BIG DATA ANALYTICS – HADOOP PERFORMANCE ANALYSIS

A Thesis

Presented to the

Faculty of

San Diego State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Ketaki Subhash Raste

Spring 2014

SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Ketaki Subhash Raste:

Big Data Analytics – Hadoop Performance Analysis

[illegible]

Carl Eckberg, Chair
Department of Computer Science

[illegible]

Robert Edwards
Department of Computer Science

[illegible]

Carmelo Interlando
Department of Mathematics and Statistics

April 15 2014

Approval Date

DEDICATION

I would like to dedicate this thesis to my family - my beloved husband Manish Sirdeshmukh for his endless support, to my loving daughter Ruta for keeping my spirit up with all the innocence, my parents Mr. Subhash Raste, Mrs. Vinaya Raste, and my brother Pushkar Raste for their constant encouragement to accomplish the thesis work. Last but not the least, this thesis is dedicated to my soon-to-arrive baby who has accompanied me through every effort and thought of this thesis.

ABSTRACT OF THE THESIS

Big Data Analysis – Hadoop Performance Analysis
by
Ketaki Subhash Raste
Master of Science in Computer Science
San Diego State University, 2014

The era of "Big Data" is upon us. From big consumer stores mining shopper data to Google using online search to predict incidence of the flu, companies and organizations are using troves of information to spot trends, combat crime, and prevent disease. Online and offline actions are being tracked, aggregated, and analyzed at dizzying rates. For example, questions like, how many calories we consumed for breakfast, how many we burned on our last run, and how long we spend using various applications on our computer, can be recorded and analyzed. We can lose weight by realizing we tend to splurge on Thursdays. We can be more efficient at work by realizing we spend time more than we thought on Facebook.

Data warehousing and data mining are related terms, as is NoSQL. With data firmly in hand and with the ability given by Big Data Technologies to effectively store and analyze this data, we can find answers to these questions and work to optimize every aspect of our behavior. Amazon can know every book you ever bought or viewed by analyzing big data gathered over the years. The NSA (National Security Agency) can know every phone number you ever dialed. Facebook can and will analyze big data and tell you the birthdays of people that you did not know you knew. With the advent of many digital modalities all this data has grown to BIG data and is still on the rise.

Ultimately Big Data technologies can exist to improve decision-making and to provide greater insights...faster when needed but with the downside of loss of data privacy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest thanks to Dr. Carl Eckberg for his constant guidance throughout the thesis work and being a chair for this thesis committee. This thesis would not have been possible without his timely suggestions despite of his busy schedule.

I take this opportunity to specially thank Mr. Saket Joshi (Manager Engineering @ OpenX) for his valuable insights on Big Data Technologies. I am grateful to Dr. Robert Edwards and Prof. Carmelo Interlando for serving on the thesis committee and their extended help. I would like to extend my thanks to the staff and all the faculty members of the Department of Computer Science.

Finally, my heartfelt gratitude to my loving friends and family for the moral support and encouragement throughout the graduation days.

CHAPTER 1

INTRODUCTION

Data has been a backbone of any enterprise and will do so moving forward. Storing, extracting and utilizing data has been key to many company’s operations. In the past when there were no interconnected systems, data would stay and be consumed at one place. With the onset of Internet technology, ability and requirement to share and transform data has been a need. This marks invention of ETL. ETL facilitated transforming, reloading and reusing the data. Companies have had significant investment in ETL infrastructure, both data warehousing hardware and software, personnel and skills.

1.1 Background, Motivation and Aim

With the advent of digital technology and smart devices, a large amount of digital data is being generated every day. Advances in digital sensors and communication technology have enormously added to this huge amount of data, capturing valuable information for enterprises, businesses. This Big data is hard to process using conventional technologies and calls for massive parallel processing. Technologies that are able to store and process exabytes, terabytes, petabytes of data without tremendously raising the data warehousing cost is a need of time. Ability to derive insights from this massive data has the potential to transform how we live, think and work. Benefits from Big data analysis range from healthcare domain to government to finance to marketing and many more [1].

Big data open source technologies have gained quite a bit of traction due to the demonstrated ability to parallelly process large amounts of data. Both parallel processing and technique of bringing computation to data has made it possible to process large datasets at high speed. These key features and ability to process vast data has been a great motivation to take a look into the architecture of the industry leading big data processing framework by Apache, Hadoop. Understand how this big data storage and analysis is achieved and experimenting with RDBMS vs Hadoop environment has proven to provide a great insight into much talked about technology.

Author of this thesis aims at understanding the dynamics involved in big data technologies mainly Hadoop, distributed data storage and analysis architecture of Hadoop, setup and explore Hadoop Cluster on Amazon Elastic Cloud. As well, conduct performance benchmarking on RDBMS and Hadoop cluster.

1.2 THESIS ORGANIZATION

The beginning chapters elaborate at length on a survey of Big Data terms and topics. But concrete experiments were made and are documented in the later chapters.

Initial chapters talk about why and how to utilize Big Data Technologies over conventional RDBMS, understand Hadoop framework for batch processing Big Data. And the later chapters aim at setting up Hadoop Cluster in the cloud, conduct performance analysis on the Hadoop cluster and compare against RDBMS.

- Chapter 2 takes a look into what exactly Big Data is all about and explains the importance of big data analytics.
- Chapter 3 will focus on the Hadoop architecture.
- Many more allied technologies have arisen to facilitate large data processing. Chapter 4 takes a look into these.
- Chapter 5 explains how to set up a Hadoop cluster on Amazon Web Services.
- Chapter 6 talks about performance analysis experiments performed on a Hadoop cluster
- Big data comes in with speed. In some business domains it's important to process this data almost instantaneously (e.g. think stock exchange, real time patient monitoring). Chapter 7 talks about real-time big data processing.
- Chapter 8 concludes the research

CHAPTER 2

WHAT AND WHY BIG DATA

The amount of data generated every day in the world is exploding. The increasing volume of digital and social media and internet of things, is fueling it even further. The rate of data growth is astonishing and this data comes at a speed, with variety (not necessarily structured) and contains wealth of information that can be a key for gaining an edge in competing businesses. Ability to analyze this enormous amount of data is bringing a new era of productivity growth, innovation and consumer surplus.

“Big data is the term for a collection of data sets so large and complex that it becomes difficult to process it using traditional database management tools or data processing applications. The challenges include the areas of capture, curation, storage, search, sharing, transfer, analysis, and visualization of this data” [2].

2.1 Big Data Attributes

The three Vs - volume, velocity and variety - are commonly used to describe different aspects of big data. See Figure 2.1 [3]. These three attributes make it easy to define the nature of the data and the software platforms available to analyze [4].

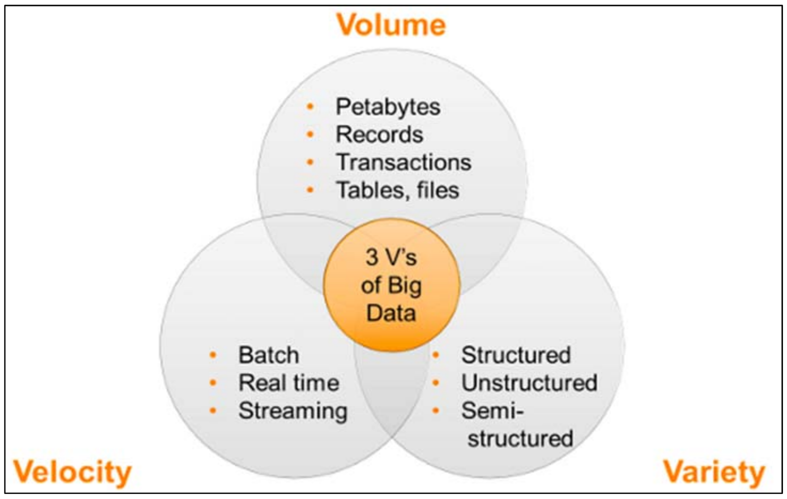


Figure 2.1. Three V’s of big data. Source: VITRIA. The Operational Intelligence Company, 2014. http://blog.vitria.com, accessed April 2014.

2.1.1 Volume

Volume is the most challenging aspect of Big Data since it imposes a need for scalable storage and a distributed approach to querying. Big enterprises already have a large amount of data accumulated and archived over the years. It could be in the form of system logs, record keeping...etc. The amount of this data easily gets to the point where conventional database management systems may not be able to handle it. Data warehouse based solutions may not necessarily have the ability to process and analyze this data due to lack of parallel processing architecture.

A lot can be derived from text data, locations or log files. For example, email communications patterns, consumer preferences and trends in transaction-based data, security investigations. Spatial and temporal (time-stamped) data absorb storage space quickly. Big Data technologies offer a solution to create value from this massive and previously unused/ difficult to process data.

2.1.2 Velocity

Data is flowing into organizations at a large speed. Web and mobile technologies have enabled generating a data flow back to the providers. Online shopping has revolutionized consumer and provider interactions. Online retailers can now keep log of and have access to customers every interaction and can maintain the history and want to quickly utilize this information in recommending products and put the organization on a leading edge. Online marketing organizations are deriving lot of advantage with the ability to gain insights instantaneously. With the invention of the smartphone era there is even further location based data generated and its becoming important to be able to take advantage of this huge amount of data.

2.1.3 Variety

All this data generated with social and digital media is rarely structured data. Unstructured text documents, video, audio data, images, financial transactions, interactions on social websites are examples of unstructured data. Conventional databases support 'large objects' (LOB's), but have their limitations if not distributed. This data is hard to fit in conventional neat relational database management structures and is not very integration friendly data and needs a lot of massaging before applications can manage it. And this leads to loss of information. If the data is lost then it’s a loss that cannot be recovered. Big Data on the other hand tends to keep all the data since most of this is write once and read many times type of data. Big Data believes that there could be insights hidden in every bit of data.

2.2 Need of Big Data Analytics

With the above-mentioned attributes of big data, data is massive, comes at a speed and highly unstructured that it doesn't fit conventional relational database structures. With so much insight hidden in this data, an alternative way to process this enormous data is necessary. Big corporations could be well resourced to handle this task but the amount of data being generated every day easily outgrows this capacity. Cheaper hardware, cloud computing and open source technologies have enabled processing big data at a much cheaper cost.

Lot of data means lot of hidden insights. The ability to quickly analyze big data means the possibility to learn about customers, market trends, marketing and advertising drives, equipment monitoring and performance analysis and much more. And this is an important reason that many big enterprises are in a need of robust big data analytics tools and technologies.

Big data tools mainly make use of in-memory data query principle. Queries are performed where the data is stored, unlike conventional business intelligence (BI) software that runs queries against data stored on server hard drive. In-memory data analytics has significantly improved data query performance. Big data analytics not just helps enterprises make better decisions and gain an edge into real-time processing, it has also inspired businesses to derive new metrics and gain new sources of revenue out of insights gained.

Note that temporal data naturally leads to Big Data, as does spatial data. Early attempts to deal with large warehouses, including non-scalar data, used so called ORDBMS [5], i.e. object relations databases. Big Data outperforms ORDBMS in various ways, including the need for more complicated backups, recovery and faster search algorithms, beyond RDBMS indexes.

Benefits of using Big Data Technologies may come at a downside of a loss of privacy of the data. In terms of privacy, some companies sell customer data to other companies, and this can be a problem.

CHAPTER 3

HADOOP ARCHITECTURE

It's hard to omit Hadoop while talking about big data. Hadoop is the open source software platform managed by the Apache Software Foundation. It's the most widely recognized platform to efficiently and cost-effectively store and manage enormous amount of data.

3.1 INTRODUCTION TO HADOOP

Formal definition of Hadoop by Apache: "The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly available service on top of a cluster of computers, each of which may be prone to failures" [6].

Hadoop was initially inspired by papers published by Google, outlining its approach to handle an avalanche of data, and has since become the standard for storing, processing and analyzing hundreds of terabytes, and even petabytes of data. Hadoop framework development was started by Doug Cutting and the framework got its name from his son's elephant toy [7].

Hadoop has drawn the inspiration from Google's File System (GFS). Hadoop was spun from Nutch in 2006 to become a sub-project of Lucene and was renamed to Hadoop. Yahoo has been a key contributor to Hadoop evolution. By 2008 yahoo web search engine index was being generated by a 10,000 core Hadoop cluster.

