
# A guide in the Big Data jungle

**Thesis, Bachelor of Science**

**Anna Ohlsson, Dan Öman**

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

**Contact Information:**
Author(s):
Anna Ohlsson,
E-mail: annaohlsson@live.se,

Dan Öman
E-mail: dan.oeman@gmail.com

University advisor:
Nina Dzamashvili Fogelström
Department of Software Engineering

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

Internet : www.bth.se/com
Phone : +46 0455 38 50 00
Fax : +46 0455 38 50 57

# Abstract

This bachelor thesis looks at the functionality of different frameworks for data analysis at large scale and the purpose of it is to serve as a guide among available tools. The amount of data that is generated every day keep growing and for companies to take advantage of the data they collect they need to know how to analyze it to gain maximal use out of it. The choice of platform for this analysis plays an important role and you need to look in to the functionality of the different alternatives that are available. We have created a guide to make this research easier and less time consuming. To evaluate our work we created a summary and a survey which we asked a number of IT-students, current and previous, to take part in. After analyzing their answers we could see that most of them find our thesis interesting and informative.

# Content

# 1 Introduction

"More data cross the internet every second then were stored in the entire internet just 20 years ago." (McAfee. A, Brynjolfsson. E, 2012)[33]

Networking, data storage and data collection capacity is rapidly developing and this leads to an expansion of Big Data in all science and engineering domains. The acceleration in growth of the amount of data that is stored partly comes from the digitization of how we interact, both with each other and with institutions that are part of our lives. Today, most of us use services/programs such as Facebook and Google that needs to handle a lot of data because of the increasingly growing base of users and information. These companies are storing a lot of information every day about us and our behavior for analyzing and stacking. These are just some examples of services that handle huge amounts of data, called Big Data. The amounts of data is so big that it can't be analyzed on one single machine in a reasonable time, which is why platforms for distributed and parallel processing are on a rise.

There is a lot to gain from correct data analytics; you can predict market trends, customer preferences and other useful business information. This could then lead to a boost in sales, increased efficiency, improved operations, better customer service and smarter management of risks. "Today's advances in analyzing Big Data allow researchers to decode human DNA in minutes, predict where terrorists plan to attack, determine which gene is most likely to be responsible for certain diseases and, of course, which ads you are most likely to respond to on Facebook. The business cases for leveraging Big Data are compelling." (Beal. V, 2014)[32]

To be able to gain as much relevant information as possible from these massive amounts of data you need to know what platforms to use and when to use them. It is hard and very time consuming to read huge texts from different resources today in order to hopefully get enough knowledge on what platform you should use for your purpose. The aim of this thesis is to map out the available platforms and their functionality to ease the research.

We believe that our "guide in the Big Data jungle" will come of use for people that want to or need to address the problems regarding the subject, both beginners and experienced developers. By the end of this thesis you will have a good overview of available tools and platforms that you can use in your data analysis.

During our research we have focused on software for distributed and parallel analysis and not included different types of hardware that are suited for analysis of large amounts of data.

## 1.1 Overview

The "Background and related work" section outlines concepts and terms regarding analytics of Big Data as well as explaining why there might be issues when analyzing Big Data and what to think about to make it right. You can also find descriptions of the platforms and frameworks that we have examined throughout this thesis.

The “Method” section defines our research questions and how we answered these. It also outlines our research methodology used to conduct our research, which is a survey and a literature study.

The “Results” we present the findings and conclusions we made about the platforms during the literature study and research. In this part you can find when a specific platform is appropriate to use.

In the “Analysis” section we discuss our findings and results from the literature study and our survey, you can also find an overview of the platforms strengths and weaknesses here.

The “Conclusion” section presents the main conclusion of our research. We point out how our work contributes to the world of Big Data and discuss what work can be done in the area in the future.

# 2 Background and related work

## 2.1 Background

The expansion of data's variety, velocity and volume often create problems. Today SVM's (Support Vector Machines)[1] are being used when data is being analyzed and pattern needs to be recognized. There could be problems with their working speed, they have a limit in how fast they can work. Is parallel the way to go? (Quora, 2015)[1]

Yes, parallel is the way to go!

Getting the right information out of the data that you have collected is a common problem, if you don't know the structure or complexity of the data it is hard to make a correct analysis of it. To make a correct analysis you need to know what you are looking for, what data is meaningful for this analysis, what tools you should use and so on.[50]

Another problem is the lack of skills regarding Big Data as it is a concept that has become more common and used in the last decade. There is an extensive collection of platforms to be considered in analytics of Big Data and it isn't always easy to know what to use for a specific type of data or algorithm. We have looked into the different platforms and tools that are available and described their functionality, strengths and weaknesses.

*Figure 1* shows an overview of the platforms that we chose to examine closer in this thesis. It includes the Big Data frameworks that we found most popular and tested after conducting our literature review. Since Hadoop is a big part of the Big Data world we chose to look at 15 of its available subprojects. It is also showing the biggest names providing wrappers for MapReduce (page 10) on the market. Types of database structures are another important part of Big Data which we have listed in *Figure 1* together with eight well known database deliverers.

[1] Supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis.

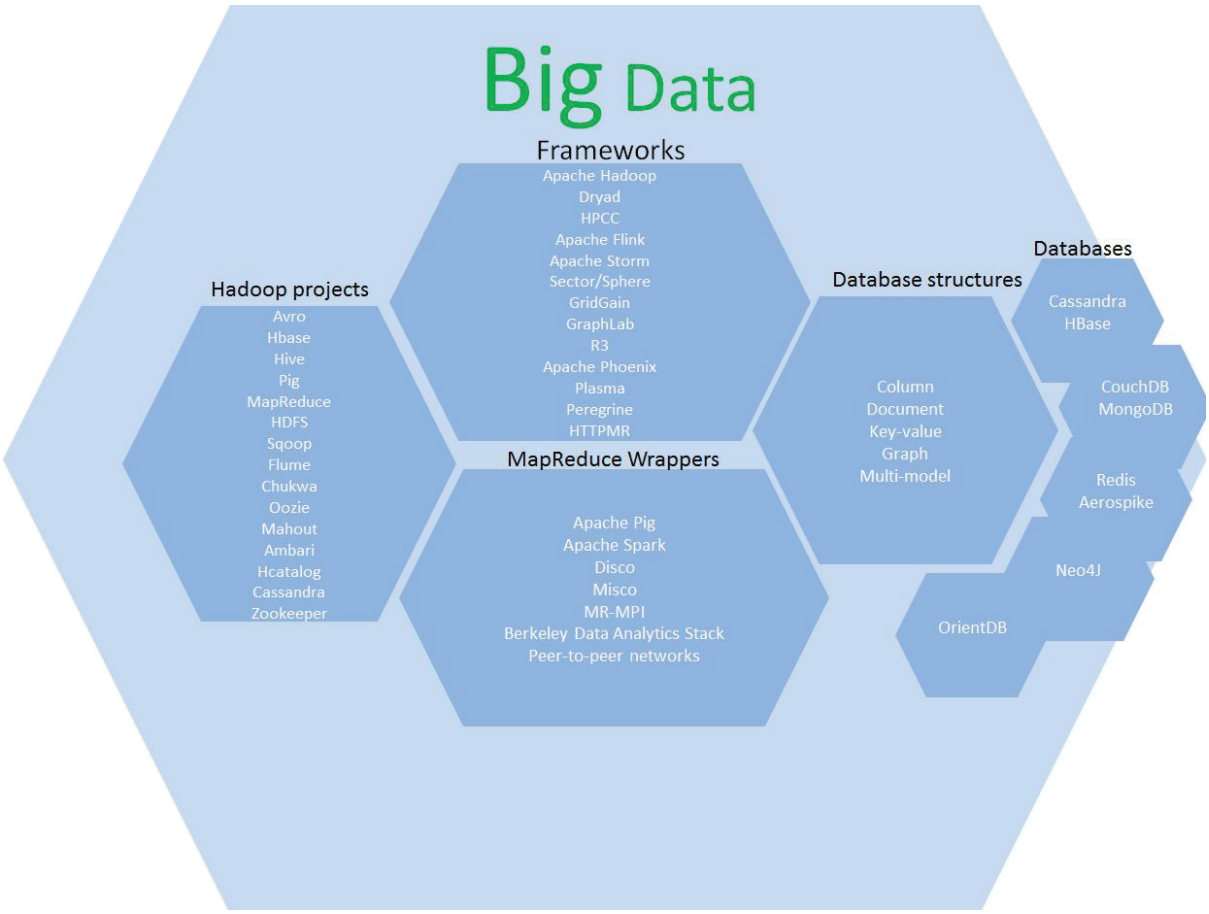



*Figure 1 Overview of Platforms and Frameworks*

### 2.1.1 What is Big Data?

It is the collection of massive amounts of information, whether unstructured or structured. (Levine, 2015)[3]

Companies around the world collects data which holds results of discoveries and analysis, this is as mentioned above the collection of information in massive amounts. It is the size, diversity and the complexity that make Big Data. These characteristics make it hard to process to extract value from it in a traditional way. It is usually described as high in:[106]

- Volume - The size of the data. The large size can come from years of storing or a constant data stream from social media, for example. The size of this data is not Terabytes (factor $10^{12}$) or Petabyte(factor $10^{15}$) but Zettabytes (factor $10^{21}$). This volume calls for scalable storage and a distributed approach to processing combined together.[106]
- Variety - The data can be in many different formats which can cause difficulties when merging or managing it. Structured data in tables or relational databases is something that belongs to the past. The unstructured data amount is growing, video's and photos can be seen everywhere. A common use of big data processing is to take unstructured data and extract ordered meaning.[106]

- Velocity - The speed of generation of data, how fast it is growing. As the data grows the analysis result will change, because of that it is important to be able to handle and analyze the data right away. You wouldn't decide to cross a road from looking at a five minute old snapshot of the traffic situation. The importance of the speed for taking data from input through to decision making is huge. Big Data solutions gives the opportunity to analyze data while it is being generated.[106]
- Variability - The inconsistency in the data. For example, there might be periodic peaks causing this. The meaning of the data can vary due to different context.[106]
- Veracity - The quality of the data might vary for different reasons. You need to know if the data that are being stored and mined meaningful for the problem being analyzed.[106]
- Complexity - The data can come from different sources and be hard to link, match, cleanse and transform across systems.[106]

Data can be:

- **Structured**, if it resides in fixed fields within a record or file.[115]
- **Unstructured,** if it is hard to interpret or unorganized.[116]
- **Multi-structured,** if it includes different data formats and types that are information collected from interactions between people and machines. Twitter and Facebook are example of this.[117]

### 2.1.2 Platforms and frameworks

We have divided this big subject into five main parts; Hadoop projects, frameworks, MapReduce wrappers, database structures and databases, because we found these parts most relevant and crucial to consider after our literature study. In this section we describe the different available platforms and frameworks for distributed and parallel analysis of Big Data.

We have color coded the platforms to developer according to *figure 2.*


*Figure 2 Legend*

**Apache Software Foundation** is an organization that aims to provide software for the public good by providing services and support for many like-minded software project communities of individuals. Apache has a lot of projects in our focus area, some of which are described in this thesis.

**MapReduce** is a highly scalable programming paradigm developed by Google, which breaks a computation into small tasks that run in parallel on multiple machines, and scales easily to very large clusters of inexpensive commodity computers. The framework consists of two functions; Map and Reduce, which are written by the user to process the data in key/value pairs. The input data is stored in partitions on every node in a cluster through a distributed file system.

#### 2.1.2.1 MapReduce wrappers

These have been created with the purpose of giving the programmer better control over the MapReduce code and contribute to the source code. There are a bunch of wrappers out there today, including:[37]

**Apache Pig** is an open source program that runs on top of Hadoop (page 12) and analyzes large data sets. Pig has a compiler that produces sequences of MapReduce programs. According to Apache, Pigs key properties are ease of programming, optimization opportunities and extensibility. Pig is compatible with Windows and Linux that have a working version of Java. User defined functions can be written in a selection of programming languages including Java, Python and Ruby. Pig uses its own language called Pig Latin which is similar to SQL and works as an abstraction of MapReduce programming in high level.[105]

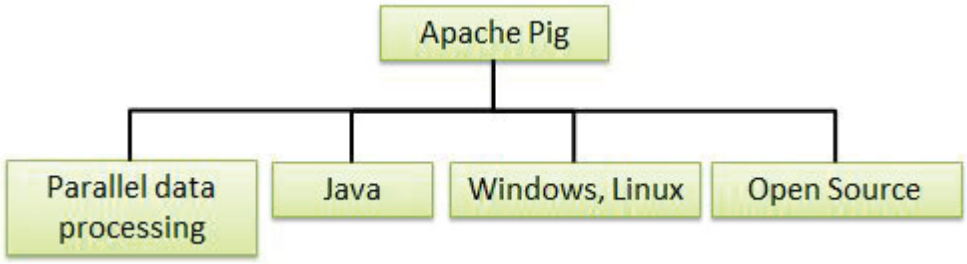



*Figure 3 Apache Pig*

**Apache Spark** is a very fast open source in-memory data-processing and cluster computing framework. Spark uses RDDs[2] which is a collection of objects that are only readable. Spark powers tools for streaming, querying, machine learning and graph computation.[118]

Some of Spark's main features are its speed, ease of use, generality and that it runs everywhere. The speed of Spark in high velocity analytics comes from the ability to stream-process large volumes of data. The framework requires a cluster manager and a distributed storage system. Available tools for cluster management are Hadoop YARN[3] or Apache Mesos[4]. For distributed storage Hadoop Distributed File System (page 13), Cassandra (page 19), OpenStack Swift[5] and Amazon S3[6] is available. Spark also has a

[2] Resilient distributed datasets is a collection of objects that you only can read. Because it is partitioned over the machines that are a part of the cluster it works really fast and is easy to rebuild if a partition is lost.
[3] YARN is a resource manager responsible for managing computing resources in clusters, then use them for user application scheduling
[4] Apache Mesos is a cluster manager that provides resource isolation and sharing across distributed applications
[5] OpenStack Swift is a object storage system
[6] Amazon S3 is an online file storage web service

local-mode which is pseudo-distributed[7] and is supposed to be used for development and testing. Spark can run on Windows, Linux and Mac OS.[118]

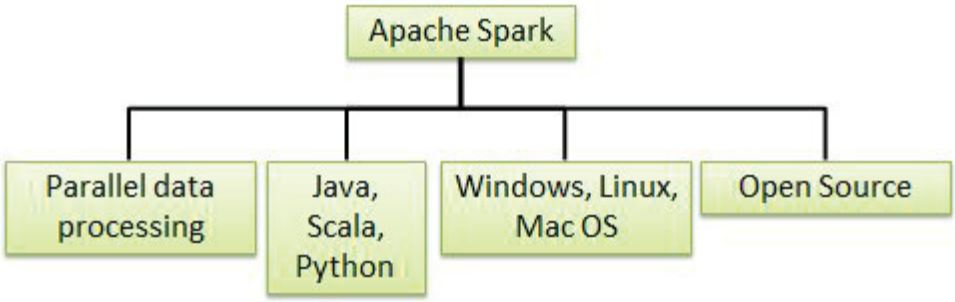



*Figure 4 Apache Spark*

**Disco** is an open source lightweight scalable framework for distributed computing based on MapReduce, developed by Nokia Research Center. It is designed for operation in clusters of commodity server machines. It has the ability of fault-tolerant scheduling and it has an execution layer. Disco also has a distributed and replicated storage layer called Disco Distributed File System (DDFS). Disco describes itself as powerful and easy, thanks to Python.[119] Supported operating systems are Linux, Mac OS X and FreeBSD.[91]

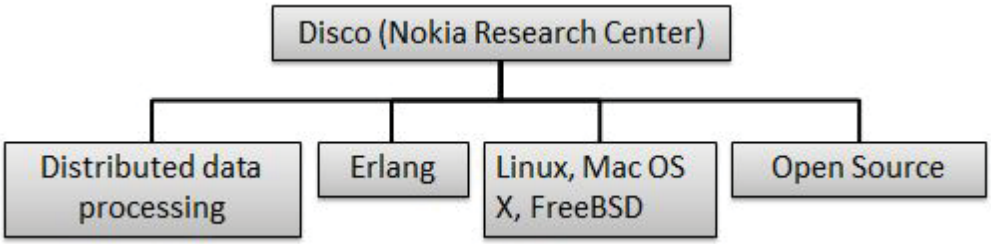



*Figure 5 Disco*

**Misco** is a distributed computing framework based on MapReduce, developed by Nokia Research Center and it is targeted at any mobile devices that support Python and network connectivity. The platform supports distributed applications and the user can create and manage applications through a browser interface. It has a built in job scheduler and fault-tolerance.[91]

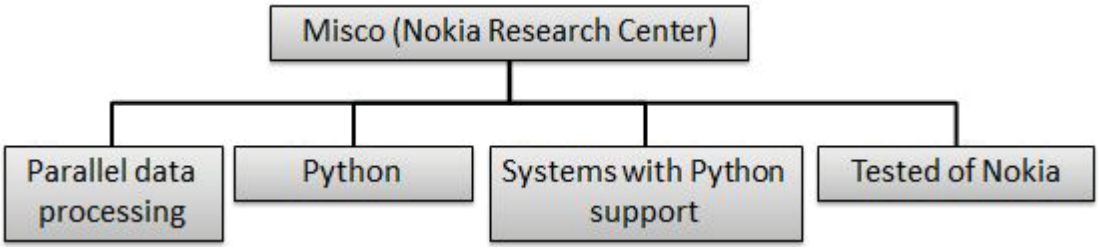



*Figure 6 Misco*

[7] The same thing as a cluster of one. All daemons run as separate processes.

**MR-MP**I (MapReduce-MPI) is an open source implementation of MapReduce written for distributed-memory parallel machines on top of standard MPI[8] message passing. It is developed by Sandia national, has an open source C++ library and it is callable from most high level languages. It uses a scripting language wrapper called OINK which is provided for easy scripting and development. MR-MPI works on any operating system which supports C++, C or Python.[91]

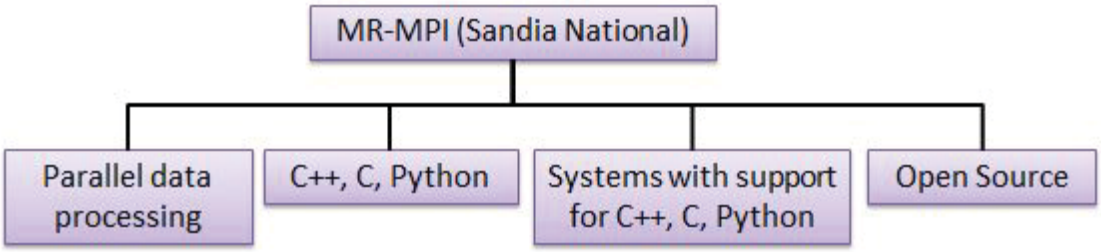



*Figure 7 MR-MPI*

#### 2.1.2.2 Big Data frameworks/projects

**Apache Hadoop** is a well known and common open source framework for distributed storage and distributed processing, and it uses message passing. It has built up a strong base from related subprojects that are all hosted by the Apache Software Foundation. Apache is providing support for a community of open source software projects, an extensive list of Hadoop projects follows ahead.[58][59] Hadoop is written in Java and compatible with GNU/Linux and windows with Java and ssh installed.[120]

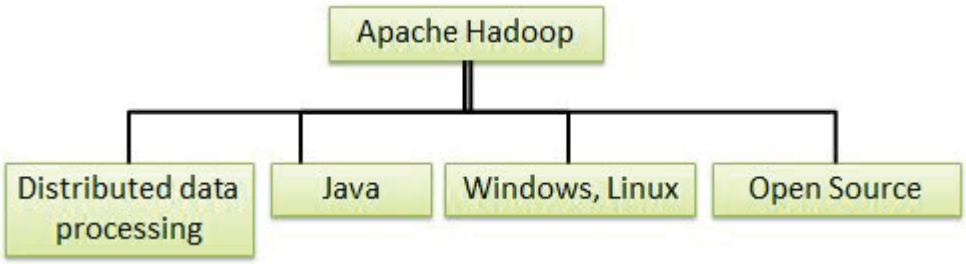



*Figure 8 Apache Hadoop*

- **Avro** is a system that performs data serialization, in other words it translates data structures or object state into a format that can be stored and constructed again in that computer environment, or in a different one.[72] Avro file format are including schema information data files.[73]

- **HBase** explained later under **Databases** on page 19.

