Linköping University
Department of Computer and Information Science

Master's Final Thesis

A Cloud Based Platform for Big Data Science

by

Md. Zahidul Islam

LIU-IDA/SaS

LIU-IDA/LITH-EX-A--14/006--SE

2013-08-29

Examiner: Professor. Kristian Sandahl
Supervisor: Peter Bunus

Copyright

Acknowledgments

I would like to thank my wife, Sandra, for her love, kindness and support she has shown during my study which has taken me to finalize this thesis. Furthermore I would also like to thank my parents for their endless love and support.

I would like to express my gratitude to my examiner Kristian Sandahl and supervisor Peter Bunus for their assistances and guidance throughout my thesis. I can’t say thanks enough for their tremendous support and help. I feel motivated and encouraged every time I attend meeting with them. Without their encouragement and guidance this thesis would not have materialize. Last but not least, I would like to thank David Törnqvist, Christian Lundquist and everyone in SenionLab for letting me to use their infrastructure to design, develop and test my prototype.

Table of Contents

List ofTables

List ofFigures

Abstract

With the advent of cloud computing, resizable scalable infrastructures for data processing is now available to everyone. Software platforms and frameworks that support data intensive distributed applications such as Amazon Web Services and Apache Hadoop enable users to the necessary tools and infrastructure to work with thousands of scalable computers and process terabytes of data. However writing scalable applications that are run on top of these distributed frameworks is still a demanding and challenging task. The thesis aimed to advance the core scientific and technological means of managing, analyzing, visualizing, and extracting useful information from large data sets, collectively known as "big data". The term "big-data" in this thesis refers to large, diverse, complex, longitudinal and/or distributed data sets generated from instruments, sensors, internet transactions, email, social networks, twitter streams, and/or all digital sources available today and in the future. We introduced architectures and concepts for implementing a cloud-based infrastructure for analyzing large volume of semi-structured and unstructured data. We built and evaluated an application prototype for collecting, organizing, processing, visualizing and analyzing data from the retail industry gathered from indoor navigation systems and social networks (Twitter, Facebook etc). Our finding was that **developing large scale data analysis platform is often quite complex when there is an expectation that the processed data will grow continuously in future**. The architecture varies depend on requirements. If we want to make a data warehouse and analyze the data afterwards (batch processing) the best choices will be Hadoop clusters and Pig or Hive. This architecture has been proven in Facebook and Yahoo for years. On the other hand, if the application involves real-time data analytics then the recommendation will be Hadoop clusters with Storm which has been successfully used in Twitter. After evaluating the developed prototype we introduced a new architecture which will be able to handle large scale batch and real-time data. We also proposed an upgrade of the existing prototype to handle real-time indoor navigation data.

CHAPTER 1

1 Introduction

1.1 Big Data

Data, data everywhere [1]–[4]. We are generating a staggering quantity of data. The growth of data is astonishing which has profound affects in businesses. For years, companies have been using their transactional data [5] to make informed business decisions. Decreasing the cost of both storage and computing power has made companies interested to store user generated content like tweets, blog posts, social networks, email, sensors, photographs and servers log messages that can be mined for useful information. Traditional database management system, such as relational database, was proven good for the structured data but in cases of semi-structured and unstructured data it breaks. However, in reality data are coming from different data sources in various formats and vast majority of these data are unstructured or semi-structured in nature. Moreover, database systems are also pushed to its limit of storage capacity. As a result, organizations are struggling to extract useful information from the unpredictable explosion of data captured from inside and outside their organization. This explosion of data is referred as "big data".

Big data is a collection of large volume of complex data that exceeds the processing capacity of conventional database architecture[6]. Traditional databases and data warehousing technologies do not scale to handle billions of lines of data and cannot effectively store unstructured and semi-structured data. In 2011, the amount of information created and replicated in the world was 1.8 zettabytes (1.8 trillion gigabytes) and it will grow by a factor of nine in just five years (Source: IDC Digital Universe Study, sponsored by EMC, June 2011.). The Mckinsey Global Institute estimates that the data is growing 40% per year and this percentage will grow more than 44% between 2009 and 2020 [7]. To tackle the challenge we must choose an alternative way to process data.

"Big Data technologies describe a new generation of technologies and architectures, designed to economically extract value from very large volumes of a wide variety of data by enabling high velocity capture, discovery and/or analysis." [8]

Google was the pioneer of many big data technologies including MapReduce computation framework, Google distributed file systems (GFS) and distributed locking services. Amazon's distributed key-value store (Dynamo) created a new milestone in big data storage space. Over the last few years open source tools and technologies including

Hadoop, HBase, MongoDB, Cassandra, Storm and many other projects has been added in big data space.

1.2 The importance of big data

We can unlock the potential of big data by combining enterprise data with data from heterogeneous data sources and analyze them. For example, retailers usually know who is buying their products from transactions logs. Combining social media and web logs data from their ecommerce websites questions like who didn't buy and why they chose not to buy, or is there any bad review in the social network which is influencing the sale can be answered? This can enable a more effective micro customer segmentation and targeted marketing campaigns as well as improve supply chain efficiencies [9].

Retail industry is adopting big data rapidly. They are collecting huge amount of data from consumer's smart phone and social media. They are employing big data analytics in Marketing, Merchandising, and Operations etc. Mckinsey's research prediction is by 2013 over half of all U.S sales will be online [7] as depicted in Figure 1.

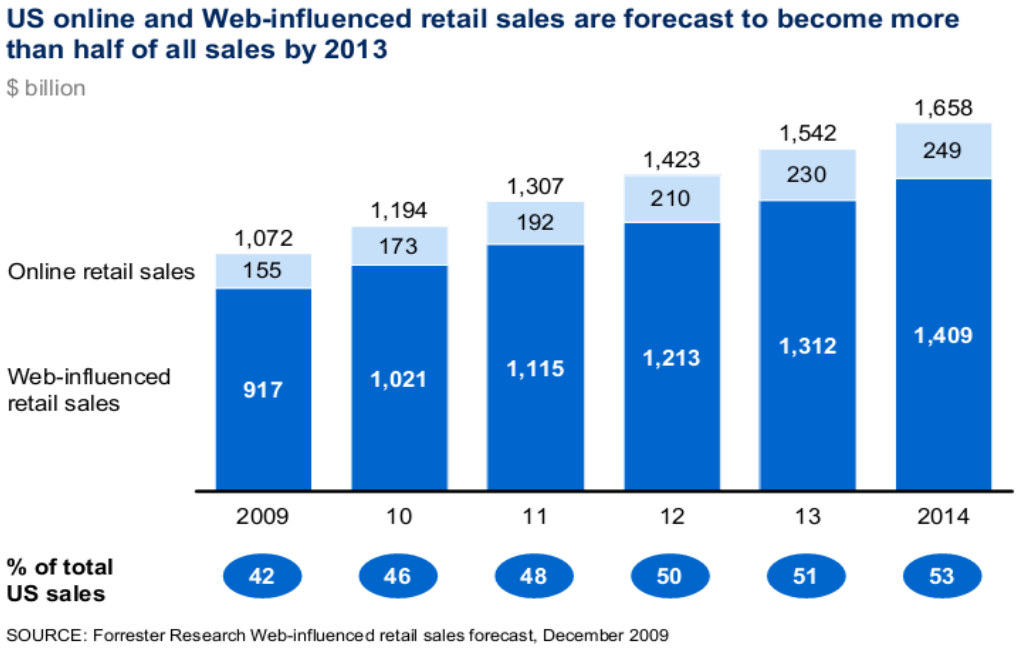

Figure 1: Online retail sales growth

1.3 Big Data user cases

The competitive edge of commodity hardware makes the big data analysis more economic. If we think about Netflix's movie streaming in 1998, it would have cost $270

to stream one movie but now it cost only $0.05. We can use big data techniques to solve various complex problems. Some of the big data use cases are given below:

1.3.1 Sentiment Analysis

It is one of the most widely discussed use case. It can be used to understand the public opinion about a particular brand, company or market by analyzing social networks data such as twitter or Facebook. It is becoming so popular that many organizations are investing huge amount of money to use some sort of sentiment analysis to measure public emotion about their company or products.

1.3.2 Predictive Analysis

Another common use case is predictive analysis which includes correlations, back-testing strategies and probability calculations using Monte Carlo simulations. Capital market firms are one of the biggest users of this type analytics. Moreover, predictive analysis is also used for strategy development and risk management.

1.3.3 Fraud Detection

Big data analysis techniques are successfully used to detect fraud by correlating point of sale data (available to a credit card issuer) with web behavior analysis (either the bank's site or externally) and cross examining it with other financial institutions or service providers.

Finally, big data provide us tools and technologies to analyze large volume of complex data to discover patterns and clues. But we have to decide what problem we want to solve.

1.4 Motivation and Problem Description

With the advent of cloud computing, resizable scalable infrastructures for data processing is now available to everyone. Software frameworks that support data intensive distributed applications such as Amazon Web Services and Apache Hadoop enable users with the necessary tools and infrastructure to work with thousands of scalable computers and process terabytes of data. However writing scalable applications that are running on top of these distributed frameworks is still a demanding and challenging task. There is a need for software tools and cloud based infrastructure for analyzing large volumes of different kinds of data (semi-structure, unstructured), aggregate the meaning from it and visualize it in human understandable format. The thesis aims to advance the core scientific and technological means of managing, analyzing, visualizing and extracting useful information from large, diverse,

distributed and heterogeneous data sets gathered from indoor navigation system, retail and different social networks.

1.5 Research Questions

The thesis project will address the following research challenges:

> How to provide software tools and a cloud based infrastructure for analyzing large volumes of data from retail industry in both semi-structured (e.g., tabular, relational, categorical, meta-data) and unstructured (e.g., text documents, message traffic) format? The project will address these issues by combining sensor structured data gathered from pedestrian indoor positioning systems and combining it with unstructured data from Twitter.

1.6 Research Methodology

1.6.1 Literature review

"Big data" is relatively new term and its meaning is subjective and unclear. In 2011, The Data Warehousing Institute (TDWI) conducted a study that showed that although most of the participants were familiar with something resembling to big data analytics yet only 18% were familiar with the term [10]. The theoretical assumptions that characterize this thesis are based on an extensive literature review from different sources. We reviewed literatures related to "Big data", "Big data analytic", "Big data tools", "Big data technologies", "Big data ecosystem", "Data visualization", "Data visualization frameworks in JavaScript", "Sentiment analysis of twit", "Sentiment analysis", "Sentiment analysis API" and "Retail analytics" from ACM digital library, IEEE Xplore and SpringerLink. We also studied numerous white papers, reports, blog posts related to above keywords from Google Scholar and Google Search. Moreover, we reviewed reports from big data market leading company's website including Google, Facebook, IBM, Amazon, Microsoft, SAP, SAS, IDC, Cloudera, DataStax, TDWI and Wikibon.

1.6.2 Prototyping

Prototyping is considered as an instrument of design knowledge enquiry in research through design[11]. A prototype is an early sample or model built to test a concept or process [12]. It can evolve in degrees of granularity from interactive mockups to fully functional system.

Throwaway and Evolutionary prototyping are two major types of prototyping in software prototyping methods[13]. In throwaway or rapid prototyping, prototypes are eventually discarded after evaluation. However, in evolutionary prototyping, prototype is continually refined and rebuilt and eventually it becomes part of the main system. In this research,

evolutionary prototyping was used. Prototype was develop using agile software development methodology [14]–[17]where end-users where involved from the beginning of the project. The prototype was continuously evaluated by the users and refined during the software development lifecycle.

1.6.3 Evaluation methods

Black box testing and white box testing methods are often use for the evaluation of the technical aspects of product [18]. Black box testing method tests the functionality of an application without knowing its internal structure. It insures that the specific inputs could generate the desired output, while white box testing method tests the internal structures of the application. Software practitioners use this method to evaluate the correctness of all possible paths of the code. It is possible to apply both black box and white box testing since the thesis is about developing a platform/tool for analyze and visualize different kinds of data (structured, semi-structured, unstructured).

We can consider sensor data and social network data (twit) as input for the analysis tool for black box testing. So, it is possible to test the tool regarding the correctness of the created output.

We can also evaluate this research from different point of views. Oates [18] has introduced three type of evaluation of a product from different points of view:

1. **Proof of concept**: It is based on developing simple prototype to show the feasibility of the solution in terms of some specific properties under specific circumstances.
2. **Proof by demonstration**: In this approach, the product will evaluate in practice (not in real-world) but applying restricted context.
3. **Real-world evaluation**: The product will be evaluated in real context not in an artificial one.

In this research, SenionLab AB, a start-up company specialized in mobile indoor positioning systems, evaluated the proof of concept and proof by demonstration. The real-world evaluation was done by a major telecom operator in Singapore and the testing is done in real context.

For the evaluation of the tool, usability evaluation method can be used. This could be done through sending a questionnaire to the telecom operator users to find out whether the system is user friendly or not. This work is considered as a future work for the thesis.

1.7 Contribution

The output of this thesis consists of the following items:

1. Application Prototype: One of the main tasks of this thesis was to design and develop an application prototype to analyze large volumes of data. We have developed a prototype that can analyze indoor navigation data, sentiment analysis on social networks data and visualize it in a human understandable ways using different kinds of graphs. The prototype is a web-based application running on Amazon EC2.
2. Visualization: One of the easiest ways to see intensity of variables in data is plotting it into heatmap. We have developed a framework that can visualize user movement pattern using heatmap.
3. Simple Object Relational Mapping (ORM): We extend the Amazon SDK and implemented an ORM to persistence data from Amazon SimpleDB. Literature Review: In this master thesis we have done an extensive literature review to understand the ecosystem of big data and its related tools and technologies. This thesis will be a useful reference for student or researcher who would like to do research on Big data ecosystem.

1.8 Demarcations

Due to the limited time available for the research the study focused primarily on big data analytics in retail industry and prototype was based on the highest priority requirements from customer with limited functionality. The prototype could be further improved in terms of functionality and scalability. However, we have proposed a better architecture in **Chapter 4** which will solve the existing limitations.

1.9 Structure of the thesis

The remaining of the thesis is structured as follows.

