**A Report submitted in partial fulfilment of the regulations governing the award of the Degree of BSc (Honours) Computer Science at the University of Northumbria at**

**Newcastle**

**Project Report**

**A performance comparison between Apache Spark and a standalone equivalent**

**Sarah Nightingale**

**2015/2016**

**General Computing Project**

**Authorship Declaration**

**Acknowledgements**

**Abstract**

Essentials of the report, nature of, context, work done, conclusions.

No more than a page

**Contents Page**

**Introduction**

**Background to project**

**Plan of Work**

This project will make a performance comparison between an Apache Spark implementation of an algorithm, and an identical machine learning algorithm implemented standalone using python. These implementations will then be tested using three datasets of varying sizes, gathering the speed of the tests, the average time taken for the data to be analysed, and the standard deviation of the speed of the test completion. The benchmarking results from these tests will then be compared to give insight into the effects of scaling on both approaches.

The results of this project will help to provide insight into reasonable approaches of analysing data of varying quantities, and into a sensible method of applying these applications.

**Analysis**

**Literature Review**

As this project aims to make a comparison between Apache Spark and a standalone version of the same algorithm, this review of relevant literature will be used to investigate the ways both of these pieces of software have been used in the past, and take into consideration the types of data they have been used to analyse. This will help to ensure all previous research is considered, the results of which can be used to enhance this report. Without the use of previous research done in this area this report will not be able to make a valid comparison, so any data gathered during this literature review must be wholly considered and brought forward into the final comparison.

**Big Data and Data Mining**

‘*It’s not easy to measure the total volume of data stored electronically, but an IDC estimate put the size of the “digital universe” at 0.18 zettabytes in 2006 and is forecasting a tenfold growth by 2011 to 1.8 zettabytes. A zettabyte is 1021 bytes, or equivalently one thousand Exabyte’s, one million petabytes, or one billion terabytes. That’s roughly the same order of magnitude as one disk drive for every person in the world.’* – Tom White

Big Data can be described as “*data that exceeds the processing capacity of conventional database systems. The data is too big, moves too fast, or doesn’t fit the strictures of your database architectures*.” (Big Data Analysis: Andrej Trnka, July 2014), however there is no set definition of what is considered big data. As technologies advance and new software is developed what may have today been considered ‘big data’ may soon become easier to deal with and new, even larger and more unordered data sets will take over as the new ‘big data’. “*There is a lot more data, all the time, growing at 50 percent a year, or more than doubling every two years*” (The Age of Big Data, Steve Lohr). With increases in data at this rate and heavier dependability from society on such information, reliable methods must be found to analyse this data successfully, and gather the information we require from it.

Big Data can be gathered from a multitude of different sources and is embedded in every part of modern day society. As quoted by McKinsey Global Institute, “*Big data—large pools of data that can be captured, communicated, aggregated, stored, and analyzed—is now part of every sector and function of the global economy*”. With the development of Smart technologies and Smart Cities the amount of diverse data being gathered is only set to increase, growing in both size, and variety.

Without being able to successfully analyse this data and draw information from it, all of this data is useless at it means nothing in its raw format. In order to gather this information from the data a technique known as data mining can be used.

*“Data mining is the extraction of implicit, previously unknown, and potentially useful information from data.”* (Data Mining: Practical Machine Learning Tools and Techniques, Second Edition: Ian D Witten, Eibe Frank)

Data Mining (also known as data discovery) is the process of gathering and analysing information from sources of data for practical uses, whether they be financial or otherwise used in society. It is often related to finding correlations within large amounts of data. Data mining has the power to harness the data gathered and put it to its full potential. According to Jiawei Han - University of Illinois ‘*Every enterprise benefits from collecting and analysing its data: hospitals can spot trends and anomalies in their patient records, search engines can do better ranking and ad placement, and environmental and public health agencies can spot abnormalities in their data’*

One way that data mining is often used is for marketing purposes, companies such as Amazon store data on each customer’s spending patterns, and which items their customers are looking at or have previously purchased. They can then use this data to advertise similar products to the customer in an attempt to drive up their profits. It also allows the companies to send targeted promotions, and shows them which items are currently popular or are predicted to become popular and make warehouse adjustments accordingly. When data mining is used effectively and to its full potential it can create a multitude of opportunities for businesses, and is therefore a powerful tool, not just in modern day commercialism, but also in other fields.

**Data Mining Algorithms**

Data Mining can be performed by a number of different techniques and algorithms, all completing different tasks and drawing information from the data in different ways. ‘*Data Mining approaches are mainly compromised of statistical and machine learning algorithms*.’ (Data Warehousing- Shirley Becker 2002). The fundamental feature of a data mining procedure is the underlying algorithm that generates the decision support model.