- **Hive** is an open source data warehouse software that facilitates querying executed via MapReduce and managing of large data sets residing in distributed storage.[121] Hive uses HDFS to store the data in chunks. The query language that Hive uses is an SQL-like language called HiveQL. It also provides indexing, including bitmap indexes

[8] A standardized and portable message-passing system designed by a group of researchers from academia and industry to function on a wide variety of parallel computers.

to accelerate queries. The system uses different storage types such as plain text and HBase. Hive supports Windows, Linux and Mac OS and requires a working version of Java and Hadoop.[102]

- **Pig** is explained earlier in the thesis on page 10.

- **MapReduce**, as mentioned on page 10, is a big part of Hadoop. The Hadoop version of MapReduce has the same features as Google's original version.

- **HDFS** stands for Hadoop Distributed File System and its main purpose is to store huge files across machines connected in a large cluster. It is developed in such a way that it stores every file as a sequence of blocks, and blocks belonging to a file are replicated for fault tolerance.[95]

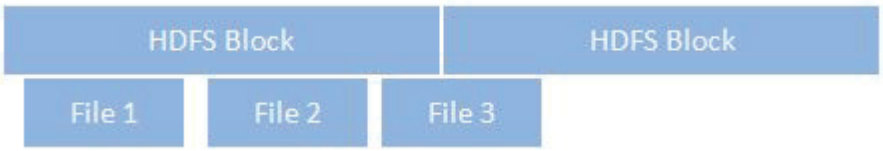



*Figure 9 HDFS*

Block size and replication factor are configurable per file. HDFS follows Master-slave structure (page 20). HDFS have one Namenode[9] but several Datanodes[10] - one per node in a cluster. Together they perform operations like opening, closing, renaming and serves read and write requests from file system clients.[95]

- **Sqoop** transfers data between relational databases and Hadoop Distributed File System or resembled system like HBase and Hive. It organizes data into tables of rows and columns that has a unique key for each row. Sqoop is working together with Oozie, described on the next page, which allows you to handle scheduling and automation of import and export tasks. The structure is connector based, meaning you can connect new external systems.[77]

- **Flume** was created because the developers wanted to collect Big Data logs and being able to flow data from a source into a Hadoop environment. The entities in Flume are called sources, decorators and sinks. Source can be any data source, sink is the target of a specific operation and a decorator is an operation on the stream that can transform it in some manner.[49]

- **Chukwa** is as Flume, devoted to Big Data log collection, but also focused on analysis. It is built on top of Hadoop's HDFS and MapReduce frameworks and take use of Hadoop's scaling ability.[81] Chukwa consists mainly of four components. Agents is one of these components and they are existing on each machine in a cluster with chukwa and broadcasts data. Collectors are there to collect the data from agents and write it to storage. The third component is MapReduce jobs, their purpose

[9] Server that manages the namespace of the file system and controls the file access
[10] Manage the storage of specific node

is to parse and archive data. HICC(Hadoop Infrastructure Care Center) is the last component which is a interface for displaying data.[81]

- **Oozie** schedules workflows and manages Hadoop jobs. Jobs could be MapReduce, Pig and Hive. These jobs have dependencies in order to achieve their goals, the dependencies are expressed as a Directed Acyclic Graph called workflow. It is based on XML in order for the graph base to specify sequences of actions execution.[48]

- **Mahout** is a collection of machine learning libraries that has a main purpose of being scalable and robust, handling clustering with ease. It produces free implementations of distributed or otherwise scalable machine learning algorithms.[87] If you want to cluster documents/data you usually need to consider a step which includes removal of stop words, stop words are words that are removed before or after the process of natural language, but Mahout has a built-in function for this.[87]

- **HCatalog** works as a management layer for Hadoop, it manages tables and storage. They have a table of abstraction that shows a relational view of data in HDFS. It gives you the opportunity of viewing all types of data in Hadoop clusters. It also allows diverse tools, including Pig (page 10) and Hive (page 12), to process any type of data elements without being obliged to know where it is physically stored in the cluster. HCatalog is built on top of Hive and is merged with Hive's Data Definition Language.[96]

- **Cassandra** explained later under **Databases** on page 19.

- **ZooKeeper** is a service that aims to maintain information about configuration, as well as a synchronization service. It started as a Hadoop subproject but has grown to stand on its own. The architecture uses clients and servers, the server part is called ensemble and has a "main-server" called leader and the rest of the ZooKeepers are called followers. By using this architecture they provide a replicated service for users and in order to have this continuously up and running it requires that most of the servers haven't crashed. The best thing about this though is that a server can rejoin after it has recovered from the crash, if the leader crashes a new leader is elected by the other servers. Znodes is ZooKeepers data registers which can be compared to files and directories. The difference between a typical file system and the one that ZooKeeper uses is that it keeps the data in-memory.[60]

**Dryad** is a general-purpose distributed execution engine for coarse-grain data-parallel applications. It uses short running processes communicating via pipes, disk or shared memory between cores and are implemented by messaging. Dryad gives the developer the control to specify an arbitrary acyclic graph to describe the application's communication patterns, and express the data transport mechanisms between the computation vertices. This control allows the programmer to optimize tradeoffs between parallelism and data distribution overheads. Dryad is flexible regarding the size of a cluster; it works with everything from multi-core single computers to data centers with thousands of computers. Fault tolerance, job monitoring and visualization, scheduling and delivering of data to where it is needed is all handled by the Dryad execution engine. Dryad provides a data-flow

programming model suited for bulk-processing of on-disk data but that doesn't make it a natural fit for in-memory computations as it doesn't expose any globally shared state and application often have to emulate shared memory access by joining multiple data streams. Even so, the flexibility of Dryad makes it usable in many different situations, it can handle database-style queries and iterative algorithms such as graph traversal.[19][57] Dryad supports a range of programming languages including C++, C# and Perl.[122] Dryad is written in C# and Windows is the only supported operating system.[91]

Dryad was a project at Microsoft Research but in 2011 they discontinued active development of Dryad to focus on implementing Hadoop.[122]

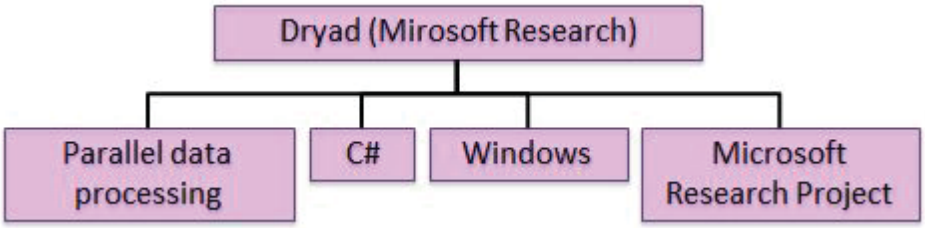



*Figure 10 Dryad*

**HPCC (High-Performance Computing Cluster) Systems** is an open source, data-intensive computing system platform which provides a distributed file system, job execution environment, online query capability, parallel application processing, and parallel programming development tools in a high-performance computing environment. It is developed by LexisNexis Risk Solutions and uses commodity clusters of Unix machines and message passing. Supported operating system is Linux and primary programming language is ECL.[91]

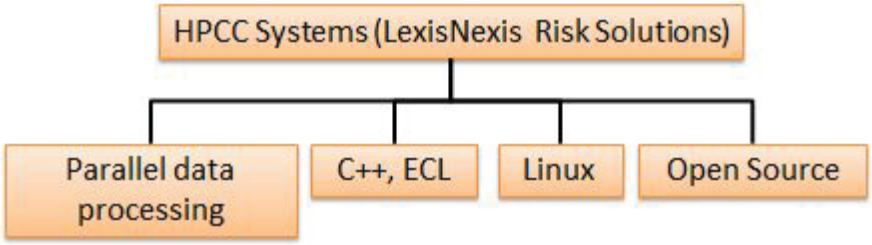



*Figure 11 HPCC Systems*

**Apache Storm** is an open source low latency stream processing framework which provides real-time analysis and continuous computing. Storm relies on ZooKeeper (page 14) to handle the coordination of the nodes in the cluster; it balances the workers and handles failed nodes automatically. Storm runs on topologies that are static and defined upfront regarding communication in the framework.[71] Storm is Java based and Windows, Linux and Mac OS are supported operating systems.[91]

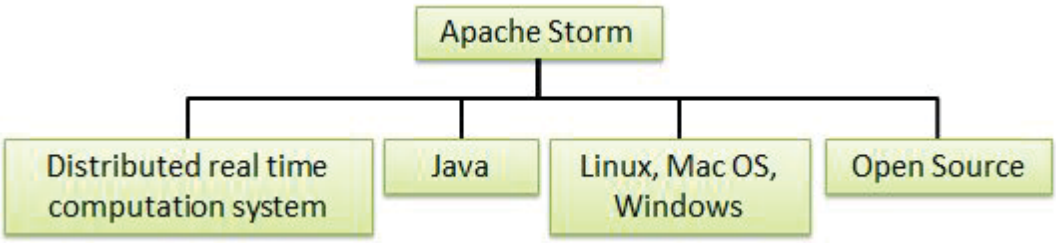



*Figure 12 Apache Storm*

**Sector/Sphere** consists of two parts; Sector is a scalable, fault tolerant and secure distributed file system with high performance using commodity computers. Sphere is a parallel in-storage data processing engine that can be used to process the data managed by Sector in parallel.[91][107[40] The open source framework handles scheduling and load balancing. Sphere applies arbitrary user-defined functions (UDF) which makes it flexible, and it is designed for data locality. Sphere supports multiple inputs and outputs and can combine multiple UDFs in different ways. Sector can be deployed over wide area networks. Sector/Sphere provide support for data intensive applications, including iterative processing which is straightforward to implement.[41][40] Sector/Sphere is based on C++ and supports Linux.[91]

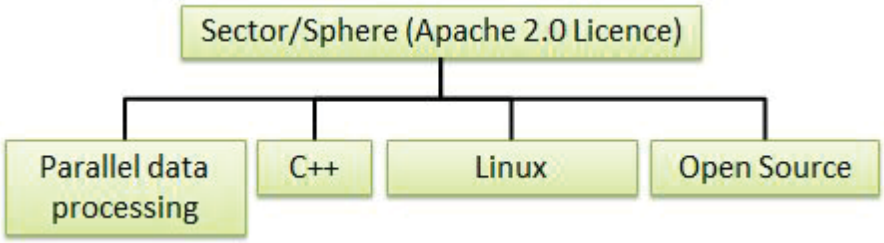



*Figure 13 Sector/Sphere*

**GridGain** is an open source middleware for development of data processing applications in distributed environments. The framework have two parts; compute grid, which provides in-memory MapReduce implementation, and in-memory data grid, which presents the capability to parallelize the data storage by storing the partitioned data in memory, closer to the application. It supports development of applications that are scalable, data-intensive and high performance. GridGain is Java based, independent from infrastructure and platform and supports Java, Scala and Groovy programming languages.[22] GridGain can run on any operating system that supports Python and its networking libraries.[91]

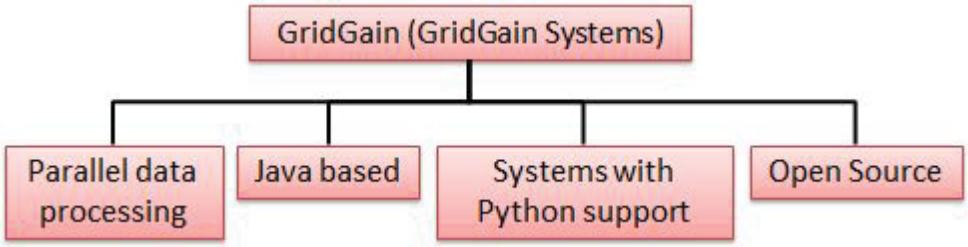



*Figure 14 GridGain*

**GraphLab Create** is an open source asynchronous parallel framework designed for large-scale graph mining. It is an innovative and flexible programming model especially fitted for graph algorithms. GraphLab improves upon abstractions like MapReduce by compactly expressing asynchronous iterative computations with sparse computational dependencies while ensuring data consistency and achieving a high degree of parallel performance. [61][62] GraphLab is written in C++ and can run on Linux and Mac OS.[91]

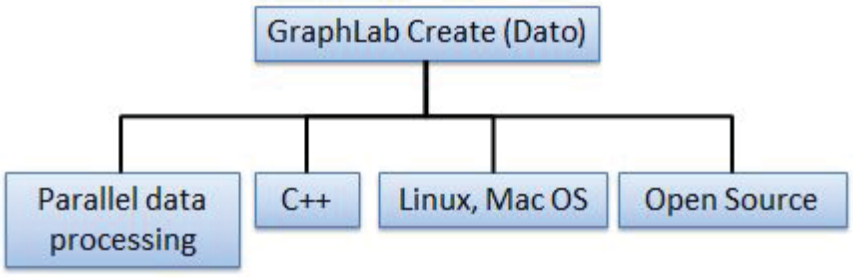



*Figure 15 GraphLab Create*

Following are six platforms which haven't been examined to the same extent as the ones already mentioned in this paper. We chose to give a short description of them but they will not be a part of the analysis or the result that we concluded. We have described how we performed our selection, seen under: *4.2 Literature focus.*

**Apache Flink** is an open source streaming dataflow engine which provides data distribution, communication and fault tolerance for distributed computations over data streams. It also provides libraries for machine learning and graph processing. Flink exploits in-memory data streaming and integrates iterative processing deeply into the system runtime, and it is compatible with Hadoop. Flink runs on all UNIX-like environments with Java installed.[8] Flink was developed from Stratosphere.

**R3** is a simple, Python based MapReduce engine using a Redis (page 20) backend. R3 has three concepts to grasp; input streams, mappers and reducers. Its purpose is to be simple and fast. R3 can run on any machine that supports Python and its networking libraries. [91]

**Apache Phoenix** is a Java based, open source, massively parallel relational database layer on top of NoSQL stores such as HBase. It is a query engine that translates SQL queries into native HBase (page 19) API calls. It is a high performance, horizontally scalable data store engine for Big Data.[6]

**Plasma** is a graph based abstraction and associated framework for distributed computing. It contains a distributed file system, key/value database and a MapReduce system. Plasma allows for concise implementations of distributed algorithms, which are implemented by issuing graph queries.[7]

The **Peregrine** framework is designed to run iterative jobs across partitions of data which supports a number of optimizations and features for MapReduce that other MapReduce frameworks are missing.[91]

**HTTPMR** is an implementation of MapReduce on a cluster of HTTP servers.[91]

Following is a table with an overview of the frameworks just described.

| Framework | Availability | Processing | Platform | OS |
|---|---|---|---|---|
| **Pig** | Open Source | Parallel data processing | Java | Windows, Linux |
| **Spark** | Open Source | Parallel data processing | Java, Scala, Python | Windows, Linux, Mac OS |
| **Disco** | Open Source | Distributed data processing | Erlang | Linux, Mac OS X, FreBSD |
| **Misco** | Tested of Nokia | Parallel data processing | Python | Systems with support for Python |
| **MR-MPI** | Open Source | Parallel data processing | C++, C, Python | Systems with support for C++, C, Python |
| **Hadoop** | Open Source | Distributed data processing | Java | Windows, Linux |
| **Dryad** | Microsoft Research Project | Parallel data processing | C# | Windows |
| **HPCC** | Open Source | Parallel data processing | C++, ECL | Linux |
| **Storm** | Open Source | Distributed real time computation system | Java | Windows, Linux, Mac OS |
| **Sector/Sphere** | Open Source | Parallel data processing | C++ | Linux |
| **GridGain** | Open Source | Parallel data processing | Java based | Systems with support for Python |
| **GraphLab** | Open Source | Parallel data processing | C++ | Linux, Mac OS |

*Table 1 Overview of frameworks*

### 2.1.2.3 Databases/Storage

**NoSQL Databases**

These databases provide users/developers with a mechanism for storage and the data retrieval will follow a different model compared to relational databases.[11] One big difference between NoSQL databases and relational databases is that NoSQL uses dynamic schemas, relational databases requires that schemas are being defined before a user can add data.

NoSQL databases can be divided into categories because of the various approaches they have been classified by. The categories are Column, Document, Key-value, Graph and Multi-model. Now a list of NoSQL databases will follow, divided into the different categories.

**Column** store databases store data tables as sections of columns of data instead of storing the data in rows (Rudra, 2012).[53]

[11] A digital database whose organization is based on the relational model of data.

- **Cassandra** is an Apache open source distributed database management system. It is designed to handle large amounts of data across many servers, to be highly scalable, eventually consistent and being a structured key-value store. All nodes in Cassandra are similar and querying is done by key or key-range. Cassandra uses protocols called CQL3 and Thrift. We among others compare CQL3 and SQL, they are very much alike, but CQL3 does have some limitations regarding the scalability. One of these limitations is that it is without any join or aggregate functions. Nowadays Cassandra is all about CQL3, it is the official interface of Cassandra. Cassandra is written in Java and supported operating systems includes Windows, Linux/Unix and Mac OS X.[20] The data formats includes structured, semi-structured and unstructured, so actually every format there is.

- **HBase** is an open source cross-platform[12] optimized database model written in Java, that you apply on top of Hadoop HDFS and it is based on Google's BigTable.[123] It is ideal to use when you require real-time read/write random access to very large datasets. It is also ideal to use when you want a MapReduce program to operate on the data being stored. HBase uses HTTP/REST protocol, sometimes it can be used with Thrift as well. It queries by push down via server side scan and get filters and HBase is Jruby-based(JIRB) shell.[74][124] HBase is optimized for multi-structured data.

**Document** based databases are designed for storing, managing and retrieving document-oriented information.

- **CouchDB** saves the data in JSON-objects in a document, which fits perfect for log data. It can be effective to use when you don't have any demands on clustering computations. CouchDB is written in Erlang and is cross-platform supported. It is designed for high traffic to let you work with huge data volumes, but this high traffic can reduce its processing speed because of shortage in functionalities. CouchDB uses HTTP/REST protocols and it is recommended to use Ruby on Rails or Django as language frameworks.

- **MongoDB,** like CouchDB, is a document oriented database. The documents are built in JSON-object-like structure with dynamic schemas called BSON. The write performance of MongoDB is high, but memory usage problems can occur with too much writes being done. It is written in C++, JavaScript and C, cross-platform supported. MongoDB is good to use when you need high availability in an environment that is unstable.[127] If the unstable environment results in node failures, MongoDB is there to automatically recover it instantly and safely. MongoDB uses a structure we call Master-Slave.

[12] A software that runs identically on different platforms, it is said to be platform independent

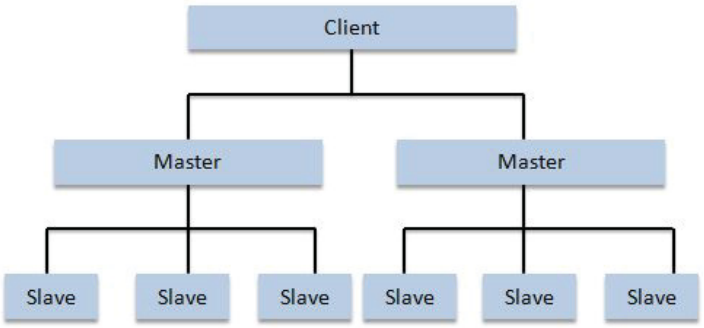



*Figure 16 Master-Slave structure*

This means that you define a Master and one or more Slaves. Master can do both writes and reads, slaves can only read as they are back-ups. All updates being done doesn't spread immediately to replicas, this keeps it more fault-tolerant. Dynamic queries, as MongoDB uses, give a deep query-ability. It also has a scalability solution built in the system if you want to partition your database, and it supports many of the biggest programming languages. It can handle data inconsistency with ease if it is configured correctly.[16]

**Key-value** is designed for storing, managing and retrieving, same as document based are, but key-value applies this on arrays instead of document-oriented information. Key-value systems treat data as single unclear collection of objects which could include different fields for every record. As the name implies it stores data as keys and values and retrieve values when a key is known.[47] Many developers imply that key-value databases handles size well and it constantly processes small reads and writes.

- **Redis** is an open-source in-memory cache and store, which stores keys with optional durability. The data can be persisted by being written to disk asynchronously. Redis is supported by a lot of programming languages, for example C, C++. Java and Python. It is also working under the Master-Slave structure.[12]

- **Aerospike** is an open source in-memory database, like Redis. It uses flash-memory (SSD) for scaling, and works with complex data types. It was developed with the idea of having market competitive persistence. It is a platform that follows the Master-Slave structure, however, Aerospike does work under an auto assign Master-Slave structure. It is written in C to be optimized for Linux users. Aerospike queries performs lookup through the use of secondary indexes and is based on value. The queries are done with a SQL-like tool called AQL.[97]

**Graph** databases use graph structures for semantic queries. With semantic queries you can retrieve explicitly and implicitly derived information, with nodes, edges and properties that represent and stores data. Every element in a graph database contains a pointer for its adjacent elements and it doesn't have to check indexes.[45]

- **Neo4J** is a native open-source database which is described as an on-disk based format. It uses HTTP/REST protocol for embedding in Java. It has its own integrated pattern-matching query language called Cypher which is inspired by SQL - using clauses to build queries.

