2015

Hadoop Based Data Intensive Computation on IAAS Cloud Platforms

Sruthi Vijayakumar
University of North Florida, sruthi.sinoj@hotmail.com

HADOOP BASED DATA INTENSIVE COMPUTATION ON IAAS CLOUD PLATFORMS

by

Sruthi Vijayakumar

A thesis submitted to the
School of Computing
in partial fulfillment of the requirements for the degree of

Master of Science in Computing and Information Sciences

UNIVERSITY OF NORTH FLORIDA
SCHOOL OF COMPUTING

May, 2015

The thesis "Hadoop Based Data Intensive Computation on IaaS Cloud Platforms" submitted by Sruthi Vijayakumar in partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Sciences has been

Approved by the thesis committee: Date

Dr. Sanjay P. Ahuja
Thesis Advisor and Committee Chairperson

Dr. Roger Eggen

Dr. Sandeep Reddivari

Accepted for the School of Computing:

Dr. Asai Asaithambi
Director of the School

Accepted for the College of Computing, Engineering, and Construction:

Dr. Mark A. Tumeo
Dean of the College

Accepted for the University:

Dr. John Kantner
Dean of the Graduate School

ACKNOWLEDGEMENT

I would like to take this opportunity to express my deepest gratitude to the people who supported me throughout the duration of this thesis. This thesis would not have been possible without the direction and support of my thesis advisor Dr. Sanjay P. Ahuja. I would also like to thank my committee members, Dr. Roger Eggen and Dr. Sandeep Reddivari, for their valuable suggestions. I would also like to thank Dr. Asai Asaithambi and Dr. Ching-Hua Chuan for attending my thesis defense and providing suggestions. Finally, I would like to thank my husband and my family for their continuous support, and encouragement during the process in achieving this important milestone. This thesis presented me a great experience on learning new and upcoming technologies on Enterprise Cloud computing platforms.

CONTENTS

FIGURES

TABLES

ABSTRACT

Cloud computing is a relatively new form of computing which uses virtualized resources. It is dynamically scalable and is often provided as pay for use service over the Internet or Intranet or both. With increasing demand for data storage in the cloud, the study of data-intensive applications is becoming a primary focus. Data intensive applications are those which involve high CPU usage, processing large volumes of data typically in size of hundreds of gigabytes, terabytes or petabytes. The research in this thesis is focused on the Amazon's Elastic Cloud Compute (EC2) and Amazon Elastic Map Reduce (EMR) using HiBench Hadoop Benchmark suite. HiBench is a Hadoop benchmark suite and is used for performing and evaluating Hadoop based data intensive computation on both these cloud platforms. Both quantitative and qualitative comparisons of Amazon EC2 and Amazon EMR are presented. Also presented are their pricing models and suggestions for future research.

Chapter 1

INTRODUCTION

According to National Institute of Standards and Technology (NIST), Cloud Computing can be defined as ‘A model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction” [Peter11]. The most important characteristics and aspects of cloud computing are:

- On-Demand self-service where no human interaction is required for server time and network storage.
- Broad network access where the users can essentially access the clouds from workstations, laptops and cell phones.
- Resource Pooling where multiple resources are pooled to server different physical and virtual resources gets assigned dynamically.
- Measured Service where computing resources such as storage, bandwidth can be monitored and reported to consumers and providers, hence providing flexibility.

There are three service models provided in the cloud:

- Infrastructure-as–a-Service (IaaS) where consumer is provided with the capability of provisioning storage, processing and networks and run arbitrary services. In this model, the consumer does not control the cloud infrastructure, storage or processing.

- Platform-as-a-Service (PaaS) wher0e consumer is provided with the capability of deploying applications on the cloud using the provider's tools and, libraries and languages. In this model, the infrastructure is controlled by the provider and the consumer only has access to deploy applications and change configuration settings related to deployment.
- Software-as–a-Service (SaaS) where consumer is provided with the capability of using provider's applications that are running on the cloud. In this case, the applications are either accessible from a web interface or a program interface. In this case, the provider controls the cloud infrastructure [Peter11].

1.1 Cloud Platforms

1.1.1 Amazon Elastic Compute Cloud (Amazon EC2)

Amazon EC2, also known as Amazon Elastic Compute Cloud, is an IaaS cloud platform that provides a web service based API for provisioning, managing, and de-provisioning virtual servers inside the Amazon cloud as shown in Figure 1. Applications residing anywhere on the Internet can launch a virtual server in the Amazon cloud and users can launch as many virtual servers as they want in the Amazon cloud. Amazon EC2 also allows users to configure security, and provide networking and scaling based on business requirements. Amazon EC2 instances can store data in Amazon S3 buckets (Amazon S3 provides an online file storage web service provided by Amazon Web Service) or Amazon EBS (Elastic Block Storage).

Amazon EC2 instances types can be classified as:

- On-Demand Instances-where the user pays for computing capacity by the hour.
- Reserved Instance (Light, Medium, and Heavy Utilization Reserved Instances) where the user pays one-time payment for the instance that they want to reserve and receive hourly discount on that instance.
- Spot Instances where the users bid on unused EC2 instances and run the instances as long the users bid does not exceed the spot price.

Each instance type varies in terms of memory capacity, available virtual cores, storage capacity and I/O performance. Users can choose the instance types based on their application needs [AWS14].

Graphic redacted, paper copy available upon request to home institution.

Figure 1: AWS EC2 Structure

1.1.2 Amazon Elastic Map Reduce (Amazon EMR)

Amazon EMR consists of multiple EC2 instances grouped in a cluster. It can process huge amount of data by splitting the computational work across multiple EC2 instances where each EC2 instance is a virtual server as shown in Figure 2. Amazon EMR cluster is managed by an open source Hadoop distribution. Amazon EMR cluster performance can be measured using Amazon CloudWatch. In order to run a job on Amazon EMR, users have to create an Amazon EMR job flow and execute it on the number of cluster nodes

they need. Amazon EMR is suitable for large cloud computing as new instances can be easily configured (added and removed) on running custom code.

Amazon EC2 is a stand-alone instance whereas Amazon EMR is a cluster of EC2 instances. Cluster management is performed by the user on each Amazon EC2 instance whereas automated Cluster management occurs in Amazon EMR. They also differ with respect to the cost variance factor. Amazon EC2 is more cost effective than EMR since Amazon charges for cluster management.

Amazon EMR pricing is the cost of running of Amazon EC2 instance plus the cost charged by Amazon for cluster management. Based on these varying factors, it is critical to establish benchmarks on both the clouds so that the user can determine whether to choose Amazon EC2 over Amazon EMR or vice-versa when it comes to Data Intensive Cloud Computing [AWS14].

Graphic redacted, paper copy available upon request to home institution.

Figure 2: AWS EMR Structure

1.2 Data Intensive Computations

Data intensive computations comprises of applications that involve high CPU usage, processing large volumes of data typically in size of hundreds of gigabytes, terabytes or petabytes. It has become critical that data intensive cloud providers offer on-demand computing instances and on-demand computing capacity. Clouds that provide on-demand computing instances and clouds that provide on-demand computing capacity like Amazon EC2 and Amazon EMR can support any computing model compatible with loosely coupled cluster. MapReduce along with Hadoop has become the dominant programming model used in data intensive cloud computing that provides on-demand computing capacity.

1.2.1 Hadoop

Apache Hadoop is an open source software project that enables distributed processing of large data sets across clusters of commodity servers. It is designed to utilize Master-slave system architecture as shown in Figure 3. Apache Hadoop is driven by two main components:

- Map Reduce - The framework that understands and assigns work to the nodes in a cluster.
- HDFS - A file system that spans all the nodes in a Hadoop cluster for data storage. It links together the file systems on many local nodes to make them into one big file system. HDFS assumes nodes will fail. Therefore, it achieves reliability by replicating data across multiple nodes.

Hadoop comes with a lot of inbuilt advantages like:

- Scalable – New nodes can be added as needed and added without needing to change data formats, how data are loaded, how jobs are written, or the applications on top.
- Cost effective – Hadoop brings massively parallel computing to commodity servers. The result is a sizeable decrease in the cost per terabyte of storage, which in turn makes it affordable to model all data.
- Flexible – Hadoop is schema-less, and can absorb any type of data, structured or not, from any number of sources. Data from multiple sources can be joined and

aggregated in arbitrary ways enabling deeper analyses than any one system can provide.

- Fault tolerant – When you lose a node, the system redirects work to another location of the data and continues processing without missing a beat [Hadoop13].

Graphic redacted, paper copy available upon request to home institution.

Figure 3: Hadoop Architecture

1.2.2 MapReduce

MapReduce is a programming model and software framework first developed by Google [MapReduce14]. This programming model helps in the processing of huge amount of data in parallel on large clusters in a reliable and a fault-tolerant manner. There are two fundamental steps associated with a MapReduce programming model. First step is the Map () function where a master node converts a set of data input into smaller set of data where individual elements are broken down into tuples (key-value pairs) as shown in Figure 4. Each of these tuples will be distributed to a slave node and these input lists

processed by the Map() function under slave nodes produces a different output list. The next step is the Reduce() function where the master node takes the output provided by each of the worker nodes and then combines them in a predefined way to provide the final output. MapReduce requires a “driver” method to initialize a job, which defines the locations of the input and output files and controls the MapReduce process. Each node in a MapReduce cluster is unaware of the other nodes in the cluster, and nodes do not communicate with each other except during the shuffling process [Hedger11].

Graphic redacted, paper copy available upon request to home institution.

Figure 4: MapReduce Architecture

1.3 Benchmarks

1.3.1 HiBench Benchmarks

HiBench is a representative and comprehensive benchmark suite for Hadoop. This benchmark suite consists of a set of Hadoop programs including both synthetic micro-

benchmarks and real-world applications. These benchmarks are used intensively for Hadoop benchmarking, tuning and optimizations. The categories of benchmarks used for this research are Micro benchmarks (Sort, WordCount, and TeraSort) which include more of unstructured data, Web Search Benchmark (PageRank) which include more of semi-structured data, and Analytical Query benchmarks (Hive Join, Hive Aggregation) which includes structured data.

Micro Benchmarks:

- Sort: This workload sorts its text input data, which is generated using the Hadoop RandomTextWriter program. Here the sorting is done automatically during the Shuffle and Merge stage of MapReduce programming model. This is an I/O bound function. The input workload for the Sort benchmark is the data size to be generated.
- WordCount: This workload counts the occurrence of each word in the input data, which is generated using the Hadoop RandomTextWriter program. This job extracts a small amount of information from a large data source. This is a CPU bound function. The input workload for the WordCount benchmark is the data size to be generated.
- TeraSort: This is a benchmark where input data are generated by the Hadoop TeraGen program that creates by default 1 billion lines, with each line 100 bytes in length. The data are then sorted by Terasort that provides its own input and output format and also its own Partitioner, which ensures that the keys are equally distributed among all nodes. This is an improved Sort program which provides

equal loads among all the nodes during the test. As a result this is a CPU bound function for the Map stage and I/O bound function for the Reduce stage. The input workload for the Terasort benchmark is the data size to be generated.

Web Search Benchmark

- PageRank: The workload contains an implementation of the PageRank algorithm on Hadoop which is a link analysis algorithm used widely in web search engines. This is a CPU bound function. The input workload to PageRank algorithm is number of Wikipedia pages.

Analytical Query Benchmarks

- Hive Join and Hive Aggregation: The workload contains queries that correspond to the usage profile of business analysts and other database users. The two tables created are User Rankings table and UserVisits table. Once source data have been generated, two of the Hive requests would be performed, a Join and an Aggregation. These tests are I/O bound functions [Wang14].The input workload for the Analytical Query Benchmark is number of records to be inserted into User Rankings table and User Visits table. The overview of benchmarks, its categories and metrics captured are shown in Table 1.

Benchmarks	Method	Metrics Measured
Data Benchmarks	Sort WordCount TeraSort	Response Time Data Size Throughput
Web Search	Page Ranking	Response Time Page Workload Throughput
Analytical Query	Hive Join Hive Aggregation	Execution Time Data Size Throughput

Table 1: Benchmarks and Metrics

1.4 Research Objectives

This study compares the performance of Hadoop based data intensive computation on two of the cloud services provided by Amazon, Amazon EC2 and Amazon EMR. Parameters such as the number of nodes, hardware and software resources and instance types vary while evaluating the performance of each cloud. Three categories of benchmarks under HiBench suite of Hadoop benchmarks such as Micro benchmarks, Web Search and Analytical query are utilized to perform the research. The literature review presents the previous work carried out on one of the cloud services provided by Amazon and also certain areas of work which are done on non-cloud platforms, all of these helped in this research. It is important to note that no previous research work exists on these Amazon cloud services to compare their performance using the HiBench benchmarks as done in this study.

Chapter 2

LITERATURE REVIEW

Currently, there are no set of existing benchmarks and experiments for evaluating cloud performance of Amazon EC2 and Amazon EMR from the perspective of data intensive computing though there have been benchmarks that have been run on local machines and clusters using Hadoop. However there exist certain studies and benchmarking of Amazon cloud service particularly Amazon EC2 with other cloud platforms such as Rackspace as discussed below.

2.1 Studies using HiBench Benchmarks

A recent paper 'HiBench: A Representative and Comprehensive Hadoop Benchmark Suite' by Intel research group, talks about a comprehensive benchmark suite for Hadoop [Huang10]. HiBench benchmarks according to the study can be divided into various categories: Data Benchmarks, Web Search Benchmark and Analytical query Benchmarks. This study on HiBench consists of a set of Hadoop programs including both synthetic micro-benchmarks and real-world applications.