- **Chapter 2** introduces and defines the concept of Big data and describes associated concepts. This chapter also describes existing NSQL databases, Big data tools and related technologies.
- **Chapter 3** describes the background of the project, technologies used, system architecture and how the system works. This chapter also summarizes some typical usages of the developed system.
- **Chapter 4** presents thee valuation of our work and proposes a new architecture to improve the system in future. This chapter also provides a roadmap for bootstrapping the performance of the existing system.
- **Chapter 5** presents the conclusion of our work and highlights some future works.

We want to conclude the chapter by saying that “Keep everything (data) if you can because you never know the signal might be hidden it those piece of data”. This is the era of big data. We have everything to attack big data problem today.

CHAPTER 2

2 Theoretical Framework

2.1 Defining "Big data"

In general we can define big data comprehensively by the three Vs (Volume, Variety and Velocity) which are commonly used to characterize different aspects of big data [19], [10], [8].

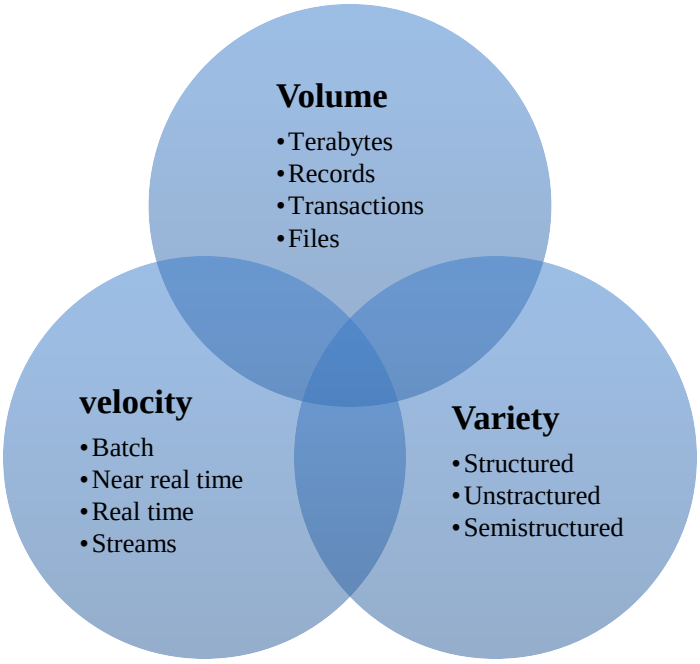

Figure 2: The three Vs of big data as defined in [20]

2.1.1 Volume

The volume in big data refers to the amount of data that is larger than the capacity of conventional relational database infrastructures. The volume introduces the first challenge of conventional IT structure. It demands for scalable storage and distributed computation. In May 2011, Mckinsey Global Institute (MGI) study found that companies in all sectors have at least 100 terabytes of stored data in United States; many of them have more than 1 petabyte [7]. We have two options for processing these large volumes of data:

- Massively parallel processing architectures. (Data warehouses or databases such as Greenplum)
- Apache Hadoop based distributed batch processing solutions.

The data warehousing approaches need a predetermined schema which is suitable for regular and slowly evolving database. However, Apache Hadoop has no constrain over the structure of the data it can process which make Apache Hadoop suitable for many applications when we deal with semi or unstructured data.

However, the choice is often influence by the other Vs where velocity comes into play.

2.1.2 Velocity

Real time analytics are becoming increasingly popular. There is a huge demand to analyze fast-moving data which is often referring as "streaming data" or "complex even processing" in the industry. There are two main reasons to consider streaming processing.

- When the input data are too fast to store in database and some level of analysis is needed when the data streams in.
- When we want to get immediate response to the data.

Both commercial and open source products can be employed to handle big data velocity problems. IBM's InfoSphere Streams is a proprietary solution. On the other hand, there are some emergent open sources such as Twitter's Storm and Yahoo's S4 which are widely use.

2.1.3 Variety

One of the other aspects of big data is the demand of analysis of semi-structured and unstructured data. Data could be in text from social networks, images or raw feeds from different sensor sources. The challenge here is to extract ordered meaning for either humans or as structured input for other applications. In relational database we need predefined schemas which result in discarding a lots of data that cannot be captured by a predefined schema. However, the underlying principle of big data is keeping everything as long as possible because the useful signals might be hidden in the bits we throw away [19].

The NoSQL databases fulfill the need for flexibility to store and query semi-structure and unstructured data. They provide enough structure to organize data without constraining fixed schema. Graph database such as Neo4j make operations on social networks data as it is graph by nature. Moreover, we can use document store, key value store, column oriented databases depending on our application.

2.2 Big data concepts

When we talk about big data there are some concepts we should know before diving into details. In this section we will describe some of the important and popular concepts:

2.2.1 Key Value Stores

In key value stores data is addressed by a unique key which is similar to dictionary or map. It provides very basic building blocks which have very predictable performance characteristics. It supports massive data storage with high concurrency. The query speed is higher than relational database [21]. Some popular key value stores are discussed in section [2.3.1].

2.2.2 Document Stores

Document stores are similar to Key-value pairs but encapsulate key-value pairs in JSON or XML like format. Every document contains a unique "ID" within a collection of documents and can be identified explicitly. Within a document, keys have to be unique too. Document database are suitable for nested data objects and complex data structures. It offers multi attribute lookups on documents [22]. The most popular use cases are real-time analytics and logging. Some most prominent document stores are described in the section [2.3.2].

2.2.3 Column Family Stores

Column family stores use Table as the data model. However, it does not support table association. All column stores are inspired by Google's BigTable[23]. BigTable is a distributed storage system for managing large volume (petabytes) of structured data across thousands of commodity servers. It is used in many Google's internal project including Web Indexing, Google Earth and Google Finance. This data model is more suitable for data aggregation and data warehouse [21]. Section [2.3.3] discussed some popular column family stores.

2.2.4 Graph Databases

Graph databases are suitable for managing linked data. As a result, applications based on many relationships are more suitable for graph databases. The data model is similar to document oriented data model, but it added additional relationships between nodes. Unlike relational databases, graph databases use nodes, relationships and key-value pairs. An use case for graph databases could be social network where each person is related with its friends, passions or interests [24]. Neo4j and GraphDB are discussed in the section [2.3.4].

2.2.5 Vertical Scaling

Traditional database systems were design to run on single machine. In order to handle growing amounts of data without losing performance can be done by adding more resources (more CPUs, memory) to a single node[25].

2.2.6 Horizontal Scaling

Recent data processing system (NoSQL) handle data growth by adding new nodes (more computers, servers) to a distributed system. The horizontal scaling approach becomes cheaper and popular due to low costs for commodity hardware[25].

2.2.7 CAP Theorem

In 2000, Professor Eric Brewer introduce CAP (**Consistency**, **Availability**, **Partition tolerance**) theorem[26] where he proved that a distributed system cannot meet the three district needs simultaneously. It can only meet any of the two.

- **Consistency** means that each client always has the same view of the data.
- **Availability** means that all clients can always read and write.
- **Partition tolerance** means that the system works well across physical network partitions.

One of the main design goals of NoSQL system is horizontal scalability. To scale horizontally, NoSQL systems need network partition tolerances which require giving up either consistency or availability[27].

2.2.8 MapReduce

MapReduce framework is the power house behind most of today's big data processing. It is a software framework that takes query over large data sets, divide it and run it in parallel over several machines. The core idea is that we write a map function that processes a key/value pair to generate a set of intermediate key/value pairs and we also write a reduce function that merges all intermediate values associated with the same intermediate key [28]. We can simplify many real world problems using this model.

Think about a scenario where we want to count the number of occurrences of each word in a large collection of documents. We could write the MapReduce code in the following pseudo-code:

```
Map (String key, String value):
// key: document name
// value: document contents
for each word w in values:
       EmitInternmediate (w, “1”);

Reduce (String key, Iterator values):
// Key: a word
// values: a list of counts
int result = 0;
for each v in values:
       result += ParseInt (v);
Emit (AsString (result));
```

The below diagram illustrates the above pseudo-code:

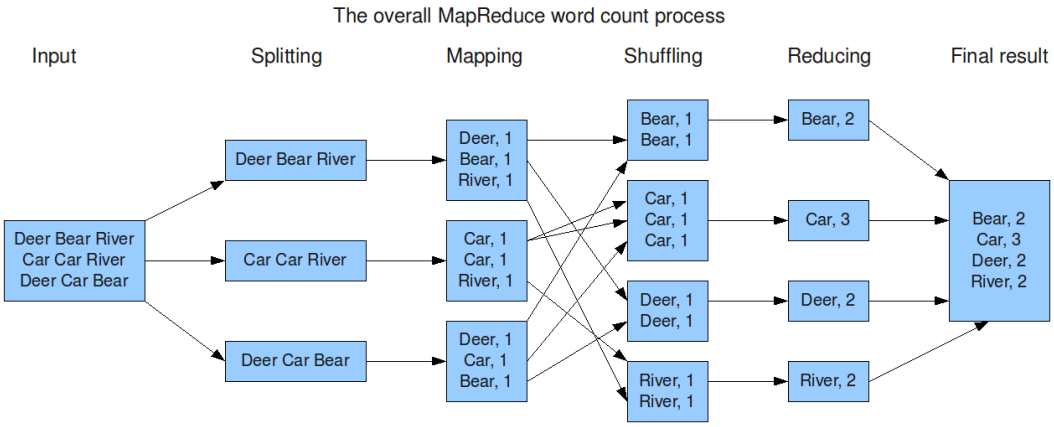

Figure 3: MapReduce from word count process [120]

There are many use cases for using MapReduce framework include [28]:

- Distributed sort
- Distributed search
- Web-link graph traversal
- Machine learning
- Count of URL access frequency
- Inverted index

2.3 NoSQL Databases

NoSQL databases are often characterized by various non-functional attributes such as consistency, scalability and performance. This aspect of NoSQL is well studied in both

academics and industries. The usage of NoSQL is always influenced by non-functional requirements of the application. The original intention of NoSQL has been modern web-scale database [29]. The following figure depicts IlyaKatsov's imaginary evaluation of the major NoSQL system families, namely, key-value stores, BigTable style databases, Document databases, Full Text Search Engines and Graph databases.

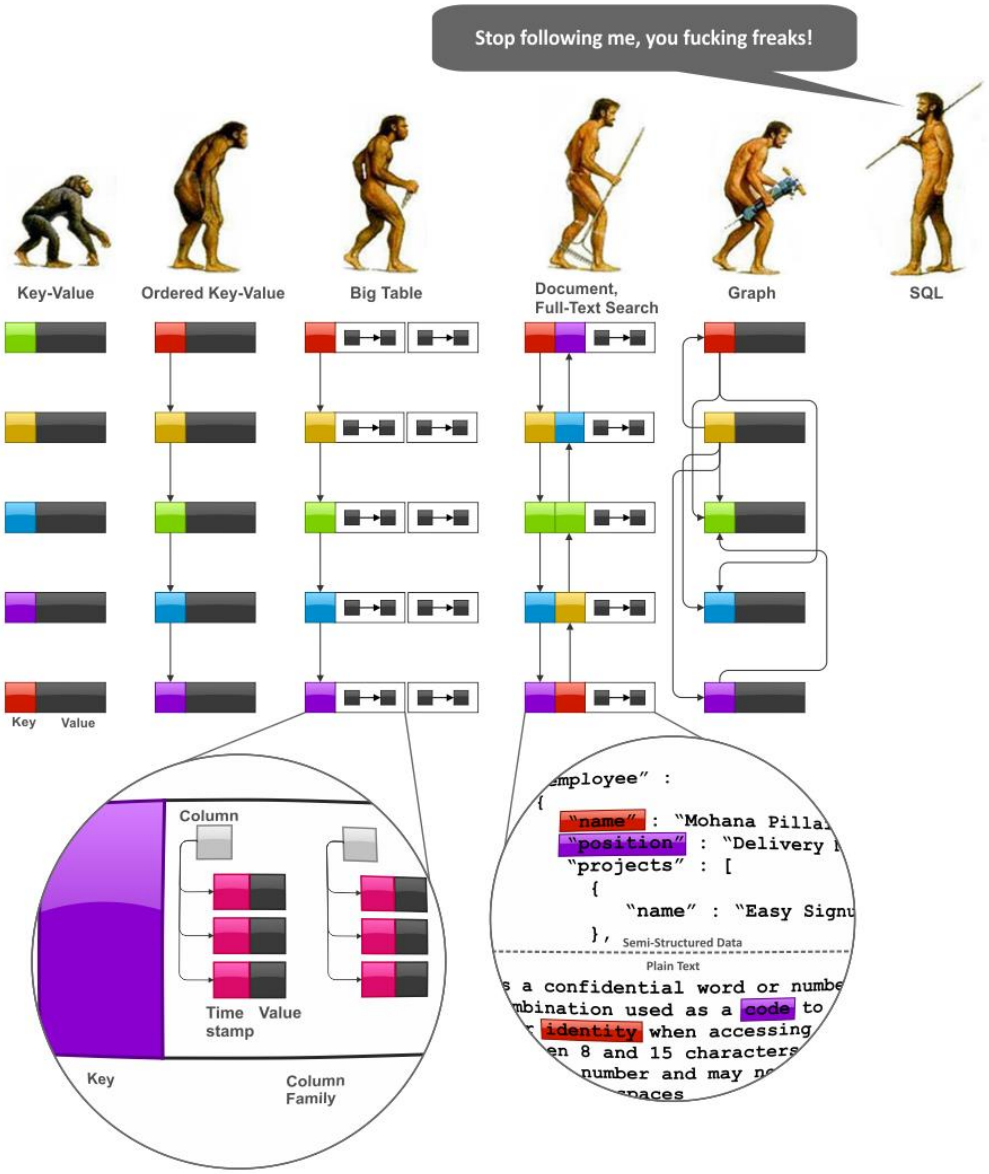

Figure 4: Evaluation of major NoSQL Systems[30]

The following sections will discuss some popular NoSQL databases based on their usage in industries.

2.3.1 SimpleDB

Amazon SimpleDB is a highly scalable, available, flexible and zero administration data store. It creates and manages multiple geographically distributed replicas of your data automatically. Developer can focus on application development because behind the scenes, Amazon SimpleDB take care high availability, data durability, schema, index management and performance tuning.