**Multi-model** is a database that can support multiple data models. Usually a database management system only follows a single data model that has specified how data can be organized, stored and manipulated. Key-value, graph, relational and document based models are all examples of what you could find in a multi-model database.[44]

- **OrientDB** is known to many people as a graph database, however, it is also mixed together with features taken from document oriented databases and features from key-value database structure. It uses binary protocol, HTTP REST/JSON or Java for embedding. Its architecture uses many masters, so called multi-master. OrientDB is compatible with graph-style and interconnected data and the query language is much like SQL but there is no support for JOINs.[128] It has support for Java, PHP, Python, Ruby, Scala, Perl along with some other more low profiled programming languages.[103]

Following is a table with an overview of the databases just described. It shows the availability, what type of data structure it supports, the category, which protocol and platform it supports.

| Database | Availablity | Data structure | Category | Protocol | Platform |
|---|---|---|---|---|---|
| **Cassandra** | Open Source | Structured, semi-structured, unstructured | Column | CQL3 | Java |
| **Hbase** | Open Source | Multi-structured | Column | HTTP/REST | Java |
| **CouchDB** | Open source | Structureed, semi-structured, unstructured | Document | HTTP/REST | Erlang |
| **MongoDb** | Open Source | Structured, semi-structured unstructured | Document | MongoDB Wire Protocol | C++, C, JavaScript |
| **Redis** | Open Source | Structured, unstructured | Key-value | RESP | ANSI C |
| **Aerospike** | Open Source | Structured, unstructured | Key-value | Aerospike Wire Protocol | C |
| **Neo4J** | Open Source | Structured, semi-structured, unstructured | Graph | HTTP/REST | Java |
| **OrientDB** | Open Source | Structured, semi-structured, unstructured | Multi-model | HTTP/REST/JSON | Java |

*Table 2 Overview of databases*

# 3 Method

## 3.1 Research questions

Both of us have an interest in analysis of Big Data and we have noticed that there is a lack of "guides" where different platforms are described. Because of this it can be time consuming to research the available options to find the platform that is best suited for the specific analysis you want to perform. To ease this research we wanted to create a guide and in the process many questions arose and we narrowed them down to:

> **RQ1.** What are the functionality, strength and weaknesses of the most common platforms for analysis of Big Data? I.e in what situations can they be used and what type of analysis are they best respectively worst suited for?
> **RQ2.** Does our thesis meet the goal to work as a guide in the Big Data jungle?

## 3.2 Research design

To answer our research questions we have read articles addressing analysis of Big Data, compared their contents and formulated descriptions of the frameworks, which you can find in section *2.1.2 Platforms and Frameworks*. We have also analysed our findings and formulated our conclusion about when a specific framework is appropriate to use in section *4.3.2 Solution proposals and their areas.* We created a summary which we handed out to ten IT students to read before answering a survey about the content of this thesis to investigate if it achieves the goal to function as a guide among the frameworks.

Our research approach was:

- Descriptive - We gathered and compared information that regarding our questions at issue. We did not change or manipulate any of the information that we found but we added our own conclusion. Our research included article search and a survey among IT students.
- Qualitative / Flexible - We wanted to get an in-depth understanding for what problems are associated with analysis of Big Data, and we wanted to make a profound research on what tools and platforms there is to facilitate these problems.

We have searched seven different databases for articles addressing the subject and we have found relevant literature for our thesis. We have read this literature to define the background of this thesis, describe the platforms, and to find in what situations they are best suited to use. The result from this research is found throughout the text and you can find the references in the end of this thesis. We have also chosen to use footnotes for short explanations of words where we found it necessary. To make the selection of articles more manageable and to make sure the information we found was up to date we focused on results from at most four years back. Research for this thesis has involved a literature study on articles addressing Big Data and available platforms for analysis of Big Data. A more

descriptive explanation and information about the different keywords and databases used in our searches can be seen under chapter 4.1 and 4.2.

To evaluate the purpose of our thesis, to create a guide in the Big Data jungle, we asked 30 IT students to take part in a survey. This survey is made to indicate how informative and helpful the content of this thesis is when trying to get an overview of the different frameworks and tools that are available for Big Data analysis. You can find the structure and the result of the survey in appendix B. To ease the reading for the survey participants we created a summary of our thesis, which you can find in appendix A

## 3.3 Survey execution

We wanted the participants to read our summary about Big Data and then answer questions related to it. The survey was created in such manner that it would be quick to carry through it. Because of time restraints we put a time limit of 48 hours to answer the questions. We believe that this is the reason that we only got 10 participants. Another reason for making the survey short is that we believe that if a survey is short, more people will actually take part in it and really try to answer the questions correctly and truthfully.

As mentioned in *3.2 Research design,* we performed the survey on IT students, either ongoing or former. We did a convenience sampling because we wanted relevant people to perform the survey. It would have been harder to analyze results from people that is without IT education since our survey consists of words and explanations that requires some knowledge in this area. The survey is divided into three parts: background, knowledge based on thesis and evaluation. The first part of the survey consists of questions about gender, age and education. This is followed by questions about frameworks and tools related to our thesis summary that they have read. The last part consists of questions about the quality and relevance of our thesis.

# 4 Literature study results

We have used seven different databases to find literature that we could use to achieve our goal to create a guide in the Big Data jungle. Throughout our research we have used several keywords and combinations of those keywords, we have search each database with each combination.

## 4.1 Databases

- **SIGSOFT** is short for Special Interest Group on Software Engineering. It's a collection of their collaborated interaction between practitioners, researchers and educators. By using their database and its constant striving for evolving and learning about new platforms we have gained more knowledge about the importance of Big Data using our keywords.[108]
- **ACM digital library** for finding papers/articles listed at SIGSOFT that is relevant to our study. This database has helped us find relevant articles for specific areas in Big Data by using our keywords.[107]
- **IEEE Xplore Digital Library** for finding conference publications from their digital library, recommended by the BTH library. Finding relevant reports by using related Big Data keywords and narrowing searches down by searching for publications not older than 2010 so that it is still relevant.[109]
- **BTH Summon** used for finding publications listed at IEEE Xplore. To find a complete and specific text BTH Summon has been of help.[110]
- **Google** to find information/statements about different platforms used in the world of Big Data. We found information/statements that we could use in our literature study by using web pages like Quora, Informationweek and other technical related web pages. By using these pages we found a lot of links and guidance to other articles and papers relevant to our thesis.[111]
- **Research at Google** for finding researches done on specific areas in Big Data. MapReduce is one example of an area that is being researched on and shared on this site. By using keywords that are part of the Big Data world we found much relevant articles to help us mapping out our knowledge.[112]
- **Google Scholar** for finding articles and papers relevant for our study.[113]

**Keywords** big data, technologies, scalability, clustering, analyzing, data, storing, parallel, software, problems, solutions, research, vertical, horizontal, platform, mining, cloud, computing, types

**Combined keywords** "big data problems", "big data technologies", "big data solutions", "big data analytics", "big data storage", "big data software", "big data research", "scalability in big data", "scalable big data analyzes", "clustering technologies", "clustering software", "clustering big data", "parallel analytics", "big data parallel technologies", "horizontal scaling", "vertical scaling", "horizontal scaling platforms", "vertical scaling platforms", "cloud computing", "data mining", "data mining for big data", "big data types"

**Specific parts** big data, hadoop, mapreduce, horizontal scaling, vertical scaling, big data analytics

We had a big base of words to search for in order to find what we want, and we have combined them in many different ways that we found relevant.

## 4.2 Literature focus

By focusing on newer studies, articles and information we have gained trust in our search results. By newer studies we mean years between 2010-2015. Though we needed to broad our searches to find some relevant information, one time we expanded the year span to 1997. After reviewing the article from 1997 we found that the information in it was still usable for our purpose.

The world of data is vast and constantly evolving, and it is constantly updated because it is still new ground.[114] Of course there are some specific keywords and standards in Big Data that will remain the same, but the platforms are either constantly popping up, being updated or removed from the competition of maintaining, storing, analyzing or handling Big Data.

We focused on articles which included tests/experiments on one specific framework or a framework in comparison to others. Because of RQ1: *What are the functionality, strength and weaknesses of the most common platforms for analysis of Big Data? I.e in what situations can they be used and what type of analysis are they best respectively worst suited for?* we chose articles and websites that focused on actual results, and recommendations. By choosing this we felt that we would get to know functionality, strengths and weaknesses of Big Data frameworks. Meaning, there are some frameworks/tools that we wont do any deeper analysis on because we did not find enough resources. However, we do mention them under *2.1.2.2 Big Data frameworks/projects.*

## 4.3 Literature review results

The vast and fast expanding data being stored needs to be processed and in the beginning of the Big Data era the number of platforms for this was very few. As a result new coming platforms for processing data have arrived. It's not easy to know which platform to use for what, mapping out the advantages and disadvantages of the data processing platforms is a help on the way.[35]

Throughout our literature review we have seen a lot of confusion about what Big Data is, more and more people are getting interested in knowing more about the buzz of Big Data. Baeta-Yates(2013) claims other factors than just volume should be considered. As an example, big can mean different things depending on the device that is being used. The importance of finding the right data to analyze for a certain problem is highlighted.[50]

As mentioned in the previous paragraph the number of people interested in Big Data is growing fast, this goes for different businesses/enterprises of course. Caldarola and Picariello (2015) points out the benefits for enterprises in analyzing Big Data, and how important it is to have comprehension of the data to be collected and to know how to handle it in order to get the correct results. They also outline new trends in information and communication technology for modern enterprises as well as some Big Data application scenarios like retail industry and public sector and how they can benefit from Big Data

analytics. One benefit can be increased sales per visit with a red line in customer behavior when it comes to shopping.[2]

By now we know that there are benefits to gain from analyzing Big Data. In a study about platforms for Big Data analytics, Singh and Reddy(2014) are mapping out different hardware platforms in a limited scope. The number of hardware platforms is high and can be hard to keep track on. The platforms they are focusing on are analyzed, both their strengths and weaknesses based on scalability and data size supported amongst other things. As we have seen there is a need for guidance among platforms, Singh and Reddy agree with us, e.g analytics of the effectiveness the different platforms bring. One of the direction they mean that other people can take in the area is mapping out what platforms serves a particular application best.[35]

Gandomi and Haider(2014) are writing about how important it is to use certain methods for analyzing data that is not organized, so called unstructured data.[55]

### 4.3.2 Solution proposals and their areas

In this part we are summarizing our findings regarding the frameworks and platforms, the areas that certain programs fit in and why this is needed in the world of Big Data. We do not focus on numbers and exact results from comparisons that have been conducted, instead we want to show what conclusions we reached for each platform after finishing the literature study. The scope of this thesis does not include measurements, it is focused on functionality and in what situation a certain platform is well suited to use. The reason for not including measurements is that the material that we have read differ from each other and not all platforms have been measured in exact numbers. However, when we have found numbers that will show when a platform works well or not we have included those results in our text.

The following parts (the rest of this chapter) combined with *5.1 Available platforms and frameworks for distributed and parallel analysis of Big Data* will answer RQ1: *What are the functionality, strength and weaknesses of the most common platforms for analysis of Big Data? I.e in what situations can they be used and what type of analysis are they best respectively worst suited for?*

**MapReduce** is a common tool to use when you are processing large volumes of data and offers quality service when you have to face the huge variety of data types that is found in Big Data problems. Even so, it has some limitations in architecture and performance which doesn't make it optimal when iterative computations or stream processing is a part of the application. There is also a lack of standardized SQL-like[13] language which doesn't make it ideal in a database context either. Even so, MapReduce comes with a lot of benefits; high scalability, fault-tolerance, simplicity and independence from the programming language and the data storage system. It handles failures automatically and hides the complexity of fault-tolerance. Message Passing Interface (MPI)[14] gives MapReduce excellent performance through its support for general reductions but the fault tolerance and flexibility is not as good as in some MapReduce wrappers. MapReduce is an important model for large-scale data-parallel applications like web indexing and data mining, and it is attractive for

[13] A special-purpose programming language designed for managing data held in a relational database management system, or for stream processing in a relational data stream management system.
[14] A standardized and portable message-passing system designed by a group of researchers from academia and industry to function on a wide variety of parallel computers.

processing of arbitrary data in parallel. As MapReduce is schema and index free it is very flexible and can work with semi-structured and unstructured data but it is not well suited for correlated data.[19][26][27]

#### 4.3.2.1 MapReduce wrappers

**Apache Pig** is easy to learn, but the easy to learn part is not always a positive thing. There are just a small amount of built-in common functions that often work well in their specific meaning, but this leads to a narrow type of analytics of data sets. Pig handles the variety of data with grace as it is said being able to handle any type of data - “Pigs eat anything!”[5] The suggestion that Fuad, Erwin and Ipung (2014) have is to use it on data that requires complex queries and a lot of joining of data in large sets. Pig is better suited for analyzing data in petabyte volume than low-volume data, which can be time consuming compared to other alternatives.[39] According to Apache, Pigs key properties are ease of programming, optimization opportunities and extensibility.[125]

**Apache Spark:** Zaharia, Chowdhury, Franklin, Shenker and Stoica (2010) believes that Spark's RDD idea, though limited, of storing information so that it can easily be rebuilt if needed could be useful. Imagine if a node in a cluster of machines fails, then the RDD can be restored. This can be done because RDDs has handles which contains enough information in order to make sense of it, the elements of a RDD doesn’t need to exist in any physical storage.[23]

Wang, Goldstone, Yu and Wang (2014) have analyzed Spark’s performance on high performance computing systems. They see Spark as a promising new MapReduce framework even though they see problems in Spark’s performance, when Spark is trying to distribute the workload it can get unevenly balanced and they could see the need of a load balancer to help Spark.[31][118] Spark powers tools for streaming, querying, machine learning and graph computation and some of its main features are its speed, ease of use, generality and that it runs everywhere. The speed of Spark comes from the ability to stream-process large volumes of data.[118]

**Disco** scales well and can efficiently process and analyze large datasets on commodity hardware. Disco supports Python which makes it very powerful and it includes tools for indexing large amounts of data points and query them in real time.[90][91] Disco minimizes the overhead of virtual machines as well as allowing for efficient sharing of memory and disk resources between the virtual machines.[92] Highlights of the Disco framework includes that it is easy to install and that it runs jobs written in any language. According to Schömig, Neuhäuser, Seidler and Bücker (2014) Disco is unreliable when pushing the cluster network to its limit and performs worse than Hadoop and MR-MPI in their tests of implementing MapReduce to generate a list of SBAC (signed-binary adder cells) truth tables.[34]

**Misco** is a very powerful platform for developing and supporting distributed applications and the user can create and manage applications through a browser interface. The system has a built in job scheduler and fault-tolerance.[38][36][91] Through experiments, Dou, Kalogeraki, Gunopulos, Mielikainen and Tuulos (2010) found that “the majority of processing is spent performing computations, and only a small fraction of the time is required for downloading,

uploading and cleanup, which is quite efficient for resource constrained devices such as the phones we are considering".[38]

**MR-MP**I (MapReduce-MPI) has one major flaw, there is no fault-tolerance.[29] Through implementing MapReduce to generate a list of SBAC (signed-binary adder cells) truth tables without any redundancies Schömig, Neuhäuser, Seidler and Bücker (2014) found that MR-MPI had the shortest run time out of three tested MapReduce frameworks (Disco and Hadoop). They also found that MR-MPI was unstable when calculating $2^{34}$ or more (key, value) pairs and that it is less stable than Hadoop MapReduce when bringing a compute cluster to its limits in terms of disk memory and network bandwidth.[34] MR-MPI works on any operating system which supports C++, C or Python.[91]

#### 4.3.2.2 Big Data frameworks/projects

Apache Hadoop is a framework that is widely used in Big Data processing. As mentioned in *2.1.2.2 Big Data frameworks/projects*, Hadoop has a wide variety of tools which makes it powerful. Yao, Varghese and Rau-Chaplin (2013) recommend developers to use Hadoop's projects to tackle the data process challenge that comes with Big Data.

- **Avro** file format includes schema information data files. Smith, Seligman, Rosenthal, Kurcz, Greer, Macheret, Sexton and Eckstein (2014) recommend developer to use Avro to fetch stored schema information from data files.[73]

- **HBase** results are specified under **Databases** on page 32.

- **Hive** is a tool to use when it comes to query and data analysis, the query executes via MapReduce. It can be quite easy to learn Hive if you have some experience in SQL since before and just do some research on the data types, functions, formats and scripts it comes with. Fuad, Erwin and Ipung (2014) are pointing out that Hive works really well on cheaper hardware machines with correct storage software. According to Liu, Liu, Liu and Lei (2013) there are some suggestions on which data analyses tasks Hive works best with, one of which is when you have complicated array operations.[56]

- **Pig** is analyzed on page 27.

- **MapReduce** is analyzed on page 26.

- **HDFS** is a common and prized tool to use.[129] Even so, there have been some questioning about HDFS performance and Shafer, Rixner and Cox (2010) did perform extensive tests and analyzed HDFS performance thoroughly. At the end of their analysis they had reached the conclusion that the biggest problem is the trade-offs between portability and performance, but the shortage that is mostly mentioned is the weak performance when dealing with small files.[94] Islam, Rahman, Jose, Rajachandrasekar, Wang, Subramoni, Murthy and Panda (2012) saw some good results in the scalability that HDFS deliver, they found it to be a perfect tool for it.[93]

- **Sqoop:** Pippal, Singh and Kushwaha (2014) needed to transfer data between relational databases somehow. They picked Sqoop because they had an idea of storing data into tables of rows so that they could identify each row with a unique key. They can assure that this is an easy way to go and it is quite simple as a command-line interface.[77][78]

- **Flume** is recommended to use when you want a reliable service for collecting, aggregating and moving big amounts of log data.[79] Flume is common to use when there is need to evaluate logs produced in heavy tested Big Data projects. Wang, Rayan and Schwan (2012) have seen good potential in Flume's highly adaptable software. If you have high priority on deployment, ease of operating and good scaling of real-time Big Data processing, then Flume is a good alternative. Although, if a big company have a lot of servers, like 100 000 servers, Flume is not be a good choice. A lot of servers equals a lot of tracing data and troubleshooting, so it would be like finding a needle in a haystack.[80]

- **Chukwa** was used by Shang, Jiang, Hemmati, Adams, Hassan and Martin (2013) to collect and analyze log data. They are pointing out that Chukwa is good to use when it comes to scaling of nodes in collection and analysis, but not as good when complex analysis needs to be done.[82]

- **Oozie** is quite good in operating cost efficiently.[84] Islam, Huang, Battisha, Chiang, Srinivasan, Peters and Neumann (2012) developed Oozie because they wanted a tool that can provide scalability, security, multi-tenancy and operability. They mean that other workflow implementations were lacking on at least one of these requirements at the time being. Geyik, Saxena and Dasdan (2014) used Oozie as a workflow scheduler in their project and they found it to be efficient for these four purposes.[85][86]

- **Mahout:** According to Moh and Bhagvat (2012) there are four steps in clustering sets of data when you use Mahout, they use these steps in their project. The first step they use is about converting input to sequence file, second step converts sequence files to vectors, third step runs an analyzer for English stop words, and the last step runs the clustering algorithm. They liked Mahout but when they performed their tests Mahout was very much in an early stage which did affect the time it took to understand it. Mahout did what they expected though, and it did work as a tool for clustering data sets.[88] Walunj and Sadafale (2013) worked with Mahout a year later and can attest that it is highly scalable. They suggest that you should use Mahout when you want to use existing algorithms and are looking for flexibility in your implementations by using those algorithms.[89]

- **HCatalog:** Lin and Ryaboy (2013) suggests that you should use HCatalog when you are working in a large team, because of the possibility of needing to alter a lot of already existing scripts. This is very time consuming but can be a good investment if you have enough team members in order to allocate time for it. If you would find that

the complexity of your platform which handles analyses grows you should consider taking advantage of what HCatalog has to offer.[68]

- **Cassandra** analyzed later under **Databases** on page 31.