Huang et al in their paper 'The HiBench Benchmark Suite: Characterization of the MapReduce-Based Data Analysis', discuss the MapReduce model used as a prominent model for large scale data analysis in the cloud [Huang10]. The authors use HiBench to

evaluate and characterize Hadoop framework in terms of speed (job running time), throughput (the number of tasks completed per minute), HDFS bandwidth, system resource (CPU, memory and I/O) utilizations, and data access patterns such as map period, average mapper time and job execution time. The authors concluded that HiBench is a new, realistic and comprehensive benchmark suite for Hadoop, which consists of a set of Hadoop programs including both synthetic micro-benchmarks and real-world applications. The HiBench suite is essential for the community to properly evaluate and characterize Hadoop, because it's workload not only represent a wide range of large-scale data analysis using Hadoop, but also exhibit very diverse behaviors in terms of data access patterns and resource utilizations.

2.2 Studies on Amazon Cloud Services vs. Other Cloud Platforms

According to the recent benchmark study on clouds by Sarda et al in 'Cloud Performance Benchmark – Amazon EC2 vs. RackSpace' (cloud based VPS), Rackspace's 512MB instance was more than twice as fast as Amazon's micro instance [Sarda11]. The study benchmarked metrics Relative CPU Performance, IO Read and IO Write, Number of Requests Apache Can Handle and Processing Power. The authors concluded that Rackspace is 3 times faster than Amazon EC2 in terms of Processing Power, Rackspace can handle 5.5 times more requests than Amazon when using Apache HTTP server, and Rackspace can write 7.6 times more data than Amazon per second and is 2.3 times faster than Amazon EC2 in terms of CPU performance.

As discussed in this chapter, there are various benchmarks comparing the performance of Amazon EC2 to other clouds and vice versa but there do not exist any studies of benchmarks that focus on comparing the performance of Amazon EC2 with Amazon EMR for data intensive computing using Hadoop, which is the focus of this thesis.

Chapter 3

RESEARCH METHODOLOGY

This study evaluates the performance of the two cloud services provided by Amazon, Amazon EC2 and Amazon EMR for Hadoop based data intensive computation. The study uses HiBench benchmark suite with Micro Benchmarks (Sort, WordCount, Terasort), Web Search Benchmark (Page Rank) and Analytical Query performance Benchmarks (Hive Join and Hive Aggregation) for the performance comparison of Amazon EC2 and Amazon EMR for data intensive computing.

The initial step uses Micro Benchmarks which includes Sort, WordCount and TeraSort benchmarks to be run on the Amazon EC2 cloud service with varying dataset sizes of 1GB, 10GB and 100GB by varying the number of nodes with each dataset size from one to eight nodes. The metrics such as response time and throughput will be measured while varying the nodes and the dataset sizes. The Web Search Benchmark which include PageRank Benchmark to be run on Amazon EC2 cloud service with varying dataset sizes of 100000,1000000,10000000 Pages by varying the number of nodes during each dataset size from one to eight nodes. The metrics such as response time and throughput will be measured while varying the nodes and the dataset sizes. The Analytical Query Benchmarks which includes Hive Join and Hive Aggregate to be run on Amazon EC2 cloud service uses two input tables, User Visits table and User Aggregate table. The number of records (1000000, 10000000, 100000000) are modified for UserVisits table

and number of records (600000, 6000000, 60000000) are also modified for User Aggregate table by varying the number of nodes during each dataset size from one to eight nodes. The metrics such as response time and throughput will be measured while varying the nodes and the dataset sizes.

The study also performs the HiBench benchmarks on the Amazon EMR in the same manner as performed on the Amazon EC2 cloud service. The average response time and throughput is computed and a graph is plotted to analyze the performance of the Amazon EMR cloud service. By comparing the performance of Amazon EC2 and Amazon EMR through their test results, conclusions are drawn for the experiments.

Chapter 4

TESTBED SETUP

4.1 Creating Clusters on the Amazon EC2 Cloud Service

StarCluster is an open source utility for creating and managing distributed computing clusters on Amazon EC2. StarCluster utilizes the Amazon EC2's web service to create and terminate on demand clusters on Amazon EC2. StarCluster enables to launch clusters on Amazon EC2 by setting up a single configuration file. StarCluster automates security groups, user accounts, provides passwordless SSH, automation of EBS volumes and provides a Queuing System. StarCluster uses an Amazon AMI to launch a cluster. StarCluster also enables to dynamically resize Clusters, create and format EBS Volumes. Below are the steps for creating a cluster using StarCluster on Amazon EC2 [StarCluster14].

StarCluster requires python packages to be installed. In order to install python package dependencies, execute the following command.

```
$ sudo python setup.py install
```

In order to install StarCluster on master node, execute the following command.

```
$ sudo easy_install StarCluster
```

Once StarCluster has been installed, the next step is to initialize StarCluster configuration file. In order to do this, type the following command and select option 2.

```
$ starcluster help
```

Options:

1. Show the StarCluster config template
2. Write config template to /home/user/.starcluster/config
3. Quit

Next step is to configure StarCluster configuration file to use AWS configuration.

```
$ vi ~/.starcluster/config
```

Below parameters needs to be modified in the StarCluster configuration files.

```
AWS_ACCESS_KEY_ID = AKIAJFPRDFCJ6MI2EKBA
AWS_SECRET_ACCESS_KEY =
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
AWS_USER_ID= 751348288379
NODE_INSTANCE_TYPE = m3.2xlarge
```

Next step is to generate an EC2 key pair in StarCluster. Execute the following command for creating a keypair named mykey and it saves the private key to ~ /.ssh/mykey.rsa

```
$ starcluster createkey mykey -o ~/.ssh/mykey.rsa
```

Next step is to modify StarCluster configuration file to modify the key name and key

value generated in previous step.

```
keyname = mykey (the key name generated from above step).
key_location = ~/.ssh/mykey.rsa (location provided for
saving private key).
```

Next step is to generate an EC2 key pair in StarCluster. Execute the following command for creating keypair named ‘mykey’ and it saves the private key to the default location which is ~/.ssh/mykey.rsa

```
$ starcluster createkey mykey -o ~/.ssh/mykey.rsa
```

StarCluster can be started once the above configurations are in place. By default, StarCluster starts a two node cluster with mater node aliased as ‘master’ and named node as ‘node001’. In order to start the cluster, execute the following command.

```
$ starcluster start pagerank100GB
```

In order to login to the StarCluster master node as root user, execute the following command

```
$ starcluster sshmaster pagerank100GB
```

In order to add notes to a StarCluster, execute the following command

```
$ starcluster addnode pagerank100GB
```

In order to get list of running nodes on StarCluster, execute the following command

```
$ starcluster listclusters
```

In order to remove nodes from a StarCluster, execute the following command

```
$ starcluster removenode pagerank100GB node001
```

In order to terminate a running StarCluster, execute the following command

```
$ starcluster terminate pagerank100GB
```

In order to get a list of available StarCluster commands, execute the following command

```
$ starcluster –help
```

4.2 Creating Clusters on the Amazon EMR Cloud Service

Amazon EMR distributes its computational work across a cluster of virtual servers on the Amazon cloud using open source Hadoop. Hadoop uses distributed processing architecture using MapReduce to split a task to run on a set of servers for processing. Amazon EMR provides a User Interface for Creating and Managing a Cluster. For creation of a cluster, the Amazon UI allows to specify the Hardware configuration, Software configuration, configuring Security and Access and specifying Bootstrap Actions. For managing of a cluster, the Amazon UI allows to add a node to existing cluster, resize a cluster, clone a cluster and terminate a cluster. A detailed explanation of creating a cluster on the Amazon EMR cloud service is available in Appendix C.

4.3 Hadoop setup

Apache Hadoop is an open source software framework for distributed storage and processing of large datasets on clusters. Its Hadoop Distributed File System (HDFS) splits files into blocks which are distributed and processed across nodes in a cluster. Apache Hadoop runs on both Linux and Windows operating systems.

4.3.1 Prerequisites

Apache Hadoop requires Java JDK as a prerequisite. We can download the latest Java JDK from Oracle’s website. To install the Java JDK use the following command.

```
$ sudo apt-get install openjdk-7- jdk
```

To verify the Java installation and version use the following command.

```
$java –version
```

To check if java class path has been set use the following command.

```
echo $JAVA_HOME
```

Apache Hadoop only supports IPv4 and hence IPv6 needs to be disabled. In order to disable IPv6, execute the following command

```
$sysctl -w net.ipv6.conf.default.disable_ipv6=1
$sysctl -w net.ipv6.conf.all.disable_ipv6=1
```

To verify IPv6 is disabled, execute the following command and if it returns a result of 1, then IPv6 has been disabled.

```
$cat /proc/sys/net/ipv6/conf/all/disable_ipv6
```

Hadoop needs to be able to establish secure shell connections without using a passphrase within all nodes in a cluster. In order to communicate with all nodes in a cluster, we must ensure that SSH is installed on all of the nodes in the cluster on port 22 [Noll11]. To verify SSH installation, execute the following command.

```
$ssh localhost
```

4.3.2 Hadoop Installation

Once all Hadoop pre-requisites have been met, install Hadoop from Apache Website on single node. Once the installation on single node is successful, then move ahead with installation of Hadoop on multi node cluster by using the command shown below.

```
$ wget "https://archive.apache.org/dist/hadoop/core/hadoop-
1.0.3/hadoop-1.0.3.tar.gz"
```

To extract the tar file and move the files into new directory called Hadoop, use the following commands.

```
$ sudo tar xzf hadoop-1.0.3.tar.gz
$ sudo mv hadoop-1.0.3 hadoop
```

Hadoop is driven by six major configuration files with the conf directory which need to be configured before we start a Hadoop Cluster by using the following commands.

```
$ vi /<hadoop folder path>/conf/core-site.xml
$ vi /<hadoop folder path>/conf/mapred-site.xml
$ vi /<hadoop folder path>/conf/hdfs-site.xml
$ vi /<hadoop folder path>/conf/haddop-env.sh
$ vi /<hadoop folder path>/conf/master
$ vi /<hadoop folder path>/conf/slaves
```

Appendix A provides a detailed explanation of the above configuration files used for Amazon EC2 and Amazon EMR. Hadoop's default configuration uses hadoop.tmp.dir as the base temporary directory for both the local file system and HDFS. Create Hadoop temp directory under Hadoop folder and set appropriate permissions using the following commands.

```
$ sudo mkdir -p /<hadoop folder path>/tmp
$ sudo chmod 750 /<hadoop folder path>/tmp
```

The Hadoop framework consists of two main layers, the Distributed File System and Execution Engine (Map Reduce Layer). When starting Hadoop cluster, the HDFS layer should be started first followed by the Map Reduce layer.

When installing Hadoop for the very first time, we need to format Hadoop Named Node using the following command.

```
$ /<hadoop folder path>/bin/hadoop namenode –format
```

Once named node has been successfully formatted, start the HDFS layer by using the following command.

```
$ /<hadoop folder path>/bin/start-dfs.sh
```

Once HDFS layer has been successfully started, start the Map Reduce layer by using the following command.

```
$ /<hadoop folder path>/bin/start-mapred.sh
```

Once Hadoop has been configured successfully on a single node, we can move ahead with configuring Hadoop on multi-node cluster. Configure first node in multi node cluster as master node, and remaining nodes in the cluster as slave nodes as shown in Figure 5. The master node's 'slaves' configuration file within the Hadoop `conf` directory specifies all the slaves nodes under that master node. Once 'slaves' file has been modified on the master node and SSH has been established between master node and slave node, we can start Hadoop on multi node cluster by executing following commands on master node [NOLL11]. Perform the startup of the multi-node cluster using below two steps. Start the HDFS layer followed by the Map Reduce layer by executing the following commands.

```
$ /<hadoop folder path>/bin/start-dfs.sh
$ /<hadoop folder path>/bin/start-mapred.sh
```

In order to stop a running Hadoop cluster, stop the Map Reduce layer followed by HDFS layer by executing the following commands

```
$ /<hadoop folder path>/bin/stop-mapred.sh
```

```
$ /<hadoop folder path>/bin/ stop-dfs.sh
```

Graphic redacted, paper copy available upon request to home institution.

Figure 5: Overview of multi-node cluster

4.4 HiBench Setup

HiBench is a comprehensive Hadoop benchmark suite developed by Intel Engineers representing real world applications. HiBench Benchmarks are classified into Micro Benchmarks, Web Search Benchmark and Data Analytics Benchmarks.

4.4.1 HiBench Installation

The primary requirement for installing HiBench benchmark is that Hadoop must be installed on the master node. Once Hadoop installation is complete, we can download the

current version of HiBench from github website, in our case HiBench-2.2 by executing the following command.

```
$ wget "https://github.com/intel-
hadoop/HiBench/zipball/HiBench-2.2"
```

The next step is to unzip HiBench downloaded file by executing the following command.

```
$ unzip HiBench-2.2
```

The next step is to rename the unzipped HiBench file to a new folder called HiBench by executing the following command.

```
$mv intel-hadoop-HiBench-4aa2ffa/ HiBench
```

The next step is to change permissions on HiBench folder by executing the following command.

```
$ chmod -R 755 HiBench.
```

Within the HiBench `bin` directory, modify the HiBench configuration file named `'hibench-config.sh'` to specify Hadoop home installation directory by executing the following command.

```
$HADOOP_HOME =/usr/local/Hadoop
```

The report of the tests run from HiBench is written to `'hibench-report'` file within the HiBench `root` directory.

Chapter 5

HARDWARE AND SOFTWARE SPECIFICATIONS

5.1 Software Specifications

The following list of software installations were performed for this research.

- Install Amazon Linux AMI on the workstations.
- Install version 1.7 of the Java JDK.
- Install Hadoop version 1.0.3 on both Amazon EC2 and Amazon EMR.
- Install Hi-Bench 2.2 on Amazon EC2 and Amazon EMR Hadoop.
- Configure SSH on all the nodes on Amazon EC2 and Amazon EMR for communication between name node and all the data nodes.
- Install Python on Amazon EC2 which is a pre-requisite for StarCluster Installation.
- Use StarCluster open source toolkit to create cluster on Amazon EC2.
- Create cluster on Amazon EMR using Amazon UI.