Amazon SimpleDB offers a very simple web service interface to create, store and query data easily and return the results. It provides support for most of the main stream programming languages including Java, PHP, Python, Ruby and .NET and over SDKs for them. Amazon SimpleDB was designed to integrate easily with other Amazon Web Services (AWS) such as Amazon EC2 and S3 to create web-scale applications. Amazon SimpleDB offers multiple geographical regions (Northern Virginia, Oregon, Northern California, Ireland, Singapore, Tokyo, Sydney, and Sao Paulo) to store data set so that user can optimize for latency, minimize costs or address regulatory requirements[31]–[40].

2.3.2 DynamoDB

Amazon DynamoDB was designed to address the core problems of relational database management problems such as performance, scalability and reliability with a few clicks in the AWS Management Console. It is a NoSQL database service which provides faster and predictable performance with seamless scalability with minimal database administration. Developers can store any amount of data and traffic for an application over a sufficient number of servers to handle the request faster. DynamoDB store all data on Solid State Drives (SSDs) and automatically replicated across all availability zones in a region to provide high availability and data durability[24], [31], [41]–[44]. The main features of DynamoDB service are:

- Seamless throughput and storage scaling
- Fast, predictable performance
- Easy administration
- Built-in fault tolerance
- Flexible
- Strong consistency
- Cost effective
- Secure
- Integrated monitoring
- Elastic MapReduce integration

2.3.3 Voldemort

Project Voldemort was build inspired by Amazon's Dynamo [44] paper at LinkedIn to support fast online read-writes. Voldemort leverages Hadoop elastic batch computing infrastructure to build its index and data files to support high throughput for batch refreshes. A custom read-only storage engine plugs into Voldemort extensible storage layer[45]. Data is automatically replicated and partitioned over multiple servers and each server contains only a subset of total data. Voldemort server handled failure transparently and there is no central point of failure. Moreover, it supports in memory caching, data versioning and pluggable serialization which supports common serialization frameworks like Protocol Buffers, Thrift, Avro and Java Serialization [46]. It is used at LinkedIn for different high-scalability storage problems. The source code is available under the Apache 2.0 license.

2.3.4 Redis

Redis is an open source, advanced key-value store[47]. It keeps the entire database in-memory and backed up on disks periodically which offer fast and predictable performance. It supports complex data structures as value with a large number of list and set operations handled quickly on the server side. We can horizontally scale up by clustering multiple machines together [48]. Redis has a rich set of commands [49] and support different programming languages [50] for application developer.

2.3.5 MongoDB

MongoDB is a high-performance, scalable, open source Document-oriented storage [51]–[53]. It support full index and automatically replicate data for scale and high availability. It support horizontal scaling and sharding is performed automatically. It offers atomic modification for faster in-place updates and support flexible aggregation and data processing using Map/Reduce operations. MongoDB has client support for most programming languages [54].

2.3.6 CouchDB

Apache CouchDB is an open source Document-oriented database with uses JSON as storage format[55]–[57]. In CouchDB documents can be access via REST API. It uses JavaScript as Query language and support Map/Reduce operations. CouchDB provides ACID[58]semantics by implementing a form of Multi-Version Consistency Control (MVCC) [59] which can handle a high volume of concurrent read and write without conflict. It was design considering bi-directional replication (synchronization) and off-line operation in mind. However, it guarantees eventual consistency which provides availability and partition tolerance. CouchDB is suitable for modern web and mobile applications.

2.3.7 Riak

Riak was designed inspired by Amazon's Dynamo database which uses consistent hashing, gossip protocol and versioning to handle update conflicts[48], [60], [61]. It has built-in MapReduce operations support using both JavaScript and Erlang. Riak extended Dynamo's proposed query model by offering Secondary Indexing and Full-text search. Many companies ranging from large enterprises to startups are using Riak in production [62].

2.3.8 Cassandra

Apache Cassandra is an open source column-oriented distributed database management system which was developed at Facebook[63], [64]. It was design inspired by Dynamo paper[44]. It was design using built-for-scale architecture to handle petabytes of data and thousands of users/operations per second. Its peer-to-peer design offers no single point of failure and delivers linear performance gains for both read and writes operations. Cassandra has tunable data consistency and read/write can be performed in any node of the cluster. It provides a simplified replication which insures data redundancy across the cluster. Its data compression can reduce the footprint of raw data by over 80 percent. Cassandra uses a SQL-like query language (CQL) and support for key development languages and operating systems. It runs on commodity hardware[65].

2.3.9 BigTable, HBase and Hypertable

Bigtable is a distributed column-oriented data storage system for managing structure data at Google[23]. It was design to scale. It can handle petabytes of data across thousands of commodity servers. Many projects including web indexing, Google Earth, Google Analytics, Google Finance, Google Docs and Orkut at Google use Bigtable as data storage.

Many NoSQL data storage was modeled after Google Bigtable[66].HBase[67] and Hypertable[68] are two open source implementation of Google's Bigtable. Data in HBase is logically organized into tables, rows and columns. It runs on top of HDFS [2.5.1] as a result it can be easily integrated with other Hadoop ecosystem's tools such as machine learning (Mahout) and system log management (Chukwa). However, HBase cannot run without Hadoop cluster and HDFS file system, a system similar with Google's Google File System (GFS).

Hypertable was developed and open sourced by search engine Zvents. It stores data in a table, sorted by a primary key. It achieved scaling by breaking tables in contiguous ranges and splitting them up to different physical machines [66]. Write and read speed in one node are up to 7 MB/s and 1 M cells/s when writing 28M column data to Hypertable[21].

2.3.10 Neo4j

Neo4j[69] is an open-source, high-performance graph database which is optimize for faster graph traversals. Unlike Relational database, Neo4j data model is a Property Graph [70]. It is massively scalable, highly available and reliable with full ACID transactions. Neo4j provides human readable expressive graph query language with very powerful graph traversal framework for high-speed queries. It is accessible via REST interface or Java API. Neo4j is supported by Neo Technology and it has a very active community.

2.4 Big Data Technologies

Table 1: Hadoop Ecosystem at a glance (Dumbill, 2012, p. 20)

	Hadoop Ecosystem
Ambari	Deployment, configuration and monitoring
Flume	Collection and import of log and event data
HBase	Column-oriented database scaling to billions of rows
HCatalog	Schema and data type sharing over pig, Hive and MapReduce
HDFS	Distributed redundant File system for Hadoop
Hive	Data warehouse with SQL-like access
Mahout	Library of machine learning and data mining algorithms
MapReduce	Parallel computation on server clusters
Pig	High-level programming language for Hadoop computations
Oozie	Orchestration and workflow management
Sqoop	Imports data from relational database
Whirr	Cloud-agnostic deployment of clusters
Zookeeper	Configuration management and coordination

2.4.1 Apache Hadoop

Hadoop is an open source implementation of MapReduce framework managed and distributed by the Apache Software foundation. It was originally developed by Yahoo! as a part of open-source search engine Nutch. Hadoop distribute MapReduce job across a cluster of machines and take care of [48]:

- Chunking up the input data,
- Sending them into each machine
- Running MapReduce code on each chunk
- Checking the code running
- Passing results either on to next processing steps or to the final output location
- Send each chunk of data to the right machine
- And finally write debugging information on each job’s progress.

Hadoop is now a day default choice for analyzing large data set for batch processing. It has a large ecosystem of related tools which makes writing individual processing steps easier or orchestrates more complex jobs. Hadoop provide a collection of debugging and reporting tools and most of them are accessible through a web interface which make easy to track MapReduce job state and drill down the errors and warning log files.

2.4.2 Hive

Hive is an open source data warehousing solution on top of Hadoop[71], [72]. Facebook created it as data warehouse solution over their large volumes of data stored in HDFS. It allows queries over data using SQL like syntax which is called HiveQL[71]. Hive query are compiled into MapReduce jobs and then execute on Hadoop. It supports select, project, join, aggregate, union and sub-queries like SQL queries. Hive is suitable for such use cases where we have predetermined structure of data. Hive's SQL like syntax makes it ideal point of integrate between Hadoop and business intelligent tools [72].

For example, if table page_views is partitioned on column date, the following query retrieves rows for just one day 2008-03-31 [73].

```
SELECT page_views.*
FROMpage_views
WHEREpage_views.date>='2008-03-01'ANDpage_views.date<='2008-03-31'
```

2.4.3 Pig

Pig is a programming language that simplifies working process with Hadoop by raising the level of abstraction for processing large datasets. It simplifies the loading, expressing transactions one data and storing the final results [19].

Pig has two main components:

- The higher level language which is use to express data flows, is called Pig Latin.
- The runtime/execution environment where all Pig Latin program run. Currently, Pig has local execution in a single JVM and distributed executions on top of Hadoop are available.

Below is the example of the "Word Count" script in Pig Latin:

```
input_lines = LOAD '/tmp/my-copy-of-all-pages-on-internet' AS (line:chararray);

-- Extract words from each line and put them into a pig bag
-- datatype, then flatten the bag to get one word on each row
words = FOREACH input_lines GENERATE FLATTEN(TOKENIZE(line)) AS word;
 -- filter out any words that are just white spaces
filtered_words = FILTER words BY word MATCHES '\\w+';
```

```
-- create a group for each word
word_groups = GROUP filtered_words BY word;

-- count the entries in each group
word_count = FOREACH word_groups GENERATE COUNT (filtered_words) AS
count, group AS word;

-- order the records by count
ordered_word_count = ORDER word_count BY count DESC;
STORE ordered_word_count INTO '/tmp/number-of-words-on-internet';
```

2.4.4 Cascading, Cascalog, Mrjob, Scalding

Cascading is an open-source application framework on top of Hadoop to develop robust data analysis and data management application quickly and easily. Developer can lay out the logical flow of the data pipeline using Java API. Cascading frameworks take care of checking, planning and executing MapReduce jobs on Hadoop cluster. The framework offers some common operation like sorting, grouping and joining and support custom processing code development.

Cascalog is a higher level query language for Hadoop inspired by Datalog[74]. It is DSL developed using Clojure (functional language run on JVM). Cascalog support query on HDFS, databases or even local data. Query can be run and test in Clojure REPL or as a series of MapReduce jobs. Cascalog is built on top of Cascading framework as a result it can take advantage of both Cascading and Clojure programming language features[75].

Mrjob[76] is a python framework with offers writing MapReduce jobs in python 2.5+ and run on a Hadoop cluster or Amazon Elastic MapReduce (EMR) [77]. It allows developer to rest MapReduce program on local machine.

Scalding is a Scala based DSL for Cascading with run on top of Hadoop [78]. Like Cascalog it takes advantage of Scala functional programming language and Cascading and offer functional style of writing data processing code.

2.4.5 S4, Flume, Kafka, Storm

Apache S4[79], [80] is a distributed, scalable, Fault-tolerant system that allows programmer to easily develop applications for processing continuous unbounded streams of data. **S4** platform hides the inherent complexity of parallel processing system for application developer. S4 runs the code across a cluster of machines by the help of Zookeeper framework which handle the housekeeping details. It was developer at Yahoo and still reliably processing thousands of search processing queries per second.

Unlike S4, **Flume** [81] was design for effectively collect, aggregate and move large amounts of log data from different sources to a central data store.

Apache **Kafka**[82] is a distributed publish-subscribe stream processing system. It was originally develop at LinkedIn. The functionality of Kafka is somewhere between S4 and Flume. It has its own persistent and offer more safeguards for delivery than S4. It can be used for log processing like Flume but keeping high throughput.

Storm [83] is an open-source distributed real-time computation system. It offers the ability to process unbounded streams of data reliably at real-time. Storm can be used for real-time analytics, online machine learning, continuous computation, distributed RPC, ETL and more. Storm was developed at Twitter.

2.5 Big Data File Storage

2.5.1 Hadoop Distributed File System

Hadoop Distributed File System (HDFS)[84] was design to support MapReduce jobs to read and write large amounts of data. HDFS is a write once at creation time file system which supports renaming and moving files and true directory structure. HDFS stores data in blocks of 64MB by default which can be configurable[48]. HDFS uses a single name node to keep track of files among client machines. Client stores data in temporary local file until it can fill a complete HDFS block and then send across the network and written to multiple servers in the cluster to ensure data durability. The potential drawback of HDFS is its single name node which is a single point of failure.

2.5.2 Amazon S3, Windows Azure Blob Storage

Amazon's S3[85] is a data storage service which allow user to store large volume of data. It offers very simple REST API to store and retrieve data. We can consider S3 as a Key-Value database service which is optimized to store large amount of data as value.

Windows Azure Blobs[86] (Binary Large Object) can store up to100 terabytes of unstructured text or binary data. Blobs are an ISO 27001 certified managed service which offers REST and managed API's.

2.6 Big Data Hosting Solutions

2.6.1 Amazon EC2

Amazon EC2[87], [31] rent computers by the hours, with different Memory and CPU configuration. EC2 offers both Linux and Windows virtualized servers that can log into as root from remote computer. Among other competitors EC2 stands out because of their

ecosystem around it. It is easy to integrate Elastic Block Storage (EBS), S3 and Elastic MapReduce with EC2 with makes it easy to create temporary Hadoop clusters. Developers can upload large data sets into S3 and then analyze using Hadoop cluster running on EC2. The pricing model of Amazon's sport instance makes it more suitable for data processing jobs than in-house Hadoop cluster.

2.6.2 Google App Engine

Google App Engine[88] provides infrastructure to run web applications developed in several programming languages including Java, Python, Ruby, Go and more. Unlike EC2, developers can scale their application easily without managing machines. It offers some powerful services such as Task Queue, XMPP, Cloud Storage and Cloud SQL. Developers can manage your application performance using web-based dashboard.

2.6.3 Heroku, Elastic Beanstalk, Windows Azure

Heroku[89] offers Ruby web applications hosting. The deployment process is very simple and easily scalable. Unlike Google's App Engine you can install any Ruby gem. Moreover, it provides real SQL databases.

Amazon Elastic Beanstalk runs on top of EC2 that offers automatically scaling cluster of web servers behind a load balancer. Unlike App Engine and Heroku, developers can directly log into the machine, debug problems and tweak the environment.

Windows Azure[90]offers both web application hosting and running windows or Linux virtual machines to build, deploy and manage applications using any programming language, tool and frameworks. One of the main benefits is developer can take advantage of Microsoft's tools and technologies in Azure platform.