A main consideration when deciding to use a data mining approach is which algorithm would be the most appropriate for the data being analysed, and the information that is needed to be interpreted. In 2006 the IEEE International Conference on Data Mining identified ‘the top 10 algorithms in data mining’. As found by them, the top data mining algorithms of the time were C4.5, the k-means algorithm, support vector machines, the Apriori algorithm, the EM algorithm, PageRank, AdaBoost, kNN: k-nearest neighbour classification, Naïve Bayes, and CART algorithms. All of these algorithms are still highly relevant in data mining and machine learning and form a good starting point for the selection of a relevant algorithm.

C4.5 was rated the number one algorithm according to the IEEE International Conference on Data Mining. It is an algorithm used for building decision trees from a set of training data, using the concept of information entropy. It was developed by Ross Quinlan in 1978 and began as a CLS-like program, which then evolved into ID3, and in turn into C4.5. C4.5 is used on classification tasks where the data to be analysed is in the form of a flat file, where ‘*all information about one object or case must be expressible in terms of a fixed collection of properties or attributes’* - J. Ross Quinlan. This restraint means that this algorithm is not particularly well suited to Big Data analysis, as this type of data can have such a varied structure.

Similar to C4.5 in the way that it takes the format of a decision tree is the CART algorithm, which is based on Classification and Regression Trees by Breiman (1984). According to IBM *‘A CART tree is a binary decision tree that is constructed by splitting a node into two child nodes repeatedly, beginning with the root node that contains the whole learning sample.’* CART differs from C4.5 in that tests in CART are always binary, however tests in C4.5 allow for two or more outcomes. Both C4.5 and CART algorithms are not ideal for the use with Big Data, due to the fact that decision trees become much more complex and difficult to understand with larger datasets. Regular pruning of these trees would be required to remove unnecessary data, but without this the predictions would become more and more inaccurate. C4.5 uses a method called pessimistic pruning. ‘Pessimistic pruning is quite fast and has been shown to provide trees that perform adequately. However, it is forced to use an estimate of error at any node in a decision tree which is not clearly sound.’ (Lawrence O. Hall et al)

Another algorithm feature in the top 10 by the IEEE is the k-means algorithm. This algorithm was designed and developed by a number of researchers, the main ones being Lloyd (1957, 1982), Forgey (1965), Friedman and Rubin (1967), and McQueen (1967). ‘K-means algorithm finds a partition such that the squared error between the empirical mean of a cluster and the points in the cluster is minimized.’ (Anil K. Jain). The K-means algorithm is a clustering algorithm, with K standing for the number of clusters which the user inputs to the algorithm. K-means attempts to classify numerical data points or observations into K-clusters. Another algorithm loosely linked to the K-means algorithm is the K-nearest neighbour classifier. This comes in at number 8 in the top 10 data mining algorithms and is used for classification and regression. K nearest neighbour or k-NN is one of the most simple machine learning algorithms and is a type of instance-based learning. It is a ‘*standard nonparametric technique used for probability density function estimation and classification’* according to W.E Henley and D. J. Hand in 1995.

Rated at number 3 in the list are Support vector machines. This is considered by the International Conference on Data Mining as *‘one of the most robust and accurate methods among all well-known algorithms’.* A support vector machine, also known as an SVM is a supervised learning model which performs classification and regression analysis. An SVM takes a set of training examples and builds a model from them, assigning new examples into one category or another. This makes this algorithm a non-probabilistic linear classifier.

At number 9 in the list is Naïve Bayes. Naïve Bayes is a statistical method for classification and can be used to solve diagnostic and predictive problems. It is a supervised learning algorithm that makes decisions based on the outcome of training data which it has been given. Naive Bayes has the ability to learn from stored data and can therefore make highly dependable and accurate decisions. It is also very quick to learn as with one pass over the data it can utilise that information. Naïve Bayes will be the algorithm used to test the functionality of Apache Spark and Map reduce in this project as it is a highly popular algorithm in data mining, and is ‘simplistic, elegant and robust’ (IEEE).

