Ministry of Education and Science of the Republic of Kazakhstan
Suleyman Demirel University

Uldana Nurkey

Comparison of Big Data analytic tools

THESIS

Presented in Partial Fulfillment for the
Degree of Master of Science in Computing Systems and Software
(degree code: 6M070400)
Department of Computer Sciences
Faculty of Engineering and Natural Sciences

Supervisor: **Bogdanchikov Andrey**

Kaskelen, 2019

Abstract

A huge repository of petabytes of data is generated each day from modern information systems and digital technologies such as scientific data analysis, social media data mining, recommendation systems, analysis on web service logs and Internet of Things along with cloud computing. The data has a huge power to directly guide us to knowledge detection. Examination of these enormous information requires a great deal of undertakings at different dimensions to extricate knowledge for making decisions by Big Data analyzing tools, and picking the correct tool requires an inside and out learning about the abilities of each. Along with these, big data examination and exploring various tools for it is a current area of research and development. This thesis is an effort to present the basic understanding of BIG DATA is and its usefulness to an organization from the performance perspective alongside with integrating it to curriculum of higher education. The basic objective is to explore platforms for analyzing big data and compare them in different perspectives. Additionally, it will open a new horizon for researchers to develop the solution, based on potential impact of big data challenges. This creates the need to incorporate the investigation of Big Data analyzing systems as a major aspect of the computing curriculum. First experiment consists from examples of analytic problems that can be solved as introduction into big data projects on Apache's tools by demonstrating how each type of system can be integrated into education via sample datasets and data analysis tasks, also analyzing their results from educational perspectives. Second experiment is conducted to compare Hadoop, Spark and Pig , as a major and modern tools in big data analytic - on iteration of supporting task, computing time for each task on each tool and Input/Output data access with real life data. Mentioned tools were chosen due to their popularity for analyzing big data. Results of this research show that various tasks require different tools and there is no all-in-one solution. Any big data problems stand in need developers to use proper tool to make job done in a way better and quicker.

Аңдатпа

Петабайт деректердің үлкен репозиторийі күнделікті заманауи ақпараттық жүйелер мен цифрлық технологиялардан, мысалы, ғылыми деректерді талдау, әлеуметтік желілерде деректерді талдау, ұсыныс жүйелері, веб-сервис журналдары және бұлтты есептеулермен бірге заттар Интернетінен алынады. Деректер маңызды мағлұматты анықтауға бағыттайтын зор күшке ие. Айтылған массивтік деректерді талдау үлкен деректерді өңдеу құралдарының көмегімен орындалып, пайдалы білімді өндіру әрі шешімдер қабылдау үшін әртүрлі деңгейдегі күшті талап етеді. Қажетті құралды таңдай алу үшін олардың әрқайсысының мүмкіндіктерін терең зерттеу керек. Сонымен қатар, үлкен деректерді және оны талдауға арналған құралдарды зерттеу қазіргі таңда маңызы жоғары зерттеу аймағы болып табылады. Бұл тезис BIG DATA туралы негізгі түсінікті қалыптастыру мен ұйымдарға тиімділігі тұрғысынан пайдалы екендігін көрсету және жоғары оқу орындарының оқу жоспарына кіріктіру мысалдарымен қамтылған. Жұмыстың негізгі мақсаты - үлкен деректерді талдау платформаларын зерттеу әрі салыстыру. Сондай-ақ, үлкен деректердің ықтимал әсеріне негізделген шешім әзірлеу үшін зерттеушілерге жаңа көкжиектер ашылады. Бұл, өз кезегінде, үлкен деректерді басқару жүйелерін зерттеуді компьютерлік бағдарламаның бір бөлігі ретінде оқыту қажеттілігін туындатады. Алғашқы эксперимент Apache құралдарын пайдала отырып үлкен деректер жобаларына кіріспе ретінде шешілуі мүмкін аналитикалық мәселелердің мысалдарынан тұрады: жүйенің әрбір түрі қандай да деректер жиынтығы мен деректерді талдау тапсырмаларын қолдана отырып, оқу жоспарына қалай кіріктірілетінін көрсете алады. Екінші экспериментте Hadoop, Spark және Pig үлкен деректерді талдаудың негізгі және заманауи құралдары ретінде қарастырылады: салыстыру тапсырмалардың итеративтілігімен, әр тапсырмаға кеткен уақытты өлшеумен, және деректерді Енгізу/Шығару бойынша жүргізіледі. Бұл құралдар үлкен деректерді талдау үшін танымал болғандықтан таңдалды. Зерттеудің нәтижесі бойынша, тапсырмалардың ерекшеліктеріне байланысты деректерді талдауға арналған әртүрлі құралдардың қолданылуы талап етілетіндігі және бірыңғай шешімнің болмауы көрсетілді. Үлкен деректермен байланысты кез-келген мәселелерде бағдарламашылардың жақсырақ әрі жылдамырақ жұмыс істеулері үшін тиісті құралды пайдалануларын талап етеді.

Аннотация

Огромное хранилище петабайта данных генерируется каждый день из современных информационных систем и цифровых технологий, таких как анализ научных данных, анализ данных в социальных сетях, системы рекомендаций, анализ журналов веб-сервисов и Интернета вещей наряду с облачными вычислениями. Данные обладают огромной силой, чтобы направлять нас к обнаружению знаний. Анализ этих массивных данных требует много усилий на разных уровнях для извлечения знаний для принятия решений с помощью инструментов анализа больших данных, а выбор правильного инструмента требует глубокого изучения возможностей каждого из них. Наряду с этим, изучение больших данных и инструментов к ним является актуальной областью для исследований и разработок. Этот тезис представляет собой попытку представить базовое понимание BIG DATA и их полезность для организации с точки зрения производительности наряду с интеграцией их в учебную программу высшего образования. Основная цель - изучить платформы для анализа больших данных и сравнить их с разных точек зрения. Кроме того, это откроет перед исследователями новый горизонт для разработки решения, основанного на потенциальном воздействии проблем с большими данными. Этот факт создает необходимость интегрировать изучение систем управления большими данными как часть компьютерной программы. Первый эксперимент состоит из примеров аналитических проблем, которые могут быть решены в виде введения в проекты больших данных с помощью инструментов Apache, демонстрируя, как каждый тип системы может быть интегрирован в образование с помощью выбранных наборов данных и задач анализа данных. Второй эксперимент проводится для сравнения Hadoop, Spark и Pig, как основных и современных инструментов в анализе больших данных, - на основе итерации задач, вычислений времени для каждой задачи для каждого инструмента и доступа к данным ввода/вывода с данными из реальной жизни. Упомянутые инструменты были выбраны из-за их популярности для анализа больших данных. Результаты этого исследования показывают, что для различных задач требуются разные инструменты, и нет единого решения. Любые проблемы с большими данными нуждаются в том, чтобы разработчики использовали соответствующий инструмент, чтобы сделать работу более качественной и быстрой.

Acknowledgements

Thanks to my thesis supervisor for constant support and to my group mates for useful discussions.

Contents

1. Introduction

1.1 Motivation

Big Data is a huge collection of data that can be gathered, transmitted, accumulated, reserved and analyzed. This reason has made big data a tempting field for research scholars in the innovative activity of using algorithmic techniques to analyze sophisticated and/or unstructured data pools.

In order to fully evaluate and analyze big data, researchers that work with a data must have a certain kind of awareness to use mighty tools and languages for analysis. Therefore, studies related to the review and analysis of the available programming languages and statistical tools, analytical solutions and visualization applications in the field of big data analysis are relevant.

Crucial changes in traditional data analyzing platforms are being made by Big data in information era. 4V's of Big data (Volume, Velocity, Variety, Variability) is increased to 5V: value is added. Since we have general knowledge about big data's other characteristics, Value is the main point which we need to consider when any kind of data is analyzed. Thus, to play out any sort of analysis on such voluminous and complex information, scaling up the hardware stages winds up fast approaching and picking the correct tool turns into a significant choice if the client's prerequisites should be fulfilled in a sensible measure of time. Enormous Data is changing science, medicine, technology, human services, business, and at last our general public itself. Common use cases are:

- information about any visited web pages that are stored in logs;
- risk modeling with unstructured data;
- analyzing social sentiment;
- image classification for health care, for example;

To pick up the important information from Big data we should choose a propelled elective approach to analyze and process it. The market is overflowed with various Big Data platforms and tools, so picking the correct tool requires an inside and out learning about the abilities of each. Every one of them guarantee to give genuine business esteem by bringing cost effectiveness, better time the executives, and investigating information to find profitable business bits of knowledge. Particularly, capacities of platforms to adjust to expanded information analysis plays crucial aspect in determining the right fitting platform to build analytic based results. That is why how to find out and use an appropriate data investigation stages or platforms and how to plan examination techniques are both significant things for the data examination process. Therefore user needs to clarify the problem itself, in order to find out suitable platform for himself. Essential points while considering which tool to choose can be formed based on simple questions like:

- How fast results need to be found?
- How much data will be processed and will its amount increase in the future?
- Does the model structure require a few repetitions or a solitary repetitions?

1.2 Aims and Objectives

The goal of our research is to compare modern big data tools based on educational as well as real practical perspectives to show their performance in different aspects. In this research main aim is focused on some modern popular analytic tools to work with Big data from Apache (Hadoop, Spark, Pig and Hive), specifically:

- Tools for processing and analyzing big data will be investigated, their characteristics will be studied based on experiments;
- To analyze and suggest for higher educations integrating exact tools for learning perspectives;
- To find out similarities, differences in usage, cases to use; Compare them;
- To try each of them on real data.

The object of the study is big data. The subject of study is development tools, languages and methods for analyzing big data. The goal of this work is to compare modern tools for working with big data to show their effectiveness in analyzing various data analyzing based tasks. While there are various works towards Big data analyzing platforms, describing their usenesses and comparing some of them respectively, in this research more horizontal scaling platforms are chosen and it is dedicated to contrast the strength and best use cases for each selected tool, providing some guidance on the reasonableness of various tools for different sorts of situations that emerge while performing huge information investigation in practice. Detailed pseudo code for all problems is given.

1.3 Thesis Outline

The first chapter is Introduction chapter, where we review related work and formulate the problem to solve. In the next chapter § 2 review of some preliminary study is given, along with brief description of each big data analyzing tool. Next,in methods and methodology part § 3 all experimental works for comparing tools are given. Finally in § 4 we summarize the results with comparison and provide open problems for future directions. And in Conclusion chapter we conclude our conclusion.

2. Preliminary studies

2.1 Overview of approaches to storing and processing big data

Big data is advancing as a significant field of research, and new discoveries and tools are always building up, this chapter isn't thorough of the considerable number of conceivable outcomes, and spotlights on giving a list of present instruments, strategies and techniques for parallelization of enormous information by crafted by different inquires about on these issue. The characteristic given by research company Gartner is often used: " "Big data" is characterized by the volume, variety and speed alongside with structured and unstructured data which is transmitted through transmission networks to processors and storage, along with the processes of transforming this data into valuable information for business". As can be seen from this definition, big data has four main characteristics: volume, velocity, variety, and value. However this characterization was given more than 15 years ago, so nowadays the list of V's is increased up to 10. Consider some of them in more detail:

1. Volume. The growing amount of data created by both people and machines places new demands on the IT infrastructure in terms of storage, processing and access.

2. Velocity or speed. It is important to realize that speed means not only the speed at which data enters the storage, but also the speed at which important information is extracted and some analysis on this data is performed.

3. Variety or diversity. The data contains a variety of information provided by different structures. With this, from accessed web page log files to the card

transactions, from photographs and recordings to the the consequences of scientific experiments one should almost certainly work.

4. Veracity or accuracy. Indicating how trustworthy the data is. Use of GPS data can be an example to show how veracity is important when this data is analyzed. GPS will show you inaccurate location while you are travelling through an urban area: satellite signals are mainly lost as they go into tall buildings or other structures. When this occurs, location data has to be merged with another data source like road data to provide proper data.

5. Value. Large amounts of data are a valuable resource. But it becomes even more valuable if it allows you to answer pressing questions or questions that may arise in the future.

It so happened that the tools that existed until recently were not able to cope with large volumes. Up to a certain point, practically the only answer to the question is "how to store and process data?" - was some kind of relational DBMS. But with the increase in volumes, there were problems with which the classical relational architecture could not cope, so the engineers had to invent new solutions.