2.7 Big Data Processing Tools

2.7.1 Drill, Dremel, BigQuery

Dremel[91] is an interactive ad-hoc query system to analyze read-only nested data at Google. It was design to be extremely scalable (thousands of CPUs) and process petabytes of data. Thousands of Google's employees are using Dremel to analyze extremely large data sets. Unlike MapReduce it can query trillion records in seconds. Dremel offers a SQL-like query language to perform ad-hoc query. **Google BigQuery**[92] hosted Dremel and offer to query very large data sets from Google Cloud Storage or local files. Currently, it only supports CSV format files.

Apache Drill[93] is an open-source implementation of Dremel. It was design to support multiple data model including Apache Avro, Protocol Buffers, JSON, BSON and more.

Moreover, it also supports CSV and TSV file formats. Unlike Google's Dremel, it provides multiple query languages such as DrQL (A SQL-like query language for nested data which is compatible with BigQuery), Mongo Query Language and others. It supports data sources including pluggable model, Hadoop and NoSQL.

2.7.2 Lucene,SolrElasticSearch

Apache Lucene was design to provide indexing and search capability on large collections of documents and Solr is a search engine server on top of Lucene. Recently both projects merged into a single project. It has highly configurable and pluggable architecture. It can handle very large amount of data and can scale horizontally across cluster of machines[94].

ElasticSearch[95] is an open source distributed RESRful search engine build on top of Apache Lucene. Unlike Solr, ElasticSearchis mainly for people in the web world. It offers schema less and document oriented data model. The configuration is painless and it can scale horizontally very well. However, it offers fewer features than Solr which is still most popular open-source search engine server.

2.8 Machine Learning

2.8.1 WEKA

WEKA[96] is a Java based framework for machine learning algorithms. It provides command line and windows interface which is design using plug-in architecture for researchers. The algorithms can be applied to a dataset using GUI tool or from Java code. WEKA offers tool for data pre-processing, classification, regression, clustering, association rules and visualization. It is extremely handy for prototyping and developing new machine learning schemes.

2.8.2 Mahout

Apache Mahout[97] is a scalable machine learning libraries. The algorithms can be applied to reasonably large data sets on a cluster of machine. Mahout offers many machine learning algorithms including collaborative filter, clustering, K-means, Fuzzy K-means clustering, classification and more. To achieve scalability Mahout was built on top of Hadoop using MapReduce paradigm. However, it can be used in a single machine or on a non-Hadoop cluster.

2.8.3 Scikits.learn

Scikits.learn[98] is a general purpose machine learning algorithms library in Python. One of the important design goals is the stay simple and efficient. It is built upon numpy,

scipy and matplotlib. Like WEKA and Mahout it also offers various tools for data mining and analysis.

2.9 Visualization

2.9.1 Gephi

Gephi is open-source network analysis and visualization software written in Java. It is very useful for understanding social network date. LinkedIn use Gephi for visualizations. Beside Gephi's best known GUI, its scripting toolkit library can be used to automated backend tasks.

2.9.2 Processing, Protovis, D3

Processing is a graphics programming language. It is a general purpose tool to create interactive web visualization. Processing offers a rich set of libraries, example and documentation for the developers.

Protovis is a JavaScript visualization library which offers various visualization components including Bar chart, line chart, and force directed layout and more. It is very difficult to build custom component in Protovis compare to Processing. D3 library is similar as Protovis. However, its design was heavily influenced by JQuery.

2.10 Serialization

2.10.1 JSON, BSON

JSON[99] (JavaScript Object Notation) is a very popular data exchange format. It is very expressive and easy for humans to read and write. Moreover, it is easy for machine to parse, read and write in almost any programming language. JSON offers a language independent structure which has a collection of name/value pairs and an ordered list of values. The name/value pairs can be compared with dictionary, hash table, keyed list or associative array.

BSON[100] is binary-encoded serialization of JSON.BSON was design to reduce to size of JSON objects and making encoding and decoding very faster. MongoDB use BSON as their data storage and network transfer format.

2.10.2 Thrift, Avro, Protocol Buffers

Apache Thrift[101], [102] is a software framework and a set of code-generator tools. It was designed to develop and implement scalable backend services. Primarily was designed to enable effective and reliable communication across programming languages.

Thrift allows developers to define data types and service interfaces in a .thrift file in a single language (IDL: Interface Definition Language). Thrift generate all the necessary code to build service that work between many programming languages including C++, Java, Python, Erlang, C# and more.

Avro[103] offers similar functionality but using different design tradeoffs. It supports rich data structures, compact, fast, binary data format. Unlike Thrift code generation is not mandatory for reading or writing data files. It is an optional optimization for statically typed languages.

Protocol Buffers[104] was design to serialize structured data like Thrift or Avro to use in communications protocols, data storage and more. Google provide a very good developer guide for protocol buffers.

2.11 Big data market leaders

Dumbill, E. presented a market survey on big data[19] which was based on some pre-assumption that data warehousing solution cannot solve the problem. The main focus was on the solution that need storage and batch data processing which work as a backend for the visualization or analytical workbench software and commercial Hadoop ecosystem.

Below figure shows Big Data solution provider employing Hadoop.

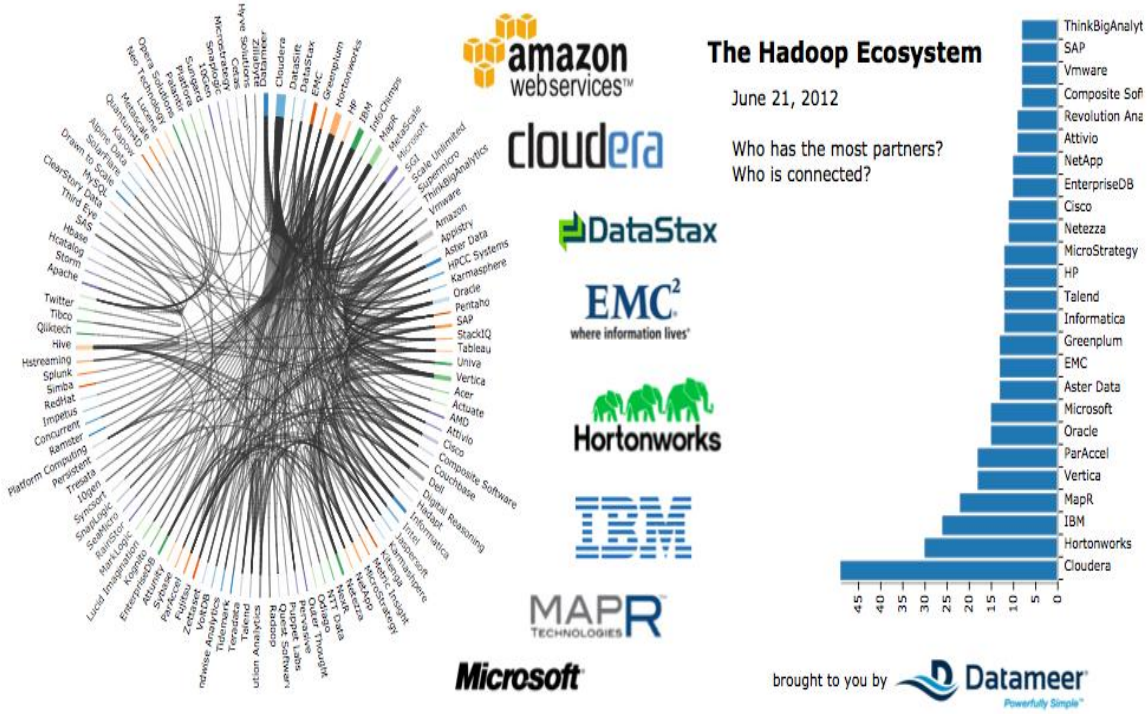

Figure 5: Big Data solution provider employing Hadoop[121]

Among Hadoop based solution providers, some companies provide Hadoop connectivity with their relational database or MPP (massively parallel processing) database. Some of the big names in this category are Oracle, Greenplum, Aster data and Vertica. On the other hand, some are Hadoop centered companies which provide Hadoop distribution and services. Below is an overview of Hadoop distributions [19]:

Table 2: Hadoop Distributors

	Cloudera	**EMG Greenplum**	**Hortonworks**	**IBM**
Product Name	Cloudera's Distribution including Apache Hadoop.	Greenplum HD	Hortonworks Data Platform	InfoSpherBigInsights
Free Edition	CDH Integrated, tested distribution of Apache Hadoop	Community Edition 100% open source certified and supported version of Apache Hadoop stack		Basic Edition An integrated Hadoop distribution
Enterprise Edition	Cloudera Enterprise Adds management software layer over CDH	Enterprise Edition Integrates MapR's M5Hadoop compatible distribution with C++ based file system and Management tools		Enterprise Edition Hadoop distribution, including BigSheets, Scheduler, text analytics, indexer, JDBC connector and security support
Hadoop Components	Hive, Oozie, Pig, Zookeeper, Avro, Hive, Flume, HBase, Sqoop, Mahout, Whirr	Hive, pig, Zookeeper, HBase	Hive, Pig, Zookeeper, HBase, Ambari	Hive, Oozie, Pig, Zookeeper, Avro, Flume, HBase, Lucene
Security	Kerboros, Role base administration and audit trails			LDAP authentication, Role base authorization , reverse proxy

Admin Interface	ClouderManager includes Centralized management and alerting	MapRadministrative interface includes Heatmap cluster administrative tools	Apache Ambari includes Monitoring, administration and lifecycle management	Administrative interfaces include Hadoop HDFS and MapReduce administration, cluster etc.
Job Management	Job analytics, monitoring and log search	JobTracker HA and Distributed NameNodeHApreventlostjobs, restarts and failover includes	Monitoring, administration and lifecycle management for Hadoop clusters	Job creation, submission, cancellation, status, logging
HDFS Access	Mount HDFS as a traditional file system	Access HDFS as a conventional network file system	REST API to HDFS	

Some other companies also provide Hadoop distributions including Microsoft, Amazon Elastic MapReduce, MapR and Platform Computing. The choice of the Hadoop technology vendor depends on the problem, existing technology and developer's skills of a particular organization.

2.12 Big data in the cloud

Big data and cloud technology play together very well. Here by cloud technology, we mean virtualized servers which are computing resource but present it as a regular server. We often call it infrastructure as a service (IaaS). Rackspace Cloud and Amazon EC2 are renowned for their IaaS. We can rent these servers per computation, install and configure our own software such as Hadoop cluster on NSQL database. However, beyond IaaS, Several companies provide application layer as service is known as PaaS. In PaaS we never have to worry about configuring software like Hadoop or NSQL instead we can concentrate on our core problems which remove our work load and maintenance burden. The key PaaS providers are VMware (CloudFoundry), Salesforce (Heroku, force.com) and Microsoft (Azure). The market leaders in big data platform service provider (IaaS/PaaS) are Amazon, Google and Microsoft [19].

Among other cloud service providers, we are going to focus on Amazon Web Service (AWS0) because both IaaS and PaaS can be used. AWS has a good support in hosting bag data processing platform. For big data processing Amazon web service provide Elastic MapReduce Hadoop service. Moreover, Amazon also offers several other services related to big data such as for distributed computing coordination SQS (Simple Queue Service), Scalable NOSQL database SimpleBD and DynamoDB, relation database RDMS and finally for blobs storage Amazon offer S3 bucket.

Below table show the offerings of different big data service providers:

Table 3: Big data service providers

	Amazon	**Google**	**Microsoft**
Product	Amazon Web Services	Google Cloud Services	Windows Azure
Big data storage	S3, Elastic Block Storage	Cloud Storage, AppEngine (Datastore, Blob store)	HDFS on Azure, Blob, table, queues
NOSQL	DynamoDB, SimpleDB	AppEngineDatastore	Table storage
RDMS	MySQL, Oracle etc.	Cloud SQL	SQL Azure
Application Hosting	EC2	AppEngine	Azure Compute
MapReduce	Elastic MapReduce	AppEngine	Hadoop on Azure
Big data analytics	Elastic MapReduce	BigQuery	Hadoop on Azure
Machine Learning	Via Hadoop + Mahout on EMR or EC2	Prediction API	Mahout with Hadoop
Streaming Processing	None	Prospective Search API	StreamInsight
Data Import	Network, Physically ship drives	Network	Network
Data sources	Public Data Sets	A few sample datasets	Windows Azure Marketplace
Availability	Public production	Some services in private beta	Some services in private beta

Now we have big data service platform to analyze huge volume of data. Let's think about a web analytics application. If we can add an IP address dataset with the logs from our website, we will be able to understand customer's locations. If we include demographic data to the mix, we can tell their socio-economic bracket and spending ability. We can

get more inside about data by adding more related datasets. This is where data marketplace comes into play.

Data Marketplaces are useful in three ways:

- They provide a centralized place where we can browse and discover the data we need. It also allows us to compare data and tell us about quality of the data.
- Most of the time the data are ready to use because the provider of the data collect and clean the data for us.
- It provides an economic model for broad access to data.

The four established data marketplaces are Windows Azure Data Marketplace, DataMarket, Factureal andInfochimps. Below table provides a comparison between theses providers.

Table 4: Data Marketplaces

	Azure	**Datamarket**	**Factual**	**Infochimps**
Data sources	Broad range	With a focus on country and industry stats	Geo-specialized, some other datasets	With a focus on geo, social and web sources
Free data	Yes	Yes	-	Yes
Free trial of paid data	Yes	-	Yes	-
Delivery	OData API	API, downloads	API, downloads for heavy users	API, downloads
Application hosting	Windows Azure	-	-	Infochimps platform
Previewing	Service Explorer	Interactive visualization	Interactive search	-
Tool integration	Excel, PowerPivot and other OData consumers	-	Developer tool integrations	-
Data publishing	Via database connection or web service	Upload or web/database connection	Via upload or web service	Upload
Data reselling	Yes	Yes	-	Yes
Launched	2010	2010	2007	2009

CHAPTER 3

3 Design and Development

3.1 Background of the project

SenionLab is an indoor positioning and navigation solution provider which radically improves navigation capabilities in environments were GPS systems doesnot work. SenionLab mainly provides indoor positioning and navigation software solutions which sensed data from the Smartphone's embedded sensors with indoor WIFI signals. As a part of the solution SenionLab is collection huge amount of semi-structured mobile sensor data every day. These data volume is growing exponentially with the increase of number of users. However, these huge amounts of geo tagged data become quite interesting when we analyze and visualize it in a meaningful ways.