Hadoop is an open source framework by Apache, and has invented a new way of storing and processing data. Hadoop does not rely on expensive, high efficiency hardware. Instead it leverages on benefits from distributed parallel processing of huge amounts of data across commodity, low-cost servers. This infrastructure stores as well as processes the data, and can easily scale to changing needs. Hadoop is supposed to have limitless scale up ability and theoretically no data is too big to handle with distributed architecture [8].

Hadoop is designed to run on commodity hardware and can scale up or down without system interruption. It consists of three main functions: storage, processing and resource management. It is presently used by big corporations like Yahoo, eBay, LinkedIn and Facebook.

Conventional data storage and analytics systems were not built keeping in mind the needs of big data. And hence no longer easily and cost-effectively support today's large data sets.

3.2 Hadoop Attributes

- Fault tolerant - Fault tolerance is the ability of the system to stay functional without interruption and without losing data even if any of the system components fail [9]. One of the main goals of Hadoop is to be fault tolerant. Since hadoop cluster can use thousands of nodes running on commodity hardware, it becomes highly susceptible to failures. Hadoop achieves fault tolerance by data redundancy/ replication. And also provides ability to monitor running tasks and auto restart the task if it fails.
- Built in redundancy - Hadoop essentially duplicates data in blocks across data nodes. And for every block there is assured to be a back-up block of same data existing somewhere across the data nodes. Master node keeps track of these node and data mapping. And in case of any of the node fails, the other node where back-up data block resides, takes over making the infrastructure failsafe. A conventional RDBMS has the same concerns and uses terms like: persistence, backup and recovery. These concerns scale upwards with Big Data.
- Automatic scale up/ down - Hadoop heavily relies on distributed file system and hence it comes with a capability of easily adding or deleting the number of nodes needed in the cluster.
- Move computation to data - Any computational queries are performed where the data resides. This avoid overhead required to bring the data to the computational environment. Queries are computed parallelly and locally and combined to complete the result set.

3.3 Hadoop Components

Let's take a look at two most important components that are the foundation to Hadoop framework.

3.3.1 Hadoop Distributed File System - HDFS

HDFS is a distributed file system designed to run on commodity hardware. HDFS has a master/slave architecture. See Figure 3.1. It's a write-once and read multiple times approach.

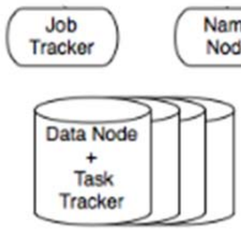


Figure 3.1. Hadoop cluster simplified visualization.

An HDFS cluster consists of a single NameNode (latest version 2.3.0 has redundant NameNode to avoid single point of failure), a master server machine that manages the file system and regulates access to the filesystem by the clients. There are multiple data nodes per cluster. As shown in Figure 3.2 [10], data is split into blocks and stored on these data nodes. NameNode maintains the map of data distribution. Data Nodes are responsible for data read and write operations during execution of data analysis. Hadoop also follows concept of Rack Awareness. What this means is a Hadoop Administrator user can define which data chunks to save on which racks. This is to prevent loss of all the data if an entire rack fails and also for better network performance by avoiding having to move big chunks of bulky data across the racks. This can be achieved by spreading replicated data blocks on the machines on different racks.

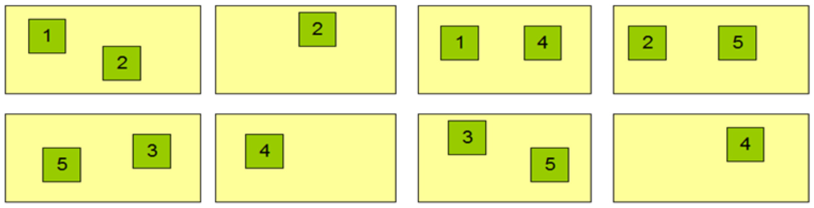


Figure 3.2. Hadoop data replication on data nodes. Source: Apache Hadoop. MapReduce Tutorial, 2013. https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html, accessed April 2014.

Refer Figure 3.3 [10], the NameNode and DataNode are commodity servers, typically Linux machines. Hadoop runs different software on these machines to make it a

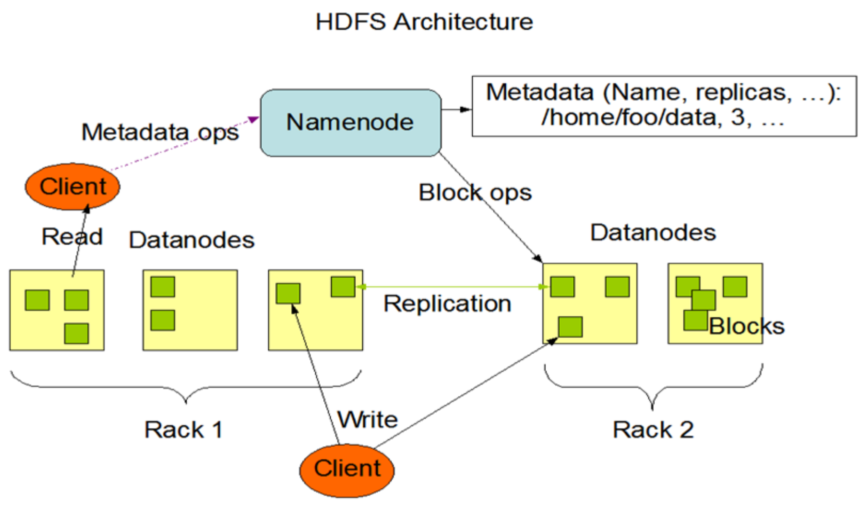


Figure 3.3. Hadoop detailed architecture. Source: Apache Hadoop. MapReduce Tutorial, 2013. https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html, accessed April 2014.

NameNode or a DataNode. HDFS is built using the Java language. Any machine that Java can be run on can be converted to act as the NameNode or the DataNode. A typical cluster has a dedicated machine that runs only the NameNode software. Each of the other machines in the cluster runs one instance of the DataNode software. The NameNode manages all HDFS metadata [11].

3.3.2 MapReduce

MapReduce is a software framework introduced by Google to perform parallel processing on large datasets...assuming that large dataset storage is distributed over a large number of machines. Each machine computes data stored locally, which in turn contributes to distributed and parallel processing. There are two parts to such a computation - Map and Reduce. Datanodes assigned to the Map phase, take raw input data and based on the type of computation required produce intermediate data that is stored locally. Reduce nodes take these intermediate outputs and combine them to derive final output which is then stored in HDFS.

Hadoop tries to collocate data and the computation. Namenode with its knowledge of how the data is distributed, tries to assign the task to the node in which the data locally resides. Programmers can write custom map and reduce functions and MapReduce function automatically takes care of distributing and parallelizing the tasks across an array of commodity machines in the cluster underneath. It as well manages inter-machine communication leaving programmers to focus on actual map-reduce functions Hadoop uses this fault tolerant, reliable, distributed, parallel computing framework to analyse large datasets distributed over HDFS.

Both Map and Reduce functions operate on data conceptualized as key - value pairs.

- Map Phase - In the Map phase, each mapper reads raw input, record by record, and converts it into Key/Value pair and feeds it to the map function. Depending upon how the user has defined the Map function, map function produces intermediate output in the form of new key/value pairs. A number of such mappers located on the cluster parallely process raw data to produce a set of intermediate key/value pairs which are locally stored on each mapper. Input data is split into M map tasks.
- map (k1, v1) -> k2, v2 (shuffle and sort - gathers all pairs with same key)
- Reduce Phase - merges all intermediate values associated with intermediate keys.
- reduce (k2, list(v2)) -> v3 (merge - combines together values for same keys - in case of queries used for thesis - reduce will sum the values)
- MapReduce illustration with word count example [10]
    - Here we try to derive word frequency with MapReduce program
    - Assume two file inputs
        - file 1: “apple banana guava watermelon mango apple”
        - file 2: “mango kiwi guava cantaloupe mango”
    - We will illustrate the following operations using the MapReduce algorithm
        - Map
        - Combine
        - Reduce
    - With a two-node cluster we would have two task nodes and that means two mappers available to distribute the first mapping task.
    - Map Phase I - split
        - mapper 1 takes file 1 as input
        - mapper 1 would produce following output in <key, value> format
          <apple, 1>
          <banana, 1>
          <guava, 1>
          <watermelon, 1>
          <mango, 1>
          <apple, 1>
        - mapper 2 takes file 2 as input
        - mapper 2 would produce following output

    <mango, 1>
    <kiwi, 1>
    <guava, 1>
    <cantaloupe, 1>
    <mango, 1>

- Map Phase II - combine
  - mapper 1 output
    <apple, 2>
    <banana, 1>
    <guava, 1>
    <watermelon, 1>
    <mango, 1>
  - mapper 2 output
    <mango, 2>
    <kiwi, 1>
    <guava, 1>
    <cantaloupe, 1>
- Reduce phase
  - Reducer will produce following final result
    <apple, 2>
    <banana, 1>
    <guava, 2>
    <watermelon, 1>
    <mango, 3>
    <kiwi, 1>
    <cantaloupe, 1>
- main method for MapReduce Java program
  - This is in line with Hadoop version 1.2.1

  -
    ```
    public static void main(String[] args) throws Exception
    {
          JobConf conf = new JobConf(WordCount.class); //
    Create a new job with the given configuration
          conf.setJobName("wordcount");
          conf.setOutputKeyClass(Text.class); //Set the key
    class for the job output data.
          conf.setOutputValueClass(IntWritable.class);     //
    Set the value class for job outputs.
          conf.setMapperClass(Map.class);
          conf.setCombinerClass(Reduce.class);
          conf.setReducerClass(Reduce.class);
          conf.setInputFormat(TextInputFormat.class);
          conf.setOutputFormat(TextOutputFormat.class);
          FileInputFormat.setInputPaths(conf,            new
    ```

```
Path(args[0]));
      FileOutputFormat.setOutputPath(conf,           new
Path(args[1]));
      JobClient.runJob(conf);
}
```

- Map function

```
public static class Map extends MapReduceBase implements
Mapper<LongWritable, Text, Text, IntWritable> {
    private   final   static   IntWritable   one   =   new
IntWritable(1);
    private Text word = new Text();

    public   void   map(LongWritable   key,   Text   value,
OutputCollector<Text,   IntWritable>   output,   Reporter
reporter) throws IOException {
       String line = value.toString();
       StringTokenizer       tokenizer       =       new
StringTokenizer(line);
       while (tokenizer.hasMoreTokens()) {
             word.set(tokenizer.nextToken());
             output.collect(word, one);
       }
    }
}
```

- Reduce function