- **ZooKeeper:** We have learned that the developers of ZooKeeper saw early potential in getting it to stand on by itself. Based on their tests, Lynden, Tanimura, Kojima and Matono (2011) suggests that ZooKeeper should be used when synchronization and communication between reduced tasks is needed. As stated by ZooKeepers own site, Lynden, Tanimura, Kojima and Matono can confirm that it reads really fast but because of the data serialization, to disk writes are a little bit more time consuming.[65] By using ZooKeeper Vernica, Balmin, Beyer and Ercegovac (2012) has tested and proven that the leader election truly works if a new leader-node is needed.[66]

**Dryad:** Isard, Budio, Yu, Birell and Fetterly (2007) examined the efficiency of Dryad and found that implementation that allows multiple jobs to cooperate when executing concurrently makes more efficient use of the resources of a large cluster. They also found that Dryad scales excellent on both small and large clusters.[57]

**HPCC (High-Performance Computing Cluster) Systems** is suited for everything from raw data processing to high-performance queries and data analysis using common language, and the optimized cluster approach results in lower system costs than other alternatives. HPCC also have a high-level fault resilience and is suited for a wide range of data-intensive applications. HPCC implements two types of data processing platforms which serves different data processing needs and can be optimized for their specific purposes to provide performance of the highest level possible.[64] HPCC Systems is fast, scalable and simple.[126] When sorting 1 TB “on the same hardware configuration a 400-node system, HPCC success is 6 minutes 27 seconds and Hadoop success is 25 minutes 28 seconds. This result showed that HPCC is faster than Hadoop for this comparison.”(Sagiroglu and Sinanc, 2013)[83]

**Apache Storm** makes it easy to process unbounded streams of data (tuples) through its real-time and fault tolerant computation method.[69][70][71] Some of the advantages Storm provides are that it can be used in any programming language, it is extremely fast and reliable and it can be utilized for online machine learning. Du, Liu, Liu and Chen (2014) finds that Storm can achieve high accuracy in their real-time computing scheme.[70]

**Sector/Sphere:** Gu and Grossman (2011) finds that Sector/Sphere consistently performs two to four times faster than Hadoop, measured by the Terasort[15] and MalStone[16] benchmarks.[41] Saranya and Sujatha (2012) also performed experiments measured by the Terasort and MalStone benchmarks and got the same result.[40]

**GridGain:** Samovsky and Kacur (2012) found that using MapReduce in GridGain reduces the time for classification model building, using the distributed cloud version of tree

[15] Samples the input data and uses map/reduce to sort the data into a total order.
[16] A stylized benchmark for data intensive computing.

algorithm.[9] Samovski and Ullbrink (2013) also performed an experiment with GridGain MapReduce as a platform of distributed classification trees. They found that the distributed version of the algorithm improves the computation time but the reduction is not linear which is a result of unbalanced data distribution, different values of variance error in learning from particular data parts.[22]

**GraphLab Create:** By targeting common patterns in machine learning GraphLab achieves a balance between low level and high level abstractions which leads to a high level of usability, expressiveness and performance. GraphLab is well suited for machine learning algorithms and it allows the user to easily design and implement efficient, scalable, parallel algorithms. The configurable consistency model guarantees high level data representation through the data graph and automatically maintained data consistency, this insulates the user from the complexities of synchronization, data races and deadlocks. GraphLab provides a collection of parallel schedulers encompassing a wide range of machine learning algorithms as well as a framework which can be used to compose custom schedules.[63] After a comparison between GraphLab and Mahout, Li, Gibson, Ho, Zhou, Kim, Buhisi, Brown and Gerber (2014) recommend GraphLab for the primary deployment environment for time-sensitive applications and Mahout for error-sensitive applications.[63]

#### 4.3.2.3 Databases/Storage

**NoSQL Databases**
This type of databases is perfect to use if you are expecting a large database.[67] Noteworthy is that NoSQL is often interpreted as Not only SQL, and it has been said that since NoSQL databases entered the scene the dominance of relational databases has dramatically decreased. This doesn't mean that NoSQL databases will be the new dominators, relational databases just won't be the obvious choice anymore.[54]

**Column:** These kinds of databases often delivers very high performance and high scalability. If you wish to compress huge amounts of data, then column store databases is the way to go. Because it is column based it uses less disk space than most other databases do.[30]

- **Cassandra:** Abramova and Bernardino (2013) have tested Cassandra. They recommend users to go for Cassandra when you think that the data size will constantly increase. They can also recommend Cassandra when it comes to read/update performance.[28] Cassandra works faster with bigger data sizes, according to Tobey and Harper (2014). Cassandra together with Apache Spark has turned out to be a good base for high velocity analytics, time between receiving and end-users outcome is timed to milliseconds.[11] Kuhlenkamp, Klems and Röss (2014) can recommend Cassandra if you want to see some good performance when it comes to reads and writes. Writes is usually much faster but Kuhlenkamp, Klems and Röss (2014) saw some good results in reads as well.[24] When you need to handle the inconsistency, variability, in data Cassandra is a tool you can count on. But you can count on

Cassandra for more reasons, the data variety that Cassandra supports is quite impressive.

- **HBase:** Bhupathiraju and Ravuri (2014) have learned during their tests that HBase is very much persistent and that it is a strictly consistent storage system with almost perfect write-in terms when it comes to I/O, and it's even better in read performance. They are pointing out that it is perfect to use when you want a system that can grow and shrink under production. Bhupathiraju and Ravuri (2014) and Vora (2011) is recommending HBase to users that want a highly scalable alternative. The architecture of HBase provides consistency in data, it is developed in such a way that the updates will be replicated asynchronously and shared between data centers.[75][76]

  HBase scales better when working with larger data volumes. When it comes to write performance HBase is standing out because of better results than Cassandra according to experiments performed by Kuhlenkamp, Klems and Röss (2014). They think that developers should use it if read performance is one of the most important factors.[24]

**Document:** Lerman (2011) means that you can actually see document-oriented databases as a subclass of key-value store databases, described in paragraph four on this page under Key-value section. The difference is that a document based database relies on internal structure in the document order to extract metadata.[52] It is perfect to use when you want to use the document id as the key and the document itself as the value.

- **CouchDB** is good to use for collecting, small changes in data on which pre-defined queries are to be run.[17] Eken, Kaya, Ilhan, Sayar, Kavak, Kocasarac and Sahin (2013) can recommend CouchDB when it comes to document management when you have replicated databases with the need of continuous repetition.[18]

- **MongoDB:** Dede, Govindaraju, Gunter, Canon and Ramakrishnan (2013) used MongoDB as back-end to Hadoop and saw some good results. By using MongoDB you separate data server availability from compute server availability. By doing this the system will have a better ability to recover failed nodes, it becomes more resilient. Using MongoDB in a cluster with nodes that is running the risk of failing is a good idea, the nodes gets the input data from the MongoDB server, therefore losing a node doesn't lead to loss of data.[15]

**Key-value** DeCandia, Hastorun, Jampani, Kakulapati, Lakshman, Pilchin, Sivasubramanian, Vosshall and Vogels (2007) that key-value databases handles size well and it constantly processes small reads and writes.[51]

- **Redis** should be used in solutions where you can afford to lose some data when servers go down. It does however scale very well and the execution time is usually good. Zhang, Tudor, Chen and Ooi (2014) did test the cluster

performance of Redis by using the PageRank algorithm.[17]Redis server operates in a single-threaded way. When you are running a long scripting-job the Redis server cannot take care of any other requests from servers, it gets unavailable. So if you have a lot of servers, or an increasing number of servers, you will see poorer performance due to waiting time. Redis performs queries by direct data retrieval commands with no query planners in the middle. Redis is optimized for reads, but writes are not far behind.[10]

- **Aerospike** is used as a session store, id-mapping and user profile store by many customers that has the need of storing huge amount of items fast with the read and write predictability. It is immediately data consistent which a lot of customers like.[98] Aerospike has turned out to be optimized for read-heavy systems and it has also shown good performance for systems that are well balanced between reads and writes.[21]

**Graph** databases should be used if you are looking for high performance online query capability. The graph is represented by the processed raw data with MapReduce in Hadoop or stream-processing in Storm.[42]

- **Neo4J** is popular by users who are in need of a self-contained web admin and a database that handles a variety of data formats. Neo4J is from the beginning optimized for reads.[99] Holzschuher and Peinl (2013) agrees and recommend Neo4J for its readability, though it does affect Neo4J to be slower overall.[100] If you are developing something that is for searching routes like public transportations, road maps or social relationships this is a good database of choice.[101]

**Multi-model:** It can be a hard decision for engineers to choose how to model and store data, which can lead to a multi-model database. It supports different types of data for different use cases and can get it all to work on one platform. So if you don't know exactly what type of data you want to model and store, multi-model database approach can be the right way to go, it gives you a certain flexibility. [43]

- **OrientDB:** Weak performances can occur on the graph-layer because it is built on top of the document oriented structure according to Barmpis & Kolovos (2012). They are also pointing out that indexing of data can take some time as well, but the performance is still good.[104] It is said to be equally good on reads and writes, but reads are very much depending on how good the cache-memory, reads from disk, is and usually ends up in a long execution time. OrientDB was usually the slowest when Abramova, Bernardino and Furtado (2014) compared different databases read-performances.[4]

[17] Google Search algorithm to measure importance of website pages

# 5 Analysis

## 5.1 Available platforms and frameworks for distributed and parallel analysis of Big Data

Following paragraphs and tables will, combined with *4.3.2 Solution proposals and their areas,* answer RQ1: *What are the functionality, strength and weaknesses of the most common platforms for analysis of Big Data? I.e in what situations can they be used and what type of analysis are they best respectively worst suited for?*

After our literature study it is clear to us that developers, scientists and companies find the world of Big Data complex and vast. If you break it down to sub-categories and describe the different frameworks functionality and give some examples of situations when they are well suited you can get a more narrow and easy perspective.

We have divided this big subject into five main parts; Hadoop projects, frameworks, MapReduce wrappers, database structures and databases, because these are the most relevant and crucial parts to consider. In this section we describe the different available platforms and frameworks for distributed and parallel analysis of Big Data. In this analysis we have looked at Hadoop as one platform and not focused at the different subprojects.

Regarding Big Data frameworks we can observe that Apache Hadoop is used by many developers globally. It has a strong support from community despite numerous of strong competitors. Developers keep supporting Apache Hadoop for its open source projects and cross-platform compatibility. By going deeper into frameworks you see that Apache Hadoop has more subprojects than any other Big Data framework.

MapReduce ends up high on the list on common Big Data programming models. In our literature study we could see a big market using MapReduce. A lot of projects have developed MapReduce wrappers in order to take advantage of MapReduce and deliver innovative and competitive products. Apache Pig is one of the wrappers used widely by companies, Yahoo uses Pig for 40% of their Hadoop jobs.[5] Analysis performed in 2014 shows that Pig suits the need of working with huge amount of data.[39]

Our search results show that Apache Spark is a big competitor on the MapReduce market, but it is still young. Nokia's wrapper called Disco has turned out to be unreliable when you are pushing the cluster to its limit according to tests performed by Schömig, Neuhäuser, Seidler and Bücker (2014).[34] Another wrapper from Nokia, Misco turned out to be a good and powerful alternative when you are developing for mobile platforms using Python. MR-MPI shows good results when compared to other MapReduce wrappers (Disco and Hadoop), for generating a list of SBAC (signed-binary adder cells) truth tables, but it does not have any fault tolerance.

There are frameworks that have a wide and general purpose and still manage to show really good results, we found that Dryad, HPCC and Sector/Sphere to fall into this category.

*Table 3* gives an overview of the functionality and properties of the frameworks that we have examined in this thesis. To make the table manageable we selected the

functionality that we find most relevant, and that is used for different types of analysis. MapReduce is in the first column in this table. This is very relevant because, as described under section *2.1.2 Platforms and Frameworks*, it is a very important part of the world of Big Data. Distributed data processing and parallel data processing follows in the next two columns. We wanted to show this in the table because it is the main focus of this thesis. As data becomes big(petabytes, zettabytes), the only way to analyze it is to spread out the processing on several machines. None of the frameworks shown in the table works as a data warehouse, but we still want to highlight this as a column because it is a part of the world of Big Data. Big Data in itself can be seen as another definition of data warehousing, it is about collecting huge amounts of data in a particular manner.[130] Iterative computations is shown in the table, it is not the main focus here but it is not unusual to use iterative algorithms when processing Big Data. Stream processing is a specific type of processing that not many frameworks supports but we believe that it will be more common as the velocity of data grows and since social media plays such a big part in our lives. Query is a common concept in computer science. If you just want to show people whose first name is Tobe, type that into a query and then you can do that.[131] This is why we have query as a column. The next column is open source. We find open source important in many aspects. Without open source a lot of development would stand still, contributors wouldn't exist in the manner they do today and maybe we wouldn't have as wide variety of available technologies as we have right now. The final three columns shows which frameworks that supports which OS.

| | MapReduce | Distributed data processing | Parallel data processing | Data warehousing | Iterative computations | Stream processing | Query | Open source | Windows | Linux | Mac OS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Apache Pig | x | | x | | x | | x | x | x | x | |
| Apache Hive | x | | | x | | | x | x | x | x | x |
| Apache Spark | x | | x | | | x | x | x | x | x | x |
| Disco | x | x | | | | | x | | | x | x |
| Misco | x | | x | | | | | | x | x | x |
| MR-MPI | x | | x | | x | | | x | x | x | x |
| Dryad | | | x | | x | | x | | x | | |
| HPPC Systems | | | x | | | | x | x | | x | |
| Apache Storm | | | | | | x | | x | x | x | x |
| Sector/Sphere | | | x | | x | | | x | | x | |
| GridGain | x | | x | | | | | x | x | x | x |
| GraphLab | | | x | | x | | | x | | x | x |

*Table 3 Summary of functionality, frameworks*

In *table 4* we have pointed out the biggest strengths and weaknesses that we found for each of the frameworks throughout our literature study. We didn't have certain parameters in mind when we created this table so there is no relation between the different frameworks weaknesses and strengths, the purpose of the table is to give a quick view of what a certain framework might be suited for or not. After our research we couldn't find any specific weaknesses for four of the frameworks and that's why these cells in the table are empty.

| | Strengths | Weaknesses |
|---|---|---|
| **Apache Hadoop** | ·Has a strong base from related sub projects | ·Iterative algorithms |
| **Apache Pig** | ·Easy to learn<br>·Good performance on large amounts of data<br>·Support for multiple common programming languages | ·Few common functions |
| **Apache Spark** | ·Versatile<br>·High speed<br>·Many tools | ·Workload balancing |
| **Disco** | ·Support for multiple common programming languages<br>·Efficient data-locality-preserving I/O | ·Unreliable when pushing the cluster network to its limit |
| **Misco** | ·Effecient processing,not much time for downloading, uploading and cleanup | |
| **MR-MPI** | ·Fast MapReduce processing | · No fault tolerance<br>·Unstable when calculating $2^{64}$ or more pairs |
| **Dryad** | ·Allows the programmer to optimize tradeoffs between parallelism and data distribution overheads<br>·Flexible | ·In-memory computations |
| **HPPC Systems** | ·Versatile<br>·Fast<br>·Simple | ·Linux is the only supported OS |
| **Apache Storm** | ·Support for multiple common programming languages<br>·Fast<br>·Reliable | |
| **Sector/Sphere** | ·Supports multiple inputs and outputs<br>·Fast | |
| **GridGain** | ·Independent from infrastructure and platform | ·Workload balancing |
| **GraphLab** | ·Graph algorithms<br>·Machine learning | |

*Table 4 Summary of strengths and weaknesses, frameworks*

- **Apache Hadoop:** From our meticulous research and literature study as seen under *4.3.2.2 Big Data frameworks/projects* and *2.1.2.2 Big Data frameworks/projects* we could map out and see how extensive and strong Hadoop is when it comes to related sub projects. Sub projects include: Avro, HBase, Hive, Pig, MapReduce, HDFS, Sqoop, Flume, Chukwa, Oozie, Mahout, HCatalog, Cassandra and Zookeeper.[58][59]
- **Apache Pig:** Based on references collected by us we could see that it is easy to learn[5], works best with large data volumes(petabyte)[39], and the support for many common programming languages together with the ease of programming[125] also makes it easy to handle. However, the common functions is on the shorter side.[39]
- **Apache Spark:** Versatile thanks to RDD's[23], the high speed comes from stream-processing of large data volumes and it is powerful thanks to the many tools it comes with[118]. Problems can occur when you are in need of balancing out workload.[31]
- **Disco:** We could see that it supports multiple common programming languages during our literature study. Articles describe it as efficient in data-locality-preserving I/O[90][91] but unreliable when you are pushing the cluster network to the limit.[34]
- **Misco:** We can summarize it as a powerful platform thanks to our literature study which points out the efficient processing, ease of installing, built in job-scheduler and fault-tolerance.[36][38][91]

- **MR-MPI:** The experiments performed and published in articles showed that MR-MPI can have short-runtime but can also be unstable with calculations.[34] The biggest flaw is the non-existing fault-tolerance as mentioned earlier in the analysis and under *4.3.2.2 Big Data frameworks/projects.*[29]
- **Dryad:** We have seen an explanation of Dryad saying that it is flexible in size of cluster.[19] We believe that this flexibility equals a usable framework for different situations. An example of this is the handling of database-style queries. However, it does have problems with in-memory computations.[19][57]
- **HPCC Systems:** is a versatile framework, meaning that is suited for everything from raw data processing to high-performance queries and data analysis.[64] It is fast, scalable and simple.[126] Unix users are the only ones that can use it though.[91]
- **Apache Storm:** can simply be summarized as fast, reliable and it has support for multiple common programming languages.[70]
- **Sector/Sphere:** It is a fast framework, based on experiments comparing it with Hadoop.[40][41]
- **GridGain:** Its independency from infrastructure and platform reduces time for model building. In this framework, as well as others before, the workload balancing is a weak factor.[9][22]
- **GraphLab:** We found it to be well suited for machine learning thanks to our literature study. It provides a collection of parallel schedulers using a wide range of machine learning algorithms.[63]

When it comes to relevant databases available for working with Big Data we noticed that we should narrow it down to NoSQL databases because of the result when we were collecting data. NoSQL is also known as Not Only SQL which uses dynamic schemas for storage, compared to relational databases which demands that schemas are being defined before a user can add data. Because NoSQL databases don't demand a predefined schema they are ideal for handling large amounts of data. NoSQL databases are divided into five categories: Column, Document, Key-value, Graph and Multi-model.

*Table 5 is* a summary of the reviewed databases that we have selected and it is based on the information we collected from different articles and websites during our literature study. It shows language properties, strengths and weaknesses. It is designed in the same way as *table 4*, there is no relation between the different strengths and weaknesses and the purpose of the table is purely to give a quick view of the databases in this thesis.

| | Written language | Query language | Strengths | Weaknesses |
|---|---|---|---|---|
| **Cassandra** | Java | **CQL** | ·Storing huge amounts of data<br>·Fast writes<br>·Highly scalable | ·No JOINs<br>·Keys have to be unique<br>·Complicated searches |
| **Hbase** | Java | **No specific language** | ·Real-time access<br>·Clever disk usage<br>·Highly scalable | ·Single point of failure<br>·No JOINs, only with MapReduce<br>·Low Security |
| **CouchDB** | Erlang | **JSON** | ·Fast reads<br>·Highly scalable<br>·Simplicity with any JSON data | ·Slow writes<br>·No complex queries<br>·Spartial compacting needed |
| **MongoDB** | C++, C#, Go | **BSON store(binary format JSON)** | ·Dynamic queries<br>·High write performance<br>·Good load-blancing | ·No JOINs<br>·Concurrency issues<br>·Memory usage |
| **Redis** | ANSI C | **Extra commands for querying** | ·Storing in various formats<br>·Fast reads<br>·Can back data to disk | ·Complex to configure<br>·Memory fragmentation issues<br>·Whole dataset always "lives" in RAM |
| **Aerospike** | C | **AQL** | ·Easy to set up queries<br>·Very fast access to data by key<br>·Handles heavy reads great | ·If a node fails, the data in that node is lost |
| **Neo4J** | Java | **Cypher** | ·Fast reads<br>·Great graph structure<br>·Self-contained web admin | ·Difficult to model tables<br>·Difficult to query on node content |
| **OrientDB** | Java | **JSON, SQL support** | ·Uses a good mix of database models<br>·Great with graphs<br>·Great on writes | ·No JOINs<br>·Reads is dependent on cache |

*Table 5 Summary of strengths and weaknesses, NoSQL databases*

## 5.2 Survey

Following you can find the answer to RQ2: *Does our thesis meet the goal to work as a guide in the Big Data jungle?*

We have summarized the results from the survey below and you can find the complete result in Appendix B. The survey had 10 participants from which the data is produced. We have reached out to 30 IT students, which gives us a participant frequency of 33%. We have accomplished that through reaching out to students at our school and some former students that we have worked with in previous projects.