One of the end-user location-based applications that SenionLab is providing to their customers is targeting shopping mall in Singapore. User can locate their position inside the shopping mall, search for a specific store or product and get inshore guidance and directions. While using navigation app users are generating large amount of data about their positions and movements. The shopping mall owners are interested in analyzing and visualizing meaningful pattern from user's data. This project (NavindoorCloud) was a test bed for the master thesis where we develop and test a cloud platform to analyze their unstructured and semi-structured data.

There are many use cases of indoor navigation data generated by end-users. However, for the time constrain of the thesis we choose to develop the most important requirements which are relevant to the scope of the thesis.

We followed "System Development Life Cycle (SDLC)" as a life cycle of system development; in which we had requirement analysis, design, implementation, testing and evaluation[108]. Furthermore, we have followed **agile** software development methodologies called **SCRUM**. However, we did not follow the pure SCRUM which consists of daily standup meeting. Instead, we stored all user stories into a project management system call **JIRA.** We had one week small iteration. Then, we selected small iteration time to get more frequent feedback from end-user.

3.2 Setting the Scope

3.2.1 Functional requirements

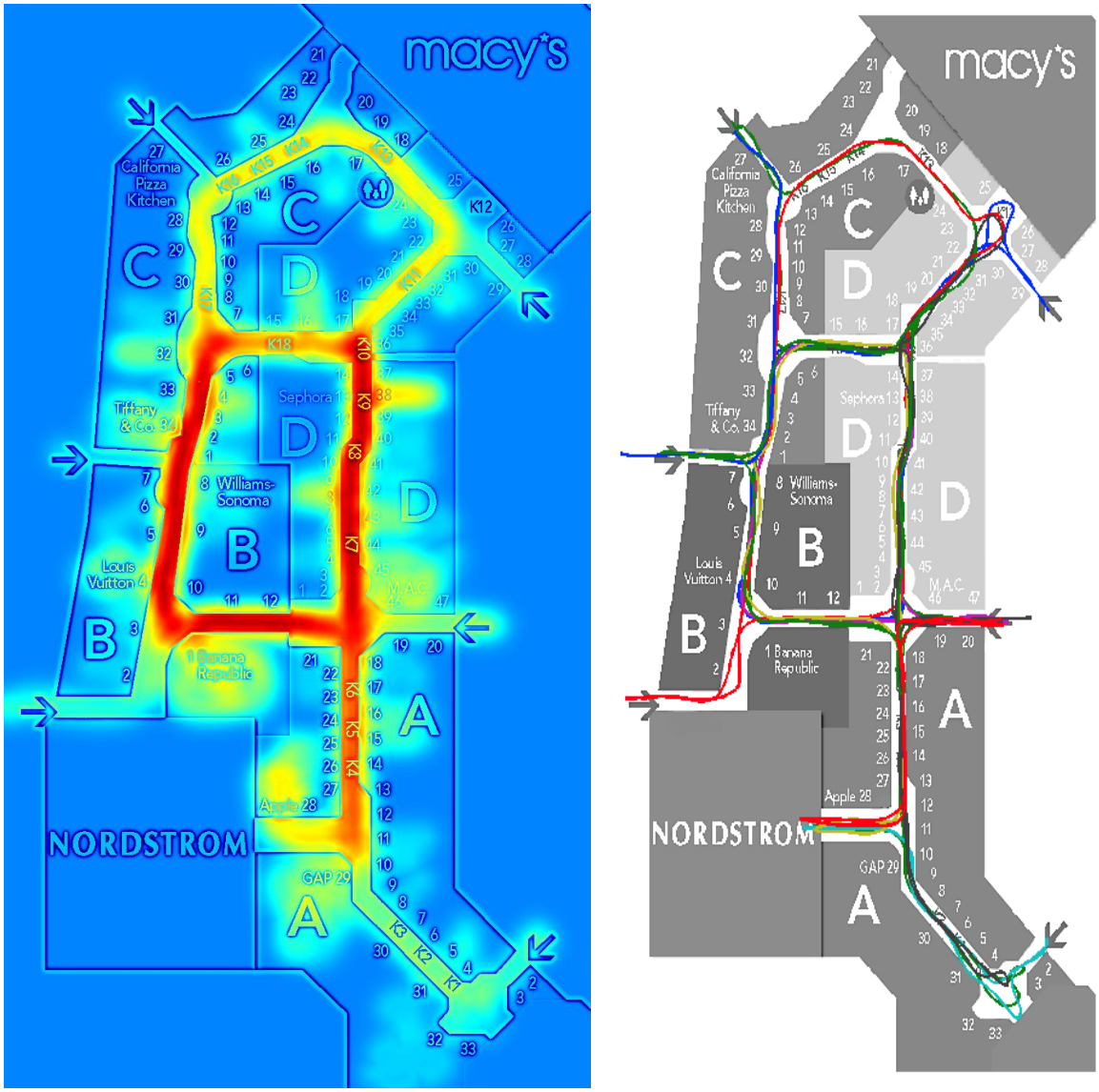

Figure 6: Heatmap of shopper's movement around the shopping mall

1. As an analyst, I want to see heatmap of shopper's movement around the shopping mall so that we can identify dominant traffic paths and low traffic or bottleneck regions.
2. As an analyst, I want to track end-user real-time so that we can assist shopper if they need any help.

3. As an indoor navigation system developer, I want to visualize mobile sensors data so that I do not have to use third party tools (example: MATLAB).
4. As an analyst, I want to understand how positive or negative are people about a shopping mall so that we can design our marketing campaign efficiently.

3.2.2 Non-functional requirements

1. The system need to be Cloud based architecture
2. The system need to utilize Amazon Web services (Amazon EC2, SimpleDB, S3, Elastic Beanstalk)
3. The system need to use third party twitter sentiment analysis service.

3.3 Technologies Used

Functional and nonfunctional requirements influenced our overall architecture of the system. The development language of choice was ASP.NET and C#. Our prototype is run on Amazon EC2 small instance machine running windows server 2008 R2, IIS7, .NET Framework 4 and ASP.NET 4.

3.4 System architecture

Figure 9 shows the higher level system architecture of NavindoorCloud. The system has three main modules in terms of functionality.

3.4.1 User Interface Layer

User interface layer is responsible for taking user input and then visualizing the results. It consists of different visualization components including Heatmap and charts.

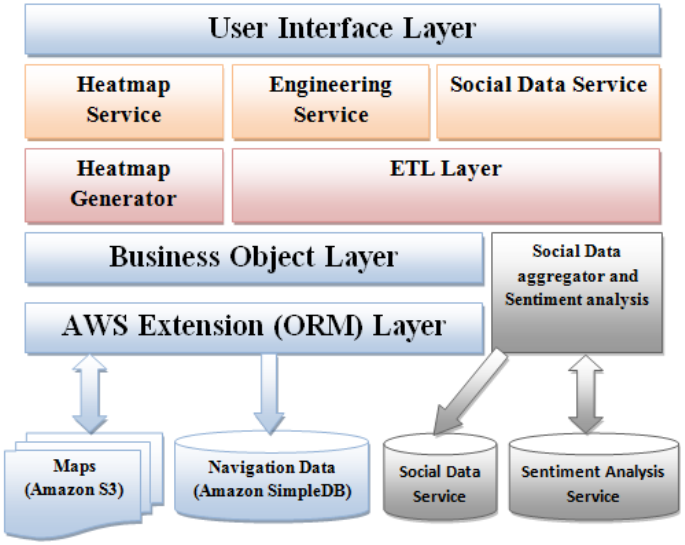

Figure 7: System Architecture of NavindoorCloud

3.4.2 Heatmap Service

When user want to visualize shopper's movement around the shopping mall then heatmap Generator generate heatmap from the data. It obtains for AWS Extension (ORM) layer for a specific shopping mall.

3.4.3 Engineering Service

When indoor navigation system developer want to plot mobile sensor (Accelerometer) data, Engineering Service read the associated mobile's sensor data from SimpleDB and visualizes using different charts.

3.4.4 Social Data Service

Social data service layer aggregation social data from different data source and analyze sentiment from them.

3.4.5 Business Object Layer

The business object layer consists of all the domain objects used in the system. Those include all shopping mall entities, heatmap entities and entities required to model the entire system. This layer was introduced to separate business logics and entity from application layer.

3.4.6 AWS Extension (ORM) Layer

The system was developed following object oriented principles. It was convenient to use object relational mapping (ORM) for data persistence. However, Amazon .NET SDK does not provide any native ORM for data manipulation. As a result we develop a simple ORM for Amazon .NET SDK. This layer performs all persistent operations.

3.4.7 ETL Layer

This layer performs ETL (Extract, Transform and Load) service which extract data from a third party micro blog data provider (**socialmention-api)**, transform it to fit for farther sentiment analysis using third party sentiment analysis web service (**Sentiment140**) and finally loading it into SimpleDB for future use.

3.4.8 Social Data Aggregator and Sentiment Analysis

This is a helper module which collects data from **socialmention-api** and supply to ETL layer for farther processing. This module also send request to **Sentiment140** API for sentiment analysis.

3.5 How the system works

In this section we will describe how the overall system works. To set the scope we will consider only two use case.

In the **first use case** we will describe how the system work when user want to visualize shopper's movement in a shopping mall.

1. When user send a query to visualize heatmap information of a shopping mall filtering by certain data range or floor number, first of all heatmap service read all associated data of a shopping mall and map information from **SimpleDB** using **AWS Extension Layer**. Primarily shopping mall data are in geo-coordinate system format (latitude and longitude).
2. **Heatmap Generator** takes those data points and converts them into Cartesian coordinate system.
3. The system then read shopping mall floor maps from S3 and generates heatmap based on Cartesian coordinate the data.
4. Finally it store heatmap into S3 and send an URL to the User interface layer for visualization.

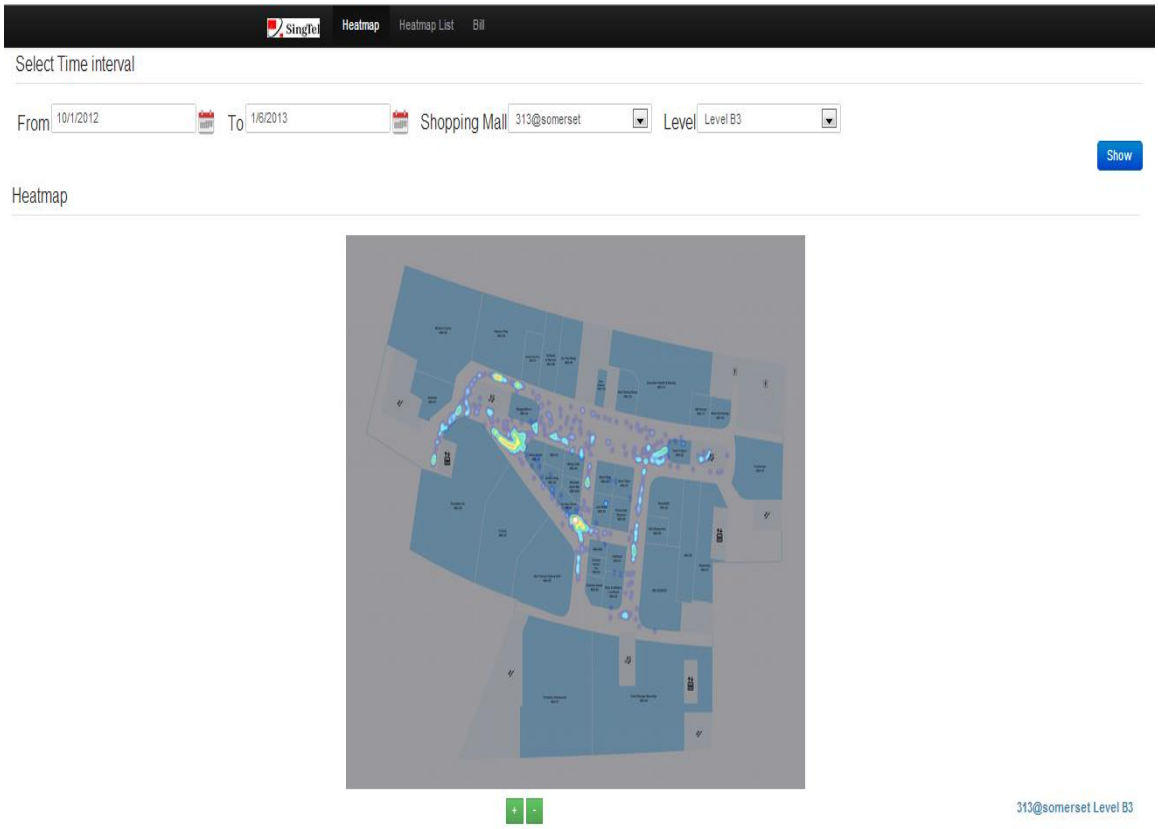

Figure 8: Heatmap Visualization

In the **second use case** we will describe how the system work when user want to understand the sentiment of shopper about a particular shopping mall.

1. When users send query to visualize sentiment of shoppers for a specific shopping mall, the **social data service layer** send a request to Social Data Aggregator and Sentiment Analysis module.
2. **Social Data Aggregator and Sentiment Analysis** collect shopping mall related information (tweet) from **socialmention-api** which is provide aggregated user generated contents from more than 100 social media including Twitter, Facebook, FriendFeed, YouTube, Digg, Google etc. However, we are only interested about Twitter data.
3. This module extracts collected data, filter and clean it if necessary. It then creates a JSON object from those data and sends it to **Sentiment140** for sentiment analysis.
4. **Sentiment140** provide services for sentiment analysis classifier. This service classified tweets into three polarities. The polarity values are:
    a. **0 : Negative**
    b. **2: Neutral**
    c. **3: Positive**

5. **Social Data Aggregator and Sentiment Analysis** module collects the result form Sentiment140 and sends it to user interface layer for visualization.