5.2 Hardware Specifications

Hardware configuration used is very critical for Hadoop based data intensive benchmark. For this purpose an M3 General Purpose Double Extra Large instance type has been chosen for both Amazon EC2 and Amazon EMR. Amazon M3 instance types provide a balance of memory, compute and network resources with its most prominent features

being SSD based storage for very fast I/O performance and High Frequency Intel Xeon E5-2670 v2 (Ivy Bridge) Processors. Provided in Table 2 are specifications for M3.2xlarge instance.

Instance Type	m3.2xlarge
Processor	Intel Xeon E5-2670 v2 2.5 GHz
Memory	30GB
Storage Drives	160 GB (2 * 80 GB SSD)
I/O Performance	High / 1000 Mbps

Table 2: AWS EC2 and AWS EMR Hardware Configuration

Chapter 6

RESULTS AND ANALYSIS

The study evaluates and compares the performance of the Amazon EC2 and Amazon EMR cloud services using HiBench benchmark suite, which includes Micro Benchmarks (Sort, WordCount, Terasort), Web Search Benchmark (Page Rank) and Analytical Query performance Benchmarks (Hive Join and Hive Aggregation). The Microsoft Excel 2010 built-in function T-TEST was used for statistical analysis of the results obtained from the benchmarks on Amazon EC2 and Amazon EMR. The T-TEST function used two datasets as input, the first dataset being Amazon EC2 and the second dataset being Amazon EMR. The p-value is measured, a p-value exceeding 0.05 is considered indicative of statistically insignificant difference between the two datasets, while a p-value not exceeding 0.05 is an indication of statistically significant difference between the two datasets.

For each benchmark, the Response Time (in seconds) and Throughput (in megabytes per sec) are measured with increasing number of nodes from 1 to 8. Graphs are then plotted for Amazon EC2 and Amazon EMR cloud services for comparing their performance. The graphs compare the performance of Amazon EC2 and Amazon EMR cloud services using each of the HiBench benchmark suites, which includes Micro Benchmarks (Sort, WordCount, Terasort), Web Search Benchmark (Page Rank) and Analytical Query Benchmarks (Hive Join and Hive Aggregation) by varying the dataset size (1GB, 10GB,

100GB) to represent data intensive computation using Hadoop. For each graph, the y-axis represents the Response Time (in seconds) and Throughput (in megabytes per second) achieved during the tests, and the x-axis represents the number of nodes tested.

6.1 Micro Benchmarks

6.1.1 Amazon EC2 and Amazon EMR Performance for Sort Benchmark

Table 3 and Table 4 show the Response time and Throughput values for Amazon EC2 and Amazon EMR respectively. Figures 6, 7and 8 present the plotted response times and throughput values for Sort benchmark performance on Amazon EC2 and Amazon EMR.

SORT						
Data size	1GB		10GB		100GB	
#nodes	EC2	EMR	EC2	EMR	EC2	EMR
1	133.411	189.75	385.191	535.754	4586.868	3157
2	74.301	116.994	210.991	303.568	2322.305	1462.162
3	53.251	95.31	153.914	240.459	1691.882	935.593
4	42.286	81.812	126.921	179.409	1194.463	749.433
5	41.282	75.251	119.811	171.431	1083.334	731.268
6	32.28	70.913	107.146	140.411	809.1	555.417
7	31.306	65.884	102.942	134.376	785.187	585.735
8	30.238	64.919	101.08	122.371	728.133	583.582
P-value	0.00000107		0.00367057		0.008840098	

Table 3: Sort: Response Time (seconds) - AWS EC2 vs. AWS EMR

SORT						
Data size	1GB		10GB		100GB	
#nodes	EC2	EMR	EC2	EMR	EC2	EMR
1	1.964130	1.380896	6.801263	4.889903	5.711348	9.306178
2	3.526689	2.239645	12.416574	8.629978	11.280689	17.916840
3	4.920781	2.749187	17.021098	10.894934	15.484059	28.000768
4	6.196767	3.202770	20.641071	14.602305	21.932199	34.956190
5	6.347476	3.482014	21.865984	15.281863	24.182016	35.824516
6	7.163937	3.695021	24.450613	18.657976	32.378199	44.725577
7	7.702596	3.977066	25.449140	19.495930	33.364282	45.844257
8	8.388389	4.036184	26.124556	21.408545	35.978593	48.120618
P-value	0.000551		0.000038		0.000058	

Table 4: Sort- Throughput (MB/seconds) – AWS EC2 vs. AWS EMR

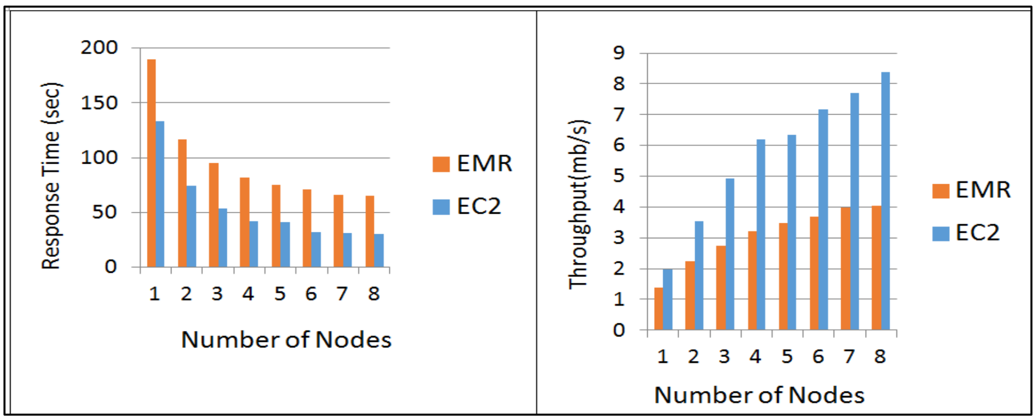


Figure 6: Sort – AWS EC2 vs. AWS EMR (1GB)

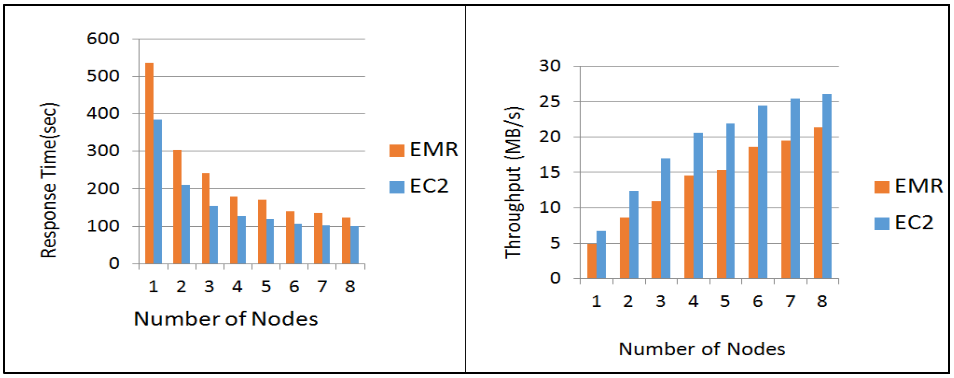


Figure 7: Sort – AWS EC2 vs. AWS EMR (10GB)

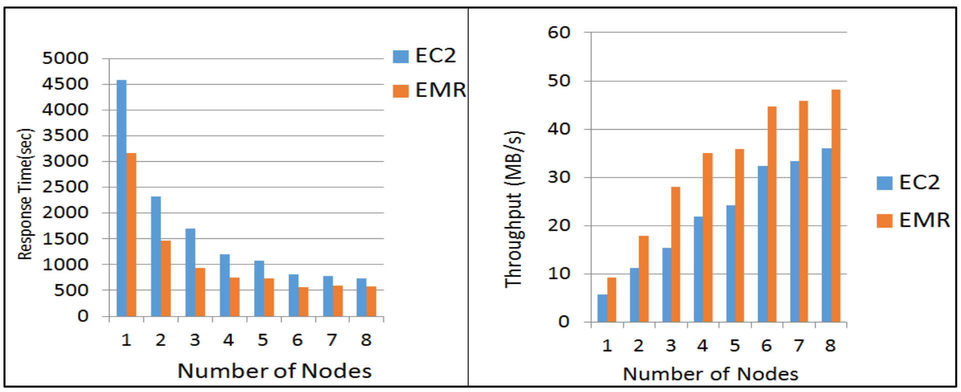


Figure 8: Sort – AWS EC2 vs. AWS EMR (100GB)

A statistical analysis was performed to determine if the difference in response time and throughput between the two cloud services offered by Amazon was significant. The differences in Response time and Throughput for each datasets (1GB, 10GB and 100GB) on varying nodes of 1 to 8 were found to be statistically significant with a p-value of less than 0.05 (Table 3 and Table 4).

Also the test results indicate that the Amazon EC2 cloud performed better than the Amazon EMR cloud service for data sizes of 1GB and 10GB but for larger data size of

100GB, Amazon EMR performed better than Amazon EC2 in terms of response times. Similar pattern is also seen in the throughput values.

From Figure 6 and Figure 7, we conclude that Amazon EC2 is performing better than Amazon EMR. Figure 8 indicates that when the datasize is increased to 100GB, performance of Amazon EMR is significantly better than Amazon EC2.

6.1.2 Amazon EC2 and Amazon EMR Performance for WordCount Benchmark

Table 5 and Table 6 show the Response time and Throughput values for Amazon EC2 and Amazon EMR respectively. Figures 9, 10 and 11 present the plotted response times and throughput values for WordCount benchmark performance on Amazon EC2 and Amazon EMR.

WORD COUNT						
Data size	1GB		10GB		100GB	
#nodes	EC2	EMR	EC2	EMR	EC2	EMR
1	131.466	216.949	416.625	465.245	2880	2779.16
2	73.357	124.694	247.451	275.105	1492.06	1445.108
3	53.338	93.914	202.412	236.128	1029.257	968.248
4	42.366	79.011	165.394	188.335	807.6	760.07
5	41.338	69.013	133.313	166.888	674.592	626.921
6	32.361	66.028	127.368	159.003	554.232	504.806
7	31.406	63.918	113.332	143.915	517.415	463.774
8	30.357	60.062	95.313	128.991	453.212	428.989
P-value	0.000409808		0.00000470076		0.000203863	

Table 5: WordCount: Response Time (seconds) – AWS EC2 vs. AWS EMR

WORD COUNT						
Data size	1GB		10GB		100GB	
#nodes	EC2	EMR	EC2	EMR	EC2	EMR
1	1.993297	1.207790	6.288044	5.630873	9.013599	9.426330
2	3.572267	2.096835	10.586970	9.522675	17.558458	18.128249
3	4.913023	2.790095	12.942692	11.094557	25.452656	27.056372
4	6.185404	3.316360	15.839488	13.909978	32.438490	34.466928
5	6.339223	3.796806	19.651169	15.697567	38.834324	41.787208
6	7.144061	3.795980	20.568402	16.454793	47.260035	51.895735
7	8.086490	4.099455	23.115768	18.203354	50.631165	56.487165
8	8.357011	4.362641	27.485823	20.309445	57.803686	61.059823
P-value	0.000338		0.004696		0.005539	

Table 6: WordCount: Throughput (MB/seconds) – AWS EC2 vs. AWS EMR

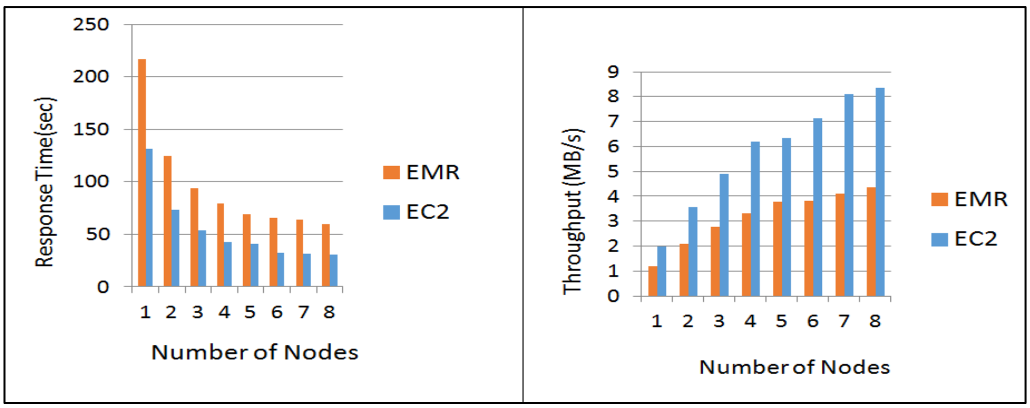


Figure 9: WordCount – AWS EC2 vs. AWS EMR (1GB)

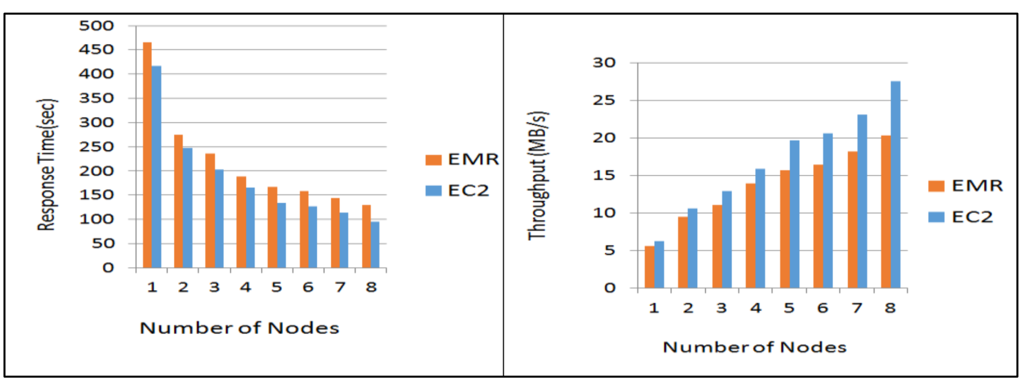


Figure 10: WordCount – AWS EC2 vs. AWS EMR (10GB)

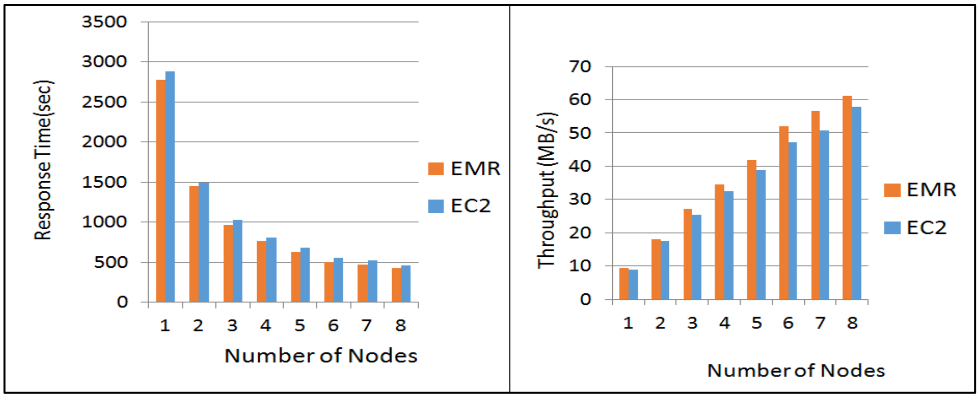


Figure 11: WordCount – AWS EC2 vs. AWS EMR (100GB)

Statistical analysis with T-Test for data sizes of 1GB, 10GB and 100GB WordCount benchmark test results indicates that the differences in Response time and Throughput for each dataset (1GB, 10GB and 100GB) on varying nodes of 1 to 8 were found to be statistically significant with a p-value of less than 0.05 (Table 5 and Table 6).