```
      public   static   class   Reduce   extends   MapReduceBase
implements Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values,
OutputCollector<Text, IntWritable> output, Reporter reporter)
throws IOException {
       int sum = 0;
       while (values.hasNext()) {
             sum += values.next().get();
       }
       output.collect(key, new IntWritable(sum));
    }
}
```

CHAPTER 4

BIG DATA ALLIED TECHNOLOGIES

The technologies that aid the entire process of cost-effectively storing and processing data, and utilizing internet technologies in a distributed way have arisen in the past few years. NoSQL and Cloud Computing are the prominent ones that enhance the potential offered by Big Data Technologies.

4.1 NoSQL Database

In past years databases always meant relational database and anything beyond that was only flatfiles. When relational databases were invented and put to use, data that these systems handled was fairly simple and straightforward, and it was easy to store objects as sets of relationships. This also helped querying the data in a very structured and defined way.

But the means and ways the data gets generated these days as well as the structure of data has drastically changed with the invention and growth of mobile and web technologies.

The need to store unstructured data, such as social media posts and multimedia, has grown rapidly. SQL databases are extremely efficient at storing structured information, and workarounds or compromises are necessary for storing and querying unstructured data.

Also, relational databases are less adaptive to changes in the data structures that may happen over the years of running the application. Agile development methods these days lead to constantly changing requirements and database schema needs to change rapidly as demands change. SQL databases [12] need to have schemas defined in advance so as to store the data without any failure. If the data is constantly changing this will lead to ALTER TABLE situation every now and then. And this is where the need for NoSQL databases, that can handle unstructured and unpredictable data, has arisen.

NoSQL database [13] is not a technology but a new way of looking at database design. Relational databases may not be the best solution for every situation in this era of unstructured data needs. Highly evolved and interactive web and smartphone applications have dramatically changed the database management needs. Agile databases are needed to process unstructured data and may be in real-time for some applications. Conventional relational databases expect data to be predictable and structured. Scaling up databases to accommodate growing data means more database servers in case of relational databases. NoSQL databases is not a replacement to SQL databases though.

A NoSQL database has the ability to distribute itself over cheaper, commodity hardware knows as clusters and can be cloud computed. NoSQL databases are looked upon as high performance, high availability and easily scalable databases. NoSQL databases do not have to adhere to table format like relational databases and may not use SQL as a language to manipulate the data. There are not schemas or joins. NoSQL databases are said to be able to handle complex, nested, hierarchical data structures.

Well known NoSQL databases either use key-value stores, column family stores, document store or graph databases. NoSQL providers either employ RESTful services to query NoSQL databases or provide querying APIs. Existing well known NoSQL databases include the following:

Document databases

- MongoDB
- CouchDB

Graph databases

- Neo4J

Key-value stores

- Redis
- MemcacheDB

Column databases

- HBase
- Cassandra

Challenges with NoSQL databases are:

- No ACID transaction support
  - Atomicity - a transaction fully completes or does not complete at all.
  - Consistency - the database will be in a consistent state before and after a transaction
  - Isolation - transactions may not interfere with each other
  - Durability - a transaction is always permanent

- Use of low level query language
- No standardized interfaces
- Enterprises already have paid the cost to build huge SQL systems

4.2 CLOUD COMPUTING

Cloud computing is one of the most widely talked about term in the IT industry these days. Technological advances have enabled devising a solution quickly and at low cost. With cloud computing comes the ability to share computing and storage resources without having to develop infrastructure from scratch every time there is a need. The speed of delivering solution has gained significant importance in the competing technological world. And enterprises feel the pressure to quickly adapt to technologies. Cloud computing comes to an aid making infrastructures easily scalable and elastic.

In simple words, cloud computing is nothing but making services or software or data available over the internet where services or software or data reside on remote machines. A client side interface is made available to access these cloud services. Cloud computing is an invention that has occurred from the need that IT organizations want to keep away from scaling infrastructure, investing money in hardware and training resources. Cloud computing usually has a front end and back end side. Front end is comprised of the client side interface and software required to access the cloud system. Whereas back end is the array of machines running different softwares depending upon the services provided by the cloud.

When the time and pressure required in creating and maintaining IT infrastructures is taken off, organizations were able to focus on the core business. Also, it keeps organizations from putting valuable resources in reinventing the infrastructure every time there is a change and the organization needs to adapt to it. Infrastructure includes application servers, database servers, middleware needed to communicate between different entities.

Cloud computing has allowed organizations to outsource overhead business applications like HR, payroll to third parties delivered via the cloud. See Figure 4.1 [14]. Cloud computing is also meant to be flexible, elastic, reliable and secured. These are features that have led to heavy adoption of cloud computing by enterprises these days.

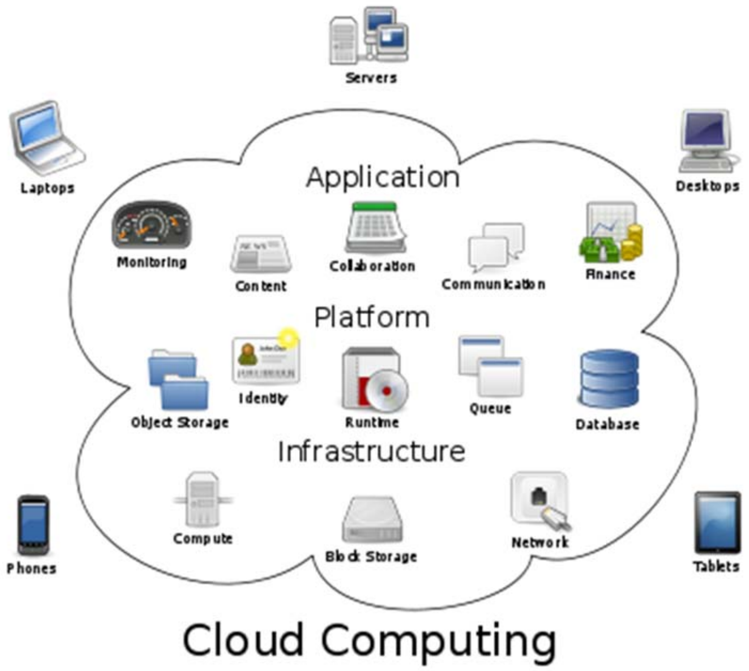


Figure 4.1. Cloud computing logical diagram. Source: S. Johnston. Seminar on Collaboration as a Service – Cloud Computing, 2012. http://www.psirc.sg/events/seminar-on-collaboration-as-a-service-cloud-computing, accessed April 2014.

Reasons to consider cloud computing

- high availability
- clients are able to access system and data from anywhere
- Data is not confined to a single machine
- Reduces the need of expensive hardware on client side, and frequent hardware updates
- Keeps organizations from worrying about space and monitoring needs for data storage devices and servers
- Optimized use of servers - it can happen that server's capacity is not fully utilized by single client or single application. The server can be configured to act as multiple servers serving multiple requests.

