Scheduling on Heterogeneous Hadoop System in Eucalyptus Private Cloud
S. Karthikeyan¹ Naresh Sammeta² B. Saravanan³
¹,²Assistant Professor ³Associate Engineer
¹,²Department of Computer Science and Engineering
¹,²RMK College of Engineering Technology, Chennai, Tamilnadu, India ³IGATE Global Solutions, Bangalore, India

Abstract—Cloud Computing is emerging as a new computational paradigm shift. Hadoop-Map Reduce has become a powerful computation model for processing large data on distributed commodity hardware clusters such as clouds. A large number of (heterogeneous) clients use the same Hadoop cluster. Map Reduce (MR) has become a popular programming model for running data intensive application on the cloud. Completion time goals or deadline of map reduce jobs set by users are becoming crucial in existing cloud based data processing environments such as Hadoop. There is a conflict between the scheduling MR jobs to meet deadline and data locality by assigning tasks to nodes that contain their input data in public cloud environments like Amazon web services. To meet a deadline, a task maybe scheduled on a node without local input data for that task causing expensive data transfer from a remote node. One of the major requirement in cloud computing is related to optimizing the resource being allocated with the objective of minimizing the costs associated with it. The research proposes an approach to schedule jobs on scalable and heterogeneous hadoop system in cloud environments with respect to the number of incoming jobs and available resources. A scheduler is proposed to address the problems which are primarily based on an efficient resource allocation and workload management strategies for large scale cloud environments. It deals with effective resource allocation strategies for achieving user satisfaction and maximizing the profit for cloud service providers.

Key words: Cloud –Hadoop Map Reduce- Resource Provisioning - Cluster - Performance Modelling

I. INTRODUCTION
Since 2007, the term Cloud has become one of the most buzz words in IT industry. Lots of researchers try to define cloud computing from different application aspects, but there is not a consensus definition on it. Among the many definitions, we choose three widely quoted as follows

A. Foster:
“A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over Internet.” As an academic representative, Foster focuses on several technical features that differentiate cloud computing from other distributed computing paradigms. For example, computing entities are virtualized and delivered as services, and these services are dynamically driven by economies of scale. Clouds are deployed in different fashions, depending on the usage scopes. There are four primary cloud deployment models.

B. Public Cloud:
Is the standard cloud computing paradigm, in which a service provider makes resources, like applications and storage, available to the general public over Internet. Service providers charge on a fine-grained utility computing basis. Examples of public clouds include Amazon Elastic Compute Cloud (EC2), IBM’s Blue Cloud, Sun Cloud, Google App Engine and Windows Azure Services Platform.

C. Private Cloud:
Looks more like a marketing concept than the traditional mainstream sense. It describes a proprietary computing architecture that provides services to a limited number of people on internal networks. Organizations needing accurate control over their data will prefer private cloud, so they can get all the scalability, metering, and agility Benefits of a public cloud without ceding control, security, and recurring costs to a service provider. Both eBay and HP Cloud Start yield private cloud deployments.

D. Hybrid Cloud:
Uses a combination of public cloud, private cloud and even local infrastructures, which is typical for most IT vendors. Hybrid strategy is proper placement of workloads depending upon cost and operational and compliance factors. Major vendors including HP, IBM, Oracle and VMware create appropriate plans to leverage a mixed environment, with the aim of delivering services to the business. Users can deploy an application hosted on a hybrid infrastructure, in which some nodes are running on real physical hardware and some are running on cloud server instances.

II. PROBLEM DESCRIPTION
By first setting up a small Eucalyptus Cloud on a few local servers this paper can answer which problems and obstacles there are when preparing the open-source infrastructure. The main priority is setting up a cloud that can deliver virtual instances capable of running Hadoop Map Reduce on them to supply a base to perform the analysis of the framework. Simplifying launching Hadoop Map Reduce clusters inside the Eucalyptus Cloud is of second priority after setting up the infrastructure and testing the feasibility of Map Reduce on virtual machines. This can include scripts, stand-alone programs or utilities beyond Eucalyptus and/or Hadoop.