Also the test results indicate that the Amazon EC2 cloud performed better than the Amazon EMR cloud service for data sizes of 1GB and 10GB but for larger data size of 100GB, Amazon EMR performed better than Amazon EC2 in terms of response times. Similar pattern is also seen in the throughput values.

From Figure 9 and Figure 10 we conclude that Amazon EC2 is performing better than Amazon EMR. Figure 11 indicates that when the datasize is increased to 100GB, performance of Amazon EMR is significantly better than Amazon EC2.

6.1.3 Amazon EC2 and EMR Performance for TeraSort benchmark

Table 7 and Table 8 show the Response time and Throughput values for Amazon EC2 and Amazon EMR respectively. Figures 12, 13 and 14 present the plotted response times and throughput values for TeraSort benchmark performance on Amazon EC2 and Amazon EMR.

TERASORT						
Data size	1GB		10GB		100GB	
#nodes	EC2	EMR	EC2	EMR	EC2	EMR
1	141.919	217.436	435.153	568.197	4772.874	4077.234
2	79.861	130.358	233.105	300.937	3706.249	2034.876
3	57.802	98.117	206.128	219.835	2041.644	1192.721
4	50.757	81.5	188.335	193.827	1501.415	1082.909
5	48.759	81.795	174.888	185.84	1397.962	1054.86
6	37.786	73.79	169.003	175.776	1151.253	949.861
7	34.767	66.426	143.915	152.785	1137.792	928.615
8	33.829	65.798	128.991	139.794	1110.719	917.558
P-value	0.000125555		0.043215098		0.01498451	

Table 7: TeraSort: Response Time (seconds) – Amazon EC2 vs. Amazon EMR

TERASORT						
Data size	1GB		10GB		100GB	
#nodes	EC2	EMR	EC2	EMR	EC2	EMR
1	7.215379	4.709429	23.531941	18.021909	38.841957	46.974741
2	12.822288	7.855296	43.852495	34.027042	42.887788	51.894589
3	17.715663	10.436527	55.826915	46.580372	50.155634	57.781176
4	20.174572	12.564426	61.498892	56.628690	68.202295	85.023325
5	21.001266	12.519112	63.098644	61.746238	73.249451	91.293951
6	27.100004	13.877229	66.755736	63.569012	79.409778	97.074445
7	29.453238	15.415662	69.441592	67.410955	89.045156	101.212000
8	30.269909	15.562794	70.872872	68.360522	92.192489	111.600519
P-value	0.000684		0.003932		0.000104	

Table 8: TeraSort: Throughput (MB/seconds) – AWS EC2 vs. AWS EMR

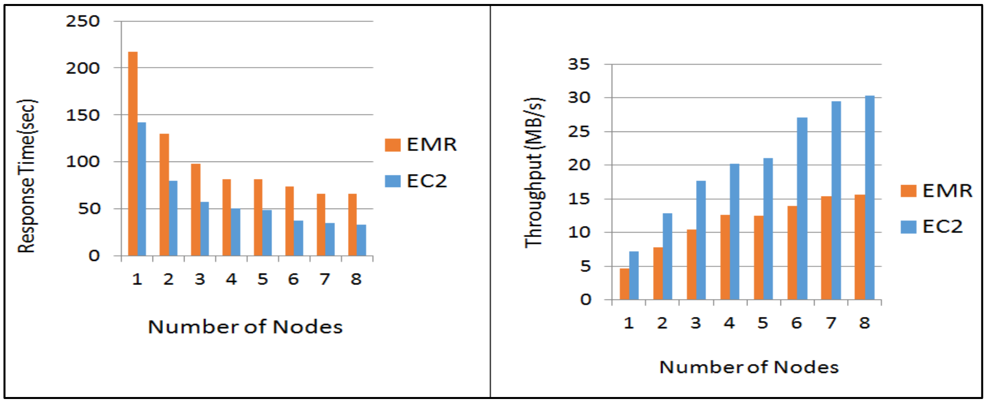


Figure 12: TeraSort – AWS EC2 vs. AWS EMR (1GB)

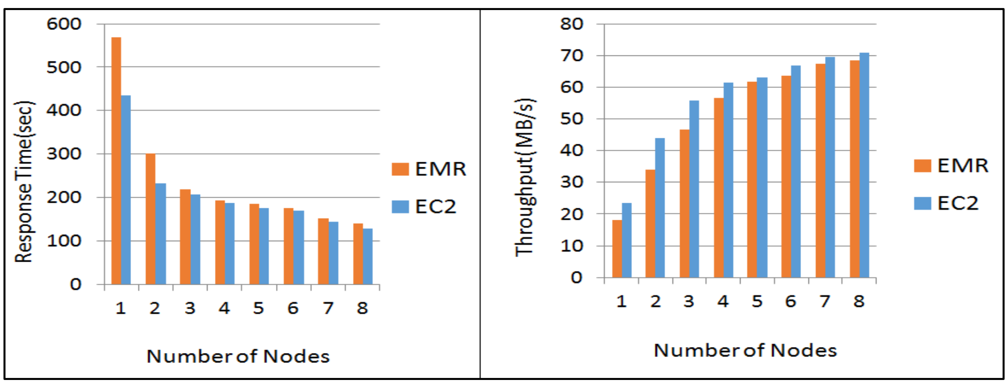


Figure 13: TeraSort – AWS EC2 vs. AWS EMR (10GB)

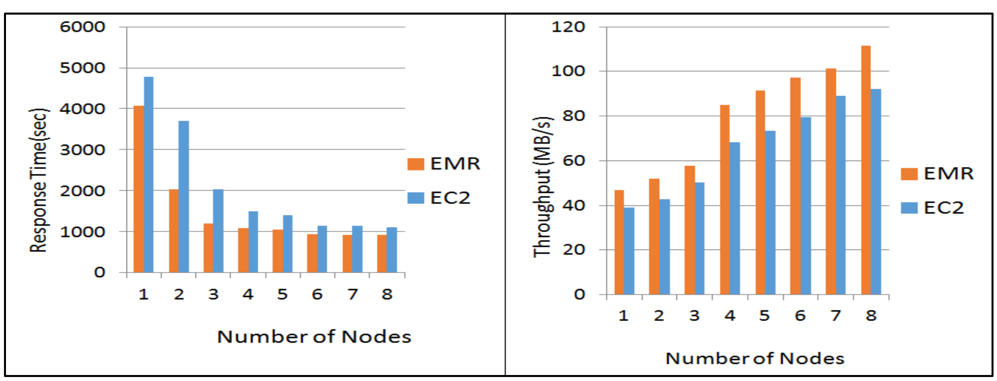


Figure 14: TeraSort – AWS EC2 vs. AWS EMR (100GB)

Statistical analysis with T-Test for data sizes of 1GB, 10GB and 100GB TeraSort benchmark test results indicate that the differences in Response time and Throughput for each datasets (1GB, 10GB and 100GB) on varying nodes of 1 to 8 were found to be statistically significant with a p-value of less than 0.05 (Table 7 and Table 8).
Also the test results indicate that the Amazon EC2 cloud performed better than the Amazon EMR cloud service for data sizes of 1GB and 10GB but for larger data size of 100GB, Amazon EMR performed better than Amazon EC2 in terms of response times. Similar pattern is also seen in the throughput values.

From Figure 12 and Figure 13 we conclude that Amazon EC2 is performing better than Amazon EMR. Figure 14 indicates that when the datasize is increased to 100GB, performance of Amazon EMR is significantly better than Amazon EC2.

6.2 Web Search Benchmark

6.2.1 Amazon EC2 and EMR Performance for PageRank Benchmark

Table 9 and Table 10 show the Response time and Throughput values for Amazon EC2 and Amazon EMR respectively. Figures 15, 16 and 17 present the plotted response times and throughput values for PageRank benchmark performance on Amazon EC2 and Amazon EMR.

PAGE RANK						
Data size	PAGES=100000		PAGES=1000000		PAGES=10000000	
#nodes	EC2	EMR	EC2	EMR	EC2	EMR
1	237.249	414.082	428.006	590.163	2635.857	2011.837
2	136.084	239.756	229.716	305.56	1385.46	1069.471
3	102.07	185.172	169.714	236.515	972.505	765.41
4	84.991	154.173	134.643	191.506	748.718	586.472
5	84.03	138.526	124.639	173.498	665.708	533.651
6	65.016	127.154	102.628	155.443	539.797	433.785
7	62.005	120.546	93.662	146.74	504.606	414.661
8	60.971	111.72	89.625	132.542	482.406	382.664
P-value	0.000855784		0.001379679		0.011394124	

Table 9: PageRank: Response Time (seconds) – AWS EC2 vs. AWS EMR

PAGE RANK						
Data size	PAGES=100000		PAGES=1000000		PAGES=10000000	
#nodes	EC2	EMR	EC2	EMR	EC2	EMR
1	0.067532	0.038692	0.433854	0.352804	0.795236	1.041898
2	0.117735	0.066826	0.808356	0.607711	1.512948	1.959968
3	0.156969	0.086524	1.094148	0.785118	2.155391	2.738570
4	0.188512	0.103921	1.379146	0.969642	2.799624	3.574133
5	0.190668	0.115659	1.489841	1.070285	3.148721	3.927902
6	0.246429	0.126003	1.809373	1.194601	3.883179	4.723298
7	0.258396	0.132911	1.982579	1.265451	4.153991	5.055042
8	0.262778	0.143411	2.071881	1.401007	4.345155	5.478013
P-value	0.000260		0.001116		0.000169	

Table 10: PageRank: Throughput (MB/seconds) – AWS EC2 vs. AWS EMR

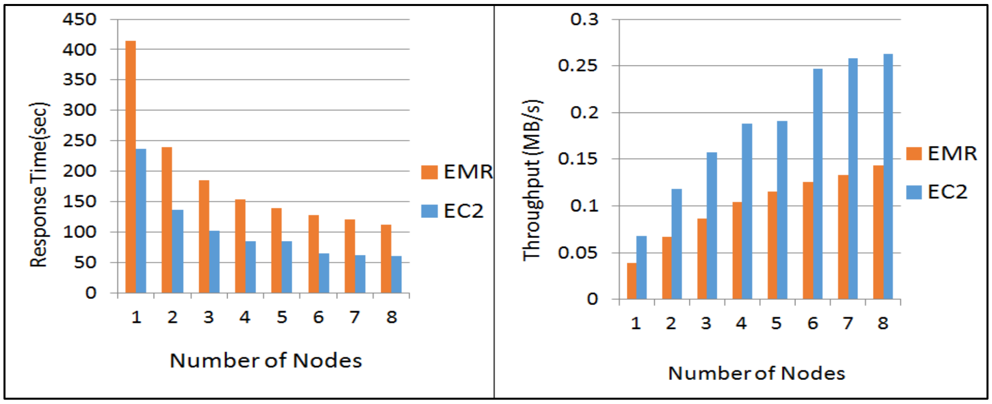


Figure 15: PageRank – AWS EC2 vs. AWS EMR (100000 Pages).

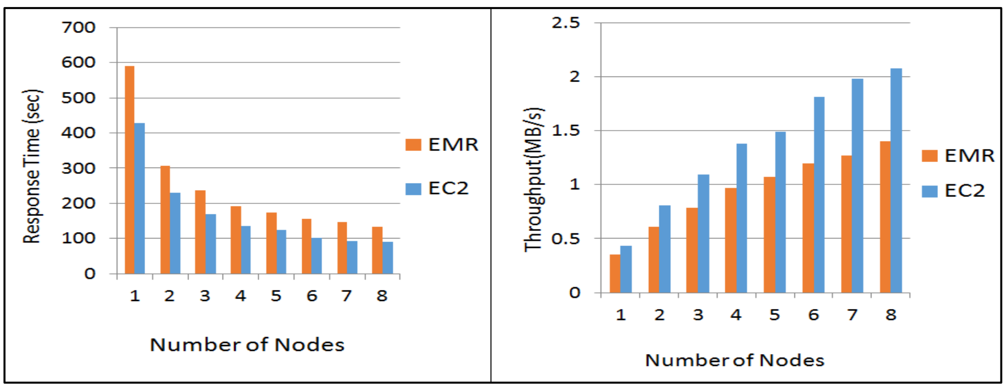


Figure 16: PageRank – AWS EC2 vs. AWS EMR (1000000 Pages).