Challenges with cloud computing

- Security and privacy - sensitive data is to be stored on the machine outside of company's network
- Maintaining transparency in service delivery and billing

- Lack of standards - there is a need of technical, management and regulatory standards to provide definitions and guidelines to cloud environment.
- Responsiveness of the service
- Integration - integrating existing in-house applications to cloud services

CHAPTER 5

AMAZON WEB SERVICES

Amazon Web Services (abbreviated AWS) is a collection of remote computing services (also called web services) that is collectively known as a cloud-computing platform. Amazon provides these services over internet [15].

5.1 WHAT IS AMAZON WEB SERVICES?

The first AWS was publicly made available in 2006 to provide online services. AWS is geographically distributed into different regions. See Figure 5.1. These regions have central hubs in the Eastern USA, Western USA (two locations), Brazil, Ireland, Singapore, Japan, and Australia. Each region comprises multiple smaller geographic areas called availability zones. Geographically distributing AWS is meant to avoid outages, provide back up and redundancy of the services and make them highly available.

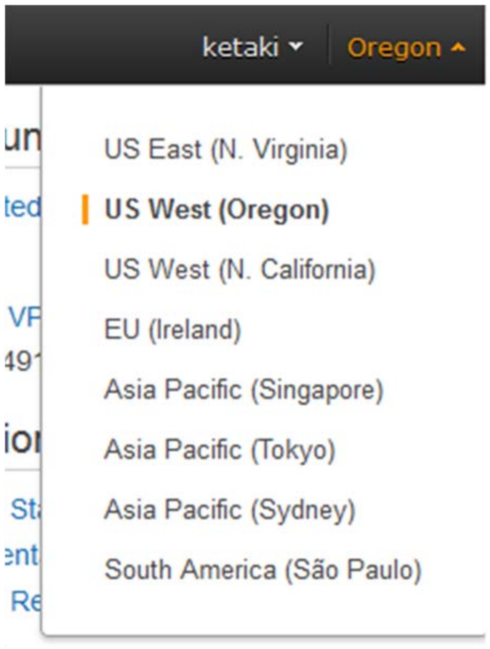


Figure 5.1. Screenshot of AWS regions.

Amazon Web Services (AWS) provides computing resources and services (see Figure 5.2) that are very easy to access, within minutes, and meant to be cost effective at pay-as-you-go pricing. For example, for this thesis purpose, Linux machine nodes were used with a rate of $0.06 per hour and amazon has charged fees based on hours of utilization. The only cost involved is the cost when the nodes are run and are in use, without any up-front purchase costs or ongoing maintenance costs. AWS also provide easy scalability which is hard to achieve with physical in house servers. It is easy to scale up the number of nodes while experimenting on AWS.

Benefits of using AWS that Amazon claims

- Low cost
- No initial purchase cost
- Flexibility
- Scalability
- High availability
- Security

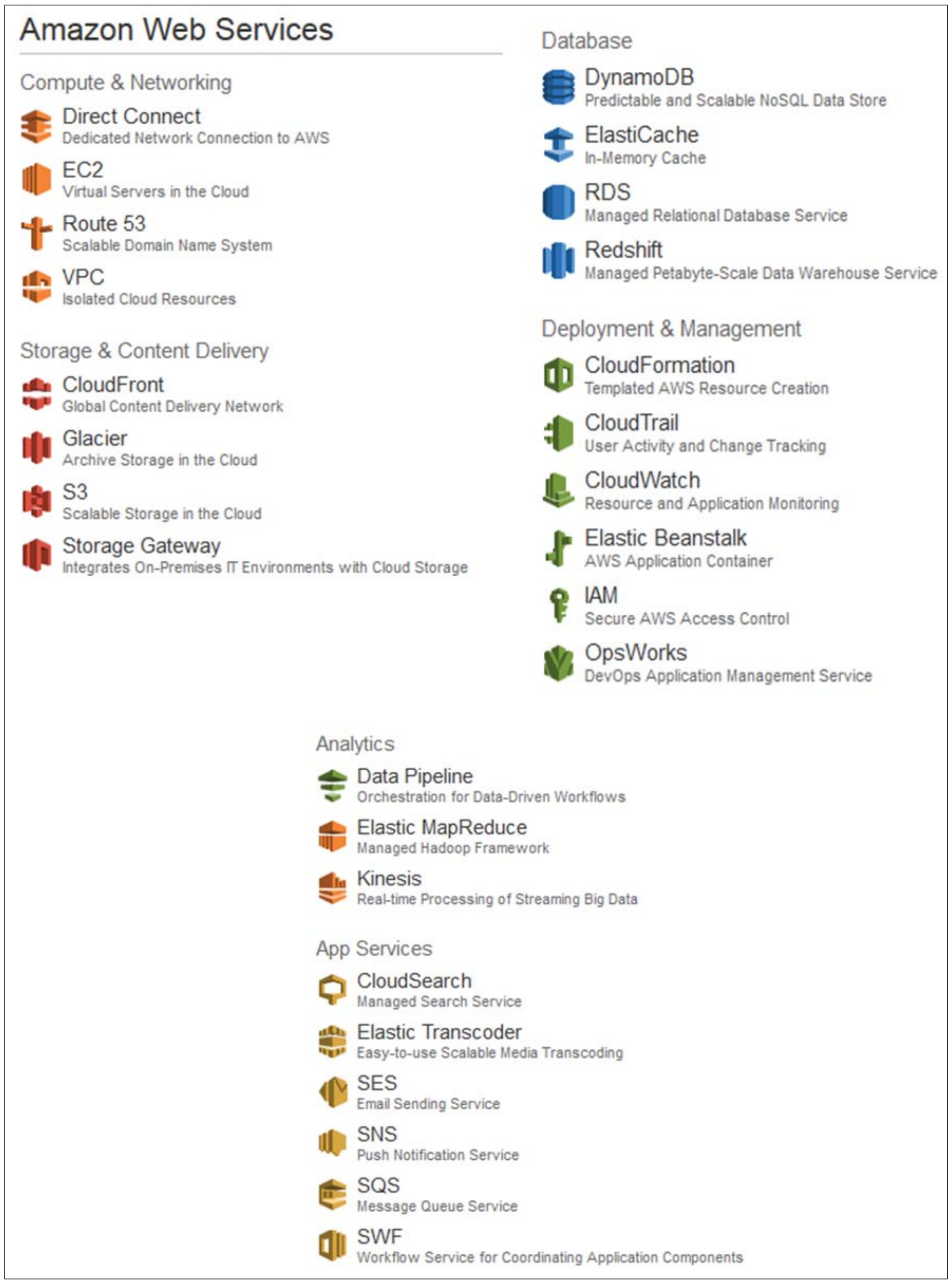
5.2 Elastic Map Reduce – EMR

Amazon Elastic MapReduce (Amazon EMR) is a web service that makes it easy to quickly and cost-effectively process vast amounts of data. Amazon EMR uses Hadoop to distribute the data and processing across a resizable cluster of Amazon EC2 instances.

It is easy launch an Amazon EMR cluster in minutes. Amazon EMR cluster comes with nodes preconfigured with hadoop APIs, hive and cluster already tuned to work with other amazon web services. This makes it easy to directly get hands on to data analysis without having to worry about other setup requirements.

Amazon EMR gives the ability to set up the cluster with tens, hundreds, thousands or few number of nodes. See Figure 5.3 [16]. At the same time its easy to scale up by quickly resizing the cluster to suit the needs. AWS also monitors the cluster to keep track of failed tasks and automatically replace poorly performing instances.

Amazon EMR also configures firewall settings to secure the cluster. There is complete control of the cluster given to clients with root access to every instance. Its easy to install additional software and customize every cluster. Amazon EMR provides different Hadoop distributions and applications to choose from.



Amazon Web Services

Compute & Networking

Direct Connect
Dedicated Network Connection to AWS

EC2
Virtual Servers in the Cloud

Route 53
Scalable Domain Name System

VPC
Isolated Cloud Resources

Storage & Content Delivery

CloudFront
Global Content Delivery Network

Glacier
Archive Storage in the Cloud

S3
Scalable Storage in the Cloud

Storage Gateway
Integrates On-Premises IT Environments with Cloud Storage

Database

DynamoDB
Predictable and Scalable NoSQL Data Store

ElastiCache
In-Memory Cache

RDS
Managed Relational Database Service

Redshift
Managed Petabyte-Scale Data Warehouse Service

Deployment & Management

CloudFormation
Templated AWS Resource Creation

CloudTrail
User Activity and Change Tracking

CloudWatch
Resource and Application Monitoring

Elastic Beanstalk
AWS Application Container

IAM
Secure AWS Access Control

OpsWorks
DevOps Application Management Service

Analytics

Data Pipeline
Orchestration for Data-Driven Workflows

Elastic MapReduce
Managed Hadoop Framework

Kinesis
Real-time Processing of Streaming Big Data

App Services

CloudSearch
Managed Search Service

Elastic Transcoder
Easy-to-use Scalable Media Transcoding

SES
Email Sending Service

SNS
Push Notification Service

SQS
Message Queue Service

SWF
Workflow Service for Coordinating Application Components

Figure 5.2. Screenshot of Amazon web services.

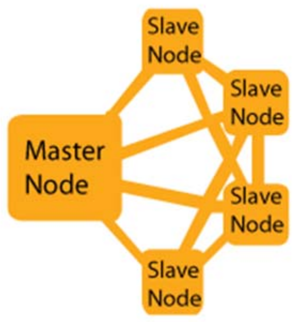


Figure 5.3. EMR simple representation. Source: Amazon Elastic MapReduce. What is Amazon EMR?, 2009. http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-what-is-emr.html accessed April 2014.

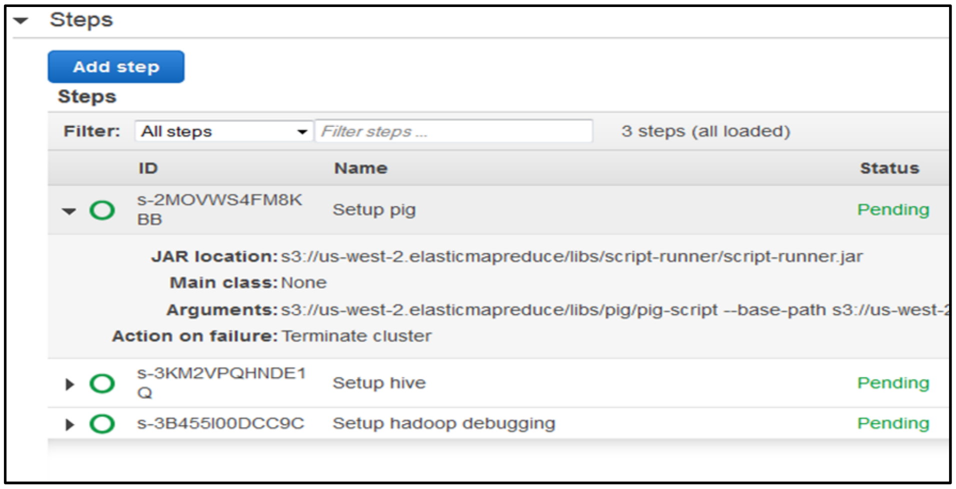
EMR Terminology [16]

- Cluster
- Nodes
- Master node - Master node manages the cluster. It keeps a track of the distribution of the MapReduce executable. Also keeps a map of distribution of the raw data. It also tracks the status of each task performed, and monitors the status of the instances. There is only one master node in a cluster.
- Core nodes - these are the slave nodes that hold distributed data in HDFS.
- Task nodes - These are additional slave nodes that can be provisioned on AWS to provide more capability to distribute the jobs
- Steps - It's a unit of work. Maximum number of steps AWS can process is limited to 256. These include any provisioning, set and debugging steps needed by Amazon EMR. AWS provide user interface to monitor the status of the steps. See Figure 5.4.

5.3 Elastic Compute Cloud - EC2

It's a virtual computing environment hosting variety of operating systems. It comes with a client-side interface to choose and launch instances from. Instances can then be customized with different softwares/ applications and comes with access management interface.

As seen from Figure 5.5 [17], EC2 comes with pre-configured machine template called Amazon Machine Image (AMI). To use, the user would select an AMI or create his own



Steps

Add step

Steps

Filter: All steps | Filter steps ... | 3 steps (all loaded)

	ID	Name	Status
▾ ○	s-2MOVWS4FM8KBB	Setup pig	Pending
▸ ○	s-3KM2VPQHNDE1Q	Setup hive	Pending
▸ ○	s-3B455I00DCC9C	Setup hadoop debugging	Pending

JAR location: s3://us-west-2.elasticmapreduce/libs/script-runner/script-runner.jar
Main class: None
Arguments: s3://us-west-2.elasticmapreduce/libs/pig/pig-script --base-path s3://us-west-2
Action on failure: Terminate cluster

Figure 5.4. Screenshot of steps monitoring user interface on AWS hadoop cluster.

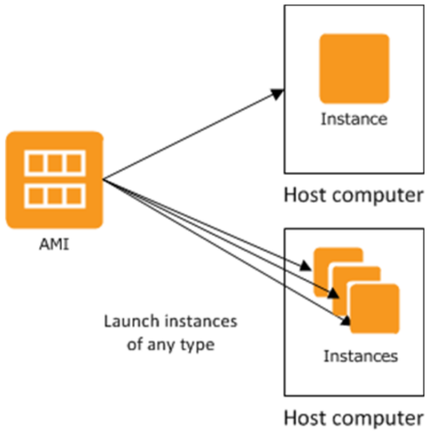


Figure 5.5. EC2 instances and AMI. Source: Amazon Elastic Compute Cloud. Instances and AMIs, 2014. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instances-and-amis.html, accessed April 2014.