2.2 Overview of general software tools for analyzing big data

There are many analytical tools for big data; they can be divided into three groups: programming languages, statistical solutions and visualization tools [1]. The choice of one or more of them depends on programming experience and knowledge in data analysis. For example, if you plan to use the R language, you must have good experience in both scientific programming and statistics. On the contrary, when using visualization tools, you can work with large ones without such specific knowledge. To readjust to expanding calls in data processing, big data platforms need to be scaled horizontally or vertically. Horizontal scaling means parallel distribution of workload across several commodity hardware, whereas vertical scaling requires installation of more memory and processors. Apaches family can excellently handle horizontally scaled hardware. Next chapters are general description about division of analytic tools.

2.3 Cloud platforms

There are many platforms for processing big data on the market, some of them are open source, such as Apache Hadoop and SciDB, while others are proprietary platforms and belong to companies such as Google, IBM, Amazon and Microsoft [1]. Depending on the features of these platforms, many platforms were implemented in the cloud (Google AppEngine, Microsoft Azure, and Amazon EC2); each of these solutions has its own ways of solving big data analysis problems (data storage, analytics, machine learning and implementation). Table 2.1 presents a comparison between the known cloud platforms for working with big data.

Table 2.1 Comparing Big Data Cloud Platforms

Features	**Amazon**	**Microsoft**	**Google**
Big Data storage	S3	Azure	Google cloud services
Big Data analytics	Elastic	MapReduce (Hadoop)	Hadoop on Azure BigQuery
Relational databases	MySQL or Oracle	SQL Azure	Cloud SQL
NoSQL databases	DynamoDB	Table storage	AppEngine Datastore
MapReduce	Elastic MapReduce (Hadoop)	Hadoop on Azure	AppEngine
Stream processing	No streaming	Streaminsight	Search API
Machine learning	Hadoop+Mahout	Hadoop+Mahout	Prediction API
Data sources	Public datasets	Windows Azure marketplace	Few datasets with examples
Availability	Public	Beta	Beta

2.4 Programming languages

For analyzing big data, several programming languages are used and they can be divided into two groups:

- High level languages;
- Low level languages.

The levels of programming languages are determined by the analytical use of these languages. In high level programming languages takes advantages by supporting various functions for solving analytical problems.

2.5 Python for Big Data

Python is one of the most famous programming languages for data analysis. It's interactivity and scientific system libraries makes it preferable for creating analytical programs and studying invisible facts in data sets. Focusing on the scientific computer community, it is easy to see how the use of the Python is increasing (since the beginning of 2000) in both industries: the creation of analytical applications and academic research [2]. Python has its own scientific ecosystem, as well as many useful libraries like *Numpy*, which is the base library for basic data structures and the main package in the Python language. Numpy provides several functions for working with multidimensional array, mathematical operators on them and many more. Also, *Panda* is python's package which makes easy to work with data.

2.6 R for Big Data

R is a universal open source programming language: for being able to analyze statistically and data analyzing [3]. In R system, it is convenient to perform any kind of statistical calculations by the use of functional syntaxes or from the scratch code because of strong debugging tools. This language has interfaces that support several programming languages. It includes in itself graphical tool which allows summary statistics. For all of its functions, R provides a wide variety of tools for statistical analysis, machine learning (linear and nonlinear modeling, classical statistical tests, time series analysis, classification, clustering) and graphical methods

2.7 SAS Software

SAS software (and programming language) is a well-known and widely used solution for accessing, transforming and analyzing data using a flexible, extensible web interface [4]. SAS analytical platform consists of a variety of analytical applications that form the structure of the platform and makes it a useful tool for scientists in the field of big data analysis [5]. The main useful analytical applications are:

1. SAS Text Miner: this is a plugin that can be added for the SAS Enterprise Miner environment, because it facilitates the solution of problems in text mining. Text Miner is able to manipulate various sources of textual data.

2. SAS Forecast Server: its automation and scalability allow organizations to make more efficient and effective decisions based on generated high-quality forecasts. This tool improves the efficiency of forecasts and allows you to choose the most important forecasts and focus your efforts on them [6].

3. SAS Model Manager: organizes work on the construction of collections of analytical models, starting with the creation, through management and monitoring, and ending with their administration [7]. SAS Model Manager provides a decision maker with a convenient web environment that facilitates user experience with management tools and support for the life cycle of models.

2.8 MS SQL Server

MS SQL Server is a very well-known solution for traditional relational databases, and has very good tools for creating ERD charts, as well as for optimizing queries using a graphical tool. MS SQL Server has an integrated Analysis Service, which has a close relationship with the Tabular Model, Multidimensional Model and the Microsoft BI stack. There are also three main services for business intelligence in MS SQL Server:

- SQL Server Integration Service (SSIS) for data collection.

- SQL Server Analysis Service (SSAS) for data analysis.

- SQL Server View Service (SSRS) for viewing data (visualization).

Microsoft SQL Server has a built-in interface to integrate with Apache Hadoop (SQL Server Hadoop connector), which is an Sqoop-based interface; The main purpose of the design of this connector is to provide an effective tool for transferring data between SQL Server and Hadoop [8]. MS SQL Server 2012 packages provide researchers with a complete set of data integration tools, visualization problem solving, a business intelligence package, and the ability to connect to Apache Hadoop and Hive through an efficient bridge.

According to a survey by KDnuggets (2016), when a comparison was made of the percentage of analysts using all analytical tools and software, it was revealed that R, SAS, Python and SQL have been the main languages for data analysis (Table 2.2). These rates demonstrate that SAS clients don't utilize R and Python in their exploration, while the mix among R and Python is the most ideal for information examination.

Table 2.2 Percentage comparison of language usage and data analysis tools

Tool	**Usage**
R	49 %
SAS	36.4 %
Python	35 %
SQL	30.6 %
R and Python	20 %
R and SQL	22 %
Python and SQL	13 %
R and SAS	6.8 %
Python and SAS	7 %
R and Python and SQL	10 %
R and Python and SAS	2 %

Also, the combination of these three tools (R, Python and SQL Server) is better than another set of tools (R, Python and SAS). Even if R and Python are similar languages, some distinctions are listed in Table 2.3

Table 2.3 Comparison of languages R and Python

Feature	Python	R
Libraries	NumPy, SciPy, Pandas, Matplotlib, Scikit Learn, etc.	Over 7500 libraries.
Compiler	Interpreted language	Interpreted language
Learning	Easy to learn	Hard to learn
IDE	PyCharm, Spyder, Anaconda	RStudio, Red-R
Speed	Slower than R	Fast processing, especially after optimization of computing algorithms.
Visualization	Pandas, ggplot libraries	ggplot2, ggvis, rgl, htmlwidgets and googleVis.

In this comparison, there is no doubt that R is the best choice. But it should be noted that other tools are sometimes better for non-professional users or for those who have no programming experience.

2.9 Hadoop Family

Apache Hadoop is an open source software suite that takes advantage of computer clusters to store and process a huge amount of data. It was initially built as Apaches projects to support web searching engine, later was inspired by Google's file system(GFS) and programming paradigm. In 2008 Hadoop broke a world record by sorting TB of data in 209 seconds with less than 1000 clusters. Hadoop consists of three important components: the repository is HDFS and resource manager YARN, also processing part MapReduce programming paradigm. HDFS is a distributed, scalable and portable file system written in Java. HDFS allows to store large files on multiple machines. It provides reliability by replicating data to other data nodes, which causes redundancy. MapReduce allows to write applications for parallel processing of huge amounts of data on large clusters. MapReduce breaks up work into independent chunks, which are processed in parallel by MapReduce tasks. Apache Spark is a unified analytic engine for large-scale data processing. Spark provides an interface for programming entire clusters with

implicit data parallelism and fault tolerance. Spark provides the same scalability and resilience as MapReduce, but works faster for a specific application due to a different data abstraction and multi-functional API. Apache Solr is an open source search platform. It is widely used for full-text search and analytics. Solr offers excellent filtering and aggregation capabilities through faceting.

2.10 Literature review of comparing Apache's Big data analytic platforms

According to paper [9] by Dr. Urmila R. Pol, author explains working principles of Hadoop Mapreduce, Pig and Hive by comparing them between each other. Author mentions that writing MapReduce code with Java would require to write several lines of code, and would take more time if user is not good familiar with Java. In this case, using another approach to write MapReduce tasks like by writing them in Pig Latin or Hive SQL language would reduce development and testing time overall. As stated in the paper, even if Pig or Hive do not run as fast as MapReduce's native Java, using them boosted data analysts productivity. It is because writing Pig script takes only 5 percent of that time which is needed to write in Java, however decreasing runtime performance. Thus, Hive and Pig are considered performance boosting tools for data analysts. Writing join functionality in java will be very complicated while using Hive SQL with several levels of nested FROM clauses. Some cases where pure Hadoop MapReduce is preferable than Pig or Hive is discussed in the article above, like:

- Job where complicated form of distributed cache is required;
- Job where optimization is needed in mapper or reducer level;
- Job where is needed some form of partitioning;
- Performance of Hadoop is much better than other two, even if they enhancing their functionality kit and etc.

Other technologies (*Hadoop*, *Spark*, Graphics Processing Unit (*GPU*) , High Performance Computing Clusters (*HPC*), Multicore processors and Field Programmable Gate Arrays (*FPGA*)) that can handle big data is reviewed in paper

[10], where authors first explained vertical and horizontal scaling, types of platforms/technologies along with their strength and weaknesses. Practical applications with general idea when scaling up and scaling out should be preferred is given along with explanation of improving systems by selecting appropriate platform. Authors claim that cases where expandable platform, which can support huge amount of data, Hadoop MapReduce and Spark is perfect, especially if there is no need for real-time responses; whereas cases where queries are processed in real-time - vertical scaling is needed. We can add that if application needs to be processed not in real-time with high velocity, Spark is the best option; but if volume matters, Hadoop can handle far more than that of Spark, since it stores data in disk memory. Which of them should be used in exact situation is given by Alex Bekker, highlighting each's dignity. Performance comparison between Hadoop and Spark was observed, where states that Spark runs in-memory 100 times faster and 10 times on disk; for 100TB of data Spark has been recorded 3 times quicker than MapReduce; also for k-means and Naive Bayes algorithms has been found to be faster. Advantage of Spark by its cyclic connection is also mentioned along with prices of each tool:Hadoop is less costly than Spark. In terms of security and fault tolerance, Hadoop is recommended to be more secure and reliable. Implementation of K-means clusterization on MapReduce, MPI, GPU, and Spark is demonstrated in article [10]. K-means algorithm include in itself characteristics like:

- Results of one iteration is waited for next iteration, and so on;
- Tasks are intensively computed;
- All local results need to be aggregated to find global;

Performing this algorithm on MapReduce showed bottleneck on accessing disk while writing centroids after every iterations. While using MPI bottleneck appeared in peer-to-peer communication. Implementation on Spark was similar to MapReduce, however global centroids are written to the memory instead of disk, which hasten processing and scale down overhead in Input/Output.

2.11 Literature review of integrating Big data analytic tools into curriculum

Paritosh Goldar, Yogesh Rai, Santosh Kushwaha, 2016 made research on ongoing analytical and all-inclusive analysis of big data parallelization procedures and strategies. They explored its one of a kind characterizations based on the investigative techniques and trials. The progressed and future methodologies of big data were investigated[11].

There are a few further future investigation of big data parallelization when it is incorporated with other essence usage of algorithms and programming. Summary from Bhandarkar, M. structures and creates Hadoop applications and more elevated amount application systems to crunch a few terabytes of information, utilizing somewhere in the range of four to 4,000 PCs. Clarification for regular issues experienced in amplifying Hadoop application performance is examined [12].

Researchers from Arizona state university shared their experience in integrating Big data analysing tools in their curriculum [13]. They defined challenges for computer science educators that raise while organizing different BDMS technologies(MapReduce, NoSQL with HBase, NewSQL) into learning units when including into the computing curricula. What is handful in this paper, authors provide class resources like in-class exercises, sample projects and programs to enable a hands-on experience with big data analysing tools. For consistent and durable data processing database systems are preferable according to the paper, while MapReduce is the right tool to analyze and get useful insights like aggregations, trends and correlations (for selling operations). As well as this, use cases for HBase is discussed: in messaging system as a right tool to run a query that quickly retrieves user's messages by accessing randomly. In the same scenario, the MapReduce framework might be the best tool to run a job that processes the entire dataset and finds total number of messages by sender location.