**Naïve Bayes Classifier**

A Naïve Bayes Classifier is based on Bayes Theorem:

**P(h|D) = P(D|h) \* P(h) / P(D)**

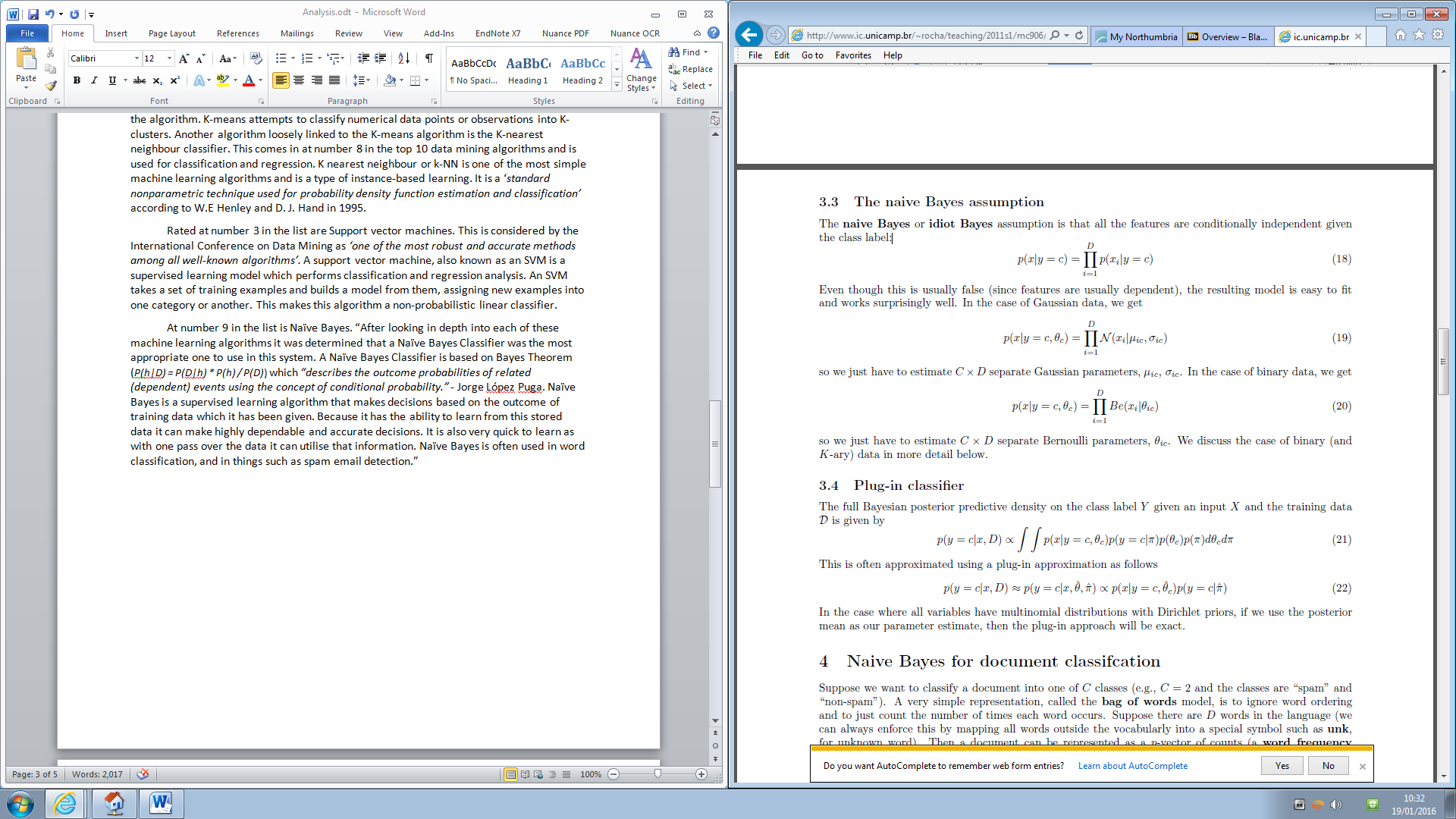
P(h) = prior probability of hypothesis h

P(D) = prior probability of training data D

P(h|D) = probability of h given D

P(D|h) = probability of D given h

Naïve Bayes “*describes the outcome probabilities of related (dependent) events using the concept of conditional probability.”* - Jorge López Puga. It was named after Thomas Bayes who proposed the initial Bayes Theorem, upon which this approach is based. Naïve Bayes is often used in word classification, and in things such as spam email detection. Naïve Bayes, also known as idiot Bayes, uses the assumption that ‘all the features are conditionally independent given the class label.

’ (Naive Bayes classifiers - Kevin P. Murphy, 2006)

Naïve Bayes utilises a Supervised Learning method, this entails using training data to teach the algorithm, allowing it to generalise class labels for the new unseen data in a rational manner. A Naïve Bayes Classifier uses prior probability to determine its result, this is due to the fact that it uses its prior knowledge and doesn’t take into account any of the observed data. This means that if 99% of the data states one thing and only 1% states the other, the system will assume any new data fits into the 99% category with a 0.01 probability of error.