III. PROPOSED SYSTEM
A. Running Map Reduce on the Cluster:
The procedure to boot a cluster and run a Map Reduce job is the following:
1) Launch VMs through euca2ools or Hybrid fox.
2) Change The Master, Slaves And The Replication Level In The Hadoop-Cc.Conf File.
3) Run Update-Cluster.

Update-cluster will do several things; first it will
connect to the master node and generate a SSH-key for the
user hadoop. That key will then be distributed across all
the slaves, as it will enable password-less-SSH for user hadoop
to all VMs. Secondly it will log in on all the VMs, use
the metadata service and fetch the IP/FQDN pairs.

Third, it will create a /etc/hosts _le containing all
the IPs/FQDNs of all the VMs in the cluster. Lastly it will
distribute the /etc/hosts _le to all nodes, distribute the
hadoop-cc.conf _le to all nodes and tell them to run the
update-hadoop.sh script.
4) Run Boot-Cluster.

This will start the Hadoop Map Reduce and HDFS
services as user Hadoop. With the SSH key pairs already
distributed for user hadoop, the master node will take care of
starting it on all the nodes.
5) Upload the _Les To HDFS
Since the database _les does not reside on
the cluster by default, they are transferred to the master
through SCP. With a SSH call to the master’s HDFS client they can
then be uploaded to the HDFS.
6) Download The JAR _Le containing The Map Reduce
Implementation.

Fetching the JAR-.le containing the job is a simple
matter of placing it in SHADOOP HOME.
7) Run the Map Reduce Job Using Bin/Hadoop Jar

This will run the Hadoop hadtest.jar with the
arguments “count articles out”. It basically means run the
count job on the articles HDFS folder and put the results in
the HDFS out folder.
8) Retrieve The Result From Out Folder.
This will be a list based on the implementation.
9) Calculate The Job Time:
This is done by visiting Hadoop’s internal web services for
the Job Tracker. By visiting http://<Master External
IP>:50030 the Map Reduce framework provides feedback
on time, number of tasks etc.

The scripts that run outside the cluster requires the
prevalence of a key pair that gives root access to the
instances. This is because they will login as root and switch
to user hadoop on the VMs and then perform the relevant
actions. There are a few shortcomings in the Eucalyptus DNS
name server for internal VMs. What it does is that the
instances cannot resolve each other through the name server
that Eucalyptus provides. To circumvent the shortcoming
the update-cluster script builds a large hosts file containing a
list of IP/FQDN-pairs of all the nodes in the cluster.
The computed hosts file is distributed to all slaves and replaces
the existing one. This lets Hadoop resolve

Them through the local file. What is also missing
in Eucalyptus is providing proper in-communication
between VMs based on their external IP. For instance,
assume a VM with 10.10.1.2 and 130.239.48.193as its
internal and external IP and another VM with
10.10.1.3/130.239.48.194 as its IPs. The first VM can only
contact the other VM by contacting the internal, 10.10.1.3,
IP but contacting through the external IP, 130.239.48.194,
will give a timeout. The reason to this is the virtual switch
inside the front-end. When booting a new instance, the
front-end adjusts its Net filter rules using the ip tables binary
in the same way as giving itself the meta data address. The
problem here is that Eucalyptus automatically does it for the
internal IPs, but the external IPs are never adjusted for the
VMs.

The boot-cluster and terminate-cluster manually
adds and removes ip tables rules to solve this issue through:

Sudo iptables -t nat -A POSTROUTING -s "<LOC_IP>" --to-source "<PUB_IP>"
and
Sudo iptables -t nat -D POSTROUTING -s "<LOC_IP>" --to-source "<PUB_IP>"

Unfortunately this requires that the scripts are run
at the front-end and with su privileges. of course, adding an
SSH command would be sufficient if it was to be run remotely, but they still require iptables changes on the front-end.