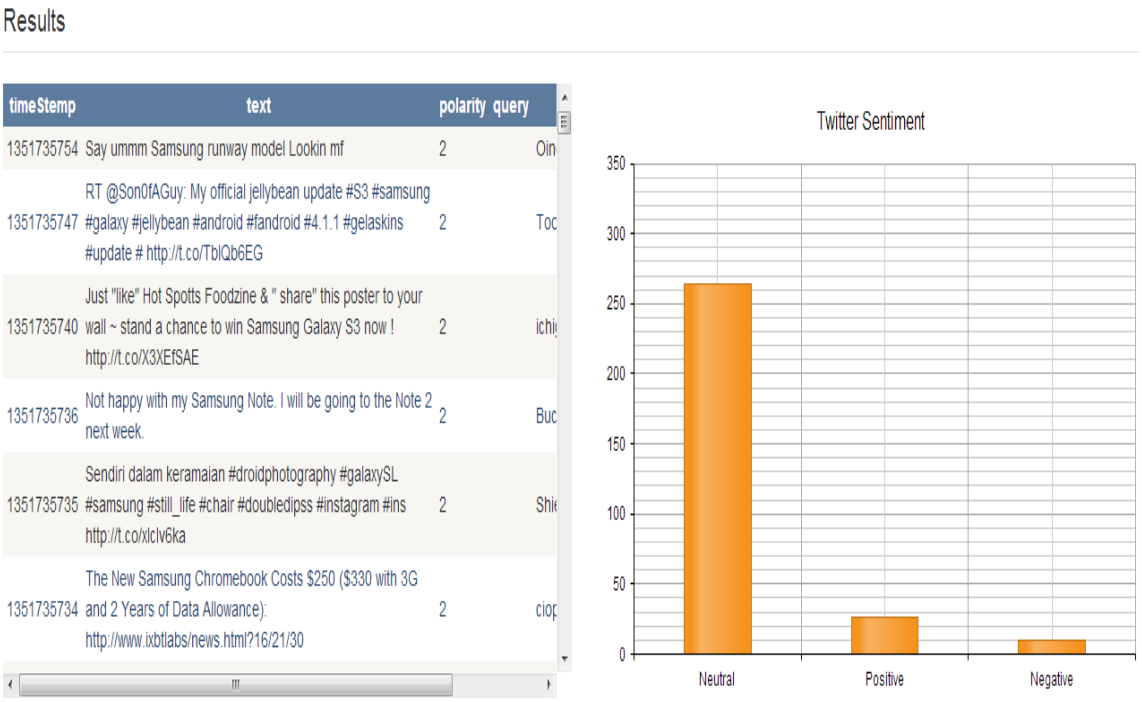

Figure 9: Twitter Sentiment visualization

3.6 Typical usage

Many analytics functionality can be built on top of NavindoorCloud. Some typical usages are:

- In-store Customer Location on Store Map
- Customer In-store Path and Dwell Time in Designated Zones
- Traffic heatmap; Real-time Visualization of Bottlenecks, or traffic around temporary fixtures, displays, etc.
- Customer count in store at any time.
- Comparison of customer count by day, hour, in-store location.
- Quickness of movement within store (to determine purchase purpose vs. browsing)
- Identify dominant traffic paths.
- Identify low traffic or bottleneck regions.
- Traffic correlation with in-store marketing campaigns.

We choose to use Amazon Web Services (AWS) which offers Platform as a Service (PaaS). As a cloud infrastructure provider AWS is a one stop place which offered everything we need to design, develop and scale-up our system. Although, Microsoft Windows Azure could fulfill our requirements; nevertheless we choose AWS because SenionLab indoor navigations system solution was built on top of AWS. Moreover, it was a non-functional requirement to use AWS infrastructure for designing and developing the prototype. We took some other design decisions during the prototype development which might be interesting. For example, we used SimpleDB for storing our data over Amazon's RDS (other relational databases offering) because of its flexibility. However, flexibility comes with price[109]. We also used Amazon S3 to store large objects such as maps, files associated with indoor navigation system and SimpleDB backup. Amazon S3 storage solution is very simple yet capable of massive scale.

CHAPTER 4

4 Discussion

4.1 Evaluation

To evaluate the quality of answers to the research questions we have to investigate validity of findings and reliability of selected methods. In other words, validity means that we have found correct data and reliability of our research methods could be proved by repeatability of research by other researchers. To increase validity of research we have selected different methods to collect data and observe facts. We reviewed the published documents on big data tools and technologies. To increase reliability and repeatability of research we tried to describe all steps and activities in this report with could be used by other researchers. Also, since the major part of our research was developing the big data platform, we have conducted black box testing which helped us to evaluate technical aspects of the developed tool.

Our prototype is up and running on the EC2 for more than six months. Real users are using it every day without any problem. To evaluate the results we have done ad hoc functional test in explorative fashion. Functional testing verifies each function of the software with the requirement specification. All functionalities of the system were tested by the developer and users of the system providing appropriate input, verifying the output and comparing the actual result with the expected results. The testing was mainly a black box testing.

Our system is stable based on its performance over time (No downtime beside maintenance or update). We were able to meet all the requirements except real-time user tracking. To implement this feature we will need a session and caching layer. Where session layer will keep track of all online users in a mall and cache navigation data in cache memory instead of SimpleDB storage. However, we were able to make a user movement simulator where we can visualize user's movement later when we already have data.

- This master thesis provides an in depth literature review on big data tools and techniques.
- This master thesis provides guidelines on how to analyze and visualize sensor data. In addition it also provides guidelines on how to analyze sentiment from micro blogs (Tweet) to discover shoppers' feelings about certain products or shopping mall.

- A web based portal was developed for unstructured and semi-structured data visualization.
- One of the main contributions of this thesis is the proposed architecture for future big data infrastructure/application development.

Computing arbitrary queries on an arbitrary dataset in real-time is problematic. Most of the times it is quite expensive or time consuming to query petabytes of data every time a user sends a request. There is no single tool or technique that provides a complete solution. Instead, we have to use different tools and techniques to build a complete Big Data System.

4.2 Lambda architecture

Lambda architecture[110]is a method for processing an arbitrary function on an arbitrary data set and returning the result with low latency. In this architecture data processing are broken into three layers: the batch layer, serving layer and the speed layer.

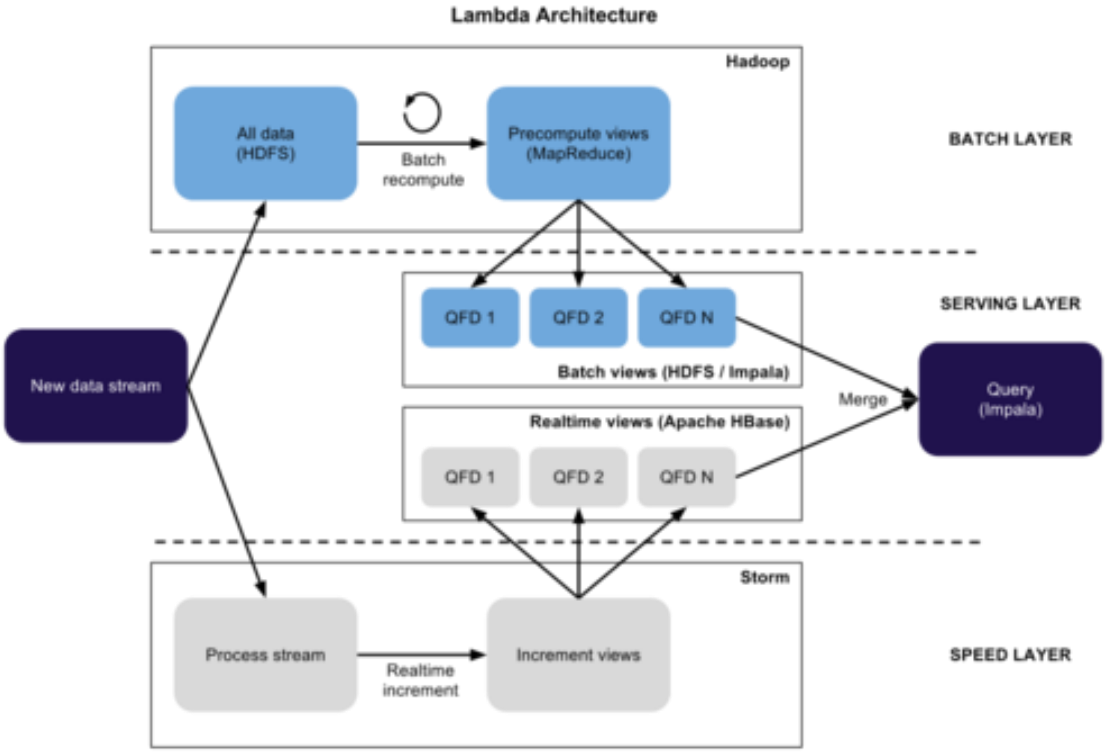

Figure 10: Lambda Architecture [111]

4.2.1 Batch Layer (Apache Hadoop)

The batch layer stores the master copy of dataset and pre-computes batch views on that master dataset. The master dataset is immutable and constantly growing. This append only dataset can be stored in HDFS. The pre-computes batch views can be computed

using MapReduce which is a continuous operation. So when data arrives it will be aggregated into the views when they are recomputed during the next MapReduce iteration.

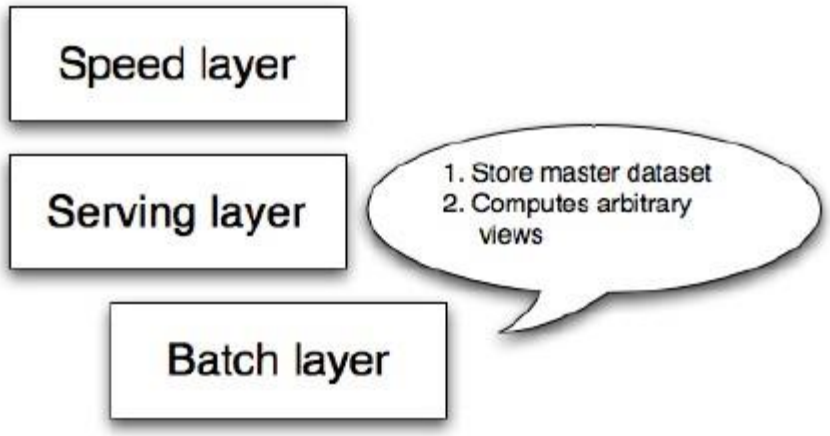

Figure 11: Batch [110]

4.2.2 Serving Layer (Cloudera Impala)

The output from batch layer is a set of batch views which are pre-computed. The serving layer is a specialized distributed database that loads in a batch views, makes them queryable, and continuously swaps in new versions of a batch view as they're computed by the batch layer. We can use Cloudera Impala to read static batch views and expose it for query by creating a table in the Hive Metastore that points to the files in HDFS [84].

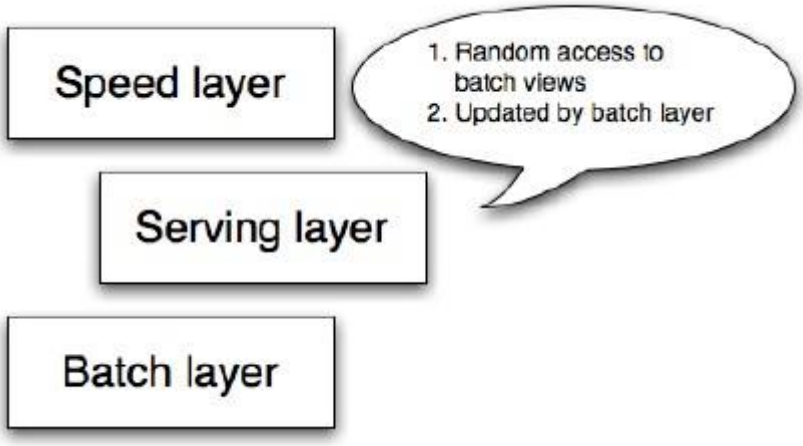

Figure 12: Serving Layer[110]

4.2.3 Speed Layer (Storm, Apache HBase)

The serving layer updates whenever the batch layer finishes pre-computing a batch view. It raises the question of what happen to the data that come to the system during the computation of a batch view. This is where speed layer come into play. Speed layer produce views based on those last few hours of data. These real-time (near real-time) views contain only the delta [112] results to supplement the batch views. The speed layer uses an incremental model where real-time views are updated only when new data is received. We can compute real-time views using Storm [83].

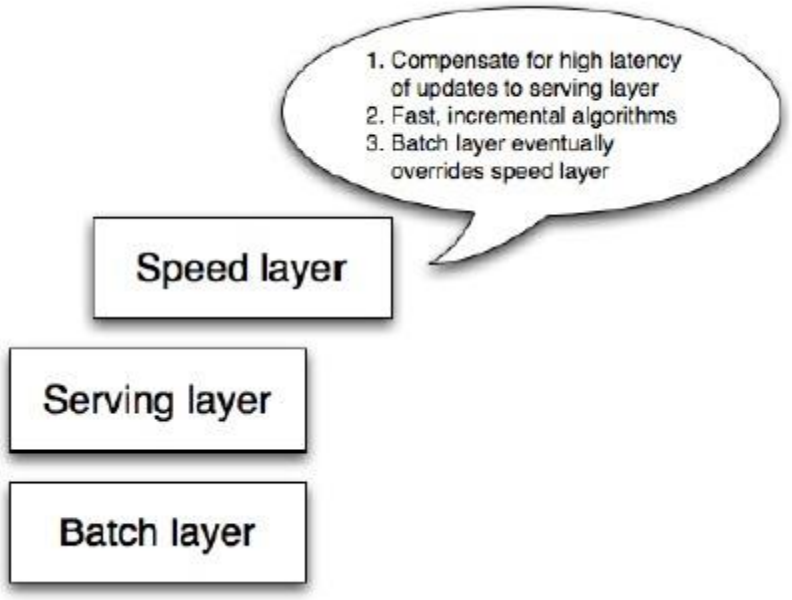

Figure 13: Speed Layer[110]

The final piece of the lambda architecture is to merge real-time and batch views and expose it for query. We can use Apache HBase for this purpose because it provides the ability for Storm to continuously increment the real-time views and at the same time can be queried by Impala merging with the batch views. Impala's ability to query both the batch views stored in the HDFS and real-time views stored in HBase make it the perfect tool for the job.

4.3 Proposed Architecture

Lambda architecture provides a roadmap to improve our framework architecture. We can fulfill all the functional and non-functional requirements using this architectural principle. Figure 13 shows our proposed architecture for NavindoorCloud with is based on lambda architecture.