custom AMI, configure security and network access permissions, then choose the type and number of instances to launch from AMI.

Features of EC2 [17]

- Provides pre-configured templates call AMI
- Any number of instances can be launched from AMI
- Various CPU, memory, storage, networking capacity choices available for instances
- Secured login using public-private key pair
- Persistent and temporary storage volumes available on EC2
- Elastic IP addresses for dynamic cloud computing

5.4 SIMPLE STORAGE SERVICE - S3

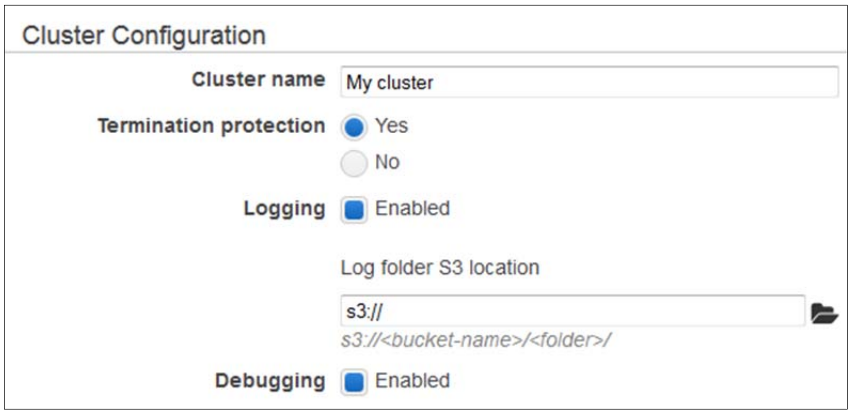
Amazon provides storage service to store data in the cloud. Data can then be downloaded locally or used with other AWS services in the cloud.

Amazon stores data in S3 buckets. Buckets resemble to domains. Amazon S3 buckets and subdomains of s3.amazonaws.com are directly mapped. REST API can access the objects saved in S3 bucket with root address as http://mybucket.s3.amazonaws.com

5.5 HOW TO SETUP HADOOP ON AWS

As part of this thesis, the author set up a Hadoop cluster on AWS, and the following steps were confirmed as correct.

1. Register with aws http://aws.amazon.com/
2. Create bucket on AWS S3 to store logs/ data
3. Create access key - Creating cluster on EMR would launch the nodes on EC2 and keys are needed to be able to access the root of these instances.
    - Go to Amazon Web Console (https://console.aws.amazon.com/ec2/v2/home?region=us-west-2#KeyPairs:) to generate AWS key. This key can be downloaded in .pem format.
    - We need Puttygen client to split .pem into public and private keys. Launch PUTTYGEN client and import the .pem key pair we created on EC2. Navigate to Conversions and "Import Key".
    - Save the private key by clicking on "Save Private Key" and click "Yes". Passphrase can be left empty.
    - Locally save .ppk file.
4. Create cluster – See Figure 5.6


Cluster Configuration

Cluster name: My cluster

Termination protection: [✓ Yes] No

Logging: [✓ Enabled]

Log folder S3 location: s3://

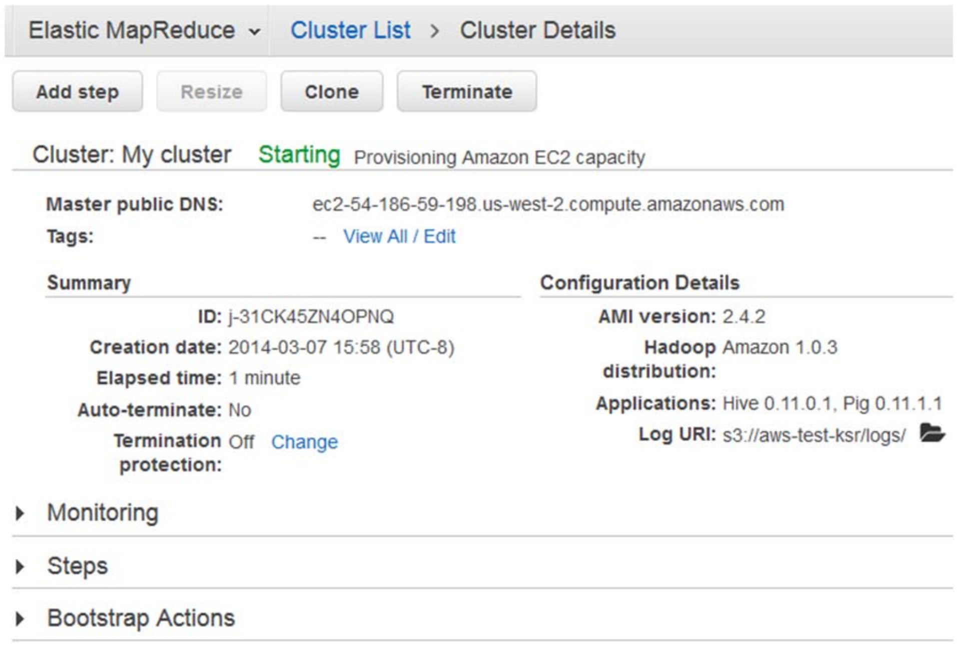
s3://<bucket-name>/<folder>/

Debugging: [✓ Enabled]

Figure 5.6. Screenshot of cluster configuration.

- Go to AWS management console and navigate to AWS EMR
- Click create cluster
- Enter cluster name
- Enter log folder s3 location which is s3://aws-test-ksr/logs for this thesis.
- For this thesis, Hadoop version is left as default which is the latest version.
- Under hardware configuration, for example, set the number of master node to one, number of core nodes to two, number of task nodes to zero.
- Under Security and Access, set the key to the previously generated AWS key pair.
- Click ‘Create Cluster’.
- This will launch one master and two slave instances on EC2. And AWS will start provisioning the nodes.
- All the nodes in the cluster would be assigned public DNS name as part of provisioning. See Figure 5.7.

5. Access root of Master node - For the thesis purpose, data is transferred to the local file system of Master node and then copied to the hive data table.
   - Install WinScp if using local windows OS.
   - Note the public DNS name of the master node. e.g. “ec2-54-186-94-179.us-west-2.compute.amazonaws.com”. This is the hostname while connecting using WinScp. See Figure 5.8
   - Username is ‘hadoop’.
   - Click on Advanced button. Under SSH-> Authentication, browse to the private key we saved previously.
   - Click Ok and Login.



Cluster: My cluster Starting Provisioning Amazon EC2 capacity

Master public DNS: ec2-54-186-59-198.us-west-2.compute.amazonaws.com
Tags: -- View All / Edit

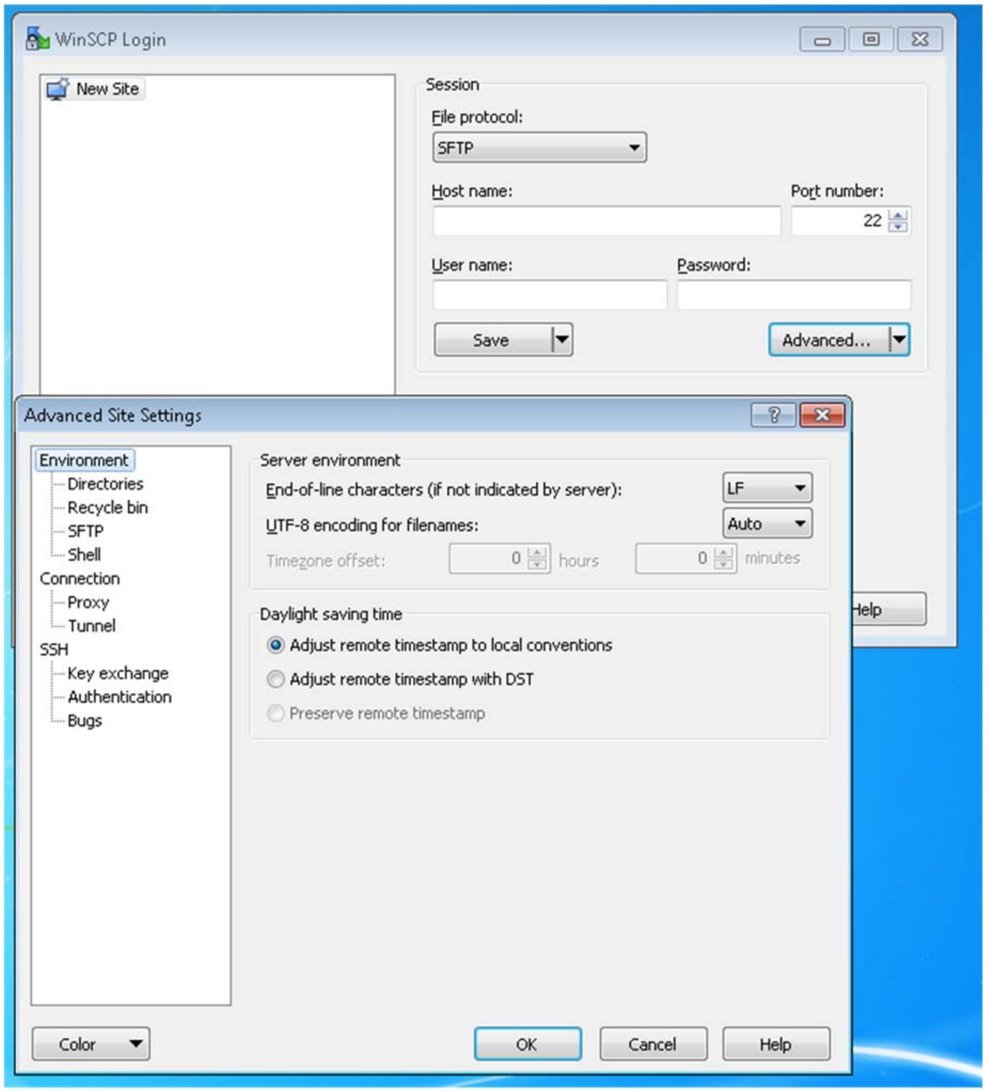
Summary
ID: j-31CK45ZN4OPNQ
Creation date: 2014-03-07 15:58 (UTC-8)
Elapsed time: 1 minute
Auto-terminate: No
Termination protection: Off Change

Configuration Details
AMI version: 2.4.2
Hadoop distribution: Amazon 1.0.3
Applications: Hive 0.11.0.1, Pig 0.11.1.1
Log URI: s3://aws-test-ksr/logs/

Figure 5.7. Screenshot of cluster details in provisioning phase.

- With correct private key, hostname and username, we should be able to access file system of the master node and transfer data to it.

6. SSH to Master Node
    - We need to access master node terminal to be able to run Hadoop/ Hive commands
    - Launch Putty.
    - Enter host name. This will be hadoop@masternodeDNS.
    - Under category-> Connection->SSH->Auth, browse to previously saved private key.
    - Click Open to connect.
7. With these steps we are setup with Hadoop cluster on AWS and now able to access local file system of the master node and able to run commands on master node terminal.



File protocol: SFTP
Host name:
Port number: 22
User name:
Password:

Server environment
End-of-line characters (if not indicated by server): LF
UTF-8 encoding for filenames: Auto
Timezone offset: 0 hours 0 minutes

Daylight saving time
[✓ Adjust remote timestamp to local conventions]
Adjust remote timestamp with DST
Preserve remote timestamp

Figure 5.8. Screenshot of WinSCP configuration.

CHAPTER 6

PERFORMANCE ANALYSIS

In this chapter we focus on two key performance evaluators, data load time and query execution time with RDBMS and with Hadoop to compare traditional RDBMS and Hadoop's distributed parallel processing architecture. ETL times in Hadoop are usually influenced by input data size, block size (hadoop distributes data to datanodes in blocks), cluster size. In the experimentation here, performance is evaluated based on input data size (increasing in gigabytes) and compared against PostgreSQL [18] as RDBMS. ETL times with increasing number of nodes is evaluated as well.

6.1 TEST SETUP

It's important to know the hardware details of the two systems that the performance analysis is conducted on. Let's take a look at the details of the RDBMS and Hadoop linux machines used for the experiments here.

6.1.1 Hadoop Setup

Hadoop cluster is set up on Amazon Web Services using Elastic Map Reduce, Elastic Compute Cloud and Simple Storage Services. Table 6.1 shows hardware details of Master and Slave nodes.

6.1.2 RDBMS Setup

RDBMS used is PostgreSQL. Table 6.2 shows hardware details of PostgreSQL server used. Figure 6.1 shows data schema used for the sample dataset.

6.1.3 Sample Dataset

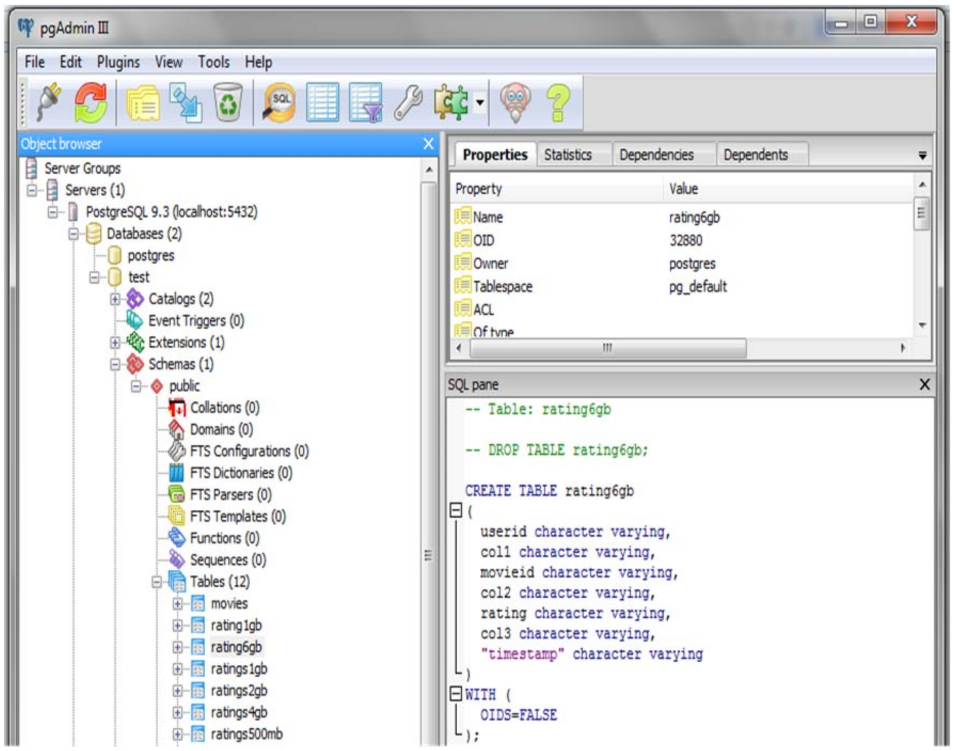
- GroupLens Research at University of Minnesota has collected and made available rating datasets from MovieLens web site. [19]
- Database used for this thesis is the MovieLens 10M database and mostly focusing on ratings table [19].
- Dataset may be used for any research purposes with conditions [20].

Table 6.1. Hadoop Cluster Setup Details

CPU	Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz
Cores	2 to 10
Cache Size	20480 KB
OS	Linux
Linux Kernel	3.2.30-49.59.amzn1.x86_64
Hadoop version	1.0.3
Hive Version	0.11.0.1
File System Size	8GB

Table 6.2. PostgreSQL Setup Details

OS	Windows 7 64 bit
CPU	Intel Core i5
RAM	6GB
PostgreSQL version	1.18.1



Property	Value
Name	rating6gb
OID	32880
Owner	postgres
Tablespace	pg_default
ACL	
Of type	