B. The Map Reduce Implementation:
To test the Map Reduce framework on the infrastructure an
embarrassingly distributed problem was chosen that could
easily use Map Reduce to distribute the tasks. The problem
chosen was to count all the links in a subset of Wikipedia
articles, listing the article title with the

Most links to it. This is a problem most suitable for
Map Reduce, as the database dumps from Wikipedia are
large XML files. The largest ness of the _les, first the
HDFS, but with XML _les there is a slight problem of tags
existing outside of file splits. When splitting _les, HDFS
and Map Reduce does not account to any specific split
locations by default, so a specific Input Split had to be
implemented. Cloud9 [15] is a Hadoop Map Reduce project
with an implementation that suits the Wikipedia databases
and Map Reduce. However, the XML parsing looks quite
like the Apache Mahout [12] implementation and both uses
the old pre-Hadoop 0.20 API.

The implementation is thus rewritten to fit the
0.21.0 API. This utilizes the Map Reduce Record Reader’s
ability to read outside of the split given, so the Input Format
can read from
a tag start to a tag end. After the XML _les have been
parsed, a Wikipedia Article is passed to a Mapper. The
mapper just calculates the link titles which are notifi-
yed and writes them to a _le that switches the key-values
and passes it through a reducer to print it out. This
implementation enables an adjustable amount of data to use,
as running the whole Wikipedia database of current articles
is a 27 GB large XML _le. Only using subset _les during
the testing shows whether there is a difference in completion
time when there is a large amount of data to process
compared to a smaller amount. Subsets used where the
article-only (no metadata, images, history or user
discussions) subsets dumped at 2011-03-17 [25]. The

All rights reserved by www.ijsrd.com 1560
subsets ranged from 2.9 GB (first three XML files), 4 GB (first 4) and 9.6 GB (first 8 files).

To calculate the time taken to process the data the following was done:

1) Start a specific amount of Hadoop VMs.
2) Configure them accordingly.
3) Upload the data and the JAR-file to the master VM.
4) Put the data onto HDFS.
5) Run the job 5 times, making sure to remove the output folder between runs.
6) Go to the JobTracker Web Service and analyse the time.
7) The max time of a job consist of adding both the Count and Sort jobs together.
8) Calculate an average of the 5 runs.

All of the nodes only uses 1 core and 512 MB RAM each and the master also contained a slave in all configurations. When running with only one node in the cluster, the Replication level was set to 1, in any other test the replication was set to 2. The size of the ephemeral HDD varied depending on the size of the database and the number of nodes in the cluster. See table 3.2.1, where the master also counts as a slave.

<table>
<thead>
<tr>
<th>DB size</th>
<th>Nr. of nodes</th>
<th>Master HDD size</th>
<th>Slave HDD size</th>
<th>Replication level</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.9 GB</td>
<td>1</td>
<td>10 GB</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>2.9 GB</td>
<td>2</td>
<td>10 GB</td>
<td>5 GB</td>
<td>2</td>
</tr>
<tr>
<td>2.9 GB</td>
<td>4</td>
<td>10 GB</td>
<td>5 GB</td>
<td>2</td>
</tr>
<tr>
<td>2.9 GB</td>
<td>8</td>
<td>10 GB</td>
<td>5 GB</td>
<td>2</td>
</tr>
<tr>
<td>2.9 GB</td>
<td>12</td>
<td>10 GB</td>
<td>5 GB</td>
<td>2</td>
</tr>
<tr>
<td>4.0 GB</td>
<td>1</td>
<td>10 GB</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>4.0 GB</td>
<td>2</td>
<td>10 GB</td>
<td>5 GB</td>
<td>2</td>
</tr>
<tr>
<td>4.0 GB</td>
<td>4</td>
<td>10 GB</td>
<td>5 GB</td>
<td>2</td>
</tr>
<tr>
<td>4.0 GB</td>
<td>8</td>
<td>10 GB</td>
<td>5 GB</td>
<td>2</td>
</tr>
<tr>
<td>4.0 GB</td>
<td>12</td>
<td>10 GB</td>
<td>5 GB</td>
<td>2</td>
</tr>
<tr>
<td>9.6 GB</td>
<td>1</td>
<td>25 GB</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>9.6 GB</td>
<td>2</td>
<td>10 GB</td>
<td>10 GB</td>
<td>2</td>
</tr>
<tr>
<td>9.6 GB</td>
<td>4</td>
<td>10 GB</td>
<td>10 GB</td>
<td>2</td>
</tr>
<tr>
<td>9.6 GB</td>
<td>8</td>
<td>10 GB</td>
<td>10 GB</td>
<td>2</td>
</tr>
<tr>
<td>9.6 GB</td>
<td>12</td>
<td>10 GB</td>
<td>10 GB</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3.2.1: The Ephemeral HDD Size of the Nodes in the Running Cluster.