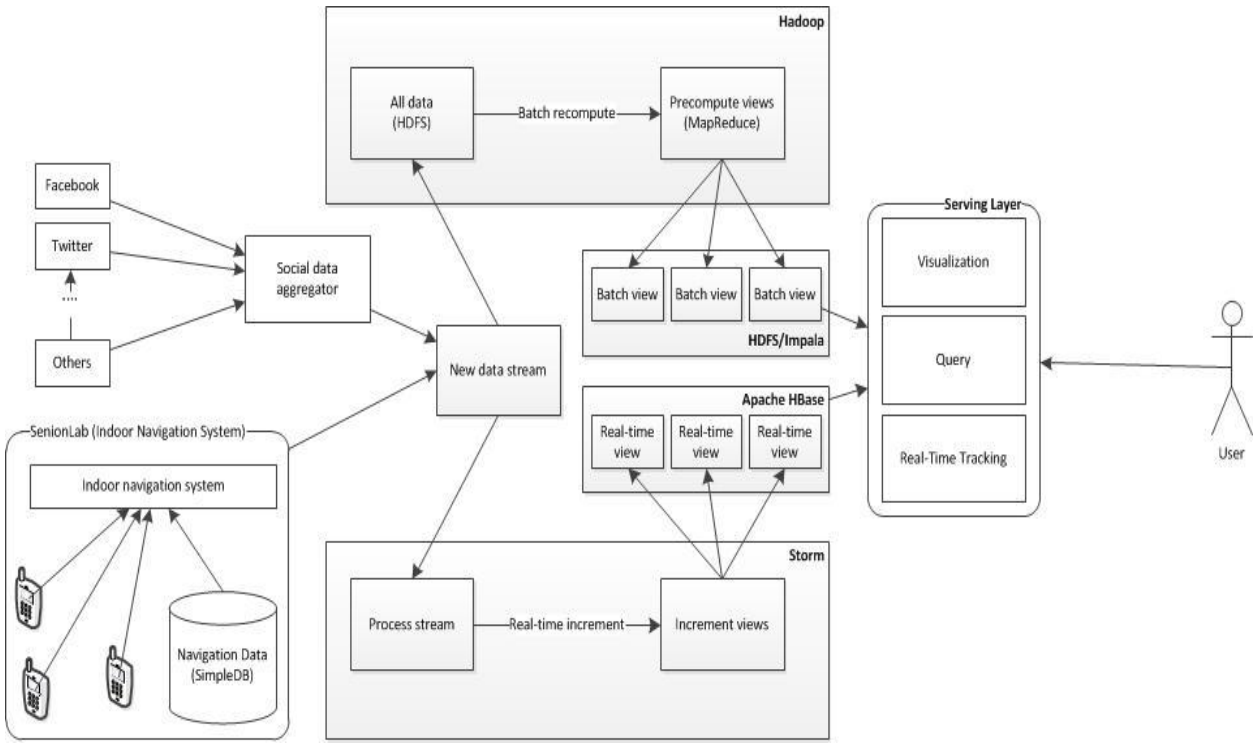

Figure 14: Proposed architecture for SenionLab big data platform

All new data coming from social medias through social data aggregator and navigation data from indoor navigation system is send to both Hadoop (the batch layer) and Storm (the speed layer). In HDFS new data is appended to the master dataset. In Storm the new data is consumed to do incremental updates of the real-time views in HBase.

In Hadoop we will run MapReduce job continuously to pre-compute batch views. The MapReduce job will include sentiment analysis from social data, transforming navigation data into Cartesian coordinate data and compute batch views by month, week and day.

The serving layer indexes the batch views produced by the batch layer and merge it with real-time views produced by Storm. Queries will be resolved by getting results from both batch and real-time views.

4.4 Quick update towards propose Architecture

The existing architecture figure 8 is works satisfactorily. However, to improve the performance and handle more load in future we will need to upgrade the system into propose architecture. It is often expensive and time consuming to upgrade into new system. Hence, we can re-engineer our existing architecture to achieve predictable performance.

Performance boost can be achieved by introducing a caching layer in between application server and database. A caching layer stores date entirely in memory and provides very quick reads and writes of data. There are many different caching servers available in market.

Memcached[113] is an open source, high-performance, distributed memory object caching system. It offers an in-memory key-value store for small chunks of data (string, objects). It also provides a very simple client API to perform data manipulation using most of the popular programming languages.

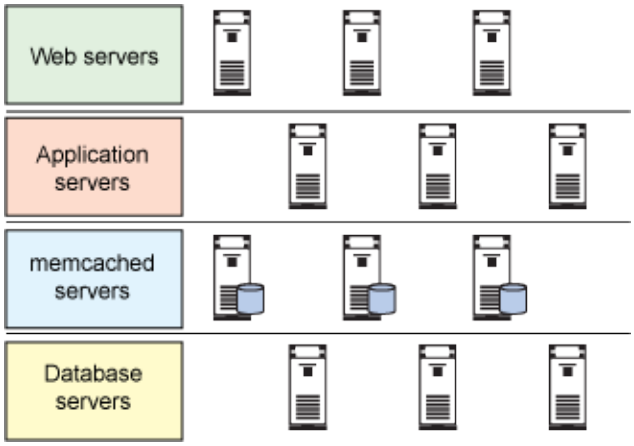

Figure 15: Sample application architecture with memcached[114]

Microsoft AppFabric[115] is also a distributed in-memory cache that runs on one or more servers to provide performance and scalability boost for applications. We can use caching to store application session state.

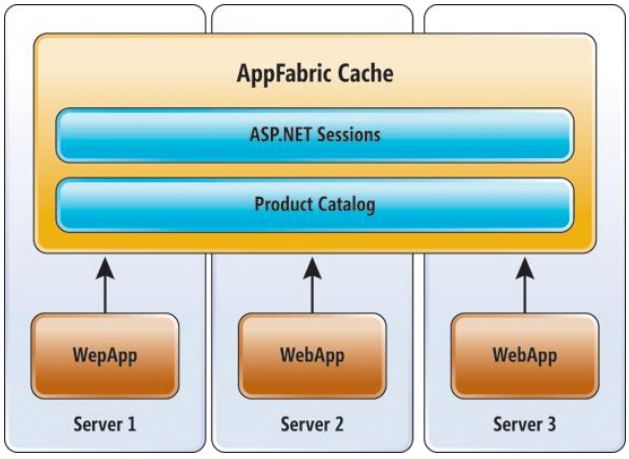

Figure 16: Sample application architecture with AppFabric[116]

Amazon offers a caching service in their AWS service offerings. Amazon ElastiCache[117] is an easy to use which can be use to add caching logic to our application. We can create Cache Clusters which might comprise of one or more Cache Nodes in a matter of minutes. We can access ElastiCache using popular Memcached protocol. Moreover, it is very easy to integrate ElastiCache layer with other Amazon's services.

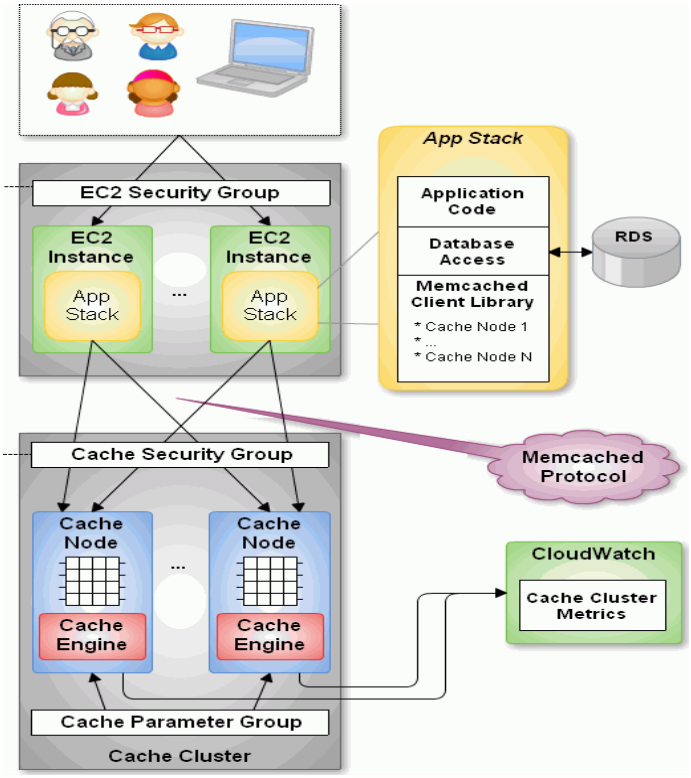

Figure 17: Sample application architecture with[118]

You can use any of above caching technology to improve the performance and scalability of your system. YouTube, Facebook, Zynga, Twitter, Reddit and many others high traffic website is using Memcached to provide high performance. However, AppFabric is closely aligned with .NET framework and makes use of several features such as persistence, monitoring and hosting of WCF and WF service. It also integrates with Internet Information Services (IIS) which provide management and monitoring tools within the IIS management console.

Our recommendation is to use Amazon ElastiCache. The main reason for using ElastiCache is add an in-memory cache to your application architecture in a matter of minutes. AWS Management Console provides a very easy to use dashboard where you can launch a Cache Cluster consisting of a collection of Cache Nodes running Memcached software within few clicks. You can scale up or down depending on the amount of memory you need by adding or deleting Cache Nodes from the cluster.

ElastiCachehas proven record in providing distributed caching service. Many well-known companies are using it every day. Esri[119]use ElastiCache to cache their users data and maps in ArcGIS (Web-based mapping application). Aribnb, BrandVerity, PBS, Tapjoy and TicketLeap are successfully using ElastiCache as distributed cache clusters to manage large number our user request.

CHAPTER 5

5 Conclusion and Future work

This work introduced architectures and concepts for implementing a cloud-based infrastructure for analyzing large volume of semi-structured and unstructured data. Moreover, we have shown how to analyze indoor navigation data and twitter stream and make meaning out of it.

Developing large scale data analysis platform is often quite complex when there is an expectation that the processed data will grow continuously in future. The architecture varies depend on your requirements. If you want to make a data warehouse and analyze data afterwards (batch processing) the best choices will be Hadoop clusters and Pig or Hive. This architecture has been proven in Facebook and Yahoo for years. On the other hand, if real-time data analytics is needed then the architecture becomes more complex. We described a proposed architecture (lambda architecture) in the literature section of the thesis.

Most of the legacy systems were not developed considering big data architectures. It is often expensive to reengineer such kind of system from scratch. We can do some work around to manage large scale and constantly growing data sets. In this literature we also proposed architecture to upgrade our prototype so that we can handle real-time data analysis and constantly growing data volume.

The main limitation of this thesis is that we only analyze indoor navigation system data and tweets from twitter. For time constrain we used third party sentiment analysis and data collection services. We were not able to test our proposed lambda architecture. Therefore, we just propose an upgrade of existing architecture of the system.

In this thesis we provide a bigger picture of big data ecosystem. We tried to provide very brief introduction of each of the tools and technologies that audience or readers need to know before doing farther research on big data analysis or developing product. The technical platform proposed by this thesis is intended to be deployed as platform for further research in big data and its surround ecosystem. There are plenty of scopes to do farther experiment with our proposed architectures.

6 References

[1] N. Criticism, “Data are Everywhere,” Narrative Analysis: Studying the Development of Individuals in Society, p. 63, 2003.

[2] M. Benda, “Data, data everywhere,” IEEE Internet Computing, vol. 1, no. 5, pp. 72–75, 1997.

[3] V. Bernhardt and B. J. Geise, “Data, data everywhere,” Victoria, 2009.

[4] L. D. Nelson, “Data, data everywhere,” Critical care medicine, vol. 25, no. 8, p. 1265, 1997.

[5] M. Shashanka and M. Giering, “Mining Retail Transaction Data for Targeting Customers with Headroom-A Case Study,” Artificial Intelligence Applications and Innovations III, pp. 347–355, 2009.

[6] “Big Data: Beyond the Hype,” DataStax Corporation, Mar. 2012.

[7] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, “Big data: The next frontier for innovation, competition, and productivity,” McKinsey Global Institute, pp. 1–137, 2011.

[8] P. Carter, “Big data analytics: future architectures skills and roadmaps for the cio,” International Data Corporation, Sep. 2011.

[9] “Oracle: Big Data for the Enterprise,” Oracle, Jan. 2012.

[10] P. Russom, “big data analytics,” TDWI Best Practices Report, 4 th Quarter 2011, 2011.

[11] D. V. Keyson and M. Bruns Alonso, “Empirical research through design,” in Proceedings of the 3rd IASDR Conference on Design Research, 2009, pp. 4548–4557.

[12] “Prototype,” Wikipedia, the free encyclopedia. 19-Dec-2012.

[13] A. M. Davis, “Operational prototyping: A new development approach,” Software, IEEE, vol. 9, no. 5, pp. 70–78, 1992.

[14] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, Agile software development methods: Review and analysis. VTT Finland, 2002.

[15] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, and R. Jeffries, “Manifesto for agile software development,” The Agile Alliance, pp. 2002–04, 2001.

[16] J. A. Highsmith, Agile software development ecosystems. Addison-Wesley Professional, 2002.

[17] R. C. Martin, Agile software development: principles, patterns, and practices. Prentice Hall PTR, 2003.

[18] B. J. Oates, Researching information systems and computing. Sage Publications Limited, 2005.

[19] E. Dumbill, Planning for Big Data. O’Reilly Media, Inc., 2012.

[20] O. Team, Big Data Now: Current Perspectives from O’Reilly Radar. O’Reilly Media, Inc., 2011.

[21] J. Han, E. Haihong, G. Le, and J. Du, “Survey on nosql database,” in Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on, 2011, pp. 363–366.

[22] R. Hecht and S. Jablonski, “NoSQL evaluation: A use case oriented survey,” in Cloud and Service Computing (CSC), 2011 International Conference on, 2011, pp. 336–341.

[23] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, “Bigtable: A distributed storage system for structured data,” ACM Transactions on Computer Systems (TOCS), vol. 26, no. 2, p. 4, 2008.

[24] R. Burtica, E. M. Mocanu, M. I. Andreica, and N. Tapus, “Practical application and evaluation of no-SQL databases in Cloud Computing,” in Systems Conference (SysCon), 2012 IEEE International, 2012, pp. 1–6.