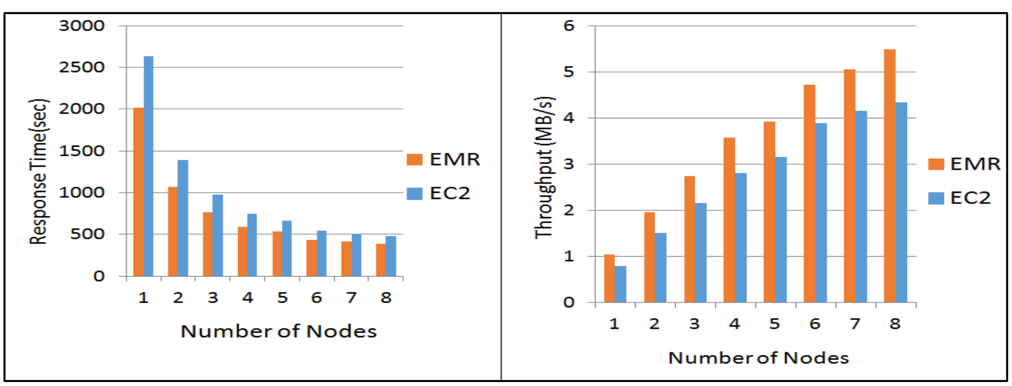


Figure 17: PageRank – AWS EC2 vs. AWS EMR (10000000 Pages).

Statistical analysis with T-Test for data sizes of 1GB, 10GB and 100GB PageRank benchmark test results indicate that the differences in Response time and Throughput for each datasets (1GB, 10GB and 100GB) on varying nodes of 1 to 8 were found to be statistically significant with a p-value of less than 0.05 (Table 9 and Table 10).
Also the test results indicate that the Amazon EC2 cloud performed better than the Amazon EMR cloud service for data sizes of 1GB and 10GB but for larger data size of 100GB, Amazon EMR performed better than Amazon EC2 in terms of response times. Similar pattern is also seen in the throughput values.

From Figure 15 and Figure 16 we conclude that Amazon EC2 is performing better than Amazon EMR. Figure 17 indicates that when the datasize is increased to 100GB, performance of Amazon EMR is significantly better than Amazon EC2.

6.3 Analytical Query Benchmarks

6.3.1 Amazon EC2 and EMR Performance for Hive Join Benchmark

Table 11 and Table 12 show the Response time and Throughput values for Amazon EC2 and Amazon EMR respectively. Figures 18, 19 and 20 present the plotted response times and throughput values for Hive Join benchmark performance on Amazon EC2 and Amazon EMR.

HIVE JOIN						
Data size	USERVISITS=1000000 PAGES=600000		USERVISITS=10000000 PAGES=6000000		USERVISITS=100000000 PAGES=60000000	
#nodes	EC2	EMR	EC2	EMR	EC2	EMR
1	212.977	438.379	325.872	496.786	1005.85	988.224
2	139.555	296.381	232.214	342.805	601.309	596.259
3	113.582	254.365	206.048	298.041	492.683	465.918
4	100.074	229.463	191.099	271.871	407.333	392.035
5	97.372	218.134	185.204	256.597	398.057	354.6
6	86.203	207.355	176.935	246.723	366.065	334.557
7	84.105	204.527	169.979	237.474	352.066	321.241
8	82.084	195.978	168.082	226.963	324.746	302.633
P-value	0.000012477		0.00021051		0.000669366	

Table 11: Hive Join: Response Time (seconds) – AWS EC2 vs. AWS EMR

HIVE JOIN						
Data size	USERVISITS=1000000 PAGES=600000		USERVISITS=10000000 PAGES=6000000		USERVISITS=100000000 PAGES=60000000	
#nodes	EC2	EMR	EC2	EMR	EC2	EMR
1	0.391939	0.190415	2.560921	1.679863	8.296048	8.444017
2	0.598145	0.281645	3.593808	2.434424	13.877358	13.994892
3	0.734924	0.328167	4.050185	2.800060	16.937017	17.909976
4	0.834124	0.363780	4.367017	3.069590	20.485893	21.285294
5	0.831647	0.382673	4.506018	3.252308	20.963280	23.532375
6	0.968343	0.402566	4.716606	3.382468	22.795351	24.942178
7	0.973575	0.408132	4.909622	3.514206	23.701750	25.976075
8	1.000152	0.425936	5.068703	3.676954	25.695714	27.573266
P-value	0.000069		0.004796		0.005515	

Table 12: Hive Join: Throughput (MB/seconds) – AWS EC2 vs. AWS EMR

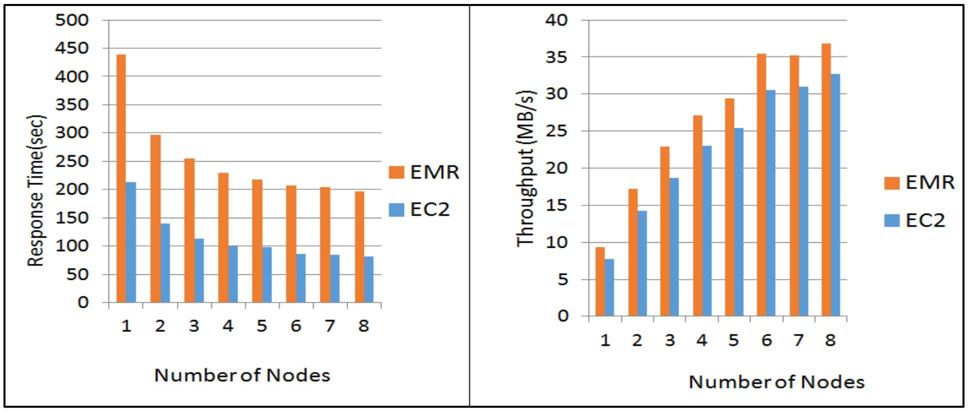


Figure 18: Hive Join – AWS EC2 vs. AWS EMR (1000000, 600000)

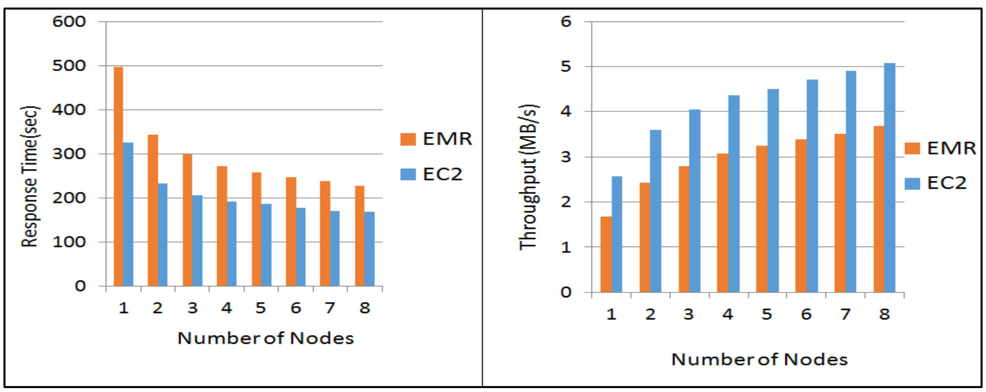


Figure 19: Hive Join – AWS EC2 vs. AWS EMR (10000000, 6000000)

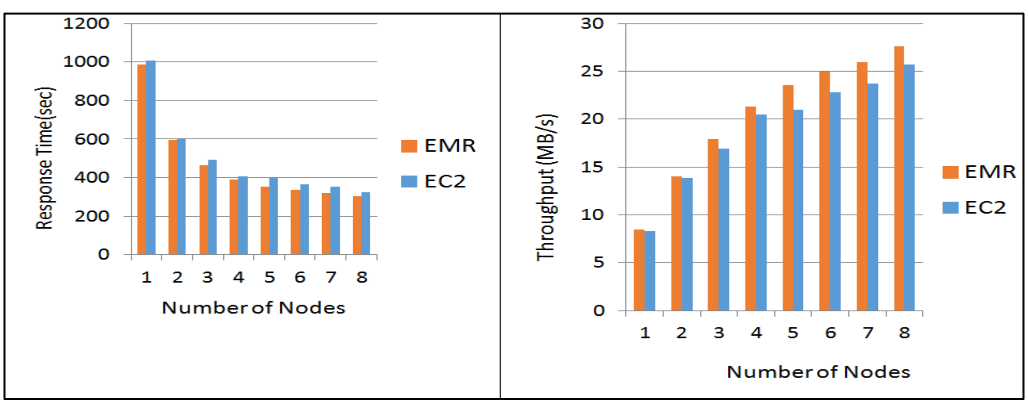


Figure 20: Hive Join – AWS EC2 vs. AWS EMR (100000000, 60000000)

Statistical analysis with T-Test for data sizes of 1GB, 10GB and 100GB Hive Join benchmark test results indicate that the differences in Response time and Throughput for each datasets (1GB, 10GB and 100GB) on varying nodes of 1 to 8 were found to be statistically significant with a p-value of less than 0.05 (Table 11 and Table 12).
Also the test results indicate that the Amazon EC2 cloud performed better than the Amazon EMR cloud service for data sizes of 1GB and 10GB but for larger data size of 100GB, Amazon EMR performed better than Amazon EC2 in terms of response times. Similar pattern is also seen in the throughput values.

From Figure 18 and Figure 19 we conclude that Amazon EC2 is performing better than Amazon EMR. Figure 20 indicates that when the datasize is increased to 100GB, performance of Amazon EMR is significantly better than Amazon EC2.

6.3.2 Amazon EC2 and EMR Performance for Hive Aggregation Benchmark

Table 13 and Table 14 show the Response time and Throughput values for Amazon EC2 and Amazon EMR respectively. Figures 21, 22 and 23 present the plotted response times and throughput values for Hive Aggregation benchmark performance on Amazon EC2 and Amazon EMR.

HIVE AGGREGATION						
Data size	USERVISITS=1000000 PAGES=600000		USERVISITS=10000000 PAGES=6000000		USERVISITS=100000000 PAGES=60000000	
#nodes	EC2	EMR	EC2	EMR	EC2	EMR
1	114.241	190.076	182.326	222.487	757.439	629.469
2	74.833	121.193	122.613	150.266	410.428	341.373
3	59.709	97.942	106.445	124.895	312.87	255.964
4	53.711	88.506	97.445	110.119	254.585	216.602
5	52.706	81.922	96.396	109.438	230.27	199.396
6	45.607	75.678	89.424	102.394	192.116	165.451
7	45.6	73.822	87.334	98.926	182.022	162.224
8	45.57	68.617	85.418	96.013	179.141	159.215
P-value	0.000352441		0.001563433		0.007024392	

Table 13: Hive Aggregation: Response Time (seconds)– AWS EC2 vs. AWS EMR

HIVE AGGREGATION						
Data size	USERVISITS=1000000 PAGES=600000		USERVISITS=10000000 PAGES=6000000		USERVISITS=100000000 PAGES=60000000	
#nodes	EC2	EMR	EC2	EMR	EC2	EMR
1	0.513319	0.308519	3.215524	2.635092	7.739821	9.313314
2	0.783640	0.483874	4.781497	3.901572	14.283730	17.173129
3	0.982132	0.598743	5.507762	4.694133	18.737631	22.903387
4	1.091808	0.662578	6.016457	5.324001	23.027447	27.065506
5	1.112627	0.715829	6.081930	5.357131	25.458994	29.401004
6	1.285814	0.774890	6.556111	5.725664	30.515119	35.433105
7	1.286011	0.794372	6.713006	5.926386	30.530057	35.268328
8	1.296395	0.854630	6.7761229	6.106191	32.725298	36.820919
P-value	0.000011		0.00000011954		0.000023	

Table 14: Hive Aggregation: Throughput – AWS EC2 vs. AWS EMR

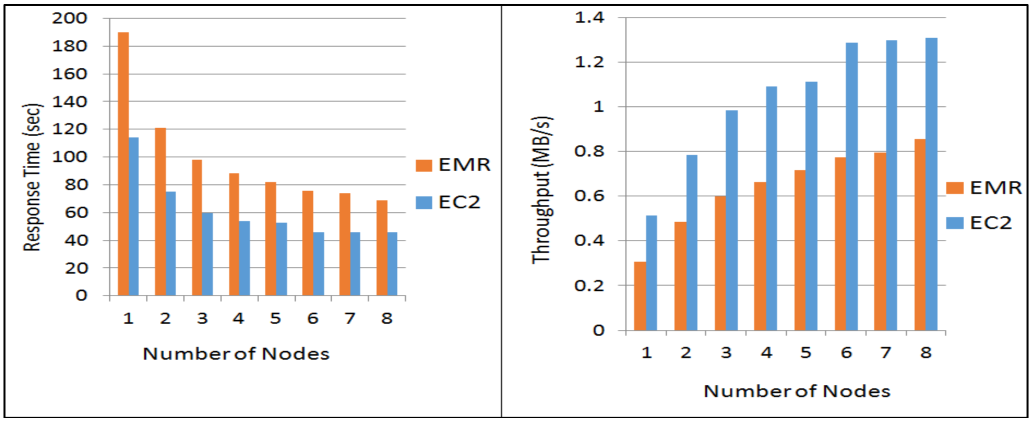


Figure 21: Hive Aggregation – AWS EC2 vs. AWS EMR (1000000, 600000)