To simplify deployment on the cluster, the JAR file was built to include any libraries it uses in its build path. This provides a relatively large file compared to without these libraries. This does however remove the need to include libraries on launch from the Map Reduce framework itself.

IV. RESULTS

This chapter details the results of running Hadoop Map Reduce jobs inside a virtual cluster. The procedure to perform the tests are explained in an earlier chapter, see section 5.5.

When running the example job and verifying the timings and map tasks run, the internal web service of Hadoop was used to procure the timings on the job time and map times. VMs run on Eucalyptus never crashed during the test period once they were properly booted. However, some things were observed and noticed during the boot phase of a VM, while most of the problems occurred. A machine running did run until the machine was terminated either inside the VM or by controlling it outside-wise. A few observations related to running the Eucalyptus cloud:

1) Booting many instances at the same time increase crash risk of VMs

When Eucalyptus receives requests of booting several instances at the same time, there is a risk that the VM either kernel panics or fails to receive an IP or properly resolve its FQDN. If a high amount of (more than three) instances are booting simultaneously then it is more likely that an instance will fail to boot. Requesting an instance one by one minimize the risk, but increases the time to completely boot all the instances in a cluster.

2) Node Controllers does not remove cached images that has been removed

Removing an image from the cloud removes it from Walrus but the Node Controllers retains a cached copy of it, even though it cannot be booted by the infrastructure. This can clout the HDDs of the NCs and sometimes requires manual removal to free up disk space.

3) Networking to a new instance is never instantly created

Even though the iptables rules and bridging is properly setup the instances are not instantly accessible. They can respond to ping, but the time it takes to access an instance through SSH usually ends at 1-2 minutes.

4) The network to instances is never at a steady speed

Transferring files through wget, curl or SCP never had a stable speed. It could go up to 20 mb/s, and in the next moment stall for 15 seconds. This was frequently observed, especially when communicating from outside the virtual cluster into a VM. If this affected the speed between Hadoop nodes in a Map Reduce job is unknown.

Hadoop Map Reduce itself is built around being redundant, so if a machine stopped working it would solve the task eventually either on the same VM or on another in the cluster. Running the Map Reduce jobs did not induce any severe crashing VMs or similar even though they could provide a high load on the machines. Some things were observed during the runs however:

5) Near linear increase in performance on few nodes

Increasing from 1 to 2 nodes is a large increase in performance. Henceforth the performance increase is not linear but still increasing.

6) Larger cluster means more repetition of same task
If the cluster is large, the likelihood of a map task being done at least twice and then dropped is increased. This is also based on the size of the database it runs on. As an example, a job of 157 tasks is done exactly 157 times on a 1-node cluster whereas on a 8-node cluster it would be done on average 165 times and having 8 tasks dropped.

7) Task getting dropped more often on large cluster

Related to the previous item, tests indicate that nodes timeout more often the larger the cluster is. Whether this is related to the Map Reduce framework or the internal networking between the nodes is unknown. This requires forces the task to be redo n Repeat or at another location. An important note here is that performance is somewhat simplified. This thesis focuses only how fast the job finishes which means that:

High performance -short job time

This means that high performance equals very low time to complete the job.

V. MAP REDUCE PERFORMANCE TIMES

Initially the testing was supposed to be made with a single size of a database, but once the initial tests of the relatively small database of 2.9 GB of Wikipedia articles were done the Map Reduce displayed results that could indicate that 12 nodes would not make an increase in performance compared to 8 nodes. So the database was increased to 4.0 GB at first and then 9.6 GB to verify the results.