### Background

**Age:** The majority of the people that participated in the survey were in the group 21-25 years old (50%). There were no one in the three groups older than 51 years (51-55, 56-60 and 61+) and no one younger than 21 years, group -20. We can also see a gap between 36-46 years (36-40 and 41-45).

**Gender:** The majority of the participants were male, only two out of ten were female.

**Education:** All participants have a relevant education, most of which is three years long. Only two participants have an education that is five years long. Three of these participants studies Software Engineering (33%) and three studies Web programming (33%), these two subjects are in majority. Two participants studies Masters of Science in Engineering: Computer Security (20%) and there is one participant that gave university as an answer. It is impossible for us to know exactly which education it is but we know that that the participant has an education in software engineering or other relevant subjects.

### Data

One of the three data questions only had one correct alternative. The other two questions had more than two correct alternatives. Percentage value for each person has been calculated on total number of correct answers against the max value if you have correct answer to all the questions. Wrong answers will be deducted from the total to make sure that participants can't put all alternatives as correct. You can have 11 right answers at most.

We drawed a line at 80% correct answers (9 right answers), a high number because of the nature of the questions, in order to see that our thesis summary gives you the information you need. In that case, 54% of the participants had enough correct answers and then 46% had insufficient number of correct answers. The ones that showed best results, three participants (33%), had 11 of 11 correct answers, the lowest point was 6. The average is 9.1 correct answers.

*Figure 17* shows number of participants (in %) that has answered 100% correctly to a specific questions. It also shows number of people that chose correct + wrong alternatives and also how many participants that chose correct answers but didn't choose enough alternatives in order to get 100% correct.

If we take a look at the ones that had correct answers we can observe that the question regarding MapReduce is the only question with 100% correctness. MapReduce wrappers, NoSQL and Big Data are all lying on 60%.

The questions about MapReduce wrappers and NoSQL have the exact same result in each category - 60% correct answers, 30 % chose correct alternatives + wrong alternatives and 10 % correct answers, but not enough alternatives chosen.

MapReduce and Big Data are the questions that stand out, MapReduce has 100% correct answers as we know. The question about Big Data has 60% correct answers, 10% chose correct alternatives + wrong alternatives and 30 % correct answers, but not enough alternatives chosen.

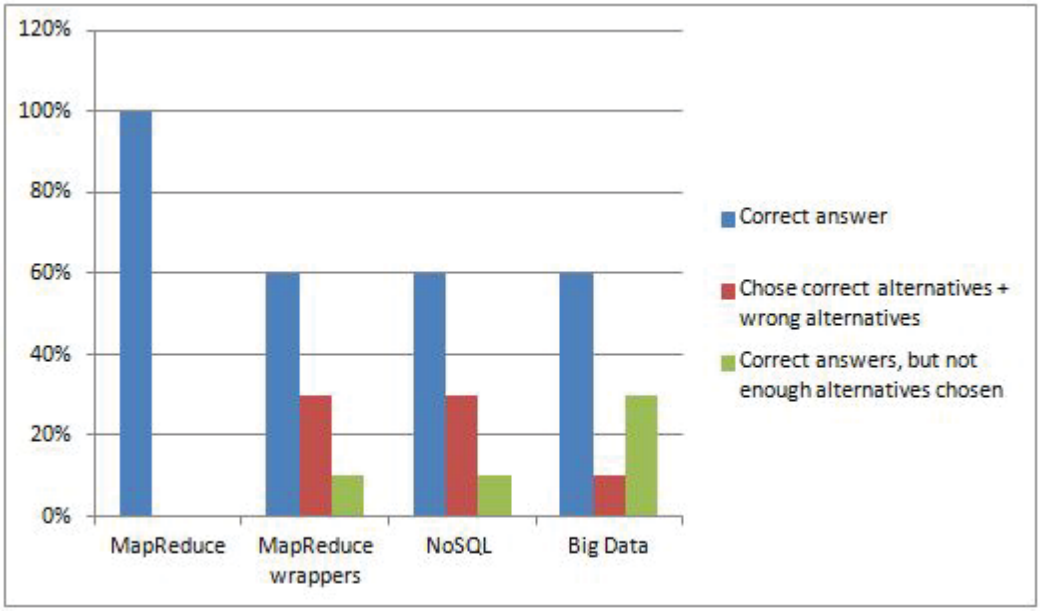



*Figure 17 Answer distribution in %*

Following are tables for each question that shows how many participants that chose a specific alternative.

**MapReduce:** All participants gave a correct answer to how MapReduce can be described, as we showed earlier.

| MapReduce can be descibed as.. | | |
|---|---|---|
| | # | % |
| A programming model that makes tasks run in parallel on multiple mamchines | 10 | 100 |
| A framework that process the data using only key | 0 | 0 |
| A programming model that only works with unstructured data | 0 | 0 |

*Table 6 MapReduce*

**MapReduce wrappers:** The correctness frequency of this question was high. Most participants knew that Apache Pig and Apache Spark are MapReduce wrappers. The alternative Google Misco is a bit sneaky, it is actually Nokia Misco.

| These are MapRwrappers: | | |
|---|---|---|
| | # | % |
| Apache Avro | 0 | 0 |
| Apache Pig | 8 | 80 |
| Apache Spark | 10 | 100 |
| Google Misco | 3 | 30 |
| Apache Flume | 0 | 0 |

*Table 7 MapReduce wrappers*

**NoSQL:** The answers to this question weren't satisfactory. All alternatives describe a type of database but not all of them are NoSQL.

.

| The NoSQL categories are.. | | |
|---|---|---|
| | # | % |
| Collection | 0 | 0 |
| Column stored | 8 | 80 |
| Distributed database | 2 | 20 |
| Document-orientated | 9 | 90 |
| Key-Value | 10 | 100 |
| Navigational | 1 | 10 |
| Relational | 2 | 20 |

*Table 8 NoSQL*

**Big Data:** The correctness here was high.

| Big Data can be described with the words.. | | |
|---|---|---|
| | # | % |
| **Security** | 0 | 0 |
| **Simplicity** | 0 | 0 |
| **Variability** | 9 | 90 |
| **Variety** | 9 | 90 |
| **Velocity** | 10 | 100 |
| **Veracity** | 9 | 90 |
| **Visuality** | 0 | 0 |
| **Vulnerability** | 1 | 10 |
| **Volume** | 8 | 80 |

*Table 9 Big Data*

## Evaluation

The feedback in the comment section was very positive and we observe that some of the participants discovered an interest in learning more about Big Data after reading our summary. As seen in table 10, 11 and 12, we also got good result in the part where the participants rated our summary. There are no answers lower than three and the majority of the participants chose the number four on these three questions, which we find satisfactory.

| How informative did you find the text? | | |
|---|---|---|
| | # | % |
| **1 (Not at all)** | 0 | 0 |
| **2** | 0 | 0 |
| **3** | 0 | 0 |
| **4** | 5 | 50 |
| **5 (Very)** | 5 | 50 |

| Do you feel that you have a good overview of the platforms and databases after reading? | | |
|---|---|---|
| | # | % |
| **1 (Not at all)** | 0 | 0 |
| **2** | 0 | 0 |
| **3** | 1 | 10 |
| **4** | 8 | 80 |
| **5 (Very)** | 1 | 10 |

| Do you feel comfortable to use our document as a tool in the world of Big Data? | | |
|---|---|---|
| | # | % |
| **1 (Not at all)** | 0 | 0 |
| **2** | 0 | 0 |
| **3** | 2 | 20 |
| **4** | 6 | 60 |
| **5 (Very)** | 2 | 20 |

*Table 10, 11, 12 Thesis evaluation*

## Validity threats

We have chosen people that have an education, ongoing or finished. We believe that this is the right way to evaluate our thesis, however the survey have several shortcomings. The number of participants is on the shorter side. We believe that if 20 IT-students, 66 % of people asked, would have performed the survey we would have felt more pleased with the total percentage of participants. However, after collecting and reviewed the survey answers

we felt that we should have aimed for a much higher number of participants in order to see clearer patterns. Asking 100 IT-students with a participation rate of 50% in the end would have made a better base for our analysis. We could probably have used age, gender and education to see patterns in a larger group of participants. Such as, does a person with the same education, age and gender describe our thesis summary as positive or negative in the same manner. More participants equals more people with the same education, age and gender. A follow up question to education, such as: *How long ago did you start your education?,* could have helped us understand if their answers is different depending on how recent they started their education. We believe that it would have been more difficult to analyze if the question would have been: *How long ago did you finish your education?*. because there are a lot of students that doesn't finish their education at stated time. People taking a three year long education can take a couple of years more to study or even longer. However, the question we have about education confirms that the person taking the survey is in our target group.

Our questions about data works in its purpose, to find out if they can answer correctly after reading our thesis summary. We could have done even more questions though, maybe two or three questions per section and have had an alternative to each question that says “I don’t know”. Now there can be participants which don’t know the answer and took a chance because the questions are mandatory. We can see one participant that has answered with one alternative on each question, except for the last question. If we would have an option for “I don’t know” this could have been a different result.

If we would have formed our survey in a different way we could have seen results that show more of people's problem solving skills. When it comes to MapReduce wrappers a question could be: “If you want a MapReduce wrapper with good load balancing, which one would you choose?”. We could have done even more complex problem solving questions for our participants, but we felt that we wouldn’t get as much attention from the participants if we did which led to the decision of not including that type of questions. We were prepared for this to be threat since the beginning of our thesis.

In addition to these issues we had to take in consideration how much time we had left before deadline and how much time people would agree to spend on answering questions after reading our summary. We could have sent out the survey earlier, but the problem is that we needed to finish our whole thesis including our conclusion (conclusion without survey result) in order to “test it” on people and see if it works as a guide.

### Summary

The result of the survey shows that we have accomplished to reach our goal, to create a guide in the Big Data jungle. Since none of the answers regarding the quality of our thesis is lower than three, we think it is safe to say that the content is informative and a good base for decision making regarding the choice of platform for analysis of large volumes of data. However, as we mentioned under “Flaws in survey” we could have done a more advanced survey with questions more related to problem solving with help from our summary.

The result of the platform part of the survey indicates that we have done a good job describing the selected platforms and tools. The comments we got on the last question were only positive which assure us that we have structured our work in a good and understandable way.

# 6 Conclusion

## 6.1 Contribution

The purpose of this thesis was to give an overview of available platforms for distributed and parallel analysis of large amounts of data, what their strengths and weaknesses are, when they are best suited to use, and so on. The importance of getting a correct result out of it is big, to get that you need to choose the right platforms for your specific needs. We have seen many available platforms and projects on the market, some of them established and some of them in early stages. Our literature study got quite extensive because of the vast market about Big Data. We almost felt overwhelmed at first, knowing when to draw the line and how to categorize different sections working with and analyzing Big Data is not easy, but thanks to our extensive research we did collect a lot of information and could systematically rinse the information in order to form our thesis. Analyses of this kind are very common in many industries, not as extensive as this thesis though. Big Data is a complex and fast growing concept with many technologies fighting for market shares. As the thesis title and it's content imply, it is a jungle. We have seen a lot of analyses and mapping of the world of Big Data, but they are quite narrow. The need for choosing the correct tool for a specific scenario is important and by using the articles and websites available today you need to do what we have done, dig, sort and collect information from many different resources. We have gathered all of this in one place, in this thesis. It is a time consuming task to do this so by using our thesis people can save precious time and focus on what's truly important, taking advantage of available frameworks and tools. It has been quite difficult to compare our thesis results with any other because there isn't any that are mentioning and analyzing as many tools as we are.

Our thesis is first of all aiming for people with an education in Software Engineering or other relevant subjects. The targeted age group is 19-65, so it is quite wide.

The thesis can be used as a foundation/introduction for people that want to start working with Big Data or people that want to just learn the basics of Big Data. It is also working as a first step into a deeper analysis for people that want to continue analyzing and learning more of the Big Data world.

To answer our research questions we have researched relevant literature and analyzed the contents to get a good picture of the different platforms. We have summarized strengths and weaknesses in Table 4 as a partly answer to RQ1: *What are the functionality, strengths and weaknesses of the most common platforms for analysis of Big Data? I.e in what situations can they be used and what type of analysis are they best respectively worst suited for?* However, we couldn't find any specific weaknesses for four of the frameworks and that's why these cells in the table are empty (row 5 - column 3, row 9 - cell 3, row 10 - cell 3, row 12 - cell 3). Together with Table 3 and 5, and paragraphs from *4.3.2 Solution proposals and their areas* and *5.1 Analyses* we do get a complete answer to RQ1: *What are the functionality, strength and weaknesses of the most common platforms for analysis of Big Data?* Table 13 gives an overview of our conclusion and result.

| | Best suited for | Worst suited for | Functionality |
|---|---|---|---|
| **Apache Hadoop** | - Solutions with demands on scalability, distributed computing or data storage | - Iterative Algorithms | |
| **Apache Pig** | - Projects with large amounts of data<br>- People with demand of a low learning curve | - Translating an existing function heavy solution. Only few common functions is existing | - MapReduce<br>- Parallel data processing<br>- Iterative computations<br>- Querying |
| **Apache Spark** | - When high speed is a demand<br>- Projects that don't have a clear picture of which tool to use | - Projects in nead of a load balancer | - MapReduce<br>- Parallel data processing<br>- Stream processing<br>- Querying |
| **Disco** | - Projects that demands preserving of data-locality | - Networks with constant growing cluster | - MapReduce<br>- Distributed data processing<br>- Querying |
| **Misco** | - Processing efficiency<br>- When there are high demands on fast downloading | | - MapReduce<br>- Parallel data processing |
| **MR-MPI** | - Projects in need of fast MapReduce jobs | - Projects with demands of good fault/error handling | - MapReduce<br>- Parallel data processing<br>- Iterative computations |
| **Dryad** | - Programmers demanding flexability<br>- Projects that values speed | - Projects in need of in-memory computations | - Parallel data processing<br>- Iterative computations<br>- Querying |
| **HPPC Systems** | - Projects with a scope of raw data processing, high-performance queries and analyzing of data. | - Windows systems, no support | - Parallel data processing<br>- Querying |
| **Apache Storm** | - Programmers that values speed and reliability | | - Stream processing |
| **Sector/Sphere** | - Projects with demand on speed | | - Parallel data processing<br>- Iterative computations |

| | | | |
|---|---|---|---|
| **GridGain** | - Projects without infrastructure | - Solutions with the need of a load balancer | - MapReduce<br>- Parallel data processing |
| **GraphLab** | - Projects working with graph algorithms<br>- Projects focusing on machine learning | | - Parallel data processing<br>- Iterative computations |
| **Cassandra** | - Huge amount of data storing<br>- Projects that prioritize fast writes<br>- Solutions in need of scalability | - JOINs in databases<br>- Projects that don't want to have unique keys | |
| **Hbase** | - Projects with the need of real-time access<br>- Where disk usage is important<br>- Demanding high scalability | - Projects in need of JOINs<br>- Where high security is a demand | |
| **CouchDB** | - Projects that prioritize fast writes<br>- High scalable solutions<br>- Solution working with JSON | - Projects that prioritize fast reads<br>- Projects in need of complex queries | |
| **MongoDB** | - Projects with demands on dynamic queries<br>- Projects with need of high write performance<br>- Projects that need to use load balancing | - Projects in need of JOINs<br>- Projects focusing on memory usage | |
| **Redis** | - Projects that are using different formats<br>- When you are in need of fast reads<br>- Projects that needs to back data to disk | - Projects in need of simple configurations<br>- Where memory fragmentation are crucial | |
| **Aerospike** | - Projects where queries should be easy to set up<br>- Projects that needs fast access to data by key<br>- Projects where reads are heavy | - Projects where data loss is not an option | |
| **Neo4J** | - Solutions where fast reads is crucial<br>- Projects with graph structure<br>- Where a webadmin should be self-contained | - Projects where tables shuld be easy to model | |
| **OrientDB** | - Projects where graphs is a high priority<br>- Projects where writes is a high priority | - Projects in need of JOINs | |

*Table 13 Summarized results*

To make sure that we could answer RQ2: *Does our thesis meet the goal to work as a guide in the Big Data jungle?* we have created a survey for current and previous students to participate in. They answered some questions to give us an idea about their knowledge in the subject after they read our thesis summary.

## 6.2 Future work

We would have wanted to see if an added alternative to the survey questions, called “I don’t know” would have generated a different result from the survey. Sadly enough we didn’t think of this alternative until the survey was completed and after that we simply didn’t have enough time to redo it. It would be interesting if a survey or test would be created in the future that focused on problem solving using the different platforms and projects explained and analyzed in this thesis.

There will be a need of continuous updates in this area as there is constant development in technologies, and new platforms keep popping up all the time. But it is not just updates that needs to be considered. We want to see future students, scientists or just people with computer science as a hobby utilizing the information we have collected in this thesis to take other steps. Performing experiments with the different frameworks and comparing their performance is a natural step to take from here. The experiments could be in high level and focused on functionality, comparing frameworks with similar functionalities as shown in *Table 3 Summary of functionality, frameworks.* Another step could be to show the summary of the thesis publicly. Then perform a survey or interviews in order to see if anyone outside the computer science world can learn what Big Data is and what tools you need. In this survey or interview a more evaluating section about if it is inspiring and if our thesis have opened a new and exciting world.

There is also different hardware to take into consideration when analyzing large amounts of data, as well as a variety of distributed file systems, which is not covered in this thesis. These areas are important and we would like to see a guide that is focusing on maybe both areas or at least one of them.

# Glossary

### 2.1.2 Cloud computing

To face the increasing amount of data a lot of people choose a cloud architecture. Cloud computing is when you have your software deployed on several different virtual machines that are working together by using online access between each other. (Ji. C, Li. Y, Qui. W, Awada. U, Li. K, 2012)[14]

### 2.1.3 Data mining

“The process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cut costs, or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.” (Palace. B, 1996)[25]

### 2.1.4 Big Data Analytics

To find the most important data and to better understand the information within Big Data organizations use Big Data analytics which is a process of collecting, organizing and analyzing large sets of data. This process is used to find patterns and other useful information. (Beal. V, 2014)[32]

### 2.1.5 Scalability

Scalability is when a system is able to handle a steady increasing amount of computation load. If you have a growing collection of data on a machine and process is getting noticeable slow, then adding machines to same computation can make it faster. (Gorton, 2014)[13]

### 2.1.6 Horizontal scaling platforms

This is when you scale out a cluster with new computers (nodes) as an example (Wikipedia, 2015)[46].

### 2.1.7 Vertical scaling platforms

This is when you are scaling up a node, like adding memory to a computer in the cluster (Wikipedia, 2015)[46].

# References


[1]Quora. (2015) What were the Top 10 problems in Machine Learning for 2013? [Online] Available from: http://www.quora.com/What-were-the-Top-10-problems-in-Machine-Learning-for-2013. [Accessed: 20th February 2015].

[2] Caldarola. E. G, Picariello. A (2015) Modern Enterprises in the Bubble: Why Big Data Matters. [Online] Available from: http://dl.acm.org/. [Accessed: 10th April 2015].

[3] Levine. P. (2015) Machine Learning + Big Data. [Online] Available from: http://a16z.com/2015/01/22/machine-learning-big-data/. [Accessed: 20th February 2015].

[4] Abramova. V, Bernardino. J, Furtado. P. (2014) Which NoSQL Database? A Performance Overview. [Online] Available from: https://www.ronpub.com/publications/OJDB-v1i2n02_Abramova.pdf. [Accessed: 19th May 2015].

[5] Pig Wiki. (2009) [Online] Available from: https://wiki.apache.org/pig/PigOverview. [Accessed: 19th May 2015].

[6] Apache Phoenix (2015) [Online] Available form: http://phoenix.apache.org/. [Accessed: 16th May 2015].

[7] Rose. J, Carzaniga. A. (2008) Plasma: a graph based distributed computing model. [Online] Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.183.933&rep=rep1&type=pdf. [Accessed: 16th May 2015].

[8] Apache Flink (2015) [Online] Available from: https://flink.apache.org/. [Accessed: 15th May 2015].

[9] Samovsky. M, Kacur. T. (2012) Cloud-based classification of text documents using the Gridgain platform. [Online] Available from: http://ieeexplore.ieee.org. Accessed: 14th May 2015].

[10] Zhang. H, Tudor. B. M, Chen. G, Ooi. B. C. (2014) Efficient In-memory Data Management: An Analysis. [Online] Available from: http://dl.acm.org/. [Accessed: 15th May 2015].