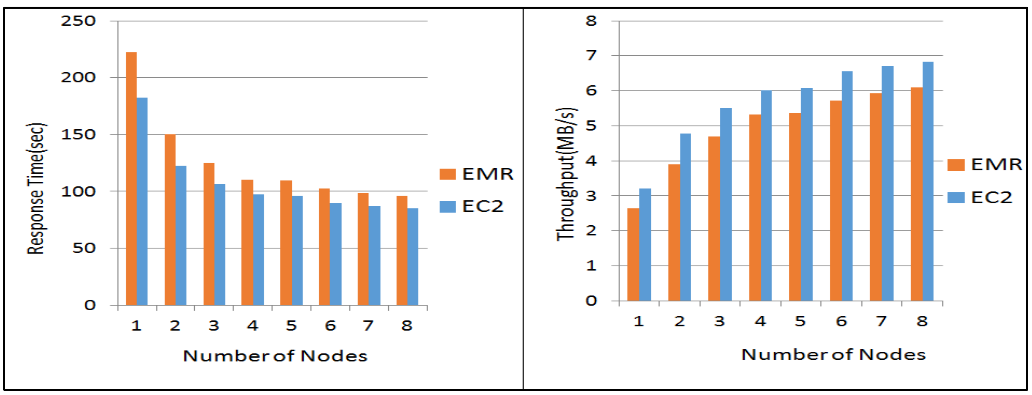


Figure 22: Hive Aggregation – AWS ECS vs. AWS EMR (10000000, 6000000)

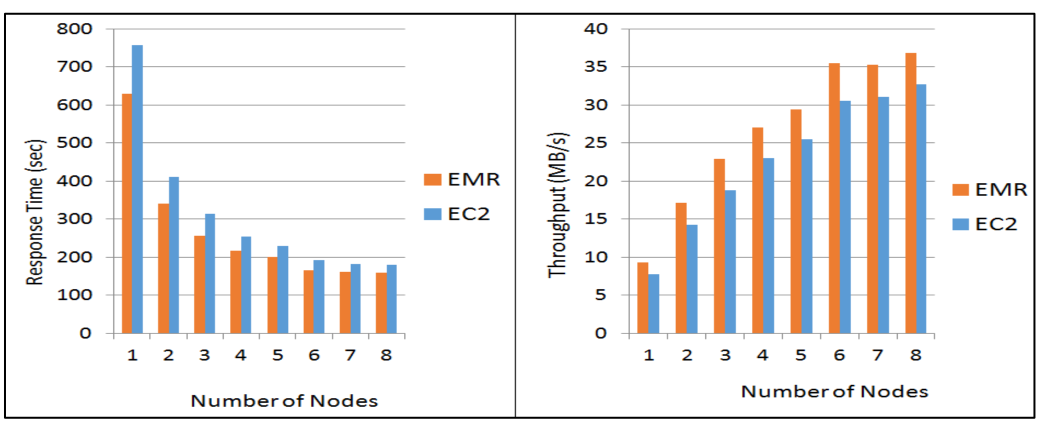


Figure 23: Hive Aggregation – AWS EC2 vs. AWS EMR (100000000, 60000000)

Statistical analysis with T-Test for data sizes of 1GB, 10GB and 100GB Hive Aggregation benchmark test results indicate that the differences in Response time and Throughput for each datasets (1GB, 10GB and 100GB) on varying nodes of 1 to 8 were found to be statistically significant with a p-value of less than 0.05 (Table 13 and Table 14). Also the test results indicate that the Amazon EC2 cloud performed better than the Amazon EMR cloud service for data sizes of 1GB and 10GB but for larger data size of 100GB, Amazon EMR performed better than Amazon EC2 in terms of response times. Similar pattern is also seen in the throughput values.

From Figure 21 and Figure 22 we conclude that Amazon EC2 is performing better than Amazon EMR. Figure 23 indicates that when the datasize is increased to 100GB, performance of Amazon EMR is significantly better than Amazon EC2.

Chapter 7

CONCLUSIONS

7.1 Benchmark Results

The Amazon EC2 and Amazon EMR cloud services were tested using the HiBench benchmark suite while the number of nodes (1 to 8) and the size of the dataset (1GB, 10GB, and 100GB) were varied. Overall, it appeared that Amazon EC2 was well suited for less data intensive applications for data size less than 100 GB. The results of datasets of 1GB and 10GB run on m3.2xlarge instance showed this behavior. When we move over to higher benchmark workloads of 100 GB, Amazon EMR preformed better than Amazon EC2. This can be attributed to the fact that Amazon EMR installation of Hadoop containing patches and improvements added to Apache Hadoop to make it work effectively on AWS. This also includes using better compression codecs and fixes to better combine and split input files and better performance tuning of running clusters on Amazon EMR. The configuration settings of Hadoop used for Amazon EMR cluster are optimized for scalability and more data intensive applications thus explaining why Amazon EMR performed better than Amazon EC2 on larger data sets.

For Sort, TeraSort, Page Rank and Hive Aggregate benchmarks, the difference in response time between Amazon EC2 and Amazon EMR and the difference in throughput

between Amazon EC2 and Amazon EMR was more significant than in WordCount and Hive Join benchmarks as the former contains more data intensive and I/O operations compared to the latter.

Certain advantages that Amazon EMR has over Amazon EC2 is that Amazon EMR can be used for large scale data processing that includes a lot of setting and configuration work as Amazon steps forward to remove that extra work out for the customers. Also Amazon takes care of cluster monitoring, resource management, cluster start-up and shutdown and even security groups management in case of Amazon EMR. In most of the cases it is hard to tune the performance of running clusters but in case of Amazon EMR, it takes care of performance tuning of the clusters while running a job or a workload. Even Hadoop is made simple and easy by Amazon EMR. Certain benefits of EMR are:

- Elastic: Amazon EMR uses a cluster of EC2 instances that are scalable. Also spins large or small job flows in minutes.
- Easy to use: Easy to run jobs quickly using the web console. No detailed configuration is required.
- Reliable: Fault tolerant service built on top of the Amazon Web Service (AWS) infrastructure.
- Cost Effective: Amazon monitors the progress of each job flow and turns off the resources when job flow is done.

From a scaling and cost perspective, for higher workloads and large number of nodes to be managed, it is better to opt for Amazon EMR than Amazon EC2 even though the cost of Amazon EMR is higher than that of EC2. Amazon EMR automatically takes care of performance tuning of running clusters, cluster monitoring, resource management and security groups management. It is also fault tolerant and it automatically retires failed tasks as well. However in Amazon EC2, all these will have to be done manually. There is less overhead in Amazon EMR compared to Amazon EC2. Whereas in case of small datasets and applications that doesn't need much scalability and need to operate on low cost, Amazon EC2 is a better option.

7.2 Pricing Models

Table 15 provides a basic insight into the pricing of Amazon EMR and Amazon EC2 for an m3.2xlarge instance. Amazon EC2 has a base price of $0.56/hr per instance whereas Amazon EMR pricing is cost of an Amazon EC2 instance which is $0.56/hr per instance plus the cost that Amazon charges for cluster management for Amazon EMR which is $0.14/hr totaling $0.70/hr. As the number of nodes is increased, the variation becomes more significant as shown in Figure 24 below. The variation becomes drastically significant when the number of nodes is multiplied by the number of hours and the price per instance.

Nodes	Amazon EC2 (per hour) m3.2xlarge instance	Amazon EMR(per hour) m3.2xlarge instance
1	$0.56	$0.70
2	$1.12	$1.40
3	$1.68	$2.10
4	$2.24	$2.80
5	$2.80	$3.50
6	$3.36	$4.20
7	$3.92	$4.90
8	$4.48	$5.60

Table 15: Pricing of Amazon EC2 vs. Amazon EMR

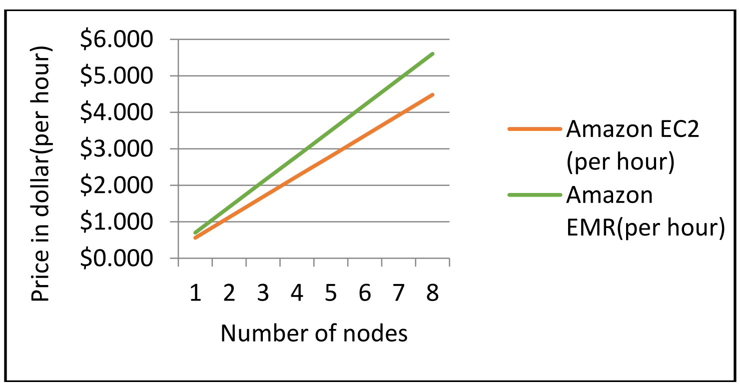


Figure 24: Pricing of Amazon EC2 vs. Amazon EMR

7.3 Future Research

This study is limited to benchmarking the two cloud services provided by Amazon, Amazon EC2 and Amazon EMR cloud services to evaluate the performance of Hadoop

on data intensive applications while varying workloads and the number of nodes in the cluster.

Extensions to this study on cloud performance include evaluating the performance of these cloud service on big data level that is, varying the sizes upto terabytes of data. This may help the research to evaluate the performance pattern of Hadoop on each node for both the cloud services thus helping in further analysis.

Also, in this thesis research we utilized m3.2xlarge instances provided by Amazon, which are high memory optimized instances. So further studies can be conducted on various instance types provided by Amazon, such as compute optimized instances (C3 instances), storage optimized instances (I2 instances) and Graphic optimized instances (G2 instances, to further explore the benchmarking on Amazon cloud platform.

Another scope of further research is in terms of new benchmarks to be used for evaluating the performance. The research utilizes HiBench benchmark suite which is a set of Hadoop benchmarks. The Hadoop performance on data intensive applications may be investigated using new benchmarks.

REFERENCES

Print Publications:

[Hedger11]
Dominique A. Hedger, “Hadoop Design, Architecture & MapReduce Performance”, DHT Technologies, 2011.

[Huang10]
Shengsheng Huang, Jie Huang, Yan Liu, Lan Yi and Jinquan Dai, “HiBench: A Representative and Comprehensive Hadoop Benchmark Suite”, Intel Asia-Pacific Research and Development Ltd., Shanghai, P.R.China, 2010.

[Huang10]
Shengsheng Huang, Jie Huang, Jinquan Dai, Tao Xie, and Bo Huang, “The HiBench Benchmark Suite: Characterization of the MapReduce-Based Data Analysis”, Conference: Data Engineering Workshops (ICDEW), Intel China Software Center, Shanghai, China, 2010.

[Peter11]
Peter Mell and Timothy Grance, “ The NIST Definition of Cloud Computing”, Recommendations of the National Institute of Standards and Technology, Special Publication 800-145, September 2011.

[Sarda11]
Kalpit Sarda, Sumit Sanghrajka, and Radu Sion,“Cloud Performance Benchmark -Amazon EC2 vs. Rackspace”, Cloud Commons Online, Stony Brook Network Security and Applied Cryptography Lab, 2011.

Electronic Sources:

[AWS14]
"What is Amazon EMR",
http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-what-is-emr., last accessed November 20, 2014.

[Hadoop13]
"HDFS Architecture Guide", http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html, last accessed November 20, 2014.

[MapReduce14]
"MapReduce: Overview", http://gppd-wiki.inf.ufrgs.br/index.php/MapReduce, last accessed November 20, 2014.

[Noll11]
Noll, M.G., "Running Hadoop on Ubuntu Linux (Multi-Node Cluster)", http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/, last revision July 7, 2011, last accessed November 20, 2014.

[StarCluster14]
"Installing StarCluster", http://star.mit.edu/cluster/docs/0.93.3/installation.html, last accessed November 20, 2014.

[Wang14]
"Hadoop Benchmark Suite (HiBench)", https://github.com/intel-hadoop/HiBench/, last accessed November 20, 2014.

Appendix A

Hadoop Configuration Files

All components in Hadoop are configured using XML files. Most common properties go in *core-site.xml* file, HDFS properties go in *hdfs-site.xml* file, and MapReduce properties go in *mapred-site.xml* file and Hadoop environment properties go into *hadoop-env.sh* file. All these files are located under Hadoop Conf sub directory under Hadoop installation directory.