```
-- Table: rating6gb

-- DROP TABLE rating6gb;

CREATE TABLE rating6gb
(
  userid character varying,
  col1 character varying,
  movieid character varying,
  col2 character varying,
  rating character varying,
  col3 character varying,
  "timestamp" character varying
)
WITH (
  OIDS=FALSE
);
```

Figure 6.1. PostgreSQL setup screenshot.

- Usage of this dataset for the purpose of this thesis has been communicated and approved from the owners of the data via email.
- Data Description:
- Data consists of 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users. Data format is CSV.
- Table 6.3 shows the data schema.

6.2 HIVE

Hive is open source data warehouse software provided by Apache [21] to be used to query and manage large distributed datasets in a SQL manner. Hive helps to conceptualize data in a structured way and provides SQL like query language known as HiveQL. Hive query language is as well capable of running MapReduce programs. HiveQL helps users keep away from writing low-level MapReduce programs. AWS supports running on Hive on EMR Hadoop cluster and comes preconfigured with Hive installation.

Hive attributes:

- A Hive table is analogous to a data table in relational databases.
- There are internal data tables and external data tables

Table 6.3. Ratings Data Schema - Sample Database Used for Performance Analysis

column name	**data type**	**note**
userid	string	
col1	string	placeholder column to hold empty field while hive parses raw input when copying to data table
movieid	string	
col2	string	placeholder column to hold empty field while hive parses raw input when copying to data table
rating	string	
col3	string	placeholder column to hold empty field while hive parses raw input when copying to data table
timestamp	string	

- HiveQL supports Data Definition (DDL) statements (CREATE, DROP) and Data Manipulation (DML) statements (INSERT, INSERT INTO) but UPDATE and DELETE queries are not supported since Hive is based on write-once, read multiple times batch processing concept
- It facilitates easy data extraction, transform and loading
- Helps to conceptualize data in a structured way
- Hive stores data in files on HDFS
- Hive queries have ability to run MapReduce without having to write custom MapReduce jobs to execute queries.
- Hive supports primitive data types like other relational databases, as well supports complex data structures like arrays, structs
- SerDe - It's a Serialize-deserialize interface to let Hive know how to process data record. Data is serialized so that hive can store it on HDFS supported format or other supported storage format. Hive deserializes data into Java object so that Hive can manipulate is while processing queries. Normally, deserialize is used while processing query like SELECT query and serializer is used while using query of type INSERT
- Partitions - Hive supports partitioning of data. This helps to efficiently run the queries when the input data records are heavily tied to a column/ attribute. Hive would save such data partitioned into different directories. Load data with partitions syntax LOAD DATA LOCAL INPATH './ratings.xls' INTO TABLE ratings PARTITION (userid='xxx')
- Buckets - Partitions can be further compartmentalized into buckets based on the key in the column. Buckets are saved as a file in the particular partition directory.

6.3 EXPERIMENTS

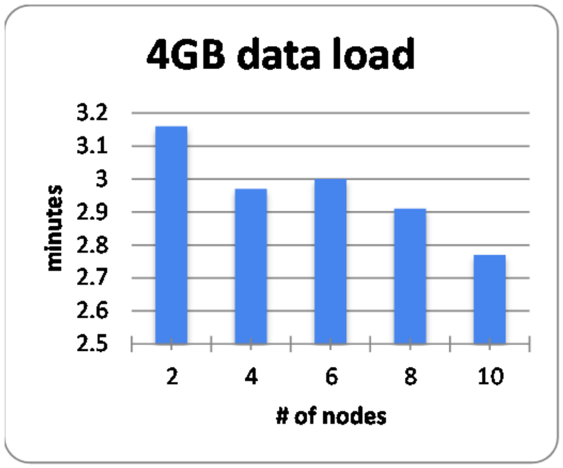
Experiment conducted are

1. Data load performance on hadoop cluster
2. Data query execution performance on Hadoop cluster
3. RDBMS vs Hadoop data load performance analysis
4. RDBMS vs Hadoop data query execution performance analysis

6.3.1 Hadoop Cluster – Data Load Performance

This experiment focuses on change in data load time in hive table with increasing number of nodes on Hadoop cluster. Experiment is done for two data sizes - 4 GB and 6 GB. See Figure 6.2.

1. WinScp to AWS hadoop master node to get access to the local file system
2. Copy csv data to the root (/home/hadoop)



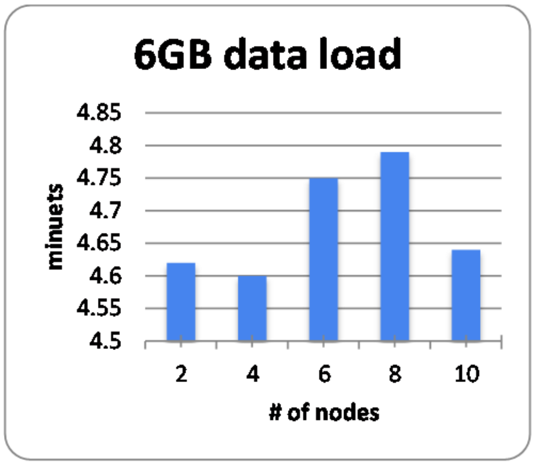


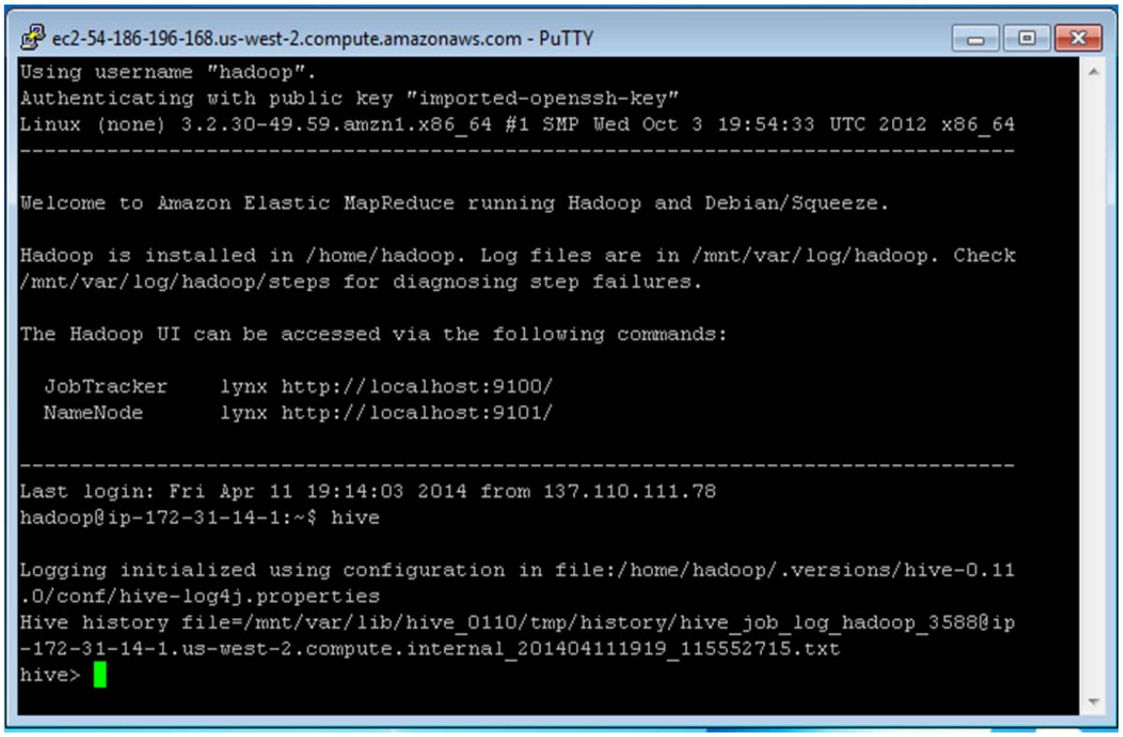
Figure 6.2. Bar graph visualization for experiment 6.3.1.

3. SSH to master node. This should launch hadoop command line.
4. To launch hive command line run command > hive on hadoop command line. See Figure 6.3.
5. Create table command > create table ratings (userid string, col1 string, movieid string, col2 string, rating string, col3 string, timestamp string) row format delimited fields terminated by ':' stored as textfile;
6. Load sample data into created table > load data local inpath './ratings4gb.csv' overwrite into table ratings; See Figure 6.4.
7. Note data load time for 4GB data size
8. Go to AWS management console
9. Select the created console and increase number of cores to 4, 6, 8 and then 10, each times re-creating hive table and re-loading it with 4GB data set.
10. There were five readings taken per data size and per node size to allow for any latencies. Average of 5 readings is considered while evaluating the performance. Refer Table 6.4.
11. Repeat above steps for 6GB data size

6.3.2 Hadoop Cluster – Data Query Execution

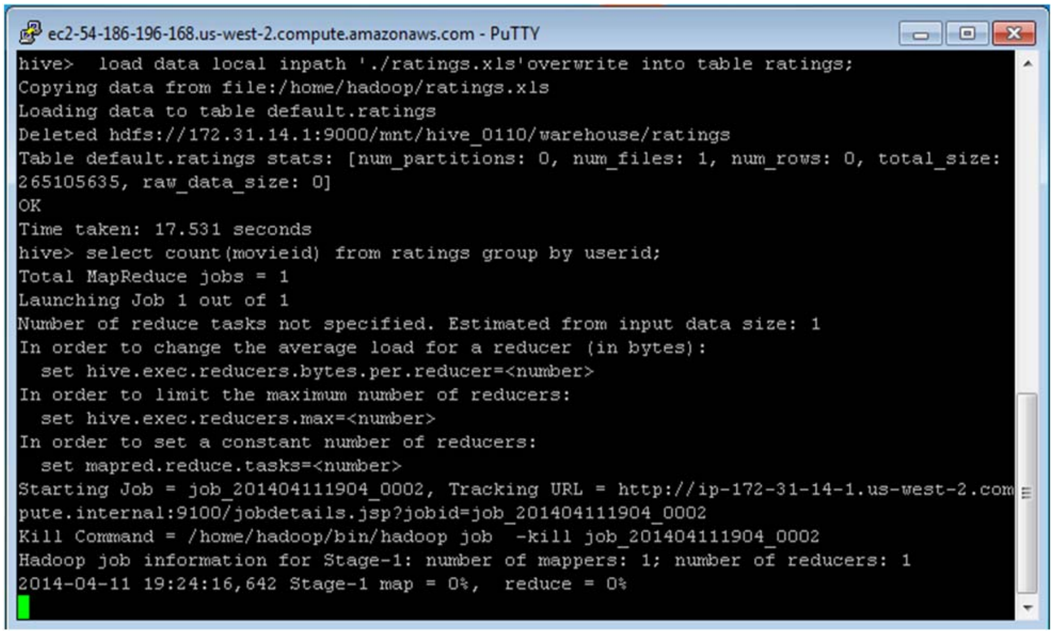
This experiment tries to analyze query execution times with increasing number of nodes. Experiment is done for two data sizes – 4GB and 6GB. See Table 6.5. Performance is evaluated for two queries. Same steps as described in 6.3.1 are carried out for this experiment.

- Part I: select count(movieid) from ratings group by userid (See Figure 6.5)
- Part II: select count(movieid) from ratings where ratings like '5'



```
Using username "hadoop".
Authenticating with public key "imported-openssh-key"
Linux (none) 3.2.30-49.59.amzn1.x86_64 #1 SMP Wed Oct 3 19:54:33 UTC 2012 x86_64
--------------------------------------------------------------------------------

Welcome to Amazon Elastic MapReduce running Hadoop and Debian/Squeeze.

Hadoop is installed in /home/hadoop. Log files are in /mnt/var/log/hadoop. Check
/mnt/var/log/hadoop/steps for diagnosing step failures.

The Hadoop UI can be accessed via the following commands:

  JobTracker    lynx http://localhost:9100/
  NameNode      lynx http://localhost:9101/

--------------------------------------------------------------------------------
Last login: Fri Apr 11 19:14:03 2014 from 137.110.111.78
hadoop@ip-172-31-14-1:~$ hive

Logging initialized using configuration in file:/home/hadoop/.versions/hive-0.11
.0/conf/hive-log4j.properties
Hive history file=/mnt/var/lib/hive_0110/tmp/history/hive_job_log_hadoop_3588@ip
-172-31-14-1.us-west-2.compute.internal_201404111919_115552715.txt
hive>
```

Figure 6.3. Screenshot - start hive console.



```
hive>  load data local inpath './ratings.xls'overwrite into table ratings;
Copying data from file:/home/hadoop/ratings.xls
Loading data to table default.ratings
Deleted hdfs://172.31.14.1:9000/mnt/hive_0110/warehouse/ratings
Table default.ratings stats: [num_partitions: 0, num_files: 1, num_rows: 0, total_size:
265105635, raw_data_size: 0]
OK
Time taken: 17.531 seconds
hive> select count(movieid) from ratings group by userid;
Total MapReduce jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201404111904_0002, Tracking URL = http://ip-172-31-14-1.us-west-2.com
pute.internal:9100/jobdetails.jsp?jobid=job_201404111904_0002
Kill Command = /home/hadoop/bin/hadoop job  -kill job_201404111904_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2014-04-11 19:24:16,642 Stage-1 map = 0%,  reduce = 0%
```

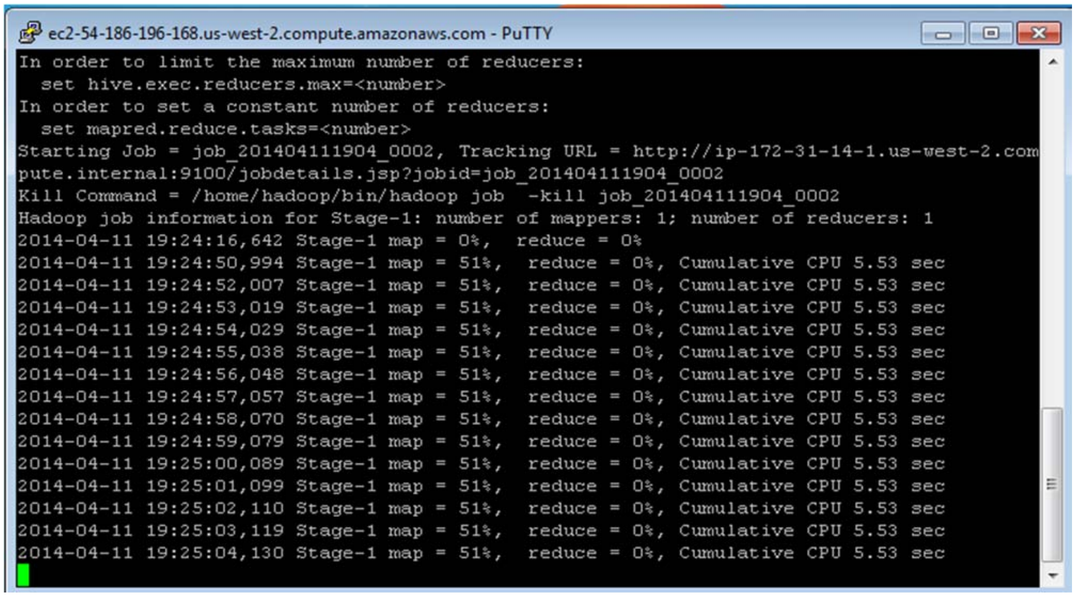
Figure 6.4. Screenshot - hive commands to create table and load data.

Table 6.4. Average Time in Minutes to Load Sample Data in Hive Table on Hadoop Cluster with Increasing Number of Nodes

data size	no of nodes	raw data load into data table time in minutes
4GB	2	3.16
	4	2.97
	6	3
	8	2.91
	10	2.77
6GB	2	4.62
	4	4.6
	6	4.75
	8	4.79
	10	4.64

Table 6.5. Average Time in Minutes to Analyse Data on a Hadoop Cluster with Increasing Number of Nodes

data size	no of nodes	select count(movieid) from ratings group by userid query time	select count(movieid) from ratings where rating like ‘5’ query time
4GB	2	5.94	4.56
	4	4.43	3.29
	6	3.64	2.95
	8	3.03	2.58
	10	3.11	2.36
6GB	2	7.6	6.17
	4	5.26	4.37
	6	4.31	3.55
	8	4.25	3.13
	10	3.46	2.83