Big data projects included in curriculum of Stetson University by using Apache Hadoop, HDFS and MapReduce; Apache Hive; Apache Spark; and other tools are discussed in Joshua Eckroth's work [14]. The principle learning objective there is that students show their ability to recognize which instrument is the most proper for explicit data examination exercises and create a model to pick right platform for each dataset. First, students were given 3GB dataset to analyze failure rate by

plotting graphics, and find unique characteristics that have a higher proportion of failures: where students had to use RStudio, which was a painful process for R because of slowing down it. Next, students were given more than 116GB XML files and needed to summarize a dataset by producing jobs with Java/Python in MapReduce. After that they were asked to write straightforward queries by using Hive, where they found that Hive is much easier to use than MapReduce for these kind of tasks. As well as this, machine learning tasks were given to be solved by using Spark Mahout VS Weka workbench. In decision matrix Hive was preferable in SQL-like queries whereas Spark was selected for image processing for big data. Theoretical differences alongside with comparison of performances of Hadoop and Spark is explained in International conference on I-SMAC 2017 as well [15]. In terms of batch and iterative processing, data storage and access rate, real time processing, working principles, and differences in computation are provided. As a practical performance evaluation authors used two data sets: Wordcount problem by changing number of words for each iteration; problem related with predicting by using logistic regression method. For both cases they used single node cluster. As was expected, performance ratio of Spark over Hadoop was 2.82 whereas in iterative queries Spark was faster by 2.17 ratio. Even if Spark is faster for iterative queries, since it saves its data in a cache, and the size of that cache is limited, that is why second experiment showed fewer performance rate.

Comparison of Hadoop and Spark was provided in many papers, for example, full-text search with more than 50GB on-disk Wikipedia data on a 20 node Amazon cluster took 20 seconds with Hadoop, while the same search took only 0.8 seconds with an in-memory RDD. Advantage of storage model in Spark (RDD) guarantees fault tolerance that minimizes network I/O.

Figure 2.1 compares execution of Spark VS Hadoop for changing quantities of iterations. In this test, 100 GB of information was processed on a cluster with 50-node; Hadoop takes a consistent time for every emphasis of around 110 seconds. Interestingly, Spark takes 80 seconds for the introductory iteration to load data in memory, yet just six seconds for each consequent cycle [16].

By taking account all provided reviews of other researches, Big data course was planned carefully to fulfill graduate degree student's knowledge and some practical exercises were selected for each tool.

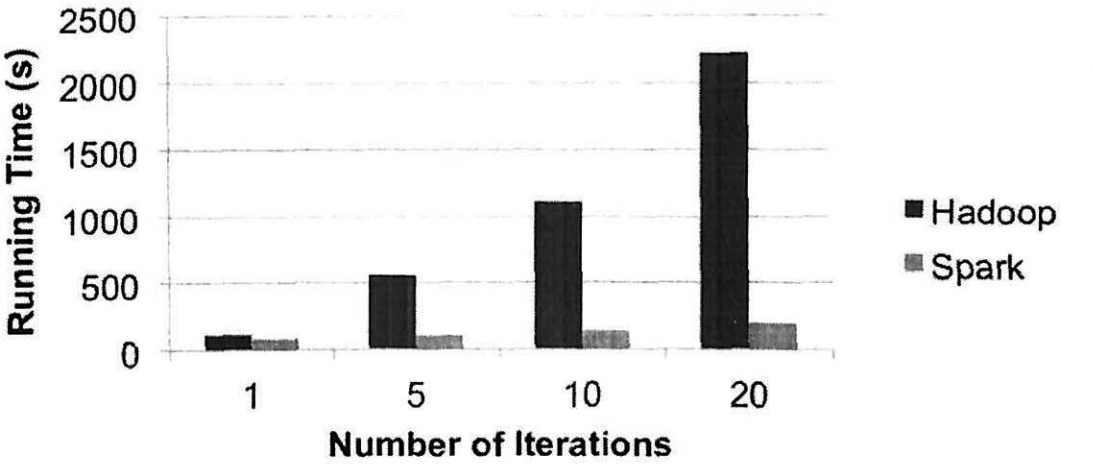

Figure 2.1: Performance of logistic regression in Hadoop VS Spark [16]

2.12 Proposed Methods

Comparing tools should be built on platform dependent and algorithm dependent, giving general view about strengths and weaknesses of big data systems based on tendency of each. However, it also should be counted that application based comparing strongly depends on problem that needs to be solved. Google can be considered a pioneer in the field of big data, which in 2003 described the distributed file system GFS [17], and in 2004 presented the computational model MapReduce [18] to the world. It was these publications that helped the developers of the free search engine Apache Nutch create the Hadoop project [19], which today has actually become synonymous with the term "big data". Before looking at Hadoop more closely, you should answer the question: "why do we need another product if many NoSQL databases provide interfaces for MapReduce calculations?". The answer can be obtained by considering the performance of modern hard drives. The average hard disk performance today is 100 MB / s, which means being able to read 1 TB of information in about 2.5 hours. Such depressing indicators can be improved by parallel reading from several disks. For example, the same 1 TB can be read from 100 discs in 2 minutes. But after all, almost all NoSQL solutions support horizontal scaling, and therefore parallel disk operations? And here the key factor is the head positioning time. In order for the update and read operations to be efficient, NoSQL databases (CouchDB, MongoDB) have to use random access structures, such as B-trees [20]. So, if you refuse to arbitrarily update the data and process the entire set consistently, you can achieve

a serious performance boost. This principle is the basis of the Hadoop architecture. Distributed file system HDFS is responsible for storing and organizing data in the Hadoop cluster. A whole ecosystem of projects has formed around Hadoop:

Apache Hadoop

Apache Hadoop is an open-source software framework for parallel processing huge sets of data on large clusters (consisting from thousands of nodes) , built from reliable commodity hardware, by using simple programming model called MapReduce. Hadoop divides files into independent blocks, then distributes them to nodes, where they are summarized separately. Data is stored in HDFS, which is a distributed file system, where all files are neighboring set of bytes. Hadoop first implements map operations to each block of data from HDFS by sorting and redistributing results depending on key values, then second part (reducer) consists from collecting data items with same keys. Reducers can aggregate the intermediate results from mapper to generate final result and write them again to HDFS. Thousands of map and reduce tasks run across different nodes in each cluster parallelly.

YARN is one more component of Hadoop which is responsible for coordinating applications runtime. Survey about parallel data processing with MapReduce is accessible in [21]. Hadoop can run distributed applications on a large cluster composed of general computing device, which uses a *Master/Slaves* structure which is depicted in Figure 2.2

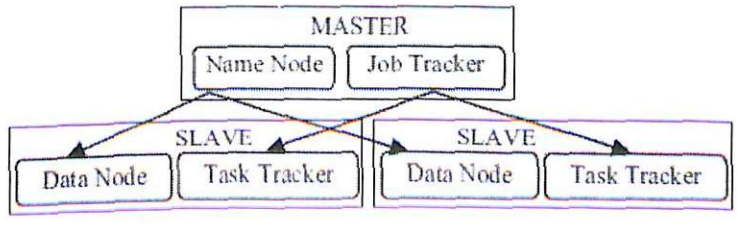

Figure 2.2: Master/Slaves structure of Hadoop cluster[22]

Master responsible for NameNode and JobTracker. JobTrackers main duty is to initiate tasks, track and dispatch their implementation. NameNode is like a bookkeeper, it keeps track of file blocks that were broken from file, by saving in memory which nodes have those blocks. In case of failure of NameNode, there exists Secondary NameNode for assisting monitoring over HDFS file blocks.

Slave is responsible for DataNode and TaskTracker. TaskTracker deals with preparation of local data and gathering result information as indicated by the states from applications and reports the states and executions to JobTracker. NameNode and DataNode are accused of satisfying HDFS assignments, while JobTracker and Task Tracker for the most part manage MapReduce tasks [22]. HDFS supervises capacity of the cluster by dividing approaching files into pieces, called "blocks", and stores every block repetitively over the pool of servers. In the regular case, HDFS stores three complete duplicates of each record by replicating each piece to three distinct servers. In Figure 2.3 two step process of MapReduce

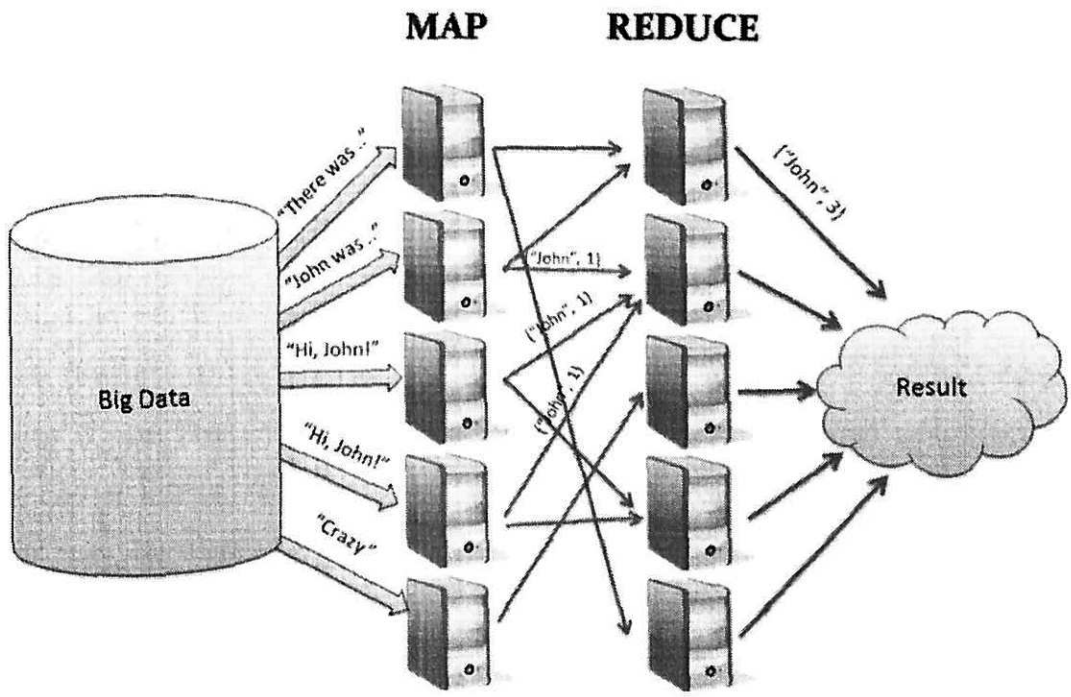

Figure 2.3: MapReduce in action [22]

is shown. There is a Mapper and Reducer phase, the mapper function will inform the cluster which data block need to be retrieved. Function of reducer is to get and collect all of the data and aggregate. Hadoop is a batch processing, therefore MapReduce works on all of data inside clusters.

Figure 2.4 shows example of using Hadoop for multimedia semantic classification to scale processing of large volumes of stored images and video.