[11] Tobey. A. P, Harper. T. (2014) Using Spark Streaming for High Velocity Analytics on Cassandra. [Online] Available from: https://spark-summit.org/2014/talk/using-spark-streaming-for-high-velocity-analytics-on-cassandra. [Accessed: 18th May 2015].

[12] Redis. (2015) Introduction to Redis. [Online] Available from: http://redis.io/topics/introduction. [Accessed: 15th May 2015].

[13] Gortin. I. (2014) Four Principles of Engineering Scalable, Big Data Software Systems. [Online] Available from: http://blog.sei.cmu.edu/post.cfm/four-principles-engineering-big-data-systems-195. [Accessed: 19th February 2015].

[14] Ji. C, Li. Y, Qui. W, Awada. U, Li. K. (2012) Big Data Processing in Cloud Computing Environment. [Online] Available from:

http://www.researchgate.net/publication/235933401_Big_Data_Processing_in_Cloud_Computing_Environments. [Accessed: 19th February 2015].
[15] Dede. E, Govindaraju. M, Gunter. D, Canon. R. S, Ramakrishnan, L. (2013) Performance Evaluation of a MongoDB and Hadoop Platform for Scientific Data Analysis. [Online] Available from: http://dl.acm.org/. [Accessed: 15th May 2015].
[16] MongoDB. (2015) Introduction to MongoDB. [Online] Available from: http://docs.mongodb.org/manual/core/introduction/. [Accessed: 15th May 2015].
[17] Couchbase. (2014) Couchbase vs. Apache CouchDB. [Online] Available from: http://www.couchbase.com/couchbase-vs-couchdb. [Accessed: 15th May 2015].
[18] Eken. S, Kaya. F, Illhan. Z, Sayar. A, Kavak. A, Kocasarac. U, Sahin. S. (2013) Analyzing Distributed File Synchronization Techniques for Educational Data. [Online] Available from: http://ieeexplore.ieee.org. [Accessed: 15th May 2015].
[19] Dobre. C, Xhafa. F. (2013) Parallel Programming Paradigms and Frameworks in Big Data Era. [Online] Available from: http://www.springer.com/gp/. [Accessed: 22nd February 2015].
[20] Apache Cassandra. (2015) Cassandra. [Online] Available from: http://cassandra.apache.org/. [Accessed: 15th May 2015].
[21] Kumar. R, Salian. S. S, Nayak. S. (2014) Comparative Analysis of Various Techniques Used In Managing Big Data. [Online] Available from: http://www.ijircce.com/upload/2014/sacaim/28_Paper%2011.pdf. [Accessed: 17th May 2015].
[22] Sarnovsky. M, Ulbrik. Z. (2013) Cloud-based clustering of text documents using the GHSOM algorithm on the GridGain platform. [Online] Available from: http://ieeexplore.ieee.org. [Accessed: 14th May 2015].
[23] Zaharia. M, Chowdhury. M, Franklin. M.J, Shenker. S, Stoica. I. (2010) Spark: Cluster Computing with Working Sets. [Online] Available from: http://static.usenix.org/legacy/events/hotcloud10/tech/full_papers/Zaharia.pdf. [Accessed: 21st February 2015].
[24] Kuhlenkamp. J, Klems. M, Röss. O. (2014) Benchmarking Scalability and Elasticity of Distributed Database Systems. [Online] Available from: http://dl.acm.org/. [Accessed: 14th May 2015].
[25] Palace. B. (1996) Data Mining: What is Data Mining? [Online] Available from: http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/datamining.htm. [Accessed: 22nd March 2015].
[26] Grolinger. K, Hayes. M, Higashino. W A, L'Heureux. A, Allison. D S, Capretz. M A.M. (2014) Challenges for MapReduce in Big Data. [Online] Available from: http://ieeexplore.ieee.org.miman.bib.bth.se/Xplore/home.jsp. [Accessed: 22nd March 2015].
[27] Pandey. S, Tokekar. V. (2014) Prominence of MapReduce in BIG DATA Processing. [Online] Available from: http://ieeexplore.ieee.org.miman.bib.bth.se/Xplore/home.jsp. [Accessed. 22nd March 2015].
[28] Abramova. V, Bernardino. J. (2013) NoSQL Databases: MongoDB vs Cassandra. [Online] Available from: http://dl.acm.org/. [Accessed: 14th May 2015].
[29] MapReduce-MPI (2015) [Online] Available from: http://mapreduce.sandia.gov/features.html. [Accessed: 14th May 2015].
[30] Techtarget. (2014) Columnar database definition. [Online] Available from: http://searchdatamanagement.techtarget.com/definition/columnar-database. [Accessed: 14th May 2015].

[31]Wang. Y, Goldstone. R, Yu. W, Wang. T. (2014) Characterization and Optimization of Memory-Resident MapReduce on HPC Systems. [Online] Available from: http://ieeexplore.ieee.org.miman.bib.bth.se/Xplore/home.jsp. [Accessed:23rd March 2015].
[32] Beal. V. (2014) Big Data Analytics. [Online] Available from: http://www.webopedia.com/TERM/B/big_data_analytics.html. [Accessed: 5th April 2015].
[33] McAfee. A, Brynjolfsson. E. (2012) Big Data: The Management Revolution. [Online] Available from: https://hbr.org/2012/10/big-data-the-management-revolution/ar. [Accessed: 5th April 2015].
[34] Schömig. M, Neuhäuser. D, Seidler. R, Bücker. M. H. (2014) Benchmarking Different MapReduce Implementations for Computer-Aided Hardware Development. [Online] Available from: http://ieeexplore.ieee.org. [Accessed: 14th May 2015].
[35] Singh. D, Reddy. C. K. (2014) A survey on platforms for Big Data analytics. [Online] Available from: http://www.journalofbigdata.com/content/2/1/8. [Accessed: 2nd April 2015].
[36] Dou. A. J, Kalogeraki. V, Gunopulos. D, Mielikainen. T, Tuulos. V. H. (2011) Scheduling for Real-Time Mobile MapReduce Systems. [Online] Available from: http://dl.acm.org/. [Accessed: 14th May 2015].
[37] Wikipedia. (2014) Dryad(programming). [Online] Available from: http://en.wikipedia.org/wiki/Dryad_%28programming%29. [Accessed: 2nd April 2015].
[38] Dou. A. J, Kalogeraki. V, Gunopulos. D, Mielikainen. T, Tuulos. V. H. (2010) Misco: A MapReduce Framework for Mobile Systems*. [Online] Available from: http://dl.acm.org/. [Accessed: 14th May 2015].
[39] Fuad. A, Erwin. A, Ipung. H.P. (2014) Processing Performance on Apache Pig, Apache Hive and MySQL Cluster. [Online] Available from: http://ieeexplore.ieee.org.miman.bib.bth.se/stamp/stamp.jsp?tp=&arnumber=7010600. [Accessed: 6th April 2015].
[40] Saranya. R. L, Sujatha. P. (2012) Data searching and implementing fault tolerance using sector sphere file system. [Online] Available from: http://ieeexplore.ieee.org. [Accessed: 14th May 2015].
[41] Gu. Y, Grossman. R. (2011) Toward Efficient and Simplified Distributed Data Intensive Computing. [Online] Available from: http://ieeexplore.ieee.org. [Accessed: 14th May 2015].
[42] Hortonworks. (2012) Big Graph Data on Hortonworks Data Platform. [Online] Available from: http://hortonworks.com/blog/big-graph-data-on-hortonworks-data-platform/. [Accessed: 14th May 2015].
[43] ArangoDB. (2015) Key Features. [Online] Available from: https://www.arangodb.com/key-features/. [Accessed: 14th May 2015].
[44] Pimentel. S. (2015) The rise of the multi model database. [Online] Available from: http://www.infoworld.com/article/2861579/database/the-rise-of-the-multimodel-database.html. [Accessed: 14th May 2015].
[45] Marzi. M. D. (2012) Introduction to Graph Databases. [Online] Available from: http://www.slideshare.net/maxdemarzi/introduction-to-graph-databases-12735789. [Accessed: 14th May 2015].
[46] Wikipedia. (2015) Horizontal and vertical scaling. [Online] Available from: http://en.wikipedia.org/wiki/Scalability#Horizontal_and_vertical_scaling. [Accessed: 6th April 2015].

[47] Seeger. M. (2009) Key-Value stores: a practical overview. [Online] Available from: http://www.slideshare.net/marc.seeger/keyvalue-stores-a-practical-overview. [Accessed: 13th May 2015].
[48] Apache Oozie. (2011) [Online] Available from: http://wiki.apache.org/incubator/OozieProposal. [Accessed: 21st May 2015].
[49] Apache Flume. (2015) [Online] Available from: http://hortonworks.com/hadoop/flume/. [Accessed: 21st May 2015].
[50] Baeza-Yates. R. (2013) Big Data or Right Data? [Online] Available from: http://ceur-ws.org/Vol-1087/paper14.pdf. [Accessed: 7th April 2015].
[51] DeCandia. G, Hastorun. D, Jampani. M, Kakulapati. G, Lakshman. A, Pilchin. A, Sivasubramanian, Vosshall. P, Vogels. W. (2007) Dynamo: amazon's highly available key-value. [Online] Available from: http://dl.acm.org/. [Accessed: 21st May 2015].
[52] Lerman. J. (2011) What the Heck Are Document Databases? [Online] Available from: https://msdn.microsoft.com/en-us/magazine/hh547103.aspx. [Accessed: 13th May 2015].
[53] Rudra. S. (2012) Column Based Database. [Online] Available from: http://www.slideshare.net/srudra25/rise-of-column-oriented-database. [Accessed 13th May 2015].
[54] NoSQL. (2015) NoSQL. [Online] Available from: http://nosql-database.org/. [Accessed 13th May 2015].
[55] Gandomi. A, Haider. M. (2014) Beyond the hype: Big data concepts, methods, and analytics. [Online] Available from http://www.sciencedirect.com/science/article/pii/S0268401214001066. [Accessed: 11th April 2015].
[56] Liu. T, Liu. J, Liu. H, Li. W. (2013) A performance evaluation of Hive for Scientific Data Management. [Online] Available from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6691696. [Accessed: 7th May 2015].
[57] Isard. M, Budio. M, Yu. Y, Birell. A, Fetterly. D. (2007) Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks. [Online] Available from: http://research.microsoft.com/pubs/63785/eurosys07.pdf. [Accessed: 7th May 2015].
[58] Nair. S, Mehta. J. (2011) Clustering with Apache Hadoop. [Online] Available from: http://dl.acm.org/. [Accessed: 7th May 2015].
[59] Wikipedia. (2015) Distributed Computing. [Online] Available from: http://en.wikipedia.org/wiki/Distributed_computing. [Accessed: 7th May 2015].
[60] Zookeeper. A. (2015) [Online] Available from: http://zookeeper.apache.org/. [Accessed: 7th May 2015].
[61] Cai. Z, Logothetis. D, Siganos. D. (2012) Facilitating real-time graph mining. [Online] Available from: http://dl.acm.org/. [Accessed on: 7th May 2015].
[62] Zhang. Y, Gao. Q, Gao. L, Wang. C. (2012) Accelerate large-scale iterative computation through asynchronous accumulative updates. [Online] Available from: http://dl.acm.org/. [Accessed: 7th May 2015].
[63] Low. Y, Gonzales. J, Kyrola. A, Bickson. D, Guestrin. C, Hellerstein. J. (2014) GraphLab: A New Framework For Parallel Machine Learning. [Online] Available from: http://arxiv.org/ftp/arxiv/papers/1408/1408.2041.pdf. [Accessed: 7th May 2015].
[64] Middleton. A. M. (2011) HPCC Systems: Introduction to HPCC (High-Performance Computing Cluster). [Online] Available from:

http://cdn.hpccsystems.com/whitepapers/wp_introduction_HPCC.pdf. [Accessed: 8th May 2015].
[65] Lynden. S, Tanimura. Y, Kojima. I, Matono. A. (2001) Dynamic Data Redistribution for MapReduce Joins. [Online] Available from: http://ieeexplore.ieee.org. [Accessed: 8th May 2015].
[66] Vernica. R, Balmin. A, Beyer. K. S, Ercegovac. V (2012) Adaptive MapReduce using Situation-Aware Mappers. [Online] Available from: http://dl.acm.org/. [Accessed: 9th May 2015].
[67] Asay. M. (2014) MongoDB, Cassandra, and HBase -- the three NoSQL databases to watch. [Online] Available from: http://www.infoworld.com/article/2848722/nosql/mongodb-cassandra-hbase-three-nosql-databases-to-watch.html. [Accessed 13th May 2015].
[68] Lin. J, Ryaboy. D. (2013) Scaling Big Data Mining Infrastructure: The Twitter Experience. [Online] Available from: http://dl.acm.org/. [Accessed 12th May 2015].
[69] Im. D-H, Cho. C-H, Jung. I. (2014) Detecting Large Number of Objects in Real-time Using Apache Storm. [Online] Available from: http://ieeexplore.ieee.org. [Accessed: 10th May 2015].
[70] Du. Y, Liu. J, Liu. F, Chen. L. (2014) A Real-time Anomalies Detection System based on Streaming Technology. [Online] Available from: http://ieeexplore.ieee.org. [Accessed: 10th May 2015].
[71] J.P. N, Raj. E. D, Badu. D, Nirmala. M, Kumar. M. (2014) Analysis on Enhancing Storm to Efficiently Process Big Data in Real Time. [Online] Available from: http://ieeexplore.ieee.org. [Accessed: 10th May 2015].
[72] Apache Avro. (2015) Apache Avro. [Online] Available from: https://avro.apache.org/. [Accessed: 10th May 2015].
[73] Smith. K, Seligman. A, Rosenthal. A, Kurcz. C, Greer. M, Macheret. C, Sexton. M, Eckstein. A. (2014) “Big Metadata”: The Need for Principled Metadata Management in Big Data Ecosystems. [Online] Available from: http://dl.acm.org/. [Accessed: 10th May 2015].
[74] Apache HBase. (2015) Apache HBase. [Online] Available from: http://hbase.apache.org/. [Accessed: 10th May 2015].
[75] Bhupathiraju. V, Ravuri. R. P. (2014) The Dawn of Big Data - Hbase. [Online] Available from: http://ieeexplore.ieee.org. [Accessed: 10th May 2015].
[76] Vora. N. M. (2011) Hadoop-HBase for Large-Scale Data. [Online] Available from: http://ieeexplore.ieee.org. [Accessed: 10th May 2015].
[77] Apache Sqoop. (2015) Apache Sqoop. [Online] Available from: http://sqoop.apache.org/. [Accessed: 10th May 2015].
[78] Pippal. S, Singh. S. P, Singh. D. (2014) Data Transfer From MySQL To Hadoop: Implementers’ Perspective. [Online] Available from: http://dl.acm.org/. [Accessed: 10th May 2015].
[79] Apache Flume. (2015) Apache Flume. [Online] Available from: https://flume.apache.org/. [Accessed: 10th May 2015]
[80] Wang. C, Rayan. I. A, Schwan. K. (2012) Faster, Larger, Easier: Reining Real-Time Big Data Processing in Cloud. [Online] Available from: http://dl.acm.org/. [Accessed: 10th May 2015].
[81] Apache Chukwa. (2015) Apache Chukwa. [Online] Available from: https://chukwa.apache.org/. [Accessed: 11th May 2015].

[82] Shang. W, Jiang. Z. M, Hemmati. H, Adams. B, Hassan. A. E, Martin. P. (2013) Assisting developers of big data applications when deploying on hadoop clouds. [Online] Available from: http://dl.acm.org/. [Accessed: 11th May 2015].
[83] Sagiroglu. S, Sinanc. D. (2013) Big Data: A review. [Online] Available from: http://ieeexplore.ieee.org. [Accessed: 11th May 2015].
[84] Apache Oozie. (2014) Apache Oozie. [Online] Available from: http://oozie.apache.org/. [Accessed: 11th May 2015].
[85] Islam. M, Huang. A. K, Battisha. M, Chiang. M, Srinivasan. S, Peters. C, Neumann. A. (2012) Oozie: Towards a Scalable Workflow Management System for Hadoop. [Online] Available from: http://dl.acm.org/. [Accessed: 11th May 2015].
[86] Geyik. S. C, Saxena. A, Dasdan. A. (2014) Multi-Touch Attribution Based Budget Allocation in Online Advertising. [Online] Available from: http://dl.acm.org/. [Accessed: 11th May 2015].
[87] Apache Mahout. (2015) [Online] Available from: http://mahout.apache.org/. [Accessed: 12th May 2015].
[88] Moh. T-S, Bhagvat. S. (2012) Clustering of Technology Tweets and the Impact of Stop Words on Clusters. [Online] Available from: http://dl.acm.org/. [Accessed: 12th May 2015].
[89] Walunj. Mr. S, Sadafale. Mr. K. (2013) An Online Recommendation System for E-commerce Based on Apache Mahout Framework. [Online] Available from: http://dl.acm.org/. [Accessed: 12th May 2015].
[90] Papadimitriou. S, Sun. J. (2008) DisCo: Distributed Co-clustering with Map-Reduce A Case Study Towards Petabyte-Scale End-to-End Mining. [Online] Available from: http://ieeexplore.ieee.org. [Accessed: 12th May 2015].
[91] Shukla. R. K, Pandey. P, Kumar. V. Big Data Frameworks: At a Glance. [Online] Available from: http://academicscience.co.in/admin/resources/project/paper/f201501301422607618.pdf. [Accessed: 12th May 2015].
[92] Bugnion. E, Devine. S, Govil. K, Rosenblum. M. (1997) Disco: Running Commodity Operating Systems on Scalable Multiprocessors. [Online] Available from: http://dl.acm.org/. [Accessed: 12th May 2015].
[93] Islam. N. S, Rahman. M. W, Jose. J, Rajachandrasekar, Wang. H, Subramoni. H, Murthy. C, Panda. D. K. (2012) High performance RDMA-based design of HDFS over InfiniBand. [Online] Available from: http://dl.acm.org/. [Accessed: 17th May 2015].
[94] Shafer. J, Rixner. S, Cox. A. L (2010) The Hadoop Distributed Filesystem: Balancing Portability and Performance. [Online] Available from: http://ieeexplore.ieee.org. [Accessed: 17th May 2015].
[95] HDFS. (2014) [Online] Available from: http://wiki.apache.org/hadoop/ProjectDescription. [Accessed: 17th May 2015].
[96] Leveranz. L. (2013) HCatalog UsingHCat. [Online] Available from: https://cwiki.apache.org/confluence/display/Hive/HCatalog+UsingHCat. [Accessed: 12th May 2015].
[97] Aerospike. (2015) Technology. [Online] Available from: http://www.aerospike.com/technology/. [Accessed: 16th May 2015].
[98] Penchikala. S. (2014) Aerospike NoSQL Database Architecture. [Online] Available from: http://www.infoq.com/articles/aerospike-qa. [Accessed: 16th May 2015].

[99] Neo4J. (2014) Top Ten Reasons for Choosing Neo4J. [Online] Available from: http://neo4j.com/. [Accessed: 16th May 2015].
[100] Holzschuher. F, Peinl. Prof. Dr. R. (2013) Performance of Graph Query Languages. [Online] Available from: http://dl.acm.org/. [Accessed: 16th May 2015].
[101] Kozhanov. E. (2013) Use Neo4J for search transport routes. [Online] Available from: http://www.slideshare.net/EugenyKozhanov/how-we-use-neo4j-fortransport. [Accessed: 16th May 2015].
[102] Apache Hive. (2015) [Online] Available from: https://cwiki.apache.org/confluence/display/Hive/GettingStarted#GettingStarted-Requirements. [Accessed: 17th May 2015].
[103] OrientDB. (2014) [Online] Available from: http://orientdb.com/orientdb/. [Accessed: 17th May 2015].
[104] Barmpis. K, Kolovos. D. S. (2012) Comparative Analysis of Data Persistence Technologies for Large-Scale Models. [Online] Available from: http://dl.acm.org/. [Accessed: 17th May 2015].
[105] Apache Pig. (2015) [Online] Available from: http://pig.apache.org/docs/r0.14.0/udf.html. [Accessed: 17th May 2015].
[106] Amodeo. L. (2014) BIG DATA 6V: VOLUME, VARIETY, VELOCITY, VARIABILITY, VERACITY, COMPLEXITY. *The Journey.* [Online] Available from: https://wydata.wordpress.com/2014/12/24/big-data-volume-variety-velocity-variability-veracity-complexity/. [Accessed: 7th June 2015].
[107] ACM digital library. [Online] Available from: http://dl.acm.org/. [Accessed: 7th June 205].
[108] SIGSOFT. [Online] Available from: http://www.sigsoft.org/. [Accessed: 7th June 2015].
[109] IEEE Xplore Digital Library. [Online] Available from: http://ieeexplore.ieee.org/Xplore/home.jsp. [Accessed: 7th June 2015].
[110] BTH Summon. [Online] Available from: http://www.bth.se.miman.bib.bth.se/bib/databaser.nsf/search.xsp/database.xsp. [Accessed: 7th June 2015].
[111] Google. [Online] Available from: https://google.com. [Accessed: 7th June 2015].
[112] Google Research. [Online] Available from: https://research.google.com/. [Accessed: 7th June 2015].
[113] Google Scholar. [Online] Available from: https://scholar.google.se/. [Accessed: 7th June 2015].
[114] Mitchell. R.L. (2014) 8 big trends in big data analytics. [Online] Available from: http://www.computerworld.com/article/2690856/8-big-trends-in-big-data-analytics.html. [Accessed: 7th June 2015].
[115] PCMag. (2015) Encyclopedia. [Online] Available from: http://www.pcmag.com/encyclopedia/term/52162/structured-data. [Accessed: 7th June 2015].
[116] Wikipedia. (2015) Unstructured data. [Online] Available from: http://en.wikipedia.org/wiki/Unstructured_data. [Accessed: 7th June 2015].
[117] Informationweek. (2014) Multi-Structured Data: The New Innovation Opportunity. [Online] Available from: http://www.informationweek.com/big-data/big-data-analytics/multi-structured-data-the-new-innovation-opportunity/a/d-id/1317838. [Accessed: 7th June 2015].