```
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapred.reduce.tasks=<number>
Starting Job = job_201404111904_0002, Tracking URL = http://ip-172-31-14-1.us-west-2.com
pute.internal:9100/jobdetails.jsp?jobid=job_201404111904_0002
Kill Command = /home/hadoop/bin/hadoop job  -kill job_201404111904_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2014-04-11 19:24:16,642 Stage-1 map = 0%,  reduce = 0%
2014-04-11 19:24:50,994 Stage-1 map = 51%,  reduce = 0%, Cumulative CPU 5.53 sec
2014-04-11 19:24:52,007 Stage-1 map = 51%,  reduce = 0%, Cumulative CPU 5.53 sec
2014-04-11 19:24:53,019 Stage-1 map = 51%,  reduce = 0%, Cumulative CPU 5.53 sec
2014-04-11 19:24:54,029 Stage-1 map = 51%,  reduce = 0%, Cumulative CPU 5.53 sec
2014-04-11 19:24:55,038 Stage-1 map = 51%,  reduce = 0%, Cumulative CPU 5.53 sec
2014-04-11 19:24:56,048 Stage-1 map = 51%,  reduce = 0%, Cumulative CPU 5.53 sec
2014-04-11 19:24:57,057 Stage-1 map = 51%,  reduce = 0%, Cumulative CPU 5.53 sec
2014-04-11 19:24:58,070 Stage-1 map = 51%,  reduce = 0%, Cumulative CPU 5.53 sec
2014-04-11 19:24:59,079 Stage-1 map = 51%,  reduce = 0%, Cumulative CPU 5.53 sec
2014-04-11 19:25:00,089 Stage-1 map = 51%,  reduce = 0%, Cumulative CPU 5.53 sec
2014-04-11 19:25:01,099 Stage-1 map = 51%,  reduce = 0%, Cumulative CPU 5.53 sec
2014-04-11 19:25:02,110 Stage-1 map = 51%,  reduce = 0%, Cumulative CPU 5.53 sec
2014-04-11 19:25:03,119 Stage-1 map = 51%,  reduce = 0%, Cumulative CPU 5.53 sec
2014-04-11 19:25:04,130 Stage-1 map = 51%,  reduce = 0%, Cumulative CPU 5.53 sec
```

Figure 6.5. Screenshot - hive run select query.

It can easily be seen from Figure 6.6 and 6.7 that query execution time significantly goes down with increased number of nodes. This is due to the fact that more number of cores are available to execute the MapReduce tasks in parallel. This is like trying to sort and shuffle x number of objects but with more resources at hands. Since Hadoop brings computation to nodes, there isn't any overhead involved in transferring the data to be analysed.

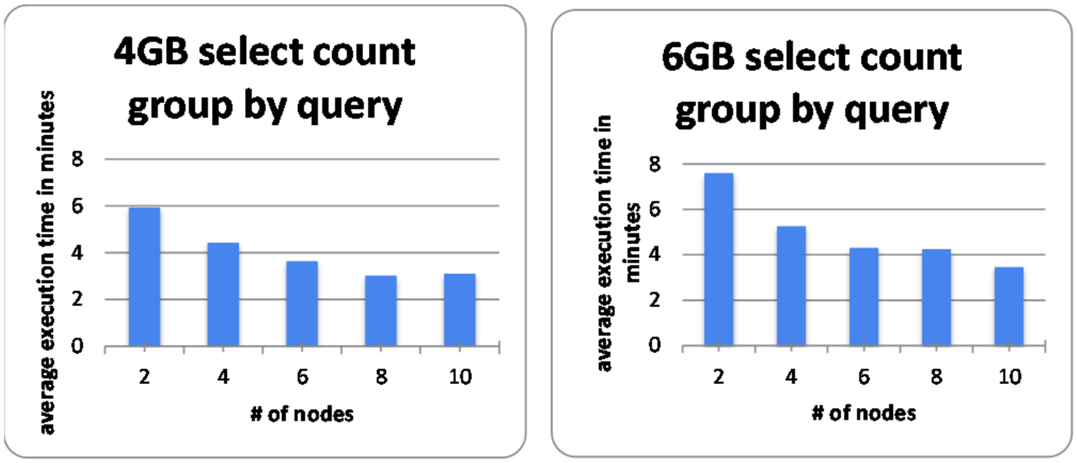
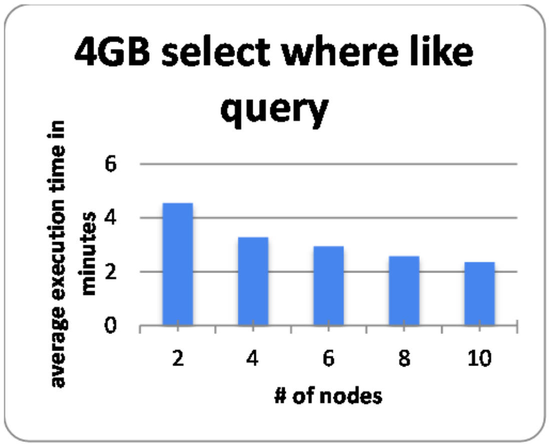


Figure 6.6. Bar graph visualization for experiment 6.3.2 part I.



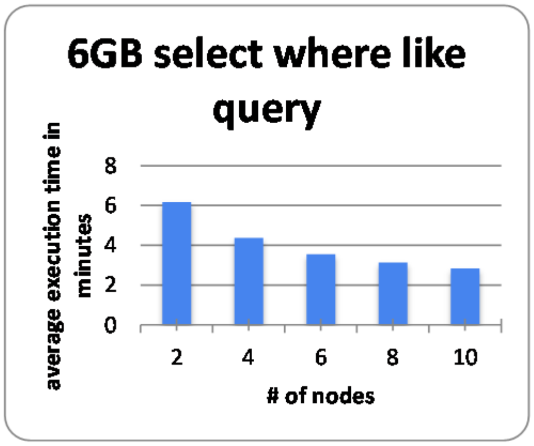


Figure 6.7. Bar graph visualization for experiment 6.3.2 part II.

6.3.3 RDBMS vs Hadoop – Data Load Performance

This experiment focuses on data load time comparison between RDBMS and Hadoop Experiment is done with increasing data size to find the optimum point where Hadoop would start to prove more efficient than RDBMS. See Table 6.6.

Table 6.6. Average Time in Minutes to Load Sample Data on RDBMS vs. Hive Table on a Hadoop Cluster

	RDBMS	**Hadoop**
datasize GB	rdbms data load time	hadoop data load time
0.5	2.78	1.39
1	3	0.83
2	5.48	1.53
4	10.9	3.16
6	15.64	4.62

Experiment 6.3.3 shows significant performance gain with Hadoop due to its distributed architecture over single Partition RDBMS. The difference in time needed to load data on RDBMS vs Hadoop seems to exponentially grow with the data size. See Figure 6.8.

6.3.4 RDBMS vs Hadoop – Data Query Execution

This experiment tries to analyse and compare query execution times between RDBMS and Hadoop. Experiment is done with increasing data sizes (See Table 6.7) to

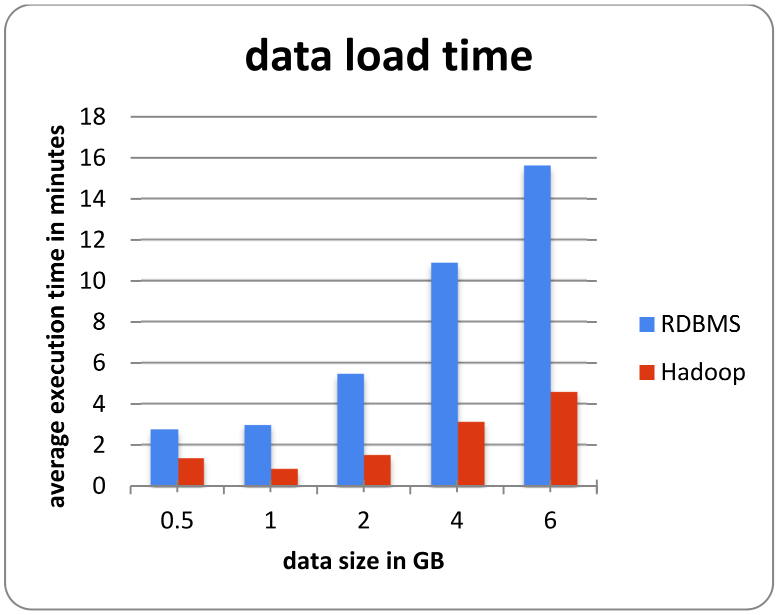


Figure 6.8. Bar graph visualization for experiment 6.3.3.

Table 6.7. Average Time in Minutes to Analyze Data on RDBMS vs. Hive Table on a Hadoop Cluster

	RDBMS	**Hadoop**	**RDBMS**	**Hadoop**
datasize GB	rdbms select count group by query time	hadoop select count group by query time	rdbms select count like timestamp query time	hadoop select count like timestamp query time
0.5	1.66	2.2	0.08	2.02
1	4.58	3.04	0.18	2.27
2	8.28	4.03	3.15	3
4	11.65	5.64	5.84	3.59
6	14.83	7.6	7.32	5.96

analyze if and where Hadoop gains efficiency over RDBMS with its distributed and parallel processing architecture. Performance is evaluated for two queries.

- Part I - select count(movieid) from ratings group by userid
- Part II - select count(movieid) from ratings where ratings like '5'

Experiment 6.3.4 endorses the fact that Hadoop runs data analysis in parallel over the nodes in the cluster. Hadoop's ability to distribute shuffle, sort activity on multiple nodes available in the cluster proves to gain performance over running data analytics query in traditional RDBMS set up. See Figure 6.9 and Figure 6.10.

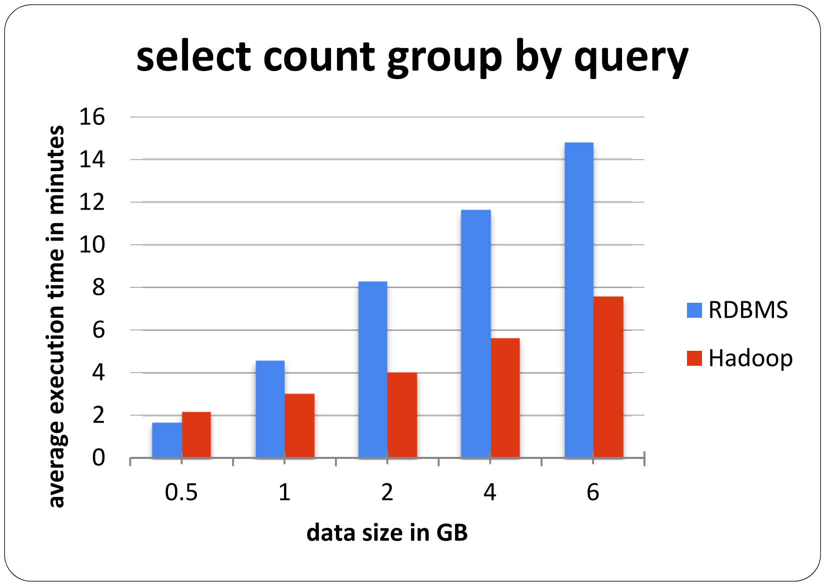


Figure 6.9. Bar graph visualization for experiment 6.3.4 part I.

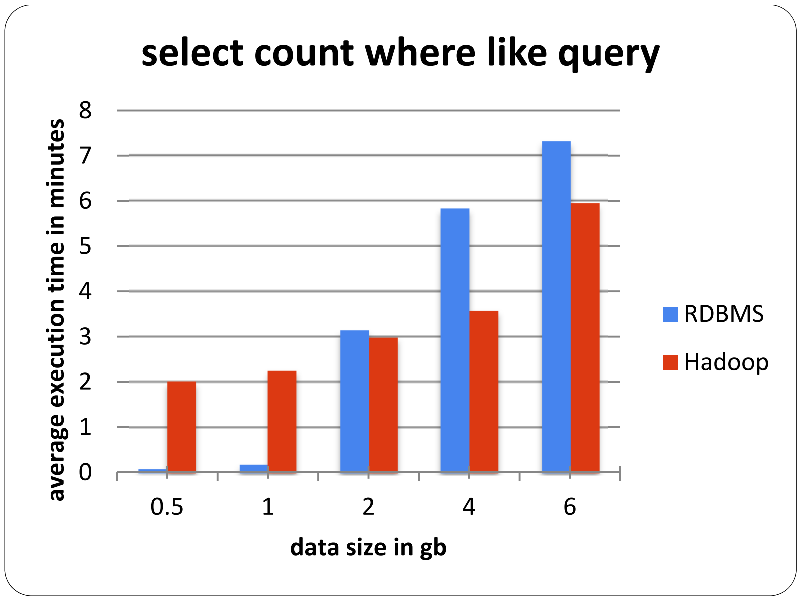


Figure 6.10. Bar graph visualization for experiment 6.3.4 part II.

CHAPTER 7

REAL TIME BIG DATA ANALYTICS

Just a few years ago, generating a result from a query on petabytes of data in less than hour was thought of nothing short of miracle. However, technological advances have made it possible to see results in under a minute. You think of a query, you get a result, and you begin your experiment.

Though Big Data analytics helps towards data-driven decisions**,** applications of big data analytics are currently bound by significant latency perceived by end users. Almost all widely used Big Data technologies are not suitable for real time analytics. Task is as difficult as finding needle in haystack in no time. Problem further exacerbates when data can be linked with other data [22].

Most of the focus on Big Data so far has been on situations where the data to be queried is already been collected and stored in a Big Data database. Real Time Big Data Analytics on the other hand attempts to analyze constantly changing data - streaming data. Events/information is continuously fed to the system and is analyzed based on insights from already collected data. It would be unreasonable to store all the events and expect RTBDA system to provide answers within milliseconds. Hence, RBDTA system works by sampling events without losing value from the data.

How fast is fast enough? Fast is a relative term, different system/scenarios may have different expectations. Getting a traffic update a two minute later might not be as bad as firing a trade 10 milliseconds later.

7.1 Applicability

Real Time Big Data Analytics finds application in myriad of fields including usual suspects like Algorithmic Trading and Healthcare. Let’s examine some of the applications in depth.

- Finance Industry: Financial industry is one of the most brutal industry where time is on the top list of the most influencing factors. It may not be advised to wait too long and hope that a competitor did not grab the opportunity in the meantime. Trading robots (algorithms) need to mine gigabytes of data and need to trigger (or decide not to trigger) a trade within milliseconds based on current market conditions. Even traders need to mine lot of data to make better baits in stock market.

- Fraud detection: RTBDA can be used to detect fraudulent transactions. If fraud detection system finds that customer made a transaction somewhere on east coast and within 5 minutes there is transaction on east coast, an alarm would trigger. This would save hundreds of millions of dollars every year.
- Traffic updates and Routing: Imagine a world where you won't be stuck in traffic at standstill wondering why you decided to take the route you are stuck on, by relying on smart phones for navigation applications that almost everyone and ability to quickly mine the enormous streaming data generated by these navigation apps. Most of these applications do quite a good job to provide multiple routes and let user select a route. Traffic situations however change all the time and most of current navigation systems don't account for traffic congestions/accidents to provide user with a better alternate route.
- System Monitoring/Log mining: Today's applications/clusters are growing more and more complex, which makes it hard to monitor and trigger alarms if error rate and/or performance of the system deviates. Log mining systems like Splunk, New Relic can receive stream of logs from thousands of nodes running application in real time. Real time feed is compared against historic data to look for any deviation from normal behavior. Any deviation above a set threshold would be considered as potential issue and alarm will be triggered