However HDFS is not intended for the kind of workloads related with observing. HDFS expects to deal with huge records and high compose rates from relatively little quantities of writers. It isn't intended for a huge number of simultaneous low-rate Inputs, and a huge number of little files. More awful, record are not no-

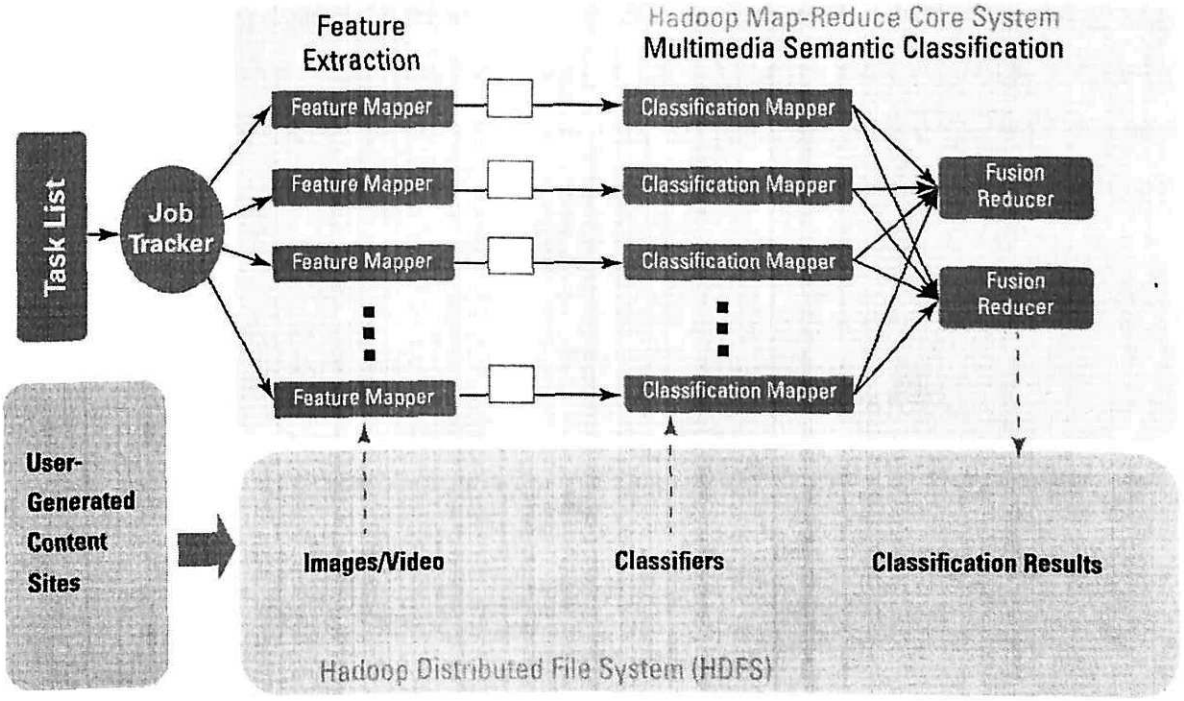

Figure 2.4: Using Hadoop to semantically classify video and images from social media sites [23]

ticeable to readers until the file is closed, and by default HDFS don't enable closed documents to be revived for composing. Accordingly, some consideration must be taken in utilizing HDFS to help processing continuously rather than batch. The best way to share information between parallel tasks in MapReduce is to write it to a distributed file system, which adds considerable overhead because of information replication and disk I/O, taking 90 % of processing time. To avoid reinitialization of tasks forward scheduling was invented, source code is extended with adding caching ability[36]. Also, since MapReduce uses high level language like Java, it might be not easy to maintain working with clusters. There are diverse coding approaches besides Hadoop MapReduce, like Pig script and Hive interface, but both works on top of Hadoop Mapreduce.

Apache Pig

Apache Pig is a SQL-like platform that was built on top of MapReduce by Yahoo company, providing developers better control on MapReduce. As java has jvm, pig uses pig runtime as an execution environment. Fact says that 10 lines of code in pig will do the same as MapReduce with 200 lines java code. However, compiler converts pig latin into MapReduce, creating consecutive set of jobs. Developers can write functions in Python, Java, Javascript and Ruby, also in Groovy to extend Pig latin code, if needed. Storing, manipulating and executing data is

allowed in Pig. Pig Latin is made up of 2 components: dataflow language Pig Latin and it's compiler, that converts script into MapReduce jobs that can be performed in Hadoop cluster. Figure 2.5 depicts architecture of Apache Pig, showing

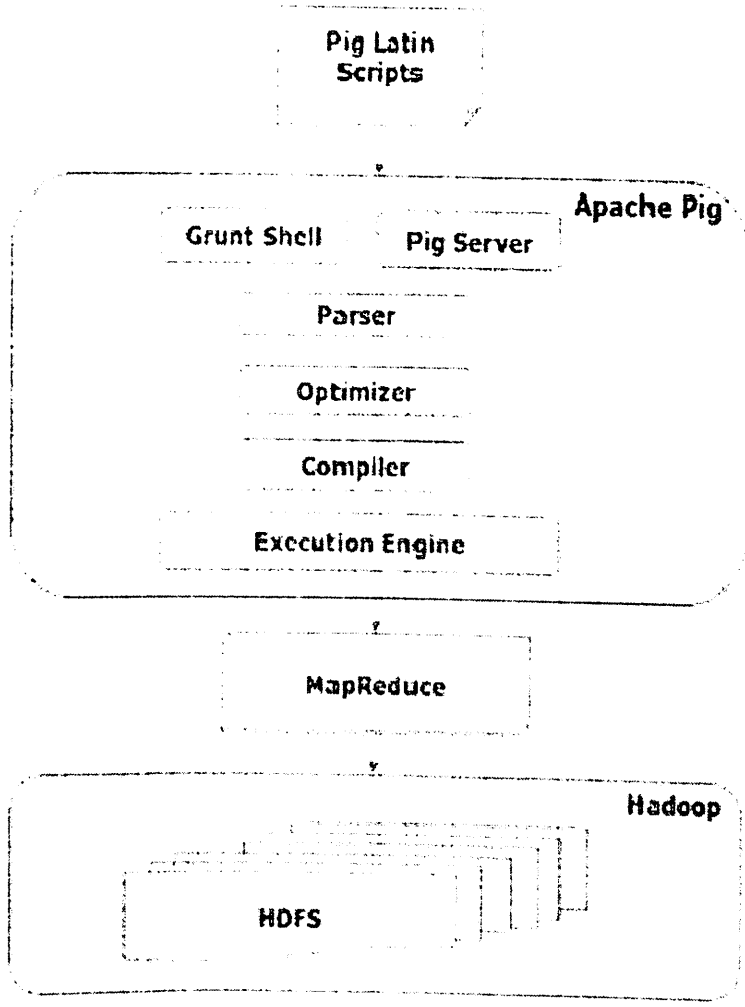

Figure 2.5: Apache Pig Architecture [24]

all components of framework.

Apache Spark

Apaches product, which also parallelly process data across clusters, was invented in 2012 at the AMPLab. Spark uses system memory, which makes it a way faster than Hadoop MapReduce, which reads and writes data to HDFS. Process of writing data to RAM is completed by using RDD(Resilient Distributed dataset). Spark Core plays significant role in coordinating scheduling, applying abstraction of RDD, optimizing and connecting Spark to proper filesystem. However, data transfer (I/O) between nodes still creates some network congestion. It runs on top of existing hadoop cluster and access HDFS, can also process structured data in Hive and Streaming data from HDFS, Flume, Kafka, Twitter [25].

1. *Speed*: Spark allows applications in Hadoop clusters to process up to 100 times more rapidly in memory, and 10 times faster even on disk. It is conceivable due to ability of Spark by diminishing number of read/write to disc. All its intermediate data is stored in cache. It uses the concept of an Resilient Distributed Dataset, which enables it to straightforwardly store information on memory and endure it to disc just when it is required. This diminishes the vast majority of the reading and writing to disc – which is the main time taking factors in data processing.

2. *Ease of use*: Spark lets you quickly write applications in Java, Scala, or Python. This encourages programmers to make and run their applications on their well-known programming language and simple to manufacture parallel programming. There are more than 80 high-level operators in built-in set, that can be used interactively to inquiry information.

3. *Integrates SQL, streaming, and complex analytics.* In addition to basic "map" and "reduce" operations, Spark maintains streaming data, SQL queries, and complex analytics such as machine learning and graph algorithms. Not just that, clients can join every one of these abilities in a solitary work process.

4. *Runs Everywhere* Spark runs on Hadoop, Mesos, standalone, or in the cloud. It can get to various information sources including HDFS, Cassandra, HBase, S3.

Spark is preferred by High Performance and visualization computing lab, where I did my internship, they proposed extension of Spark - Vispark for MapReduce processing on GPU clusters. Significant pattern in the parallel programming network has been the utilization of fine-grained parallel processors which are regularly utilized as GPU quickening agents on every hub of a group to help its all out computational power. Research Internship at UNIST gave an opportunity to work with GPU accelerators for Spark, where the main research areas are: GPU-accelerated large-scale biomedical image processing.
Improving Input/Output time optimization for GPU workers in SPARK was my task, for researched topic "Vispark: GPU- accelerated distributed visual computing using Spark" [26] where for iterative processing was used GPU. In this research

GPU workers are introduced as managers of GPU context and caching data in-memory of GPU. Basically, Spark's CPU workers are generated and stopped every time for each Spark operations, whereas recommended system with GPU workers stay until entire Spark runtime. For this reason, time is reduced by sharing data by several Spark operations in GPU memory without CPU- GPU and CPU-JVM data transfers.

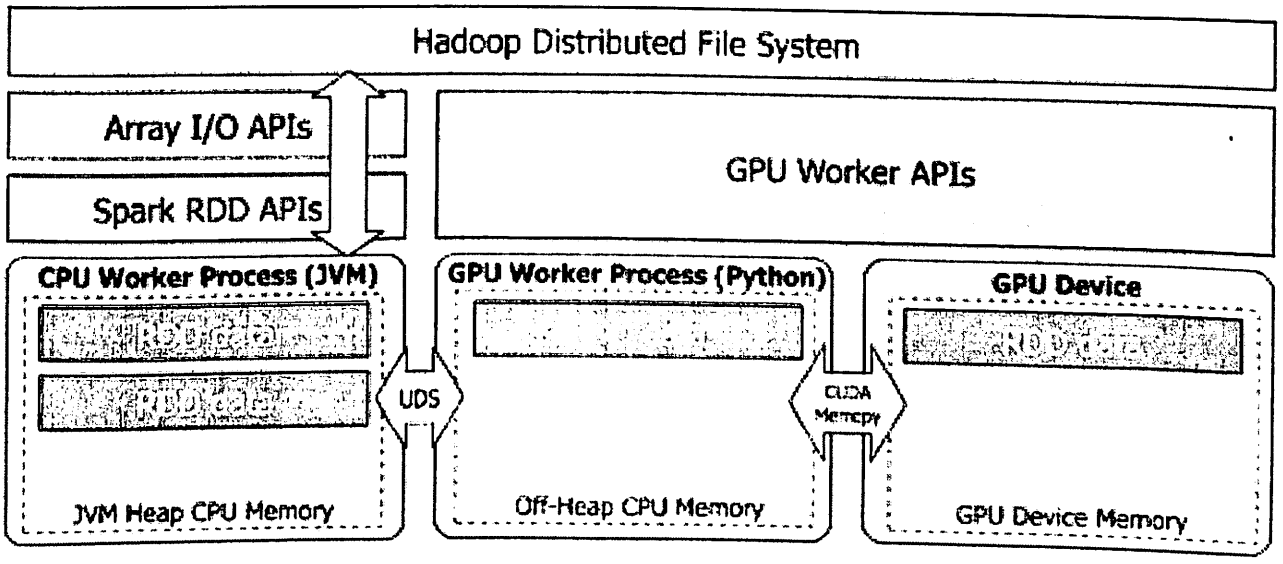

Figure 2.6: Architecture of proposed system with single GPU node, arrows- data flow between disk to worker processes and GPU[26]

In current system which is depicted in Figure 2.6, iterative processing is done in each GPU memory by modifying content of GPU memory directly by rewriting existing objects in GPU with new ones. Accordingly, GPU enlarges memory of Spark by providing additional memory and can be maintained separately by user, which in case of iterative execution of GPU workers may produce performance bottleneck. To avoid bad pipelining in each GPU worker, my task was by implementing multithreading (pthreads) via socket, parallelize tasks (in this case 4) for each GPU worker so that after socket is accepted through Network TCP/IP each of the following steps will be done parallelly for each task:

1. Recv command data (only MSGSIZE) (Network TCP/IP)
2. Check command (GPU_APIs) (CPU)
3. Recv all data (MB , GB lt;2GB) (Network TCP/IP)
4. GPU

When we have used 8 threads in each 8 nodes, with 16 GB data with different partition of it (16, 32, 64, 128) multithreaded GPU worker we achieved average 14 % speed up. The best result while using 8 thread is 44 % for 64 partition. However the main bottleneck is in HDFS-CPU IO, approach with only multithreading GPU workers was not enough.

Apache Hive

Hive was created by Facebook for programmers who are fluent with SQL to work in Hadoop environment, also uses MapReduce to analyze data stored in HDFS. It can process structured data in tables. However, unlike Hadoop developing map and reduce tasks, Hive offers a query language based statements and automatically translates them into one or more MapReduce jobs. Whereas Hadoop Streaming reduces the required code/compile/submit cycle, Hive requires only the composition of *HiveQL* statement.

