, Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor, *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, 2001, 1–14.
[21] C. A. Waldspurger, Memory Resource Management in VMware ESX Server, *in Proceedings of the 5th Symposium on Operating Systems Design & Implementation (OSDI02)* 36 (2002), 181–194.
[22] T. White, *Hadoop: The Definitive Guide. Storage and Analysis at Internet Scale*, 3rd Edition, O'Reilly Media / Yahoo Press, 2012.
[23] `http://developer.yahoo.com/hadoop/tutorial/`
[24] `http://hadoop.apache.org/common/docs/stable/`
[25] `http://developer.yahoo.com/hadoop/tutorial/module7.html`
[26] *Hadoop Virtual Image Documentation*, `http://code.google.com/intl/ro-RO/edu/parallel/tools/hadoopvm/index.html`
[27] *Nutch presentations* `http://wiki.apache.org/nutch/Presentations`
[28] J. Bank and B. Cole, Calculating the Jaccard Similarity Coefficient with Map Reduce for Entity Pairs in Wikipedia, `http://www.infosci.cornell.edu/weblab/papers/Bank2008.pdf`

(M. Coşulschi, M. Gabroveanu, A. Sbîrcea) Department of Computer Science, University of Craiova, 13 A.I. Cuza Street, Craiova, 200585, Romania
E-mail address: `mirelc@inf.ucv.ro, mihaiug@inf.ucv.ro, sbirceaadriana@yahoo.com`