[25] S. Tiwari. 2011. Professional NoSQL. John Wiley & Sons.
[26] E. Brewer, “A certain freedom: thoughts on the CAP theorem,” in Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing, 2010, pp. 335–335.
[27] “Visual Guide to NoSQL Systems - Nathan Hurst’s Blog.” [Online]. Available: http://blog.nahurst.com/visual-guide-to-nosql-systems. [Accessed: 27-Dec-2012].
[28] J. Dean and S. Ghemawat, “MapReduce: Simplified data processing on large clusters,” Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.
[29] “NOSQL Databases.” [Online]. Available: http://nosql-database.org/. [Accessed: 27-Dec-2012].
[30] “NoSQL Data Modeling Techniques,” Highly Scalable Blog. [Online]. Available: http://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques/. [Accessed: 27-Dec-2012].
[31] D. Robinson, Amazon Web Services Made Simple: Learn how Amazon EC2, S3, SimpleDB and SQS Web Services enables you to reach business goals faster. Emereo Pty Ltd, 2008.
[32] “Amazon SimpleDB.” [Online]. Available: http://aws.amazon.com/simpledb/. [Accessed: 27-Dec-2012].
[33] J. Varia, “Architecting for the cloud: Best practices,” Amazon Web Services, 2010.
[34] J. Varia, “Cloud architectures,” White Paper of Amazon, jineshvaria. s3. amazonaws. com/public/cloudarchitectures-varia. pdf, 2008.
[35] M. S. Ramanathan, M. S. Goel, and M. S. Alagumalai, “Comparison of Cloud Database: Amazon’s SimpleDB and Google’s Bigtable.”
[36] A. S. D. B. Team, Query 101: Building Amazon SimpleDB Queries. 2008.
[37] A. S. D. B. Team, Query 201: Tips and Tricks for Amazon SimpleDB Query. 2008.
[38] J. G. Baron and R. Schneider, “Storage Options in the AWS Cloud,” Amazon White Papers, 2010.
[39] P. Chaganti and R. Helms, Amazon SimpleDB Developer Guide. Packt Pub Limited, 2010.
[40] M. Habeeb, A Developer’s Guide to Amazon SimpleDB. Addison-Wesley Professional, 2010.
[41] “Amazon DynamoDB.” [Online]. Available: http://aws.amazon.com/dynamodb/. [Accessed: 27-Dec-2012].
[42] R. Yanggratoke, “The Amazon Web Services.”
[43] J. G. Baron and R. Schneider, Storage Options in the AWS Cloud. 2010.
[44] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, “Dynamo: amazon’s highly available key-value store,” in ACM SIGOPS Operating Systems Review, 2007, vol. 41, pp. 205–220.
[45] R. Sumbaly, J. Kreps, L. Gao, A. Feinberg, C. Soman, and S. Shah, “Serving large-scale batch computed data with project voldemort,” in Proceedings of the 10th USENIX conference on File and Storage Technologies, 2012, pp. 18–18.
[46] “Project Voldemort.” [Online]. Available: http://www.project-voldemort.com/voldemort/. [Accessed: 27-Dec-2012].
[47] “Redis.” [Online]. Available: http://redis.io/. [Accessed: 27-Dec-2012].
[48] P. Warden, Big Data Glossary. O’Reilly Media, 2011.
[49] “Redis - Command reference.” [Online]. Available: http://redis.io/commands. [Accessed: 27-Dec-2012].
[50] “Redis - Clients.” [Online]. Available: http://redis.io/clients. [Accessed: 27-Dec-2012].
[51] “MongoDB.” [Online]. Available: http://www.mongodb.org/. [Accessed: 27-Dec-2012].
[52] K. Chodorow and M. Dirolf, MongoDB: the definitive guide. O’Reilly Media, 2010.

[53] P. Membrey, E. Plugge, and T. Hawkins, The definitive guide to MongoDB: the noSQL database for cloud and desktop computing. Apress, 2010.
[54] “Drivers - MongoDB.” [Online]. Available: http://www.mongodb.org/display/DOCS/Drivers. [Accessed: 27-Dec-2012].
[55] “Apache CouchDB.” [Online]. Available: http://couchdb.apache.org/. [Accessed: 28-Dec-2012].
[56] J. C. Anderson, J. Lehnardt, and N. Slater, CouchDB: The Definitive Guide: Time to Relax. O’Reilly Media, 2010.
[57] “CouchDB - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/CouchDB. [Accessed: 28-Dec-2012].
[58] “ACID - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/Atomicity,_consistency,_isolation,_durability. [Accessed: 28-Dec-2012].
[59] “Multiversion concurrency control - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/Multi-Version_Concurrency_Control. [Accessed: 28-Dec-2012].
[60] “Riak - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/Riak. [Accessed: 28-Dec-2012].
[61] “Riak Documents.” [Online]. Available: http://docs.basho.com/index.html. [Accessed: 28-Dec-2012].
[62] “Basho | Our Production Users.” [Online]. Available: http://basho.com/company/production-users/. [Accessed: 28-Dec-2012].
[63] D. Featherston, “Cassandra: Principles and Application,” University of Illinois, 2010.
[64] “The Apache Cassandra Project.” [Online]. Available: http://cassandra.apache.org/. [Accessed: 28-Dec-2012].
[65] “Big Data Management for the Enterprise,” DataStax Corporation, Mar. 2012.
[66] A. Khetrapal and V. Ganesh, “HBase and Hypertable for large scale distributed storage systems,” Dept. of Computer Science, Purdue University, 2006.
[67] “HBase - Apache HBase™ Home.” [Online]. Available: http://hbase.apache.org/. [Accessed: 29-Dec-2012].
[68] “Documentation | Hypertable - Big Data. Big Performance.” [Online]. Available: http://hypertable.com/documentation/. [Accessed: 29-Dec-2012].
[69] “What is a Graph Database - Neo4j: The World’s Leading Graph Database.” [Online]. Available: http://www.neo4j.org/learn/graphdatabase. [Accessed: 29-Dec-2012].
[70] “Start - Neo4j: The World’s Leading Graph Database.” [Online]. Available: http://www.neo4j.org/index?utm_expid=2722339-1&utm_referrer=http%3A%2F%2Fwww.neo4j.org%2Flearn%2Fneo4j. [Accessed: 29-Dec-2012].
[71] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, “Hive: a warehousing solution over a map-reduce framework,” Proceedings of the VLDB Endowment, vol. 2, no. 2, pp. 1626–1629, 2009.
[72] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, and R. Murthy, “Hive-a petabyte scale data warehouse using hadoop,” in Data Engineering (ICDE), 2010 IEEE 26th International Conference on, 2010, pp. 996–1005.
[73] “Karmasphere | Hive Queries and Tables.” [Online]. Available: http://karmasphere.com/hive-queries-on-table-data. [Accessed: 14-Oct-2012].
[74] “Datalog - Wikipedia, the free encyclopedia.” [Online]. Available: http://en.wikipedia.org/wiki/Datalog. [Accessed: 29-Dec-2012].
[75] “nathanmarz/cascalog · GitHub.” [Online]. Available: https://github.com/nathanmarz/cascalog. [Accessed: 29-Dec-2012].

[76] “mrjob — mrjob 0.4-dev documentation.” [Online]. Available: http://mrjob.readthedocs.org/en/latest/. [Accessed: 29-Dec-2012].
[77] “Amazon Elastic MapReduce (Amazon EMR).” [Online]. Available: http://aws.amazon.com/elasticmapreduce/. [Accessed: 29-Dec-2012].
[78] “Scalding (scalding) on Twitter.” [Online]. Available: http://twitter.com/scalding. [Accessed: 29-Dec-2012].
[79] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, “S4: Distributed stream computing platform,” in Data Mining Workshops (ICDMW), 2010 IEEE International Conference on, 2010, pp. 170–177.
[80] “S4: Distributed Stream Computing Platform.” [Online]. Available: http://incubator.apache.org/s4/. [Accessed: 29-Dec-2012].
[81] “Welcome to Apache Flume — Apache Flume.” [Online]. Available: http://flume.apache.org/. [Accessed: 29-Dec-2012].
[82] “Apache Kafka.” [Online]. Available: http://kafka.apache.org/. [Accessed: 29-Dec-2012].
[83] “Storm, distributed and fault-tolerant realtime computation.” [Online]. Available: http://storm-project.net/. [Accessed: 29-Dec-2012].
[84] “HDFS Architecture Guide.” [Online]. Available: http://hadoop.apache.org/docs/hdfs/current/hdfs_design.html. [Accessed: 30-Dec-2012].
[85] “Amazon Simple Storage Service (Amazon S3).” [Online]. Available: http://aws.amazon.com/s3/. [Accessed: 29-Dec-2012].
[86] “Data Management - Features.” [Online]. Available: http://www.windowsazure.com/en-us/home/features/data-management/. [Accessed: 30-Dec-2012].
[87] J. Murty, Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB. O’Reilly Media, Incorporated, 2008.
[88] “What Is Google App Engine? - Google App Engine — Google Developers.” [Online]. Available: https://developers.google.com/appengine/docs/whatisgoogleappengine. [Accessed: 30-Dec-2012].
[89] “Heroku | Cloud Application Platform.” [Online]. Available: http://www.heroku.com/. [Accessed: 30-Dec-2012].
[90] “Windows Azure: Microsoft’s Cloud Platform | Cloud Hosting | Cloud Services.” [Online]. Available: http://www.windowsazure.com/en-us/. [Accessed: 30-Dec-2012].
[91] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis, “Dremel: interactive analysis of web-scale datasets,” Proceedings of the VLDB Endowment, vol. 3, no. 1–2, pp. 330–339, 2010.
[92] “Google BigQuery — Google Developers.” [Online]. Available: https://developers.google.com/bigquery/. [Accessed: 30-Dec-2012].
[93] “DrillProposal.” .
[94] “Apache Lucene - Welcome to Apache Lucene.” [Online]. Available: http://lucene.apache.org/. [Accessed: 30-Dec-2012].
[95] “elasticsearch - - Open Source, Distributed, RESTful, Search Engine.” [Online]. Available: http://www.elasticsearch.org/. [Accessed: 30-Dec-2012].
[96] “Weka 3 - Data Mining with Open Source Machine Learning Software in Java.” [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/. [Accessed: 30-Dec-2012].
[97] “Apache Mahout: Scalable machine learning and data mining.” [Online]. Available: http://mahout.apache.org/. [Accessed: 30-Dec-2012].
[98] “scikit-learn: machine learning in Python — scikit-learn 0.12.1 documentation.” [Online]. Available: http://scikit-learn.org/stable/. [Accessed: 30-Dec-2012].
[99] “JSON.” [Online]. Available: http://www.json.org/. [Accessed: 30-Dec-2012].
[100] “BSON - Binary JSON.” [Online]. Available: http://bsonspec.org/. [Accessed: 30-Dec-2012].

[101] M. Slee, A. Agarwal, and M. Kwiatkowski, “Thrift: Scalable cross-language services implementation,” Facebook, Jan, 2007.
[102] “Apache Thrift.” [Online]. Available: http://thrift.apache.org/. [Accessed: 30-Dec-2012].
[103] “Apache Avro™ 1.7.3 Documentation.” [Online]. Available: http://avro.apache.org/docs/current/. [Accessed: 31-Dec-2012].
[104] “Developer Guide - Protocol Buffers — Google Developers.” [Online]. Available: https://developers.google.com/protocol-buffers/docs/overview. [Accessed: 31-Dec-2012].
[105] A. A. B. X. I. Vovsha and O. R. R. Passonneau, “Sentiment analysis of twitter data,” ACL HLT 2011, p. 30, 2011.
[106] B. Pang and L. Lee, Opinion mining and sentiment analysis. Now Pub, 2008.
[107] H. Saif, Y. He, and H. Alani, “Alleviating Data Sparsity for Twitter Sentiment Analysis,” Making Sense of Microposts (# MSM2012), 2012.
[108] “Systems development life-cycle,” Wikipedia, the free encyclopedia. 07-Feb-2013.
[109] “Limits - Amazon SimpleDB.” [Online]. Available: http://docs.aws.amazon.com/AmazonSimpleDB/latest/DeveloperGuide/SDBLimits.html. [Accessed: 14-Feb-2013].
[110] N. Marz and J. Warren, Big Data Principles and best practices of scalable realtime data systems. MEAP.
[111] “tumblr_menrfnvuvW1rab9oo.png (PNG Image, 500 × 340 pixels).” [Online]. Available: http://media.tumblr.com/tumblr_menrfnvuvW1rab9oo.png. [Accessed: 11-Feb-2013].
[112] “What is delta differencing (delta differential)? - Definition from WhatIs.com.” [Online]. Available: http://searchstorage.techtarget.com/definition/delta-differencing. [Accessed: 11-Feb-2013].
[113] “memcached - a distributed memory object caching system.” [Online]. Available: http://memcached.org/. [Accessed: 23-Feb-2013].
[114] “memcached and Grails, Part 1: Installing and using memcached.” [Online]. Available: http://www.ibm.com/developerworks/web/library/j-memcached1/index.html. [Accessed: 23-Feb-2013].
[115] “AppFabric for Windows Server.” [Online]. Available: http://msdn.microsoft.com/en-us/windowsserver/ee695849.aspx. [Accessed: 23-Feb-2013].
[116] “MSDN Magazine: Windows Azure Cache - Real-world Usage and Integration.” [Online]. Available: http://msdn.microsoft.com/en-us/magazine/ff714581.aspx. [Accessed: 23-Feb-2013].
[117] “Amazon ElastiCache.” [Online]. Available: http://aws.amazon.com/elasticache/#functionality. [Accessed: 23-Feb-2013].
[118] “elasticache_4.png (PNG Image, 517 × 766 pixels) - Scaled (54%).” [Online]. Available: http://cdn77.vbtechsupport.com/wp-content/uploads/2011/08/elasticache_4.png. [Accessed: 23-Feb-2013].
[119] “ArcGIS - Mapping and Spatial Analysis for Understanding Our World.” [Online]. Available: http://www.esri.com/software/arcgis. [Accessed: 23-Feb-2013].
[120] “MapReduce for word count process.” [Online]. Available: “http://blog.jteam.nl/2009/08/04/introduction-to-hadoop/. [Accessed: 23-Feb-2013].
[121] “The Hadoop Ecosystem, Visualized in Datameer.” [Online]. Available: “http://datameer2.datameer.com/blog/wp-content/uploads/2012/06/Hadoop-Ecosystem-Infographic-21.png. [Accessed: 23-Feb-2013].