[118] Spark. (2015) [Online] Available from: https://spark.apache.org. [Accessed: 7th June 2015].
[119] Disco Project. (2015) [Online] Available from: http://discoproject.org/. [Accessed: 7th June 2015].
[120] Hadoop. (2015) Hadoop: Setting up a single node cluster. [Online] Available from: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/SingleCluster.html. [Accessed: 7th June 2015].
[121] Hive. (2015) [Online] Available from: https://hive.apache.org/. [Accessed: 7th June 2015].
[122] Microsoft Research. (2015) Dryad. [Online] Available from: http://research.microsoft.com/en-us/projects/dryad/. [Accessed: 7th June 2015].
[123] Wikipedia. (2015) BigTable. [Online] Available from: http://en.wikipedia.org/wiki/BigTable. [Accessed: 7th June 2015].
[124] JRuby. (2015) [Online] Available from: http://jruby.org/. [Accessed: 7th June 2015].
[125] Apache Pig. (2015) Welcome to Apache Pig! [Online] Available from: https://pig.apache.org/. [Accessed: 7th June 2015].
[126] HPCC Systems. (2015) [Online] Available from: http://hpccsystems.com/. [Accessed: 7th June 2015].
[127] MongoDB. (2013) [Online] Available from: http://www.mongodb.org/about/introduction/. [Accessed: 9th June 2015].
[128] OrientDB. (2015) [Online] Available from: http://orientdb.com/why-orientdb/. [Accessed: 9th June 2015].
[129] Apache Hadoop. (2014) [Online] Available from: https://hadoop.apache.org/. [Accessed: 9th June 2015].
[130] Oracle. (2015) Data Warehousing and Big Data [Online] Available from: http://www.oracle.com/technetwork/database/bi-datawarehousing/overview/index.html. [Accessed: 14th June 2015].
[131] SQL. (2015) [Online] Available from: https://en.wikipedia.org/?title=SQL. [Accessed: 15th June 2015]

# Appendix A

---

## Summary for survey

It is clear that developers, scientists and companies find the world of Big Data complex and vast. If you break it down to sub-categories and connect frameworks/projects to different areas that they belong to you see a more narrow and easy perspective.

We have divided this big subject into five main parts; Hadoop projects, frameworks, MapReduce wrappers, database structure and databases, because these are the most relevant and crucial parts to consider.

Big Data is the collection of massive amounts of information, whether unstructured or structured. Companies around the world collects data which holds results of discoveries and analysis, this is as mentioned above the collection of information in massive amounts. It is the size, diversity and the complexity that make Big Data. These characteristics make it hard to process to extract value from it in a traditional way. It is usually described as high in:

- Volume - The size of the data. The large size can come from years of storing or a constant data stream from social media, for example. This volume calls for scalable storage and a distributed approach to processing combined together.
- Variety - The data can be in many different formats which can cause difficulties when merging or managing it. A common use of big data processing is to take unstructured data and extract ordered meaning.
- Velocity - The speed of generation of data, how fast it is growing. As the data grows the analysis result will change, because of that it is important to be able to handle and analyze the data right away. You wouldn't decide to cross a road from looking at a five minute old snapshot of the traffic situation. The importance of the speed for taking data from input through to decision making is huge.
- Variability - The inconsistency in the data. For example, there might be periodic peaks causing this. The meaning of the data can vary due to different context.
- Veracity - The quality of the data might vary for different reasons. Is the data that are being stored and mined meaningful for the problem being analyzed?
- Complexity - The data can come from different sources and be hard to link, match, cleanse and transform across systems.

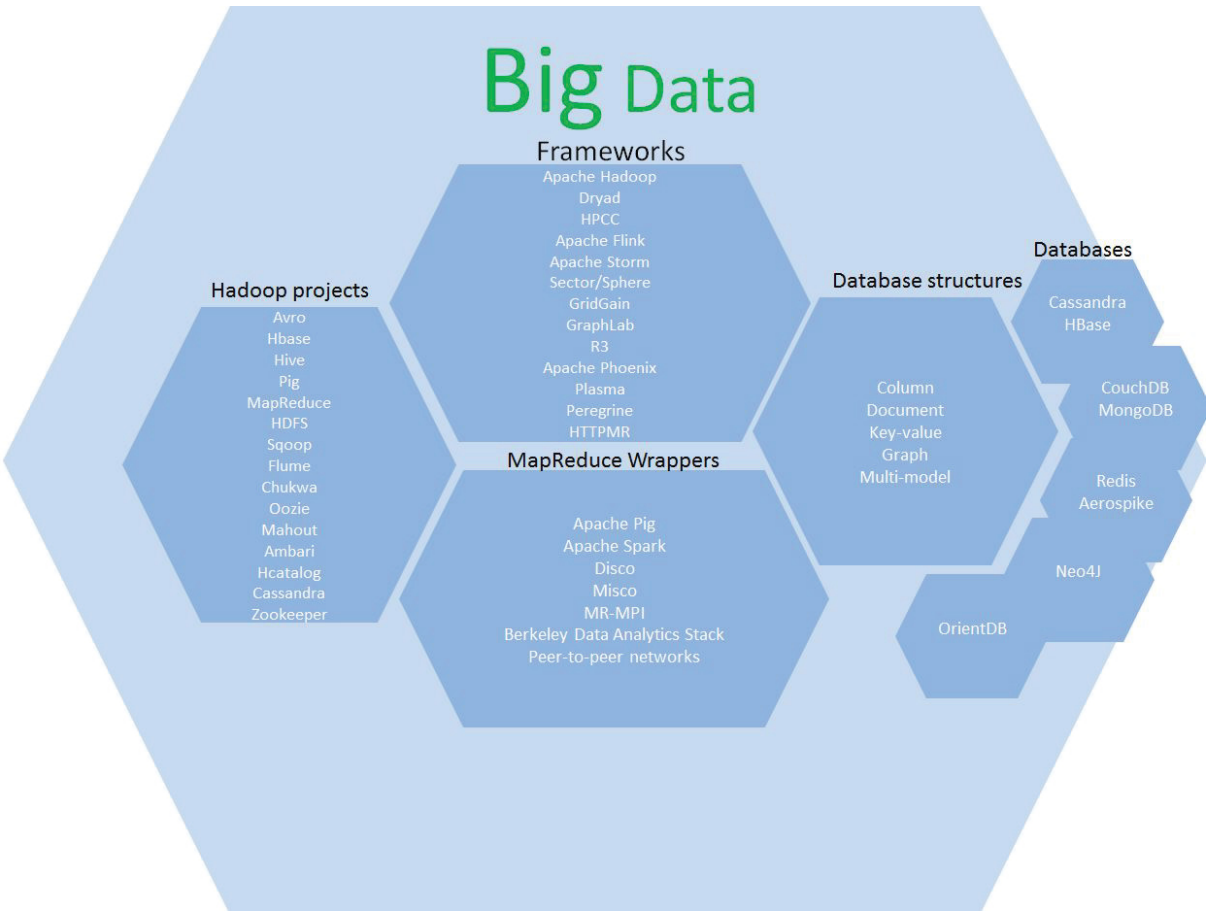


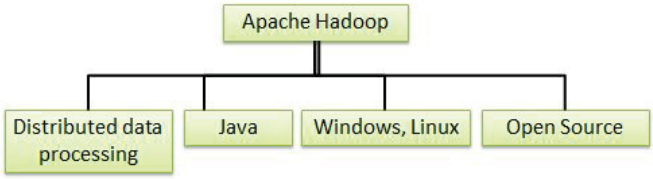

*Figure 1 Overview of platforms*

Two things that plays very big role in the world of Big Data is:

**Apache Software Foundation** is an organization that aims to provide software for the public good by providing services and support for many like-minded software project communities of individuals. Apache has a lot of projects in our focus area.

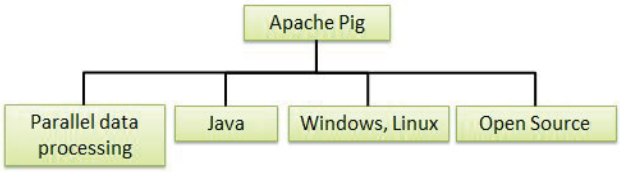
**MapReduce** is a highly scalable programming paradigm developed by Google, which breaks a computation into small tasks that run in parallel on multiple machines, and scales easily to very large clusters of inexpensive commodity computers. The framework consists of two functions; Map and Reduce, which are written by the user to process the data in key/value pairs.

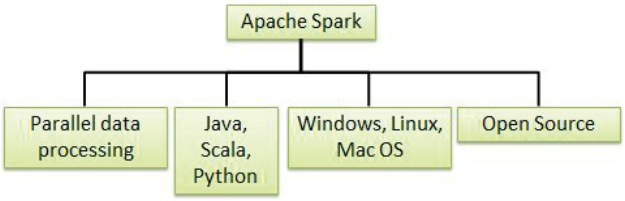

We found a lot of frameworks and MapReduce wrappers that you can use when you are working with Big Data, we selected the most popular ones and some others that are mentioned everywhere but still are alien-territory:

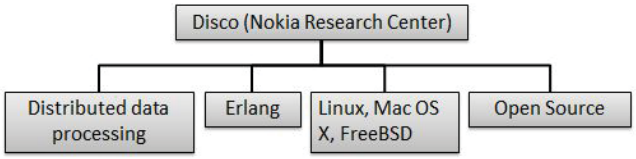


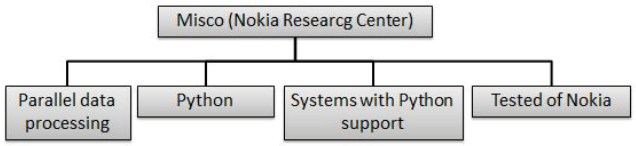

- **Apache Hadoop** is the most common framework for distributed storage and distributed data processing. It had built up a strong base from related subprojects, all hosted by the Apache Software Foundation.

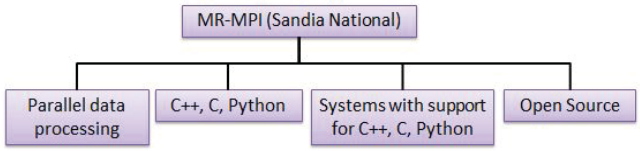
- **Apache Pig** is a tool for analyzing large data sets in a high level language which runs on top of Hadoop. It is easy to learn and runs better on large amounts of data than on small amounts. It comes with a few built in functions which leads to a narrow type of data analytics.
- **Apache Spark** is a cluster computing framework that uses resilient distributed datasets.[18] Some of its main features are its speed, ease of use, high performance and generality.
- **Disco** is a lightweight scalable framework for distributed computing based on MapReduce. It is easy to install and it support multiple common programming languages.
- **Misco** is a distributed computing framework based on MapReduce. Misco spends the majority of the processing time on performing computations, and only a small part is spent on downloading, uploading and cleanup.
- **MR-MPI** (MapReduce MPI) is an implementation of MapReduce written for distributed-memory parallel machines on top of standard MPI[19] message passing. It is callable from most high level programming languages but doesn't have any fault tolerance.
- **Dryad** is a general-purpose distributed execution engine for coarse-grain, data-parallel applications. Dryad is very flexible and allows the user to optimize tradeoffs between parallelism and data distribution overheads.
- **HPCC Systems** is a data-sensitive computing system platform which provides a distributed file system, job execution environment, online query capability, parallel application processing, and parallel programming development tools in a high-performance computing environment.

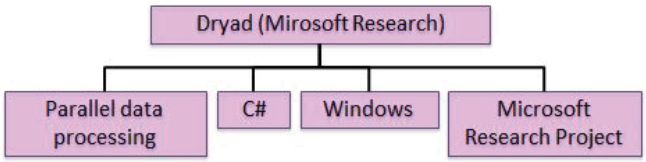


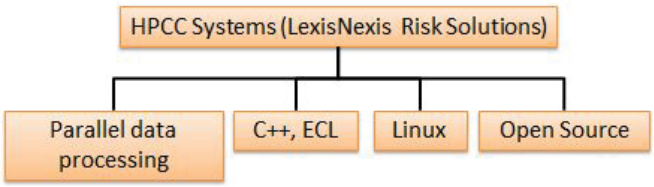



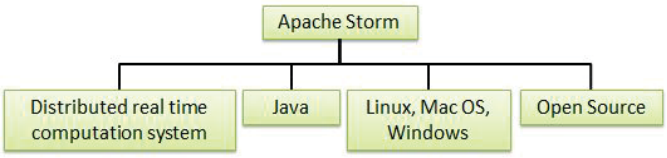



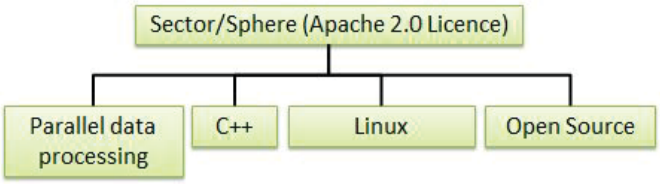




[18] Resilient distributed datasets is a collection of objects that you only can read. Because it is partitioned over the machines that are a part of the cluster it works really fast and is easy to rebuild if a partition is lost.
[19] A standardized and portable message-passing system designed by a group of researchers from academia and industry to function on a wide variety of parallel computers.

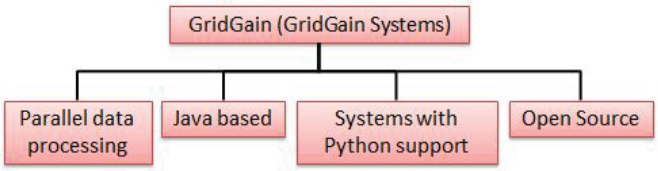
- **Apache Storm** is a low latency stream processing framework which provides real-time analysis and continuous computing. Storm can be used in any programming language and it is fast and reliable.



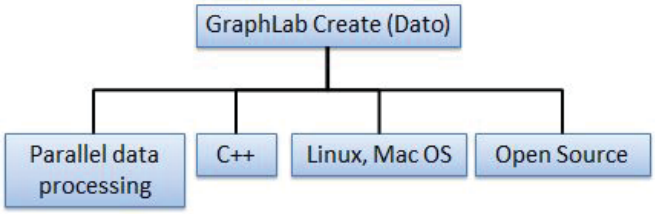

- **Sector/Sphere** consists of two parts; Sector is a scalable, fault tolerant and secure distributed file system with is high performance using commodity computers. Sphere is a parallel in-storage data processing engine that can be used to process the data managed by Sector in parallel.




- **GridGain** is a middleware for development of data processing applications in distributed environments. The framework have to parts; compute grid, which provides in-memory MapReduce implementation, and in-memory data grid, which presents the capability to parallelize the data storage by storing the partitioned data in memory, closer to the application.




- **GraphLab** is an asynchronous parallel framework designed for large-scale graph mining. It is an innovative and flexible programming model especially fitted for graph algorithms. GraphLab improves upon abstractions like MapReduce by compactly expressing asynchronous iterative computations with sparse computational dependencies while ensuring data consistency and achieving a high degree of parallel performance.




Below follows a summary of our findings regarding relevant frameworks that we have examined in our thesis.

*Table 1* gives an overview of the functionality and properties of the frameworks that we have been examined in this thesis. To make the table manageable we selected the functionality that we find most relevant, and that is used for different types of analysis. We created this table as a sort of a summary of our findings regarding those frameworks.

In *table 2* we have pointed out the biggest strengths and weaknesses that we found for each of the frameworks that we have been looking at. We didn't have certain parameters in mind when we created this table so there is no relation between the different frameworks weaknesses and strengths, the purpose of the table is to give a quick view of what a certain framework might be suited for or not.

| | MapReduce | Distributed data processing | Parallel data processing | Data warehousing | Iterative computations | Stream processing | Query | Open source | Windows | Linux | Mac OS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Apache Pig | x | | x | | x | | x | x | x | x | |
| Apache Hive | x | | | x | | | x | x | x | x | x |
| Apache Spark | x | | x | | | x | x | x | x | x | x |
| Disco | x | x | | | | | x | | | x | x |
| Misco | x | | x | | | | | | x | x | x |
| MR-MPI | x | | x | | x | | | x | x | x | x |
| Dryad | | | x | | x | | x | | x | | |
| HPPC Systems | | | x | | | | x | x | | x | |
| Apache Storm | | | | | | x | | x | x | x | x |
| Sector/Sphere | | | x | | x | | | x | | x | |
| GridGain | x | | x | | | | | x | x | x | x |
| GraphLab | | | x | | x | | | x | | x | x |

*Table 1 Summary of functionality, frameworks*

| | Strengths | Weaknesses |
|---|---|---|
| **Apache Hadoop** | ·Has a strong base from related sub projects | ·Iterative algorithms |
| **Apache Pig** | ·Easy to learn<br>·Good performance on large amounts of data<br>·Support for multiple common programming languages | ·Few common functions |
| **Apache Spark** | ·Versatile<br>·High speed<br>·Many tools | ·Workload balancing |
| **Disco** | ·Support for multiple common programming languages<br>·Efficient data-locality-preserving I/O | ·Unreliable when pushing the cluster network to its limit |
| **Misco** | ·Effecient processing,not much time for downloading, uploading and cleanup | |
| **MR-MPI** | ·Fast MapReduce processing | · No fault tolerance<br>·Unstable when calculating $2^{64}$ or more pairs |
| **Dryad** | ·Allows the programmer to optimize tradeoffs between parallelism and data distribution overheads<br>·Flexible | ·In-memory computations |
| **HPPC Systems** | ·Versatile<br>·Fast<br>·Simple | ·Linux is the only supported OS |
| **Apache Storm** | ·Support for multiple common programming languages<br>·Fast<br>·Reliable | |
| **Sector/Sphere** | ·Supports multiple inputs and outputs<br>·Fast | |
| **GridGain** | ·Independent from infrastructure and platform | ·Workload balancing |
| **GraphLab** | ·Graph algorithms<br>·Machine learning | |

*Table 2 Summary of strength and weaknesses, frameworks*

Regarding Big Data frameworks we can see that Apache Hadoop is the most common one, it is used by many developers globally. It has a strong support from community despite numerous of strong competitors. Developers keeps supporting Apache for its open source projects and cross-platform compatibility. By going deeper into frameworks you see

that Apache Hadoop has more subprojects than any other Big Data framework. The majority of searches regarding Big Data frameworks results in Apache Hadoop.

MapReduce ends up high on the list on common Big Data programming models. In our literature study we could see a big market using MapReduce. A lot of projects have developed MapReduce wrappers in order to take advantage of MapReduce and deliver innovative and competitive products. Apache Pig is one of the wrappers. It was more common a couple of years back than it is now, it has more and tougher competition now. It is still used widely by companies and analysis performed not that long ago shows that Pig suits the need of working with huge amount of data.

Our search results showed that Apache Spark is a big competitor on the MapReduce market, but it is still young. Nokia's wrapper called Disco, which other contributors could see some really good scaling results in, has however shown a lot of instability when you are pushing the cluster to its limit. Nokia's Misco turned out to be a good and powerful alternative when you are developing for mobile platforms using Python. MR-MPI shows poor results compared to other MapReduce wrappers, it is very time expensive.