1. hadoop-env.sh

```
# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_DATANODE_HEAPSIZE="1064"
export HADOOP_JOBTRACKER_HEAPSIZE="6758"
export HADOOP_NAMENODE_HEAPSIZE="3276"
export HADOOP_TASKTRACKER_HEAPSIZE="839"
export HADOOP_OPTS="$HADOOP_OPTS -server"

# The java implementation to use.  Required.
# export JAVA_HOME=/usr/lib/j2sdk1.5-sun

# Extra Java CLASSPATH elements.  Optional.
# export HADOOP_CLASSPATH=

# The maximum amount of heap to use, in MB. Default is 1000.
# export HADOOP_HEAPSIZE=2000

# Extra Java runtime options.  Empty by default.
# export HADOOP_OPTS=-server

# Command specific options appended to HADOOP_OPTS when specified
export HADOOP_NAMENODE_OPTS="-Dcom.sun.management.jmxremote $HADOOP_NAMENODE_OPTS"
export HADOOP_SECONDARYNAMENODE_OPTS="-Dcom.sun.management.jmxremote $HADOOP_SECONDARYNAMENODE_OPTS"
export HADOOP_DATANODE_OPTS="-Dcom.sun.management.jmxremote $HADOOP_DATANODE_OPTS"
export HADOOP_BALANCER_OPTS="-Dcom.sun.management.jmxremote $HADOOP_BALANCER_OPTS"
export HADOOP_JOBTRACKER_OPTS="-Dcom.sun.management.jmxremote $HADOOP_JOBTRACKER_OPTS"
```

2. mapred-site.xml

```xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
   <name>mapred.child.java.opts</name>
  <value>-Xmx2048m</value>
</property>
<property><name>mapred.reduce.tasks.speculative.execution</name><value>true</value></property>
<property><name>mapred.output.committer.class</name>
<value>org.apache.hadoop.mapred.FileOutputCommitter</value></property>
<property><name>mapred.tasktracker.map.tasks.maximum</name><value>12</value></property>
<property><name>mapred.map.tasks.speculative.execution</name><value>true</value></property>
<property><name>mapred.task.tracker.http.address</name><value>0.0.0.0:9103</value></property>
<property><name>mapred.userlog.retain.hours</name><value>48</value></property>
<property><name>mapred.job.reuse.jvm.num.tasks</name><value>20</value></property>
<property><name>io.sort.factor</name><value>48</value></property>
<property><name>mapred.reduce.tasks</name><value>7</value></property>
<property><name>tasktracker.http.threads</name><value>80</value></property>
<property><name>mapred.reduce.parallel.copies</name><value>20</value></property>
<property><name>hadoop.job.history.user.location</name><value>none</value></property>
<property><name>mapred.job.tracker.handler.count</name><value>64</value></property>
<property><name>mapred.map.output.compression.codec</name>
<value>org.apache.hadoop.io.compress.DefaultCodec</value></property>
<property><name>mapred.output.direct.NativeS3FileSystem</name><value>true</value></property>
<property><name>mapred.reduce.tasksperslot</name><value>1.75</value></property>
<property><name>mapred.tasktracker.reduce.tasks.maximum</name><value>4</value></property>
<property><name>mapred.compress.map.output</name><value>true</value></property>
<property><name>mapred.output.compression.codec</name>
<value>org.apache.hadoop.io.compress.GzipCodec</value></property>
<property><name>mapred.job.tracker.http.address</name><value>0.0.0.0:9100</value></property>
<property><name>mapred.local.dir</name>
<value>/mnt/var/lib/hadoop/mapred,/mnt1/var/lib/hadoop/mapred</value></property>
<property><name>mapred.job.tracker</name><value>master:9001</value></property>
<property><name>io.sort.mb</name><value>200</value></property>
</configuration>
```

3. core-site.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<value>/mnt/var/lib/hadoop/s3,/mnt1/var/lib/hadoop/s3</value></property>
<property><name>io.compression.codecs</name>
<value>org.apache.hadoop.io.compress.GzipCodec,
org.apache.hadoop.io.compress.DefaultCodec,
org.apache.hadoop.io.compress.BZip2Codec,
org.apache.hadoop.io.compress.SnappyCodec</value></property>
<property><name>hadoop.metrics.defaultFile</name>
<value>/home/hadoop/conf/hadoopDefaultMetricsList</value></property>
<property><name>hadoop.metrics.list</name>
<value>TotalLoad,CapacityTotalGB,UnderReplicatedBlocks,
CapacityRemainingGB,PendingDeletionBlocks,PendingReplicationBlocks,
CorruptBlocks,CapacityUsedGB,numLiveDataNodes,
numDeadDataNodes,MissingBlocks</value></property>
</configuration>
```

4. hdfs-site.xml

```xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property><name>dfs.datanode.max.xcievers</name><value>4096</value></property>
<property><name>dfs.datanode.https.address</name><value>0.0.0.0:9402</value></property>
<property><name>dfs.datanode.du.reserved</name><value>536870912</value></property>
<property><name>dfs.namenode.handler.count</name><value>64</value></property>
<property><name>io.file.buffer.size</name><value>65536</value></property>
<property><name>dfs.block.size</name><value>134217728</value></property>
<property><name>dfs.data.dir</name>
<value>/mnt/var/lib/hadoop/dfs,/mnt1/var/lib/hadoop/dfs</value></property>
<property><name>dfs.secondary.http.address</name><value>0.0.0.0:9104</value></property>
<property><name>dfs.replication</name><value>1</value></property>
<property><name>dfs.https.address</name><value>0.0.0.0:9202</value></property>
<property><name>dfs.http.address</name><value>0.0.0.0:9101</value></property>
<property><name>dfs.datanode.http.address</name><value>0.0.0.0:9102</value></property>
<property><name>dfs.datanode.address</name><value>0.0.0.0:9200</value></property>
<property><name>dfs.name.dir</name>
<value>/mnt/var/lib/hadoop/dfs-name,
/mnt1/var/lib/hadoop/dfs-name</value></property>
<property><name>dfs.datanode.ipc.address</name><value>0.0.0.0:9201</value></property>
</configuration>
```

Appendix B

Cluster setup on Amazon EC2

StarCluster open source utility is used to create cluster on Amazon EC2. Below are the steps for creating a cluster using StarCluster on Amazon EC2 [StarCluster14].

Below screenshot shows installation of StarCluster.

```
root@ip-172-31-42-159:/home/ec2-user
login as: ec2-user
Authenticating with public key "imported-openssh-key"
Passphrase for key "imported-openssh-key":

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2014.09-release-notes/
7 package(s) needed for security, out of 18 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-42-159 ~]$ sudo su
[root@ip-172-31-42-159 ec2-user]# sudo easy_install StarCluster
Searching for StarCluster
Reading https://pypi.python.org/simple/StarCluster/
Best match: StarCluster 0.95.6
Downloading https://pypi.python.org/packages/source/S/StarCluster/StarCluster-0.
95.6.tar.gz#md5=efec9d81232f933af4609e81aa11152e
Processing StarCluster-0.95.6.tar.gz
Writing /tmp/easy_install-oxchhu/StarCluster-0.95.6/setup.cfg
Running StarCluster-0.95.6/setup.py -q bdist_egg --dist-dir /tmp/easy_install-ox
chhu/StarCluster-0.95.6/egg-dist-tmp-E4a7tZ
warning: no previously-included files found matching 'requirements.txt'
warning: no previously-included files matching '*' found under directory 'docs/s
phinx/_build'
Adding StarCluster 0.95.6 to easy-install.pth file
Installing starcluster script to /usr/bin

Installed /usr/lib/python2.6/site-packages/StarCluster-0.95.6-py2.6.egg
Processing dependencies for StarCluster
Searching for iso8601>=0.1.8
Reading https://pypi.python.org/simple/iso8601/
Best match: iso8601 0.1.10
Downloading https://pypi.python.org/packages/source/i/iso8601/iso8601-0.1.10.tar
.gz#md5=23acb1029acfef9c32069c6c851c3a41
Processing iso8601-0.1.10.tar.gz
Writing /tmp/easy_install-FvaslK/iso8601-0.1.10/setup.cfg
Running iso8601-0.1.10/setup.py -q bdist_egg --dist-dir /tmp/easy_install-FvaslK
/iso8601-0.1.10/egg-dist-tmp-LaJoR9
zip_safe flag not set; analyzing archive contents...
Adding iso8601 0.1.10 to easy-install.pth file

Installed /usr/lib/python2.6/site-packages/iso8601-0.1.10-py2.6.egg
Searching for scp>=0.7.1
Reading https://pypi.python.org/simple/scp/
Best match: scp 0.8.0
Downloading https://pypi.python.org/packages/source/s/scp/scp-0.8.0.tar.gz#md5=7
240f8717a8ba8abc1b935e16d4bd992
Processing scp-0.8.0.tar.gz
Writing /tmp/easy_install-CzgUnw/scp-0.8.0/setup.cfg
Running scp-0.8.0/setup.py -q bdist_egg --dist-dir /tmp/easy_install-CzgUnw/scp-
0.8.0/egg-dist-tmp-yWzW4V
```

StarCluster open source utility launching a two node cluster (master and slave node001).

```
[root@ip-172-31-42-159 ec2-user]# starcluster start pagerank100GB
/usr/lib64/python2.6/site-packages/Crypto/Util/number.py:57: PowmInsecureWarning
ity.
  _warn("Not using mpz_powm_sec.  You should rebuild using libgmp >= 5 to avoid
StarCluster - (http://star.mit.edu/cluster) (v. 0.95.6)
Software Tools for Academics and Researchers (STAR)
Please submit bug reports to starcluster@mit.edu

>>> Using default cluster template: smallcluster
>>> Validating cluster template settings...
>>> Cluster template settings are valid
>>> Starting cluster...
>>> Launching a 2-node cluster...
>>> Creating security group @sc-pagerank100GB...
Reservation:r-e780b5cd
>>> Waiting for instances to propagate...
2/2 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Waiting for cluster to come up... (updating every 30s)
>>> Waiting for all nodes to be in a 'running' state...
2/2 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Waiting for SSH to come up on all nodes...
2/2 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Waiting for cluster to come up took 1.573 mins
>>> The master node is ec2-54-164-189-145.compute-1.amazonaws.com
>>> Configuring cluster...
>>> Running plugin starcluster.clustersetup.DefaultClusterSetup
>>> Configuring hostnames...
2/2 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Creating cluster user: sgeadmin (uid: 1001, gid: 1001)
2/2 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Configuring scratch space for user(s): sgeadmin
2/2 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Configuring /etc/hosts on each node
2/2 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Starting NFS server on master
>>> Configuring NFS exports path(s):
/home
>>> Mounting all NFS export path(s) on 1 worker node(s)
1/1 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Setting up NFS took 0.032 mins
>>> Configuring passwordless ssh for root
>>> Configuring passwordless ssh for sgeadmin
>>> Running plugin starcluster.plugins.sge.SGEPlugin
>>> Configuring SGE...
>>> Configuring NFS exports path(s):
/opt/sge6
>>> Mounting all NFS export path(s) on 1 worker node(s)
1/1 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Setting up NFS took 0.019 mins
>>> Installing Sun Grid Engine...
0/1 |                                                                  |   0%
```

StarCluster open source utility successfully launching eight nodes in a cluster.

```
root@ip-172-31-42-159:/home/ec2-user
1/1 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Setting up NFS took 0.018 mins
>>> Updating SGE parallel environment 'orte'
8/8 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Adding parallel environment 'orte' to queue 'all.q'
[root@ip-172-31-42-159 ec2-user]# starcluster addnode pagerank100GB
/usr/lib64/python2.6/site-packages/Crypto/Util/number.py:57: PowmInsecureWarning
: Not using mpz_powm_sec.  You should rebuild using libgmp >= 5 to avoid timing
attack vulnerability.
  _warn("Not using mpz_powm_sec.  You should rebuild using libgmp >= 5 to avoid
timing attack vulnerability.", PowmInsecureWarning)
StarCluster - (http://star.mit.edu/cluster) (v. 0.95.6)
Software Tools for Academics and Researchers (STAR)
Please submit bug reports to starcluster@mit.edu

>>> Launching node(s): node008
Reservation:r-6e3b0f44
>>> Waiting for instances to propagate...
1/1 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Waiting for node(s) to come up... (updating every 30s)
>>> Waiting for all nodes to be in a 'running' state...
9/9 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Waiting for SSH to come up on all nodes...
9/9 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Waiting for cluster to come up took 1.187 mins
>>> Running plugin starcluster.clustersetup.DefaultClusterSetup
>>> Configuring hostnames...
1/1 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Configuring /etc/hosts on each node
9/9 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Configuring NFS exports path(s):
/home
>>> Mounting all NFS export path(s) on 1 worker node(s)
1/1 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Setting up NFS took 0.018 mins
1/1 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Configuring scratch space for user(s): sgeadmin
1/1 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Configuring passwordless ssh for root
>>> Configuring passwordless ssh for sgeadmin
>>> Running plugin starcluster.plugins.sge.SGEPlugin
>>> Adding node008 to SGE
>>> Configuring NFS exports path(s):
/opt/sge6
>>> Mounting all NFS export path(s) on 1 worker node(s)
1/1 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Setting up NFS took 0.018 mins
>>> Updating SGE parallel environment 'orte'
9/9 |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| 100%
>>> Adding parallel environment 'orte' to queue 'all.q'
[root@ip-172-31-42-159 ec2-user]#
```

Terminating an eight node StarCluster.

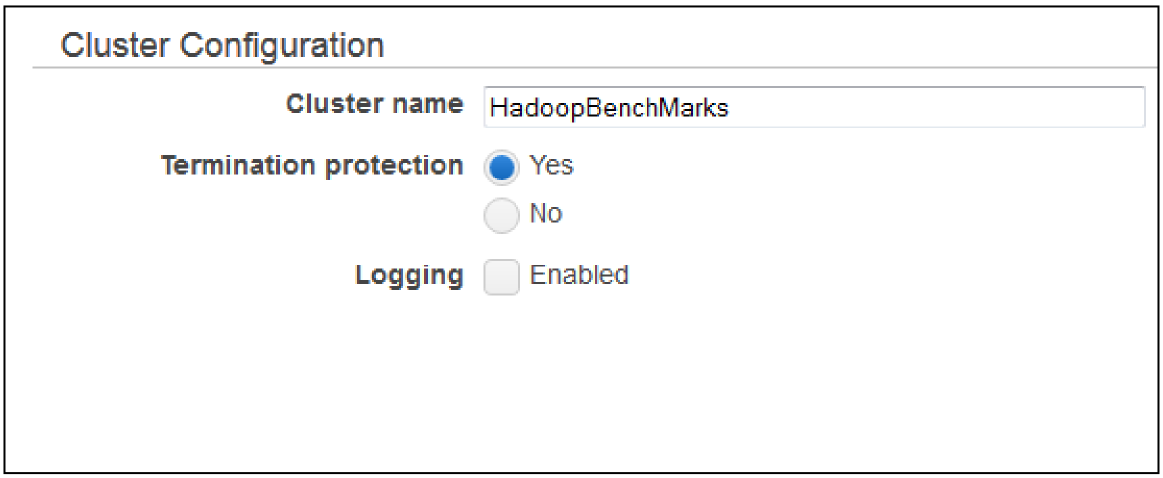
```
[root@ip-172-31-42-159 ec2-user]# starcluster terminate pagerank100GB
/usr/lib64/python2.6/site-packages/Crypto/Util/number.py:57: PowmInsecureWarning: Not using mpz_powm_sec.
 You should rebuild using libgmp >= 5 to avoid timing attack vulnerability.
  _warn("Not using mpz_powm_sec.  You should rebuild using libgmp >= 5 to avoid timing attack vulnerabilit
y.", PowmInsecureWarning)
StarCluster - (http://star.mit.edu/cluster) (v. 0.95.6)
Software Tools for Academics and Researchers (STAR)
Please submit bug reports to starcluster@mit.edu