7.2 APACHE SPARK AS RTDBA

Hadoop's reliance on persistent storage to provide fault tolerance and its one-pass computation model make MapReduce a poor fit for low-latency applications and iterative computations, such as machine learning and graph algorithms.

Apache Spark is open source project developed in AMPLab at University of Berkley. Apache Spark™ is a fast and general engine for large-scale data processing. Spark runs programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Spark is built on top HDFS, however, unlike Hadoop which use disk heavily, spark works off i memory data. Spark stores data in Resilient Distributed Dataset - RDD, which lives primarily in memory. With growing data volume one might it limit of available main memory, in such case spark will either spill it disk or recompute partitions that can't fit in memory.

Spark also differs from Hadoop by providing numerous primitive operations like map, reduce, sample, join and group-by. All these operations can run parallel with respect to RDDs.

Spark provides better performance than hadoop in following scenarios [23].

- Iterative algorithms: Spark allows users and applications to explicitly cache data by calling the cache() operation. As a result data is now available in memory providing dramatic improvement for subsequent queries that access same dataset repeatedly.
- Streaming data: Spark provides an API to work with data streams. With low-latency processing provided by Spark, with its API for streams, Spark provides perfect opportunity to build systems to process Real Time Streaming Data.
- Reuse intermediate results across multiple computations instead of every time saving those to disk and accessing at a later point. Unfortunately, in most current frameworks, the only way to reuse data between computations (e.g., between two MapReduce jobs) is to write it to an external stable storage system, e.g., a distributed file system. RDDs provide fault tolerance by logging the transformations used to build a dataset (its lineage) rather than the actual data. If a partition of an RDD is lost, the RDD has enough information about how it was derived from other RDDs to recompute and recover, often quite quickly, without requiring costly replication.
- Unlike Hadoop, Spark is not only a Map/Reduce framework; it is not limited to iterative Map and Reduce phases that need an implicit group-by in between. The extra Map step would require serialization, deserialization and disk IO calls for each iteration. RDDs essentially are cached in-memory hence helps to avoid frequest serialize/deserialize, IO and the overhead of spinning up the task instances. These distinctions allow a more efficient use of resources and are an important step forward for a certain class of Big Data problems.

CHAPTER 8

CONCLUSION

Big data has become highly prevalent in organization’s day-to-day activities. Amount of big data and rate at which it’s growing is enormous. And big data technology is sure to soon knock on the door of every enterprise, organization, and domain.

RDBMS, even with multiple partitioning and parallelizing abilities fails to easily and cost-effectively scale to growing data needs. At the same time it expects data to be structured and is not so capable of storing and analyzing raw unstructured data which is common to encounter with the advent of wearable technologies, smartphones, and social networking websites.

Hadoop is the most widely accepted and used open source framework to compute big data analytics in an easily scalable environment. It’s a fault tolerant, reliable, highly scalable, cost-effective solution that’s supports distributed parallel cluster computing on thousands of nodes and can handle petabytes of data. Two main components HDFS and MapReduce contribute to the success of Hadoop. It very well handles storing and analyzing unstructured data. Hadoop is a tried and tested solution in the production environment and well adopted by industry leading organizations like Google, Yahoo, and Facebook.

Though previous versions of Hadoop did not have real time data analytics component, Apache has recently introduced Spark as a solution to real time big data analytics. Spark relies on Resilient Distributed Data and is said to provide results in split of a second.

Many domains, like finance, social networking, healthcare, security, log mining are adopting big data technologies with a promise to gain insights from the ability to easily mine large amounts of data.

As a future work it will be interesting to setup real time big data analytics engine and see how differently it handles data that Hadoop MapReduce, benchmark its performance against distributed batch processing architecture and understand how it helps overcome the challenges in batch processing big data analytics system.

REFERENCES

[1] V. Mayer-Schoönberger and K. Cukier. *Big data – a revolution that will transform how we live, work, and think*. Eamon Dolan/Houghton Mifflin Harcourt, Chicago, Illinois 2013.

[2] Wikipedia. Big data, 2014. http://en.wikipedia.org/wiki/Big_data, accessed April 2014.

[3] VITRIA. The Operational Intelligence Company, 2014. http://blog.vitria.com, accessed April 2014.

[4] E. Dumbill. What is Big Data? An Introduction to the Big Data Landscape, 2012. http://strata.oreilly.com/2012/01/what-is-big-data.html, accessed April 2014.

[5] M. Stonebraker, P. Brown, and D. Moore. *Object-relational DBMSs, tracking the next great wave*. Morgan Kauffman Publishers, Inc., San Francisco, California, 2 edition, 1998.

[6] Apache Hadoop. What Is Apache Hadoop?, 2014. http://hadoop.apache.org/, accessed April 2014.

[7] Wikipedia. Apache Hadoop, 2014. http://en.wikipedia.org/wiki/Apache_Hadoop, accessed April 2014.

[8] T. White. *Hadoop – the definitive guide*. O'Reilly Media, Inc., Sebastopol, California, 1 edition, 2009.

[9] V. S. Patil and P. D. Soni. Hadoop Skeleton and Fault Tolerance in Hadoop Clusters, 2011. http://salsahpc.indiana.edu/b534projects/sites/default/files/public/0_Fault%20Tolerance%20in%20Hadoop%20for%20Work%20Migration_Evans,%20Jared%20Matthew.pdf, accessed April 2014.

[10] Apache Hadoop. MapReduce Tutorial, 2013. https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html, accessed April 2014.

[11] Apache Hadoop. HDFS Architecture Guide, 2013. http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html, accessed April 2014.

[12] K. Kline, D. Kline, and B. Hunt. *SQL in a nutshell, a desktop quick reference*. O'Reilly Media, Sebastopol, California, 3 Edition, 2008.

[13] P. J. Sadalage, and M. Fowler. *NoSQL distilled, a brief guide to the emerging world of polygot persistence*. Addison-Wesley, Reading, Massachusetts, 3 edition, 2013.

[14] S. Johnston. Seminar on Collaboration as a Service – Cloud Computing, 2012. http://www.psirc.sg/events/seminar-on-collaboration-as-a-service-cloud-computing, accessed April 2014.

[15] Wikipedia, Amazon Web Services, 2014. en.wikipedia.org/wiki/Amazon_Web_Services, accessed April 2014.

[16] Amazon Elastic MapReduce. What is Amazon EMR?, 2009. http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-what-is-emr.html accessed April 2014.

[17] Amazon Elastic Compute Cloud. Instances and AMIs, 2014. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instances-and-amis.html, accessed April 2014.

[18] B. Stinson. *PostgreSQL – essential reference*. Sams Publishing, Indianapolis, Indiana, 2001.

[19] GroupLens. MovieLens, 2014. http://grouplens.org/datasets/movielens/, accessed April 2014.

[20] R. Davies. Summary, n.d. http://files.grouplens.org/datasets/movielens/ml-10m-README.html, accessed April 2014.

[21] E. Capriolo, D. Wampler, and J. Rutherglen. *Programming hive*. O'Reilly Media, Inc., Sebastopol, California, 2012.

[22] M. Barlow. *Real-time big data analytics, emerging architecture*. O'Reilly Media, Inc., Sebastopol, California, 2013.

[23] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, and I. Stoica. Resilient Distributed Datasets, A Fault Tolerant Abstraction for In-Memory Cluster Computing, n.d. http://www.cs.berkeley.edu/~matei/talks/2012/nsdi_rdds.pdf, accessed April 2014.