Some of the frameworks have a wide and general purpose and still manage to show really good results, we found that Dryad, HPCC and Sector/Sphere to fall into this category.

When it comes to relevant databases available for working with Big Data we noticed that we should narrow it down to NoSQL databases because of the result when we were collecting data. NoSQL is also known as Not Only SQL which uses dynamic schemas for storage, compared to relational databases which demands that schemas are being defined before a user can add data. Because NoSQL databases don't demand a predefined schema they are ideal for handling large amounts of data. NoSQL databases are divided into five categories: Column, Document, Key-value, Graph and Multi-model.

- **Column** stores tables as columns divided into sections instead of storing data in rows. These types of databases should be used if you wish to compress huge amounts of data. Because it is column based it uses less disk space than most other databases do.
    - **Cassandra** is known for handling big amounts of data, so big that it doesn't fit on server but still has a friendly interface. It works faster with bigger data volumes and we recommend Cassandra for users that think the data volume will grow drastically. Writes is usually much faster than reads and querying is done by key or key-range.
    - **HBase** is ideal to use when you require real-time read/write random access to very large amount of data. It is also compatible with MapReduce programs so that it can operate on the data being stored. HBase scales better when working with larger data volumes, writes faster than reads and queries by pushing down via server side scan.

- **Document** oriented databases are designed for storing, managing and retrieving document-oriented information. It is perfect for when you want to use the document id as key and the document itself as the value.
    - **CouchDB** saves the data in a document as Json-objects which fits perfect for log data. Good to use for collecting, making small changes in data on which pre-defined queries are to be run. Queries grabs data from stored procedures

called views, a view is made of a map function and can have a reduce function. CouchDB has better performance when it comes to reads compared to writes.
    - **MongoDB** builds the data in Json-object-like structure with dynamic schemas called Bson. Dynamic queries, as MongoDB uses, give a deep query-ability. MongoDB is perfect to use when you need high availability in an environment that is unstable and it has high write performance which is good for systems with massive update rates.

- **Key-value** oriented databases are designed for the same purpose as document based are, but key-value applies this on arrays instead of document-oriented information. Many developers implies that key-value databases handles size well and that it constantly processes small reads and writes.
    - **Redis** stores data in-memory with keys of optional durability, but this data can be persisted by being written to disk asynchronously. Querying is done by direct data retrieval commands with no queries or query planners in the middle, performed by its key. Redis performance best when it comes to reads, writes is often referred to as ok.
    - **Aerospike** stores data using SSD-devices as a block device, it can also store in-memory though. Aerospike queries perform lookup through the use of secondary indexes and is based on value. Queries are done with an SQL-like tool called AQL. It has turned out to be optimized for read-heavy systems, but has also shown performance for well balanced systems between read and write.

- **Graph** databases uses graph theory[20] to store, map and query relationships. It is a collection of nodes and edges, nodes are defined by a unique identifier, a set of edges both outgoing and incoming and key-value pairs. Graph databases are used for analyzing interconnections, perfect for social media software.
    - **Neo4J** stores data on-disk and uses an own pattern-matching query language called Cypher which is inspired by SQL - using clauses to build queries. Popular by users who are in need of a self-contained web admin. Neo4J is optimized for reads.

- **Multi-model** is database type that can support multiple data models. Developers having a hard time to choose how to model and store data usually chooses this type of database. You can customize queries by using the different data models, do it as you see fit.
    - **OrientDB** is known as a graph database, however it is a mix with features taken from document oriented databases and from key-value databases. Query language is much like SQL but doesn't have any JOINs. Perfect for graph-style and interconnected data. Equally good on read and writes, but reads are very much depending on how good the cache-memory is.

[20] Involves the ways in which sets of vertices can be connected by lines or edges.

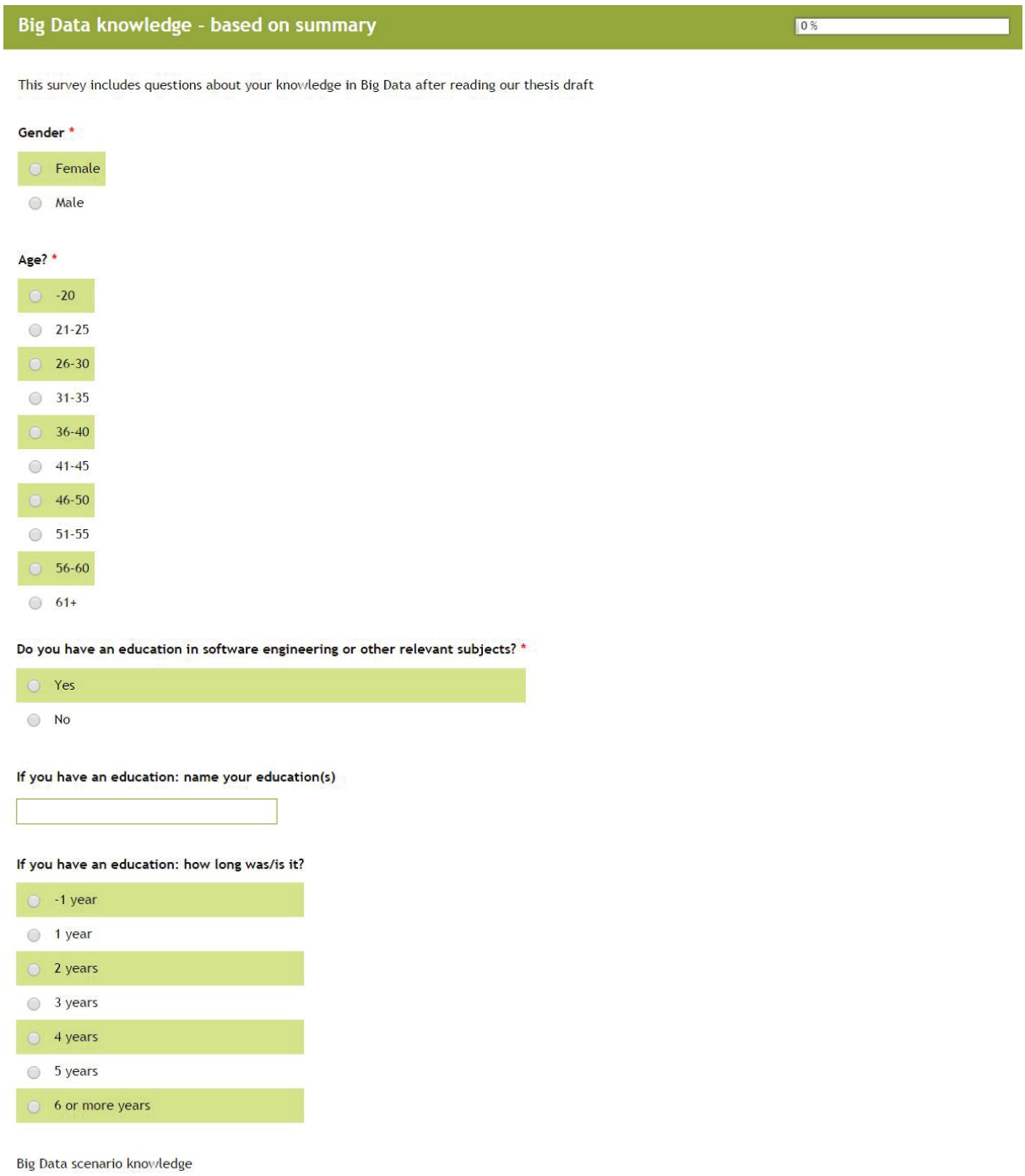
*Table 3 is* a summary of the reviewed databases. It shows language properties, strengths and weaknesses. It is designed in the same way as *table 2*, there is no relation between the different strengths and weaknesses and the purpose of the table is purely to give a quick view of the databases in this thesis.

| | Written language | Query language | Strengths | Weaknesses |
|---|---|---|---|---|
| **Cassandra** | Java | CQL | -Storing huge amount of data<br>-Fast writes<br>-Highly scalable | -No JOINs<br>-Keys have to be unique<br>-Complicated searches |
| **Hbase** | Java | No specific language | -Real-time access<br>-Clever disk-usage<br>-Highly scalable | -Single Point of Failure<br>-No JOINs, only with MapReduce<br>-Security problem |
| **CouchDB** | Erlang | JSON | -Fast reads<br>-Highly scalable<br>-Simplicity with any JSON data | -Slow writes<br>-No complex queries<br>-Spatial compacting needed |
| **MongoDB** | C++, C#, Go | BSON store(binary format JSON) | -Dynamic queries<br>-High write performance<br>-Good load-balancing | -No JOINs<br>-Concurrency issue<br>-Memory usage |
| **Redis** | ANSI C | Extra commands for querying | -Storing in various formats<br>-Fast reads<br>-Can back data to disk | -Complex to configure<br>-Memory fragmentation issue<br>-Whole dataset always "lives" in RAM |
| **Aerospike** | C | AQL | -Easy to set up queries<br>-Very fast access of data by key<br>-Handles heavy reads great | -If a node fails, the data on that node is lost |
| **Neo4J** | Java | Cypher | -Fast reads<br>-Great graph-structure<br>-Self-conatined web admin | -Difficult to model tables<br>-Difficult to query on node contents |
| **OrientDB** | Java | JSON, SQL support | -Uses a good mix of database models<br>-Great with graphs<br>-Great on writes | -No JOINs<br>-Reads is dependent on cache |

*Table 3 Summary of strength and weaknesses, NoSQL databases*

# Appendix B

## Survey: Big Data knowledge - based on summary



**Big Data knowledge - based on summary** 0 %

This survey includes questions about your knowledge in Big Data after reading our thesis draft

**Gender** *

- Female
- Male

**Age?** *

- -20
- 21-25
- 26-30
- 31-35
- 36-40
- 41-45
- 46-50
- 51-55
- 56-60
- 61+

**Do you have an education in software engineering or other relevant subjects?** *

- Yes
- No

**If you have an education: name your education(s)**

**If you have an education: how long was/is it?**

- -1 year
- 1 year
- 2 years
- 3 years
- 4 years
- 5 years
- 6 or more years

Big Data scenario knowledge

**MapReduce can be described as ... ***

- ○ A programming model that makes tasks run in parallel on multiple machines
- ○ A framework that process the data using only key
- ○ A programming model that only works with unstructured data

**These are MapReduce wrappers ***

Choose the once that you know is MapReduce wrappers

- ☐ Apache Avro
- ☐ Apache Pig
- ☐ Apache Spark
- ☐ Google Misco
- ☐ Apache Flume

**The NoSQL categories are ... ***

Choose the once that you know are correct

- ☐ Collection database
- ☐ Column stored
- ☐ Distributed database
- ☐ Documented-oriented database
- ☐ Key-value database
- ☐ Navigational database
- ☐ Relational database

**Big Data can be described with the words ... ***

Choose the once that you know are correct

- ☐ Security
- ☐ Simplicity
- ☐ Variability
- ☐ Variety
- ☐ Velocity
- ☐ Veracity
- ☐ Visuality
- ☐ Vulnerability
- ☐ Volume

**How informative did you find text? ***

Not at all ○ ○ ○ ○ ○ Very

**Do you feel that you have a good overview of the platforms and databases after reading? ***

Not at all ○ ○ ○ ○ ○ Very

**Do you feel comfortable to use our document as a tool in the world of Big Data?** *

Tools: Frameworks, wrappers and databases

Not at all ○ ○ ○ ○ ○ Very

**Comments** *

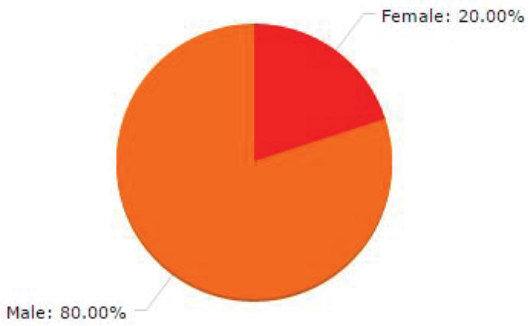
Please give us feedback on what was good and what we can improve

# Survey answers

1. **Gender** *

2 (20.0%): Female

8 (80.0%): Male




2. **Age?** *

- (0.0%): -20

5 (50.0%): 21-25

3 (30.0%): 26-30

1 (10.0%): 31-35

- (0.0%): 36-40

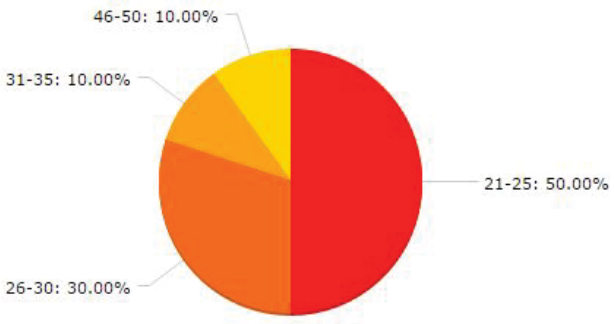
- (0.0%): 41-45

1 (10.0%): 46-50

- (0.0%): 51-55

- (0.0%): 56-60

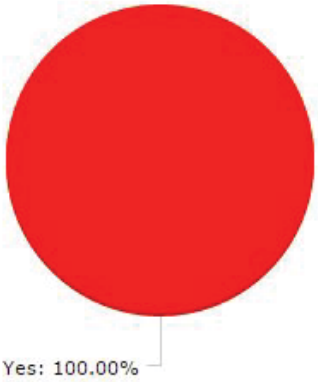
- (0.0%): 61+



3. **Do you have an education in software engineering or other relevant subjects?** *

10 (100.0%): Yes

- (0.0%): No




4. **If you have an education: name your education(s)**

- Software engineering
- Masters of Science in Engineering: Computer Security
- Webbprogrammering
- master of science in engineering:computer security
- university
- Web programming
- Software engineering
- Software Engineering
- Web programming

5. **If you have an education: how long was/is it?**

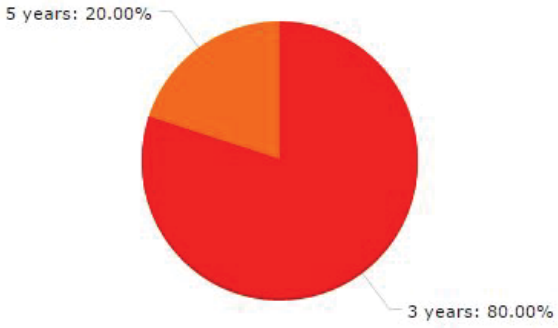
- (0.0%): -1 year
- (0.0%): 1 year
- (0.0%): 2 years

8 (80.0%): 3 years

- (0.0%): 4 years

2 (20.0%): 5 years

- (0.0%): 6 or more years

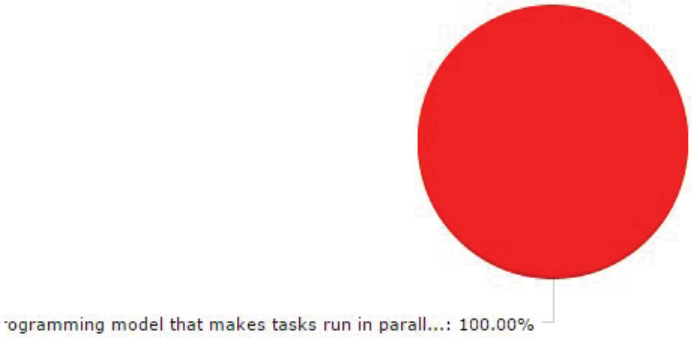


6. MapReduce can be described as ... *

10 (100.0%): A programming model that makes tasks run in parallel on multiple machines

- (0.0%): A framework that process the data using only key

- (0.0%): A programming model that only works with unstructured data




7. These are MapReduce wrappers *

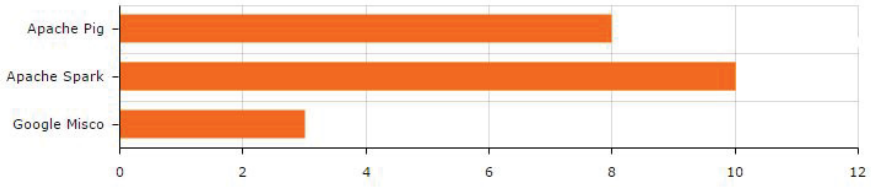
- (0.0%): Apache Avro

8 (80.0%): Apache Pig

10 (100.0%): Apache Spark

3 (30.0%): Google Misco

- (0.0%): Apache Flume




8. The NoSQL categories are ... *

- (0.0%): Collection database

8 (80.0%): Column stored

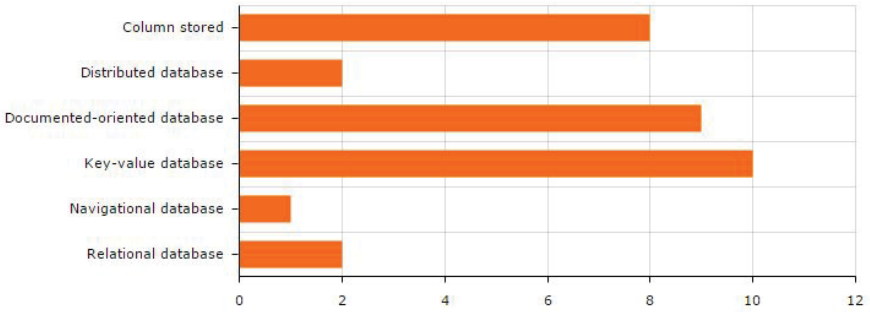
2 (20.0%): Distributed database

9 (90.0%): Documented-oriented database

10 (100.0%): Key-value database

1 (10.0%): Navigational database

2 (20.0%): Relational database




9. Big Data can be described with the words ... *

- (0.0%): Security

- (0.0%): Simplicity

9 (90.0%): Variability

9 (90.0%): Variety

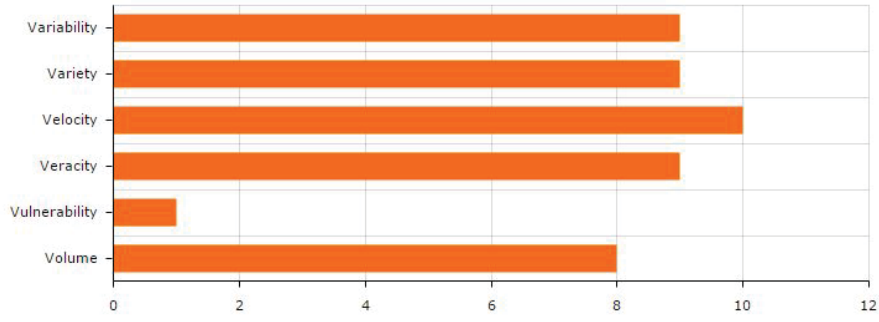
10 (100.0%): Velocity

9 (90.0%): Veracity

- (0.0%): Visuality

1 (10.0%): Vulnerability

8 (80.0%): Volume



10. How informative did you find text? *

| | (1) | | (2) | | (3) | | (4) | | (5) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Σ | % | Σ | % | Σ | % | Σ | % | Σ | % | | Ø | ± |
| Not at all | - | - | - | - | - | - | 5x | 50,00 | 5x | 50,00 | Very | 4,50 | 0,53 |

11. Do you feel that you have a good overview of the platforms and databases after reading? *

| | (1) | | (2) | | (3) | | (4) | | (5) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Σ | % | Σ | % | Σ | % | Σ | % | Σ | % | | Ø | ± |
| Not at all | - | - | - | - | 1x | 10,00 | 8x | 80,00 | 1x | 10,00 | Very | 4,00 | 0,47 |

12. Do you feel comfortable to use our document as a tool in the world of Big Data? *

| | (1) | | (2) | | (3) | | (4) | | (5) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Σ | % | Σ | % | Σ | % | Σ | % | Σ | % | | Ø | ± |
| Not at all | - | - | - | - | 2x | 20,00 | 6x | 60,00 | 2x | 20,00 | Very | 4,00 | 0,67 |

13. Comments *

**Antal deltagare: 10**

- Bra jobbat, den är förstålig och bra uppdelad!
- The report was good in general, but you used both indents and empty lines were used to indicate paragraphs at some points, but you should only use one. Otherwise great!
- This felt like a good start when it comes to understand the concept of big data. But I would like to read the whole thesis
- Bra uppbyggnad och enkla figurer!
- Very good introduction to Big Data!
- A good summary that i think can be very useful
- Very good
- It is informative, got interested in big data
- It was a good description but I want to learn more before using it as a real tool
- Interesting subject and good introduction to big data