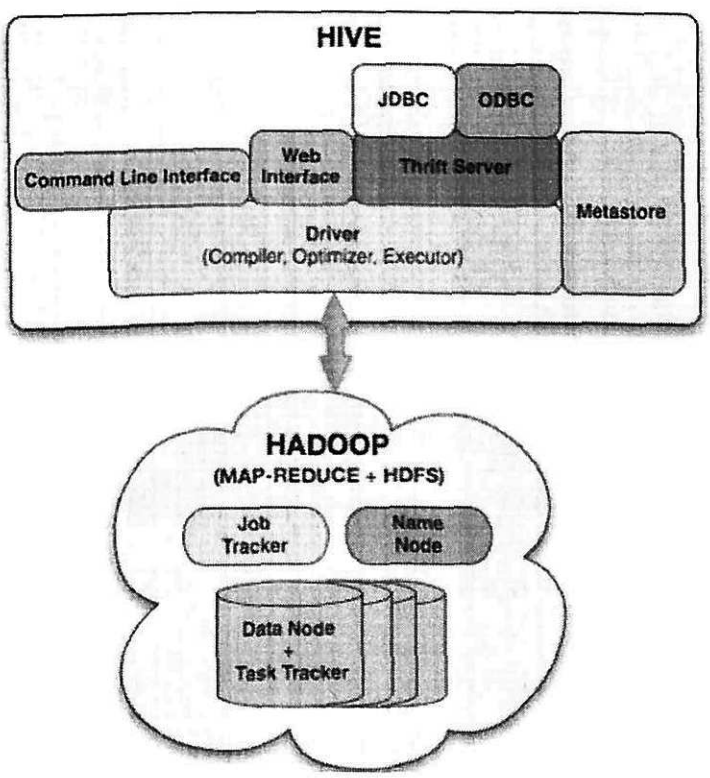

Figure 2.7: Hive Architecture[27]

Figure 2.7 shows internal design of Hive. Hive also provides a declarative query language (the SQL-like HiveQL), which allows you to focus on which operation you need to carry out versus how it is carried out, that supports basic operations like select, join, project, aggregate and union all. Therefore, anyone with a familiarity with SQL can use Hive. Users can load data from external sources and insert data operators (DML), respectively. Hive provides web based UI,thus

user expects visualization of analyzed data via graphs, dots , etc. Drivers creates a session for a query and sends them to compiler, which in turn gets data from metastore. Based on Direct acyclic graph generated by compiler, next stages will be map/reduce job or operations on hdfs.

Figure 2.8 apparently shows how Hive was built on top of Hadoop MapReduce.

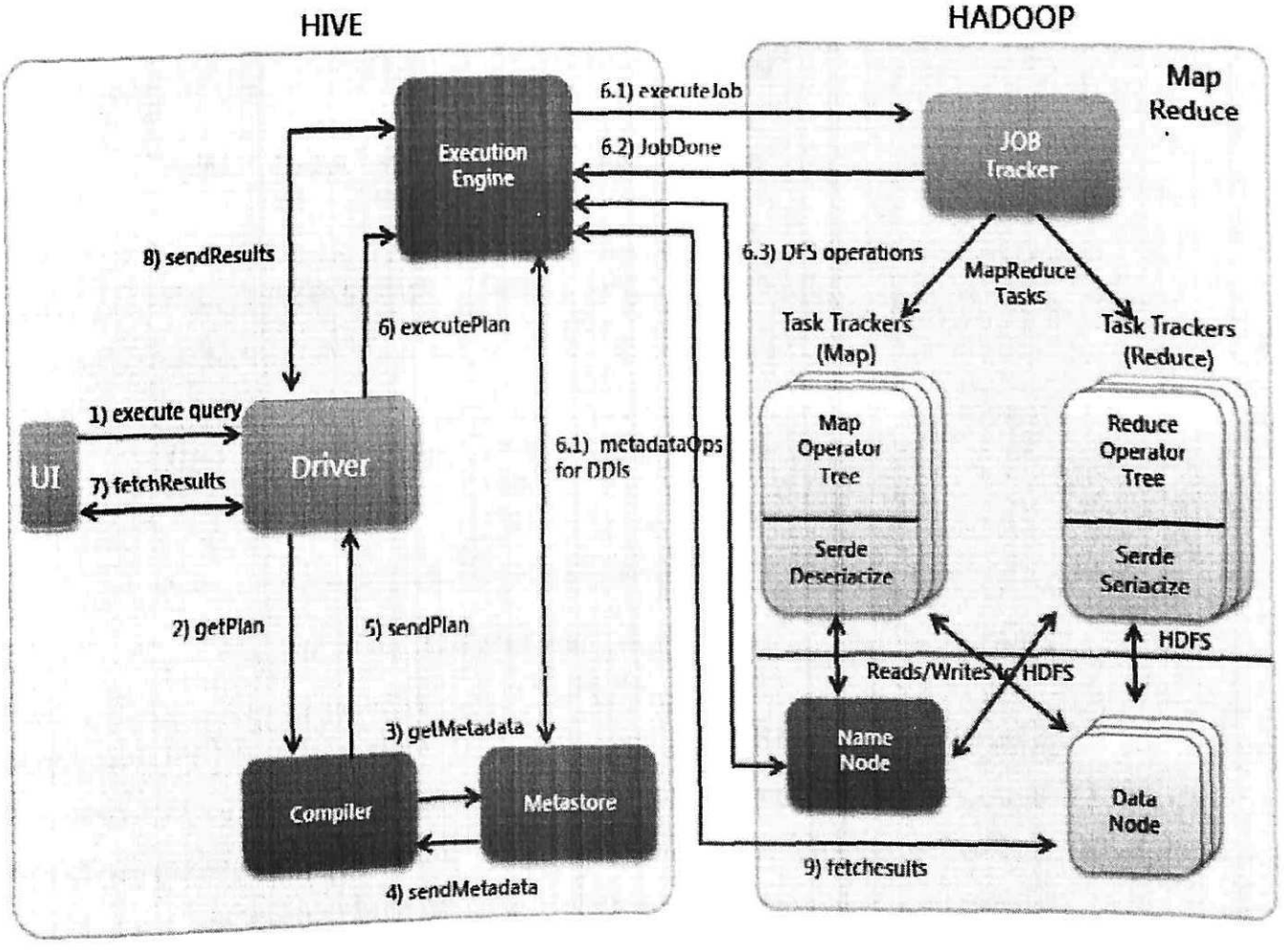

Figure 2.8: Hive job execution flow [27]

Next Figure 2.9 represents theoretical comparison of each big data analysing tool by their characteristics. It shows more qualitative relation between tools than quantitative, comparing characteristics like *I/O performance for iterative processing, real-time processing, language, abstraction level, code efficiency and development effort.* MapReduce, Pig and Hive are not basically used for real-time processing tasks, because they are slow in transferring data Input/Output between nodes. Therefore they receive only 2 out of 5.

All of the frameworks can be scaled up to tens of thousands of nodes and fault tolerant. MapReduce, Pig and Hive are not designed for iterative processing, because data has to be written onto disk after each iteration, which creates a

	Language	*Abstraction level*	*Development effort*	*Code efficiency*	*Data access/ data I/O performance*	*fault tolerance*	*real-time processing*	*iterative task support*
MapReduce	Compiled	Low	Challenging	High	External memory	5	2	3
Spark	Compiled	Medium	Hard	High	In-memory cache	5	5	5
Pig	scripting	High	Easy	Less	External memory	5	2	3
Hive	SQL-like	High	Easier than Pig	Less	External memory	5	2	3

Figure 2.9: Theoretical comparison of Big data tools

huge bottleneck while disk I/O. However MapReduce has developed tools and frameworks, one of it is improved by Hadoop development, which is for improving iterative tasks, therefore they received 3 out of 5. Decision between Hadoop and Spark lays between whether user needs an off-the-shelf instruments or he needs optimization of cluster performance.

Both Pig and Hive are wrappers for MapReduce, but their uniqueness is depicted in Table 2.1 below. Hive is utilized more by specialists and software engineers, and it is only used to inquiry and analyze enormous data sets stored in the Hadoop storage. Hive is most appropriate for batch processing jobs as opposed to working with web log information and append-only information. Hive can't work for Online Transaction Processing work types since it doesn't give questioning continuously and row- level updates.

Table 2.1: Comparing Pig and Hive

	Pig	**Hive**
Architecture of platform	Data flow language	SQL like language
Applications where used	Programming purposes	Report creation
Area of usage	Client side	Server side
Avro files supporting	Yes	No

3. Research design and Methodology

3.1 Selection of software tools for analyzing big data for different tasks

Linux is the official development platform for Hadoop. Hadoop can run in local mode, pseudo-distributed mode and in multi node clusters. All experiments in this thesis were carried out in pseudo and fully distributed mode. Installation instruction of Hadoop version 3.2.0 is fully given in [28]

3.2 Data and experimental work

This section includes experimental work 1- for comparing mentioned tools on real life data from kaggle using Amazon clusters, followed by experimental work 2- integrating these tools into Computer science curriculum for higher education.

3.2.1 Experimental work 1

Data has been processed in Amazon EMR cloud cluster, which provides managed frameworks like Hadoop, Spark, Presto and HBase with Jupiter based notebook. Amazon EMR is used due to its reliability, easiness, flexibility and elasticity, also to set same conditions for compared platforms. Instances that where applied for master is shown in Figure 3.1 - m4.large, has 4 cores with 16 GB RAM , and for workers-c4.xlarge with 4 cores and 8 GB RAM each. Furthermore no tuning was done on MapReduce, Spark and Pig, so that they had default settings. However, since Spark is not using all available cores that were given, it needs to be set

manually to use them, therefore some additional improvements on clusters were done. All provided codes are on Pig,since Pig latin is the most compact compared to other tools.

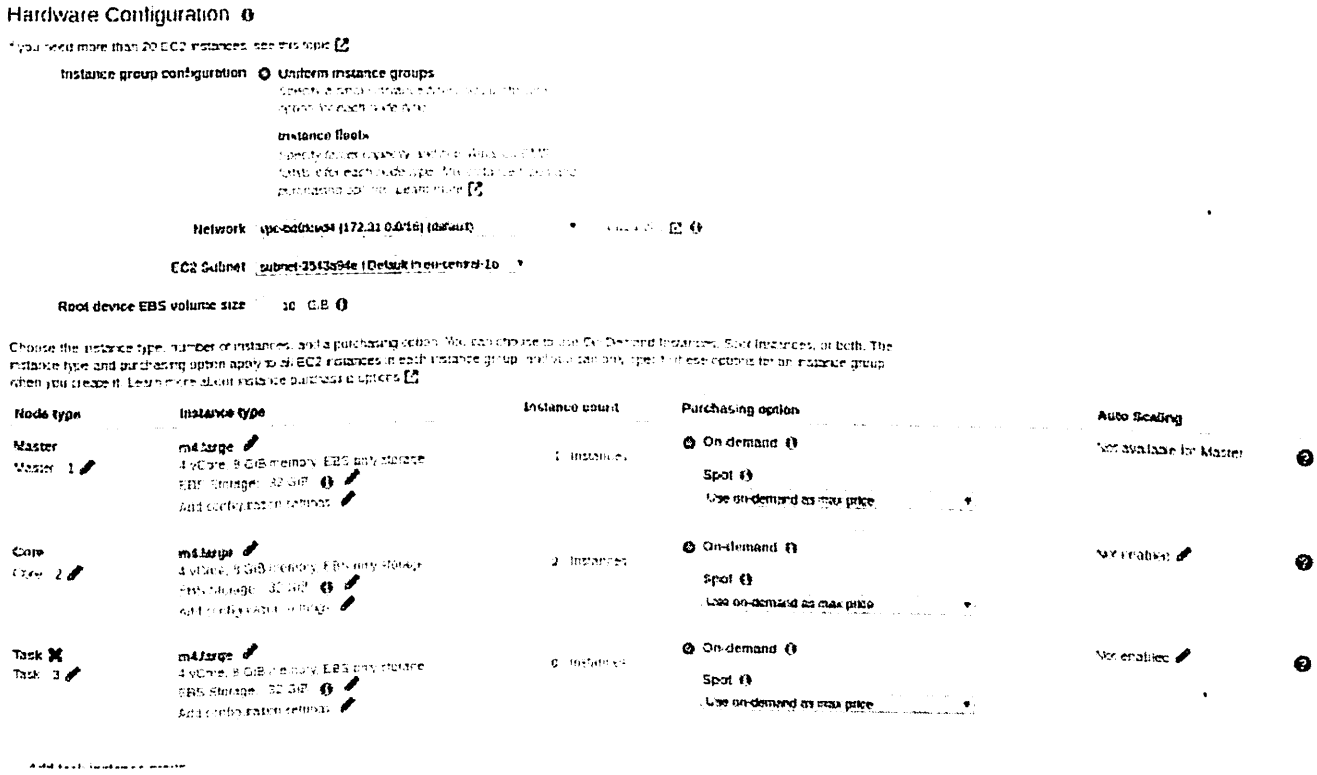

Figure 3.1: Selecting instances

Before starting comparison mentioned tools,lyric dataset was preprocessed by:

- deleting new lines in each row of song, so that each song takes one row;
- all stop words and common words were deleted, like chorus, bridge, etc.