Naïve Bayes remains one of the most relevant and preferred algorithms for use in machine learning, as can be shown by its consistent inclusion in literature in this field, spanning from its conception in the 1950’s up until today, with papers such as ‘[Risk classification with an adaptive Naive Bayes kernel machine model](http://www.tandfonline.com/doi/abs/10.1080/01621459.2014.908778) – J Minnier, M Yuan et al’ and others of a similar topic having been written in the past year. Naïve Bayes also remains more popular than boosted trees, Support Vector Machines and Max Entropy as while it can often be out performed by these techniques it is also much less computationally intensive and it requires a much smaller amount of training data. Another benefit Naïve Bayes has over these techniques is that it can learn its training data in a significantly smaller amount of time, making it very efficient. This also makes it good for use in real time systems as there is very little delay whilst the system is trained.

One type of Naïve Bayes is the Multinomial Naïve Bayes Classifier; this is a specific instance of a Naïve Bayes classifier which assumes a multinomial distribution for all of the pairs. This can be a rational assumption in some circumstances, for example for word counts in documents. Multinomial Naïve Bayes is often used for word frequency information in articles or news stories. ‘*Since every news article is dated, and thus has a number, the number token in the multi-variate Bernoulli event model is uninformative. However, news articles about earnings tend to have a lot of numbers compared to general news articles. Thus, capturing frequency information of this token can help classification*.’

Another well-known variation of Naïve Bayes is the Bernoulli model of Naïve Bayes. This is also most commonly used in text classification, however this method uses binary elements, using the value of 1 if an item is present in a text document, or 0 if the item is not present.

**Parallel Processing**

Parallel Processing is a method of processing in which multiple processes or programs are run at the same time, with the end goal of reducing the time taken to run the program. This is opposed to serial processing which is used on older computer systems, where each process was run one after another, leading to long processing times whilst waiting for the operations to complete.

Parallel processing has been able to advance further in the past couple of years, expanding from usage solely in advanced computer systems to now being used generally, thanks to the evolution of multi-core processors. Parallel processing has also advanced in an effort to minimise power consumption and heat generation

**Types of Database**

Traditional databases, such as relational databases, struggle to analyse such large amounts of data as they cannot effectively scale to meet the required storage sizes, and cannot handle such varied data types.

One reason for this is that relational databases are very strictly structured, with all the data being stored in columns and rows with each column containing a certain datatype. Relational databases can allow for Referential Integrity, as tables are related to each other using a combination of primary and foreign keys. This is important in maintaining the validity of the data in this kind of database. In non-relational databases however the schema is so flexible that there is no such thing as the referential integrity that occurs in relational databases. In these kinds of databases, the data does not require JOINs between objects as it is highly de-normalised.

**Summary of Current Systems**

**Hadoop**

When considering which pieces of software to use in relation to Big Data, the option that appears the most often is Hadoop. Hadoop was created by Doug Cutting and has its origins in Apache Nutch, a web search engine. It is an open source framework which is used for the distributed storage and distributed processing of extremely large data sets. According to the Hadoop website:

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

Hadoop is an important piece of software for many reasons, the main ones being its ability to store and process huge amounts of any kind of data. This is particularly important with Big Data and the Internet of things as the data set needing to be analysed may contain a large range of data types and a huge volume of data. Hadoop also possesses a large amount of computing power, with the processing power increasing as more nodes are added. One of the main benefits of using Hadoop in a Big Data scenario is the flexibility that it offers. Because of the structure of the databases, there is no need to pre-process the data before it’s stored. This is an advantage of Hadoop over relational databases as it allows you to store as much data as you choose to and then choose what you want to do with it later. Hadoop can also store the unstructured data that is associated with big data, it does this in a fairly inexpensive manner whereas storing this kind of data in a relational database would be very expensive. Another way Hadoop is a cost efficient solution for handling Big Data is that it allows a system to grow freely and easily, requiring very little administration to do so.

Hadoop currently consists of four core modules:

Hadoop Common – the basic libraries and utilities used by the other modules of Hadoop,

Hadoop Distributed File System – ‘’*HDFS is a file system that is designed for use for MapReduce jobs that read input in large*