The increased size did not however display any significant changes from the initial analysis of the database; when the size of the cluster where put to the maximum of 12 virtual instances the gain in performance would be marginal. Figure 5.1, 5.2 and 5.3 shows the performance curves based on the size of database and number of nodes in the cluster.

As such, the only noticeable difference when nearing the max number of virtual nodes available is that the max/min-times of completing the job would draw closer to the average compared to the 8 node clusters. On a cluster with only few nodes in it, the variations on how long time it takes to complete the job were very high compared to a 8 or 12 node cluster.

The map times on the jobs did not make any considerable changes based on how large the database or nodes there are in the cluster. The only noticeable difference would be that the larger the database the higher the maximum time of an—individual map time would be observed. However, that could point at a problem regarding the network, where the JobTracker would be unable to contact the TaskTracker of the individual map and thus dropping it from the job, redoing it later or on another node. See figures 5.4, 5.5 and 5.6.

Fig. 5.1: Runtimes on a 2.9 GB database.

Fig. 5.2: Runtimes on a 4.0 GB database.

Fig. 5.3: Runtimes on a 9.6 GB database.

Fig. 5.4: Map task times on a 2.9 GB database.
Scheduling on Heterogeneous Hadoop System in Eucalyptus Private Cloud
(IJSRD/Vol. 3/Issue 02/2015/389)

V. CONCLUSION

While Hadoop Map Reduce is designed to be used on top of physical commodity hardware servers, testing has shown that running Hadoop Map Reduce in private cloud supplied by Eucalyptus is viable. Using virtual machines gives the user the ability to supply more machines when needed, as long as it is not reaching the physical upper limits of the underlying host machines. While setting up the cloud and Hadoop can prove problematic at first, it should not pose a problem to someone experienced in scripting, command line interfaces, networking and administration in a UNIX environment.

Using Hadoop in a virtual cluster provides an added benefit of reusing the hardware to something completely different when no Map Reduce job is running. If the cloud contains several different images, it is quite viable to use a private cloud as a mean to give more computing power if needed and use the hardware to something else when it is not.

VI. FUTURE WORK

The size of both the private Eucalyptus cloud and Hadoop’s virtual cluster is very small compared to the size they are designed towards. Having a large cloud spread on large racks and preferably under several geographically separated availability zones would definitely increase the difficulty of maintenance and setup process. However, adding more Node Controllers to the already existing cluster should not pose a considerable problem, as the major of issues regarding the cloud network is already set up.

Hadoop should preferably run on larger VMs with more RAM and HDD available to them. What would be interesting to see is if the cluster shows the same decline in performance increase when the cluster is reaching 1,000 or more virtual machines. Also interesting to see would be if the Hadoop cluster recognizes a VM that relies on a different physical rack (but in the same virtual) as a node inside the same rack. The rack-awareness inside the Hadoop cluster relies on the reverse DNS and FQDNs which means that Eucalyptus first needs an upgrade in how it handles iptables, virtual switching and DNS lookups.

As the VMs runs on three physical servers, a comparisation between the performance of the virtual cluster and running Hadoop straight on top of the three servers would also be interesting. Dropping the cloud service, and with it the ability to switch VMs depending on need, would give fewer Hadoop nodes but each with higher individual performance. If this gives greater performance, and to what degree, than running more virtual nodes would be a good comparison.

The way to launch, check and terminate Hadoop nodes can easily be more streamlined. Either as complete stand-alone software utilizing the open API of Eucalyptus or adding more scripts to use in a CLI environment. The fact that the API that Eucalyptus uses is a web service implies a lot of different approaches and ways of interacting with VMs.

Using the storage features found in Eucalyptus, like it’s storage controller and Walrus, to simplify the booting of a new image could be an interesting take on how to actually load the test data locally to the virtual cluster. Hadoop’s Map Reduce uses HDFS as a file storage, but it can run directly on top of Amazon’s S3. If it uses the same API calls, then it should be able to run on top of Walrus instead. However, as Walrus is accessed through the network, it can pose a limitation in terms of bandwidth (depending on size of the database).

REFERENCES