First task is to find the most repeated (beloved) word for each singer, was done by simple Wordcount example in Figure 3.2 with Pig Latin language. Keys were singer names and values were words from their song lyrics. In Pig Latin GROUP keyword is used as a mapper.

As second task checking lyrics for plagiarism was chosen. That is necessity is to find pairs of song that have similar lyrics. Main concern of this problem is how to compare two documents efficiently, as a direct comparison is not an option, due to possible small changes in the song. There are some approaches to solve this issue, such as MinHash, SimHash, Word2Vec. However the purpose of

```
REGISTER bigdata.jar;
all_data = LOAD 'songdata_tabbed.csv' USING PigStorage('\t') as
          (singer:chararray, song:chararray, lyrics:chararray);
tokenized = FOREACH all_data GENERATE singer,
                    kz.sdu.bdt.pig.Tokenizer(lyrics) as tokens;
flat = foreach tokenized generate singer, flatten(tokens);
grp = group flat by ($0, $1);
cnt = foreach grp generate flatten(group) as (singer, word), COUNT($1);

g = group cnt by singer;
result = foreach g {
    prods = order cnt by $2 desc;
    top_prods = limit prods 1;
    generate flatten(top_prods);
};
--sorted = order result by $0;

dump result;
```

Figure 3.2: Popular word among singers with Pig

this research is to compare tools and implementation of such advanced algorithms might interfere with results.

This is the reason why simple approach with Jaccard Similarity was chosen. Jaccard similarity [29] is equal to division of the size of the intersection of A and B sets to the size of their union, where elements of the set are words in songs. Despite

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Figure 3.3: The Jaccard Similarity algorithm

that Jaccard similarity works perfect for most of the time, this approach cannot be used alone while comparing text, because it does not take into consideration position of words in songs. For example: sentences "Alex loves Beth" and "Beth loves Alex" are considered exactly similar by plain Jaccard similarity algorithm, but are not. To address this issue Shingling of lyrics is used. Song is divided in

shingles of 5 consecutive words. However it is easy to see that final size of shingled song is much bigger, so to slightly reduce memory usage all sentences are hashed to some Integer. For purposes of this research simple java String.hashCode() is used. Thereby second task is splitted into two different subtasks: Shingling of lyrics and finding Jaccard similarity of song regarding all other songs in dataset.

Shingling songs

Continuous words are grouped into one object, k-shingled is an object from sequential k words, it is also can be described as Kgram, if there is taken consecutive 2 words then bigram, and so on. The arrangement of k-shingles of dataset with n words is considered as n k. It takes space O(kn) to store them all. In case if k is little, this is certifiably not a high overhead. Moreover, the space goes down as things are rehashed. Algorithm for shingling is simple:
First, lyrics converted to lowercase. Then all non-letter characters are removed. Third, sub-sentences of 5 words are created. Finally, these sentences are hashed and outputted.
Code below 3.4 is in Pig script, which allows hashing shingles despite it's amount.

```
REGISTER bigdata.jar;
all_data = LOAD 'songdata_tabbed.csv' USING PigStorage('\t')
    as (singer:chararray, song:chararray, lyrics:chararray);
min_hashed = FOREACH all_data GENERATE CONCAT(singer, '|', song),
                kz.sdu.bdt.pig.MinHasher(lyrics) as hashes;

dump min_hashed;
```

Figure 3.4: Hashing shingles

Jaccard similarity

To calculate jaccard similarity, *cartesian product* was used for all hashed shingles in joiner, and only those lyrics that got jaccard similarity higher than 0.7 have been taken into account as the most similar pairs. Pig and Spark has predefined function for finding cartesian product, whereas in MapReduce it needs to be implemented from scratch.

```
REGISTER bigdata.jar;
all_data = LOAD 'song_hashes' USING PigStorage('\t')
                as (song:chararray, hashes:chararray);
all_data2 = FOREACH all_data GENERATE song as song2, hashes as hashes2;
crossed = CROSS all_data, all_data2;
jac_sim = FOREACH crossed GENERATE song, song2,
    kz.sdu.bdt.pig.JaccardSimilarity(hashes, hashes2)
                                        as jaccard_similarity:double;
result = FILTER jac_sim BY song != song2 AND jaccard_similarity > 0.7;
dump result;
```

Figure 3.5: Jaccard similarity with Pig

3.2.2 Experimental results

Popular word among singers with Pig

Figure 3.6 shows time distribution in seconds. As it can be seen launching distributed jobs on small dataset provides no advantages over local mode. This might happen due to overhead of running task on distributed systems. Very likely that situation changes drastically on huge datasets.

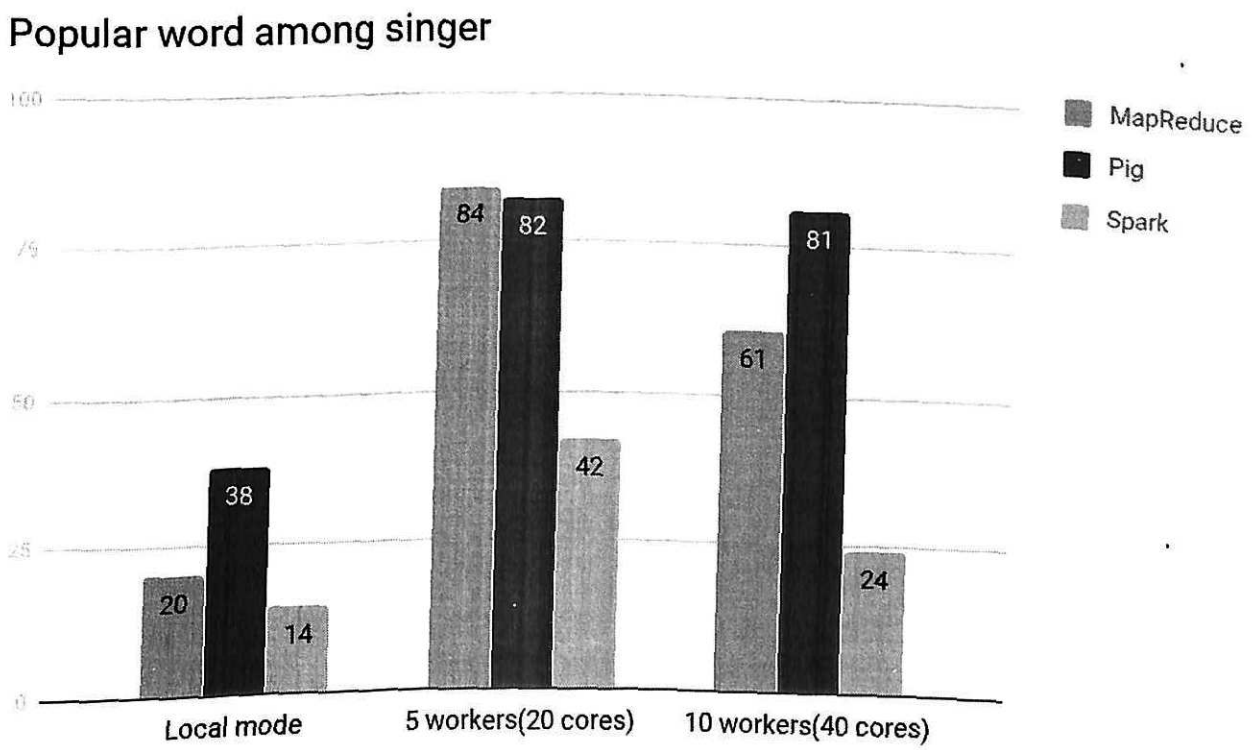

Figure 3.6: Time for "Popular word among singer" task

Table 3.1 shows the sample of the result of running first task, which lists the most

repeated word in all songs for every singer. There are totally 643 singers in the dataset.

Table 3.1: Result of "Popular word among singer" task

n Sync	love(298)
ABBA	love(189)
Ace Of Base	love(112)
Adam Sandler	like(74)
Adele	love(161)
Aerosmith	yeah(401)
Air Supply	love(514)
Aiza Seguerra	love(60)
Alabama	love(367)
Alan Parsons Project	know(78)

Shingling songs

Figure shows that due to overhead of running jobs on distributed systems all tools perform better on local mode. However it can be seen that with increase of worker cores processing time decreases for all tools except Pig.

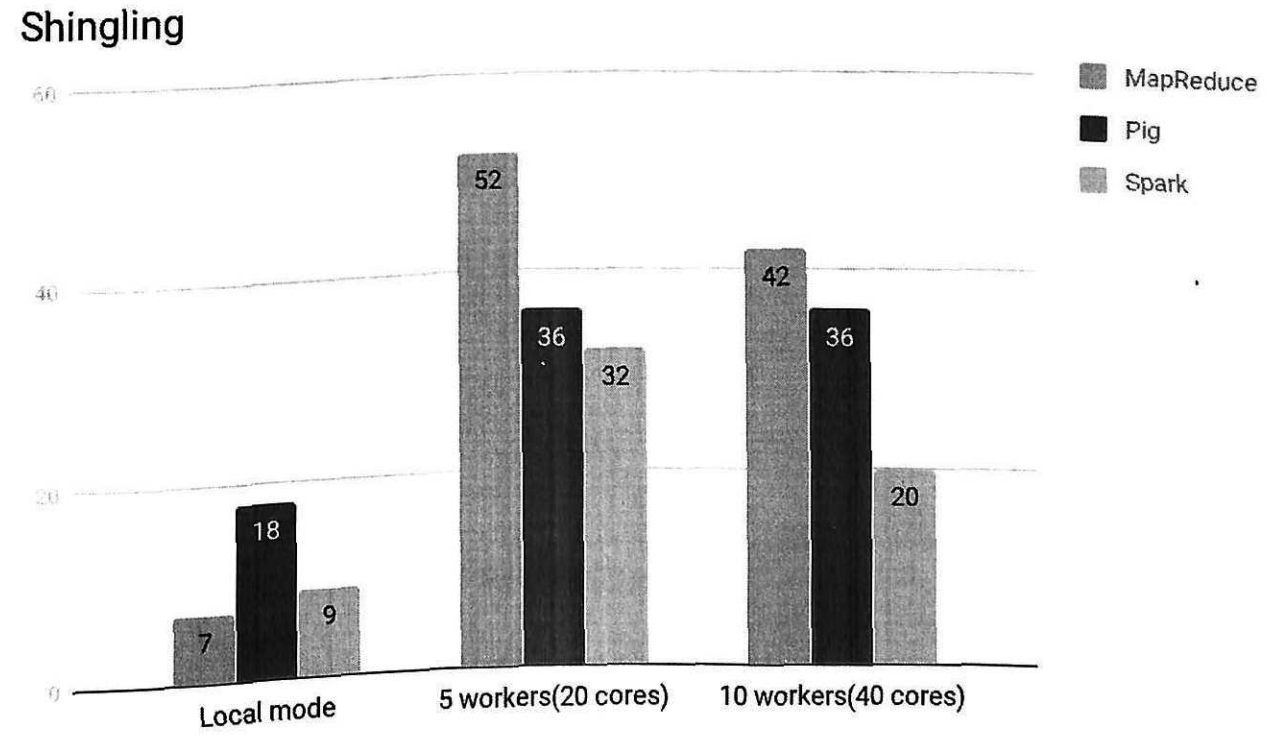

Figure 3.7: Time for Shingling task

Jaccard similarity

While previous two tasks took seconds for calculation, finding Jaccard similarity takes hours because each job needs to compare each hashed shingled sets with others, by applying [29] formula. It means that this task requires a lot of calculations. This is where power of distributed systems shines. Figure 3.8 shows that distributed systems perform more that 2 times better for MapReduce and Spark. More processing power added, less time required to process task. First surprise here is that MapReduce performs better on 10 workers than Spark, but Spark outperforms MapReduce on lesser workers. Reason might be that Spark limits processing power to some percent of available cores, whereas MapReduce takes all free processors.
Second interesting outcome is that Pig takes same time to process the task regardless cluster size. Problem is in default configuration where Pig launches only two reducers to complete job. This nullifies all advantages of distributed processing. Therefore Pig requires additional configuration to run smoothly.