Terminate EBS cluster pagerank100GB (y/n)? y
>>> Running plugin starcluster.plugins.sge.SGEPlugin
>>> Running plugin starcluster.clustersetup.DefaultClusterSetup
>>> Terminating node: master (i-9c937770)
>>> Terminating node: node001 (i-9d937771)
>>> Terminating node: node002 (i-fcc32710)
>>> Terminating node: node003 (i-d0da3e3c)
>>> Terminating node: node004 (i-842ace68)
>>> Terminating node: node005 (i-9505e179)
>>> Terminating node: node006 (i-600feb8c)
>>> Terminating node: node007 (i-2115f1cd)
>>> Terminating node: node008 (i-8f1df963)
>>> Waiting for cluster to terminate...
>>> Removing security group: @sc-pagerank100GB
[root@ip-172-31-42-159 ec2-user]#
```

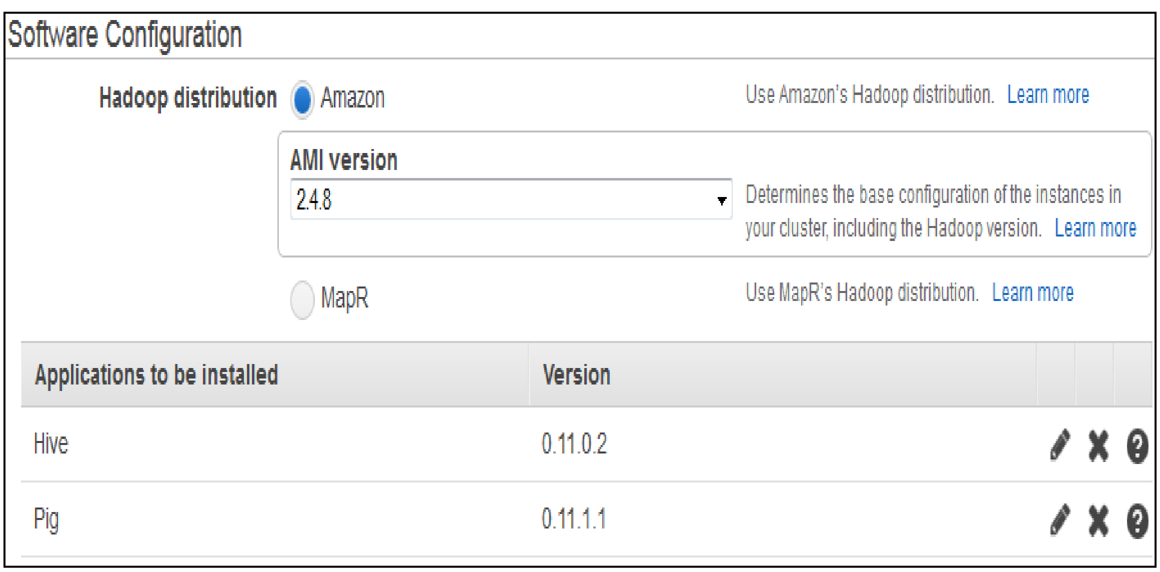
Appendix C

Cluster setup on Amazon EMR

1. Open the Amazon EMR console.
2. Click Create Cluster and Enter Cluster Name under Cluster Configuration.

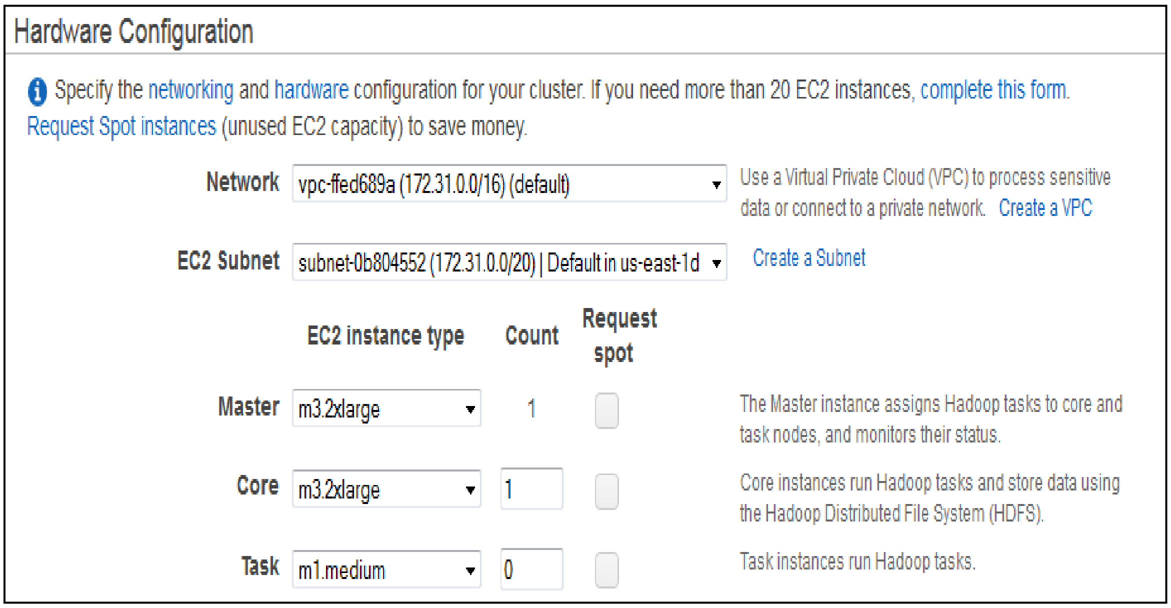


3. Under Software configuration, choose Hadoop 1.0.3 and 2.4.8 AMI.

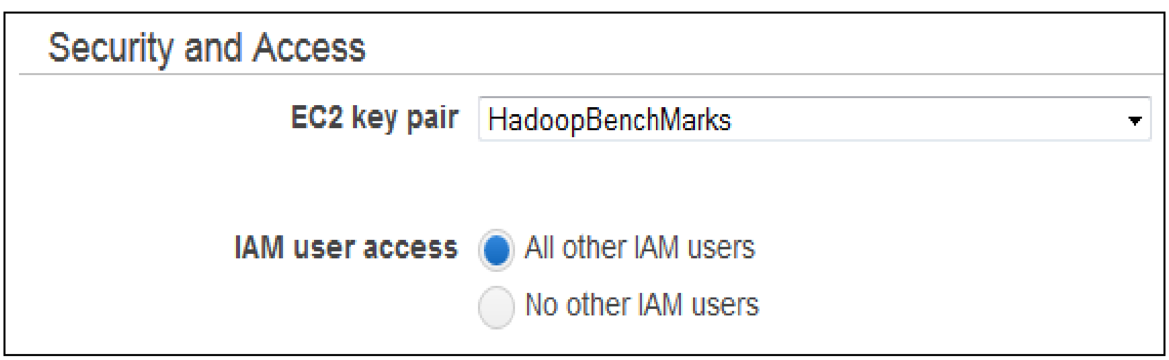


4. Under File System configuration, use the default settings for Server Side Encryption.

5. Under Hardware configuration, specify the EC2 master instance type and core instance type as m3.2xlarge.



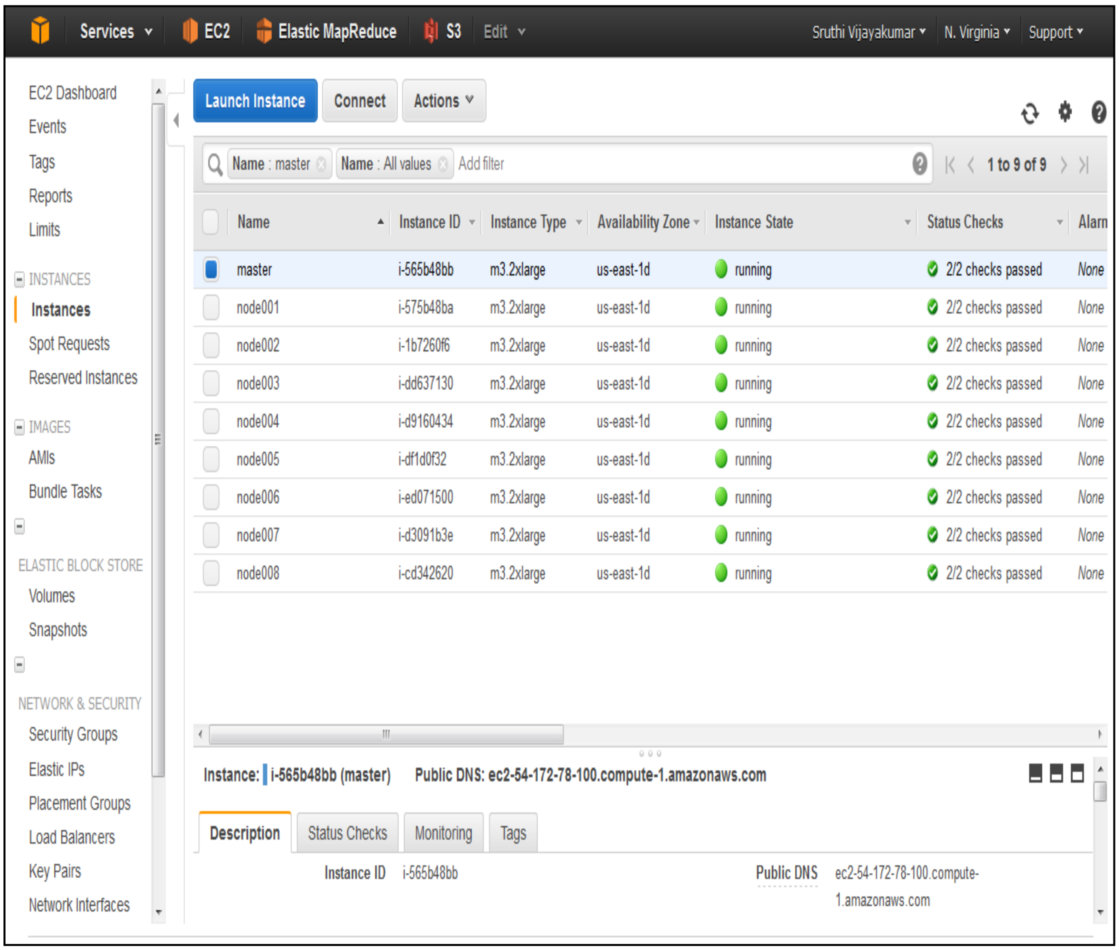
6. Under Security and Access, select the EC2 key pair used for the experiments.



7. Click Create Cluster

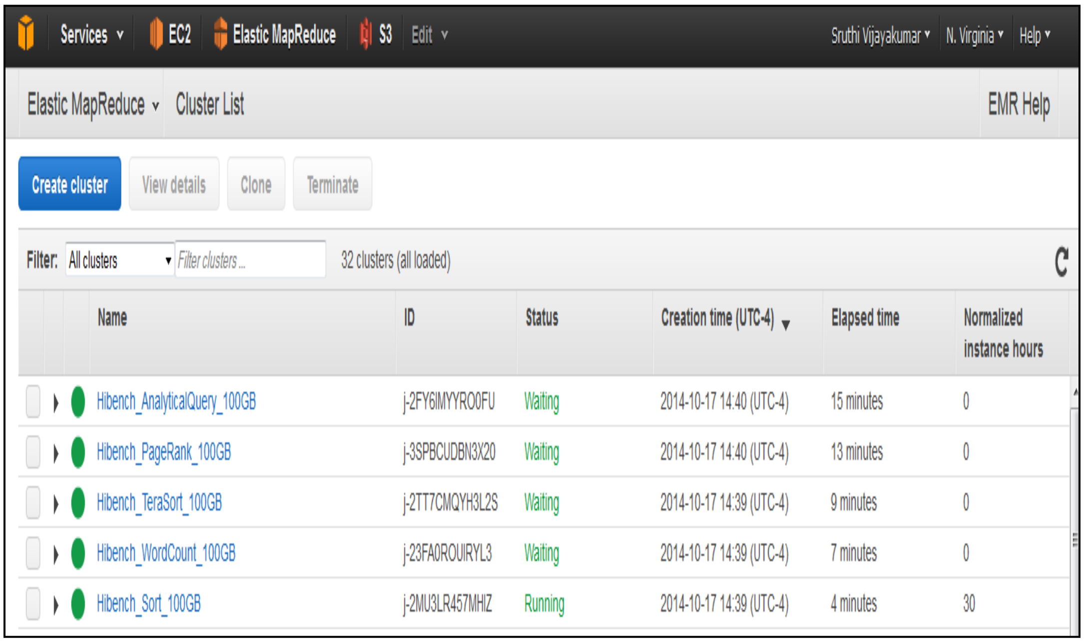
Appendix D

Amazon EC2 Screenshot



Appendix E

Amazon EMR Screenshot



VITA

Sruthi Vijayakumar received a Bachelor of Engineering in Information Technology from Amrita School of Engineering, India in 2011 and expects to receive the Master of Science degree in Computer and Information Sciences from the University of North Florida in May 2015. Dr. Sanjay Ahuja of the University of North Florida is Sruthi's thesis advisor. Sruthi worked as a Systems Engineer for two years at Infosys Limited, India for Telstra as client. Her work experience includes working on technologies like Java, J2EE, Oracle, Jasper Reporting and Android application development. She currently works as a Business Analyst for Allstate Benefits in Jacksonville, Florida. Sruthi aspires to work for a fortune 500 company where she can utilize her knowledge and skill set to the best possible extent.