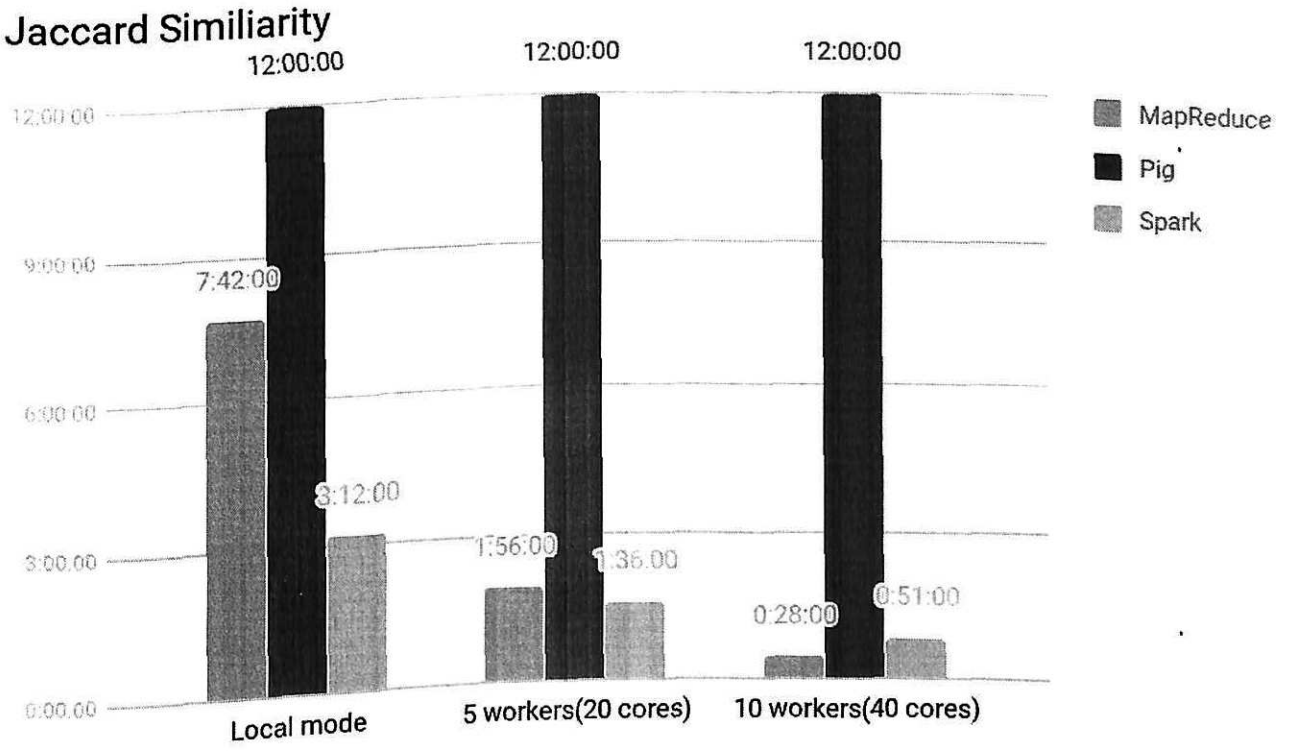

Figure 3.8: Time for Jaccard Similarity

Sample of results of finding plagiarism between singers is given in Table 3.2

Table 3.2: Result of running Jaccard similarity task

ABBA - We Wish You A Merry Christmas	Harry Belafonte - We Wish You A Merry Christmas
Kenny Chesney - Pretty Paper	Randy Travis - Pretty Paper
Rihanna - Golden Girl	Chris Brown - Golden Girl
Lady Gaga - Nature Boy	Natalie Cole - Nature Boy
Lady Gaga - White Christmas	Vince Gill - White Christmas
Lady Gaga - Winter Wonderland	Faith Hill - Winter Wonderland
Johnny Cash - Cindy	Nick Cave - Cindy
Avril Lavigne - Imagine	The Beatles - Imagine
Backstreet Boys - Cinderella	Lionel Richie - Cinderella
James Taylor - Santa Claus Is Coming To Town	Harry Connick, Jr. - Santa Claus Is Coming To Town
Rihanna - We Found Love	Coldplay - We Found Love

All three tasks with amount of workers with time consumed is written in table below. Green labels indicate the fastest tool for each task. So in firsts simple word

	Popular word among singer	Min hash	Jaccard similarity
Local mode			
MapReduce	20s	7s	7.7h
Pig	38s	18s	12h
Spark	14s	9s	3.2h
5 workers(20 cores)			
MapReduce	84s	52s	1.8h
Pig	82s	36s	12h
Spark	42s	32s	1.6h
10 workers(40 cores)			
MapReduce	61s	42s	28m
Pig	81s	36s	12h
Spark	24s	20s	51m

Figure 3.9: Comparison by time taken for processing on each tool

count task Spark calculates faster than other two. In constructing hashes from shingles Spark works faster while using 5-10 cores due to its iterative processing ability. In the last task where computing is needed Spark performed better in local

mode and with 20 processing cores, whereas with 40 cores Hadoop MapReduce processed data 2 times faster than Spark. Red labels indicate the worst tool in each situation. Data from Figure 3.9 shows code amount for each mentioned tool for every task. MapReduce takes a lot effort from data analyser, it requires writing more than twice of code of Spark, and around 18 times more lines of code than it could be written in Pig for current task. This could be because Pig is scripting language, and needs minimum number of code lines.

Table 3.3 shows code amount for each mentioned tool for every task. MapReduce takes a lot effort from data analyser, it requires writing more than twice of code of Spark, and around 18 times more lines of code than it could be written in Pig for current task. This could be because Pig is scripting language, and needs minimum number of code lines. Also, code written for Spark was in Java, if Python would have been selected, number of code lines decrease.

Table 3.3: Amount of code needed for each tool

Code amount	Popular word among singer	Min hash	Jaccard similarity
MapReduce	148	88	128
Pig	16	5	7
Spark	56	34	43

3.2.3 Experimental work 2

For the following in-class exercises was used pseudo-distributed mode of Hadoop which allows running MapReduce jobs on a single node where each Hadoop daemon runs on a different Java process. Total number of tasks was 15 for one semester, which is the duration of this course. The MapReduce programs that will be presented in the next sections make use of three datasets:

- sdu-portal.log file;
- 2 files with information about vectors;
- twitter dataset;
- 2 files about airlines : 2008 (csv) , airports.csv.

Sdu.portal.log file contains which pages are being accessed, by whom, and when (date, by using which browser) from my.sdu.edu.kz web server. Each line of the

Twitter dataset contains the following comma separated values about twit ID, date and time, sender username, twit itself . Airline on-time performance dataset consists of flight arrival and departure details for all commercial flights within the USA, from October 1987 to April 2008. There are nearly 120 million records in total, and takes up 1.6 Gb of space compressed.

1. Implementing MapReduce(in stand alone mode) program to:

- Count URL access frequency of sdu-portal.log;
  Mapper and Reducer is executed in one java file, so that output of mapper goes to reducer directly. In mapper stage by using Pattern and Matcher classes, all HTTP codes are found like in Figure 3.10: In Shuffle and Re-

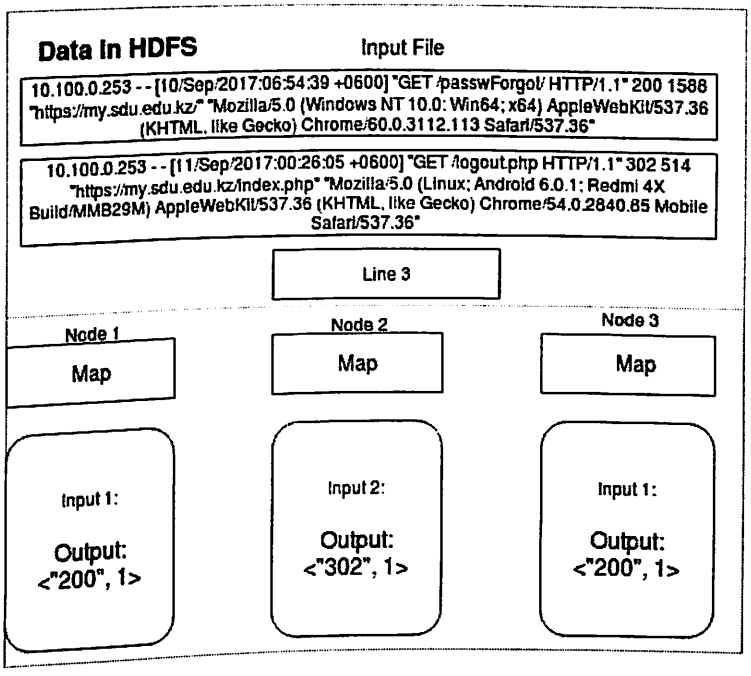

Figure 3.10: Input and Output of Map task, shuffle stage

duce stages all same keys are collected and their values summed up. Since frequency should be counted, second Reducer task is to divide each counted value to the total URL numbers.

- Find in which period of time portal is loaded most, output is a list of all hours from most accessed to least

- After extracting Date, hours are sent to Reducer with their values, where each hours value will be counted.

2. Inner product of two sparse vectors

Input files: two vectors data (A.txt, B.txt)

Output of the first MapReduce job should be multiplications of corresponding elements.

To solve this problem, the first step is preprocessing in mapper step by joining two sparse vectors data into one file as key and value. In Reducer, list is used where it checks if there exists key: Next task is to calculate sum of all elements

```
#!/usr/bin/python3

import sys

key=''
result={}

for l in sys.stdin:
        rows=l.strip().split(",")
        if len(rows)>1:
                key=rows[0]
                value=float(rows[1])

        if result.get(key):
                result[key]=result[key]*value
        else:
                result[key]=value

for i in result:
        print(str(i)+ ','+str(result.get(i)))
```

Figure 3.11: Reducer code for vector

by using aggregate function.

3. Twitter dataset [30] Join trending word count and total word count and percentage for each day from twitter dataset. Here Streaming API is used to write in Python. Hadoop Streaming enables commands to be used as mappers and reducers.

- Given twitter data needs to be extracted by each word in that date, so that each line would contain date and word from twit separately, and number 1(to count in reducer).
- The job of Reducer will be to count the occurrence of word for each date(*out1*).
- Next mapper's task is to take output from previous reducer (date, word, occurrence of word for that day) as input; and map only words with their values. Reducer is the same, however counts total occurrence of that word without date(*out2*).
- Last mapper sends two inputs (*out1, out2*) to reducer, and by joining by

key as words, divide the number of each word by days to the total number of that word.

4. task 9: By using Pig LOAD sdu-portal.log data and find link with maximum number of accesses.

First by using load command in Pig SQL-like scripting language we load log file separating by blank spaces. Then from each line we take url address name (10'th field) and group line by value of url address field. In Pig it is needed to generate group (as group_name) where count function can be applied to taken url address names. Final step is sorting: order last output by result from count function above in descending order. There are total 5 lines of code.

5. task 9.2 :LOAD 2008_flights.csv and output distinct flights sorted by flight distance

By using Pig LOAD command 2008.csv file is loaded with comma delimiter. From all fields only need to select origin of flight, destination of flight and distance of flight converted to integer. Pig has built in function DISTINCT to show only unique flights. The output of DISTINCT is just ordered by distance field in descending order. Total 5 lines are used to solve this problem, including loading and showing(dump) the results.

6. task 10.1 by using Hive- Create Table of Airlines and Load Hive 2008.csv; Find number of outgoing flights for each airport with airport name.

First part of task is - sql query to create new external tables with all fields from 2008.csv and airports.csv. with correct data types. These two tables can be listed in one sql, which then goes to python code to be parsed and joined through *iata* field and saved in dictionary. As a key value pair name of airport outgoing flights

```
for line in sys.stdin:
    line=line.strip().split(',')
    if len(line)>7:
        Origin,FlightNum=line[16], line[9]
        dict_lict[Origin]=dict_lict.get(Origin,0)+1
    else:
        iata, airport=line[0],line[1]
        if dict_lict.get(iata):
            dict_lict[airport]=dict_lict.pop(iata)


for key, value in sorted(dict_lict.items(), key=lambda item:(item[1], item[0]))[::1]:
    print("%s:%s" % (key, value))
```

Figure 3.12: Sample pyspark code for finding number of outgoing flights

from this airport goes.

7. Implementing similar tasks like 5,6 by using Spark (+ Find all unique flights and their distances + average number of flights per day.)
After loading 2008.csv file into pyspark we group all flights by origin and count number of flights (N) for each origin place:

- To find average number of flights that flew every day we divided total amount of flights for each origin place(N) into 365.
- To find unique flights and their distance we need to select Origin , destination and distance from all data, then use function distinct()

8. By using api HiPi, find RGB histograms for each images, sort histograms by color intensity

3.2.4 Experimental results

To discover to which extent learning objectives were achieved and to evaluate the integration of proposed tools to curriculum of master and PhD students, we used an end-of-class survey for participants in Big Data analytics course. While this was a small class, the results are promising. The survey provided students with a number of detailed questions on the module. First, we asked about their levels of proficiency on Java, Linux, and Python to understand level of undergraduate studies on Hadoop MapReduce. As given in Table 3.4 below, students enrolled the course with mostly intermediate knowledge to understand the programming aspects of MapReduce and Linux file system for HDFS data locality.

Table 3.4: Level of Proficiency

Java	Python	Linux
60% intermediate	46% intermediate	77% intermediate
15% advanced	15% advanced	

Next, the survey asked to estimate the time students spent on completing each assignment to determine whether the workloads of the assignments were appropriate.

Table 3.5: Time to Complete (1: less than 3 hours, 2: 4 to 8 hours, 3: 1 to 2 days, more than 3 days).

Activity	**Time taken**
LetterCount based on WordCount	100% -less than 8 hours;
2. Count URL access frequency with Java; 3. Inner product of two sparse vectors	75%-less than 8 hours(1,2):
4. Period of time when portal is loaded most	84%-less than 8 hours
5. Working with twitter dataset with Python	67%-less than 8 hours
6. RGB histograms for each images, sort histograms 33%-1-2 days; 50%-more than 3 days	25%-less than 8 hours;
7. Counting URL access frequency using Pig; 8. Distinct flights sorted by flight distance using Pig; 9. Number of outgoing flights for each airport with airport name using Hive; 10. Average number of flights per day using Spark	84%-less than 8 hours
11. All unique flights and their distances using Spark (67% within 3 hours , 33% 4-8 hours)	100%-less than 8 hours
12.Find the shortest backup flight for each unique flight using Spark 42%- more than 1 day	58% less than 8 hours;

As well as this, for each task was a question about complexity level of tool they needed to use exact for that task and complexness of task implementation itself. First three tasks were easy relatively, tool usage might took more time for implementing. For task 4 and 5, students (85%) responded this task as moderate/difficult, however tool usage was easy. Most difficult assignment for students was task 6, where 92% of them agreed with complexness of tool and assignment itself. The score distributions for the next 4 assignments were very similar: simplicity of assignment and tool usage in tasks 7,8,9,10,according to students, was 92%. Assignment where students were asked to find the shortest backup flight for each unique flight using Spark was considered complex for implementing(83%), however easy for tool usage(92%).

In the open ended questions, we asked them to write their opinion about which of the given assignments would be preferable to solve in Hadoop MapReduce, Apache Pig or Apache Hive, or Apache Spark based on their experience. As a preferable assignments for Hadoop MapReduce questions students voted mostly for tasks 1-6; also for the questions "In your opinion, which task is better to solve with Apache Pig/Hive/Spark?", we received similar responses like:"all tasks which related to joining and counting huge data set", "Data preprocessing", "When you want to get faster computing ", " Almost all", "When you have tables with data", "Where data is structured".

The last option in survey was to evaluate their knowledge on each learned big data tool after passing this course. Student feedback about the course was consistently high in measures of relevance and usefulness, quality of the lecture materials and appropriateness of assignments. More interestingly, students reported strong growth during the course: half of them felt themself as intermediate user of Hadoop and Spark.

4. Comparing Results

Pig looks as total outsider on these tasks. The only advantage of Pig over other tool is abstraction. Pig latin very concise language and needs a few lines of code for all tasks. Pig can be considered only in cases when there is no programmers available. Spark is considered best all around tool to process big data due to its in-memory caching capabilities. However results of research show that in some cases MapReduce can outperform Spark. Spark provides greater abstraction level than MapReduce, thus requires much less code to complete tasks. Even though Spark showed worse results in some cases, for most tasks it performed the best. Adding here abstraction level and amount of languages available for working with Spark, makes Spark most preferable tool for processing such tasks out of compared ones. However, all experiments were done on tools without any tuning. Adjusting launching parameters can radically change whole picture. Optimizing settings for tools and comparing them for performance with new conditions could be the next future work.

Examples are used in this paper were solved by master and PhD students in standalone and pseudo-distributed mode of Hadoop. Since the size of datasets were different, comparison by time is not efficient. Discussed examples gives good opportunity to learn big data analytic tools. Graduating students may gain employment in companies that already use BDMSs or are eager to use them to get better insights from the large datasets they manage. Thus, it is important to integrate the study of these systems as part of the computing curricula.

However, course suffers from some limits like: absence of truly big datasets of TB or larger; resources like multiple nodes. Nevertheless, we feel that students gain practical experience with big data tools, and their feedback indicates that they agree.

This course differs from similar courses at other institutions in its wide range of

topics from small data analysis and visualization to big data processing. For a more focused course including data analysis components, review Linh B. Ngo [9], developed a course focused on Map-Reduce. Here some future works based on existing research is discussed. Map/reduce job of Hadoop are long running jobs that take minutes or hours to complete.. Chosen tools to solve this problem are essential, due to the fact that proper tools reduce cost of the project and ease of support. Also, implementation of each tool will be discussed in comparison of others. Future works will be dedicated to solving more analytical problems using other Big data tools and comparing their results.

5. Conclusion

The paper contains relevant research related to the review and analysis of available programming languages and distributed analytical solutions in the field of big data analysis. The object of the study was big data. The subject of the study were tools for analyzing big data, The aim of the work was to compare modern tools for working with big data, in order to compare their performance and effectiveness in integrating to computing curriculum. To achieve this goal, in the work, the following tasks were formulated and solved:

- Reviewed software tools for analyzing big data;
- Reviewed approaches to storing and processing big data;
- The selection of software tools for the analysis of big data for diverse tasks;
- The analysis of the process of preparing and processing data;
- Preferred platform for students for learning big data;
- Completed testing and comparison of data analysis tools.

The work consists of their introduction, three chapters, conclusion and list of references. In the introduction, the object, purpose and objectives of the study are stated. In the first chapter of the work, a review of programming languages and software tools for analyzing and visualizing big data was made. The chapter describes various types of tools for analyzing big data in different areas. The analysis made it possible to determine which of them is more popular than others. It was noted that R is a common programming language for use in data analysis, whereas Spark is considered best all around tool to process big data. However results of research show that in some cases MapReduce can outperform Spark. The second chapter provides an overview of Apache platforms, the process of using

big data analysis tools for analyzing big data, by making theoretical comparison between selected tools. In the third chapter all experimental work is discussed:

- Overview of problems that are solved using Hadoop, Pig, Hive and Spark;
- Qualitative research was applied by collecting survey from master and PhD students about solving given analytical tasks and learning outcomes from the course;
- Performance comparison of platforms by giving them the same initial settings on same problem and their results;

Based on survey, structured data, data in tables are easier analyzed using Hive and Pig. Hadoop and Spark are preferred more than other two by students who have more programming skills. Sometimes Spark showed worse results in some cases, however for most tasks it performed the best. Hadoop was better performing for tasks where less iteration but more memory is needed. Adding here abstraction level and amount of languages available for working with Spark, makes Spark most preferable tool for processing such tasks out of compared ones. In conclusion, the results of the study are summarized and conclusions on the work are formed.

References

[1] T. Siddiqui and M. Al Kadri. "Big data analytics on the cloud". In: *International Journal of Emerging* (2015).

[2] P. Chen and C. Zhang. "Data-intensive applications, challenges, techniques and technologies: a survey on big data". In: *Information Sciences* 275, 2014 (2014), pp. 314–347.

[3] A. Gramfort F. Pedregosa G. Varoquax. "Scikit-learn: machine learning in python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[4] N. Matloff. *Art of R programming.* 373. No Starch Press, 2011.

[5] D. Toomey. *R for Data Science.* 347. Packt Publishing, 2014.

[6] *SAS 9.4 intelligence platform: data administration guide.* 407. SAS Institute Inc, 2015.

[7] SAS Institute Inc. *Getting Started with SAS Enterprise Miner™ 14.1.* 94. SAS Institute Inc, 2015.

[8] SAS Institute Inc. *SAS Forecast Server.* 4. SAS Institute Inc, 2015.

[9] Dr. Urmila R.Pol. "Big Data Analysis: Comparison of Hadoop MapReduce, Pig and Hive". In: *IJIRSET* 5.6 (2016).

[10] Dilpreet Singh and Chandan K. Reddy. "A Survey on Platforms for Big Data Analytics". In: *Journal of Big Data* 1.1 (2014), p. 8.

[11] Santosh Kushwaha Paritosh Goldar Yogesh Rai. "A Review on Parallelization of Big Data Analysis and Processing". In: *IJETCSE* 23.4 (2016), pp. 60–65.

[12] Srinidhi H K. G. Srinivasa Siddesh G. M. *Network Data Analytics: A Hands-On Approach for Application Development.* 398. Springer International Publishing, 2018.

[13] J. M. Reed Y. N. Silva S. W. Dietrich. "Integrating big data into the computing curricula". In: *45th ACM Technical Symposium on Computer Science Education* (2014), pp. 139–144.

[14] Joshua Eckroth. "Teaching future big data analysts: Curriculum and experience report". In: *IEEE International Parallel and Distributed Processing Symposium Workshops* (2017), pp. 346–351.

[15] Eeti Jain A.V. Hazarika. *Performance Comparision of Hadoop and Spark Engine.* I-SMAC, 2017, pp. 671–674.

[16] H. Simpson. Dumb Robots, 2012.

[17] D. Toomey. *R for Data Science.* 347. Packt Publishing, 2014.

[18] E.Derclaye D. Gervais. "The scope of computer program protection after sas: are we closer to answers?" In: *European Intellectual Property Review* 34 (2012), pp. 565–572.

[19] SAS Institute Inc. *SAS ® Model Manager.* 4. SAS Institute Inc, 2015.

[20] D. Sarkar. *Microsoft Sql Server 2012 with Hadoop.* 83. Packt Publishing, 2013.

[21] Choi H Lee K-H Lee Y-J. "Parallel data processing with MapReduce:a survey". In: *ACM SIGMOD Record* 40.4 (2012), pp. 11–20.

[22] Huang Lu. "Research on Hadoop Cloud Computing Model and its Applications". In: *Third International Conference on Networking and Distributed Computing* (2012), pp. 61–64.

[23] Bruce Brown Dirk deRoos Paul C. Zikopoulos. *Hadoop for Dummies.* 2014.

[24] *Apache Pig - Architecture.* `https://www.tutorialspoint.com/apache_pig/apache_pig_architecture.htm`.

[25] *Apache Spark.* `http://spark.apache.org/`.

[26] Won-Ki Jeong Woohyuk Choi. "IEEE 5th Symposium on Large Data Analysis and Visualization". In: *School of Electrical and Computer Engineering* (2015).

[27] P. Sudheer N. Pushpalatha. "Data processing in Big Data by using Hive Interface". In: *International Journal of Advance Research in Computer Science and Management Studies* 3.4 (2015), pp. 272–272.

[28] *Apache Hadoop.* `https://hadoop.apache.org/docs/r3.2.0/hadoop-project-dist/hadoop-common/SingleCluster.html`.

[29] *Apache Hadoop.* `https://neo4j.com/docs/graph-algorithms/current/algorithms/similarity-jaccard/`.

[30] *Twitter Dataset.* `https://www.kaggle.com/kazanova/sentiment140`.