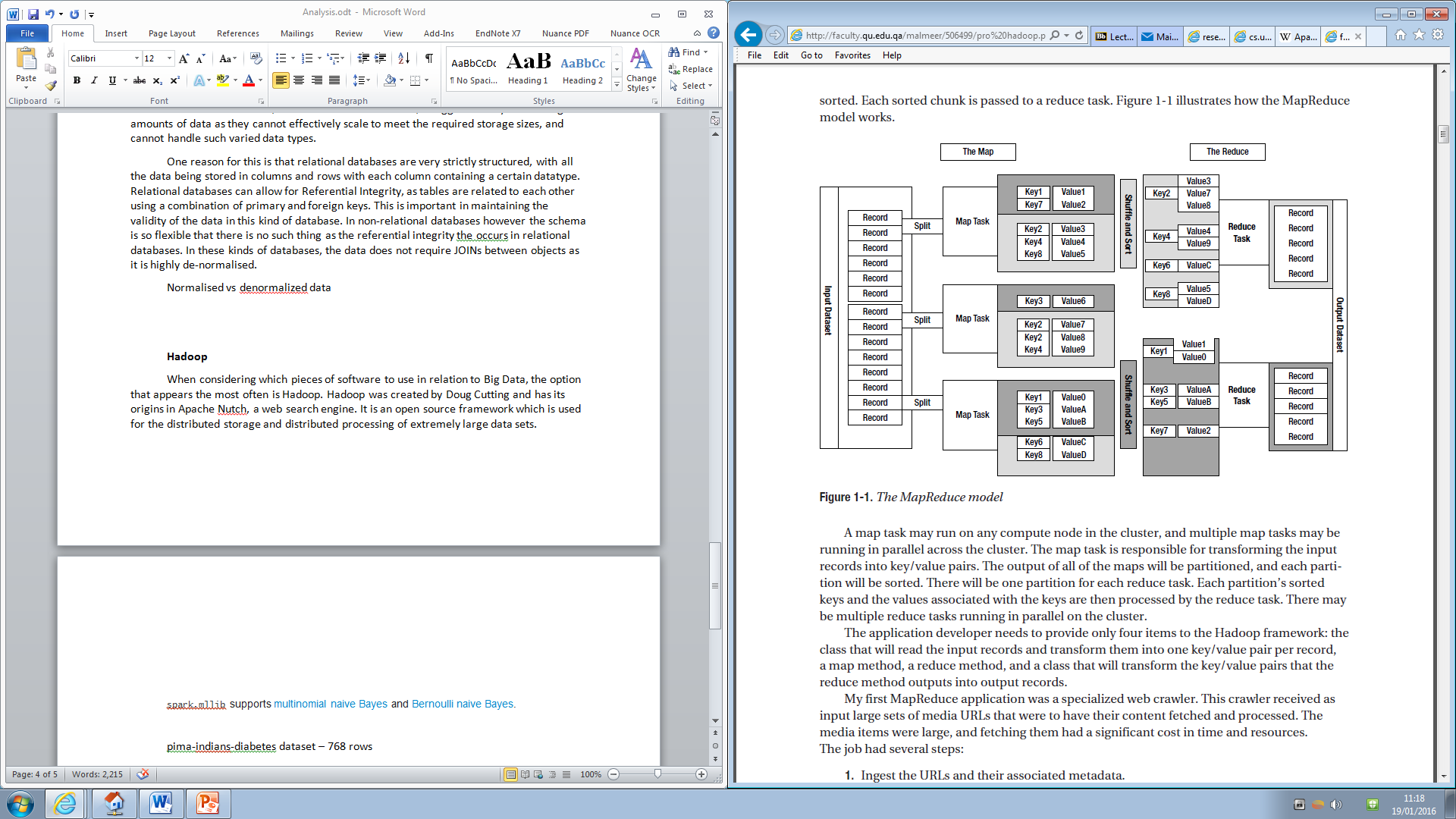
*chunks of input, process it, and write potentially large chunks of output.*’ (Pro Hadoop),

YARN (Yet Another Resource Negotiator) – the resource management handler for processes running on Hadoop,

MapReduce – ‘*a programming model and an associated implementation for processing and generating large data sets*.’ (MapReduce: Simplified Data Processing on Large Clusters: Jeffrey Dean, Sanjay Ghmeawat 2004).

**MapReduce**

MapReduce is one of the current pieces of available software that will be investigated during this project. MapReduce was created to allow for the processing of very large data sets using a parallel, distributed algorithm on a cluster. It works by allowing the user to specify a map function and a reduce function. The map function *‘processes a key/value pair to generate a set of intermediate key/valuepairs’* (MapReduce: Simplified Data Processing on Large Clusters) and the reduce function performs a summary operation, such as merging all of the values associated with the key specified in the map function. ‘*MapReduce automatically parallelizes and executes the program on a large cluster of commodity machines.’* – (MapReduce: A Flexible Data Processing Tool) The MapReduce run-time system handles all of the partitioning of the data, schedules the execution of the program across multiple machines, handles any failures that occur during execution, and deals with all inter machine communication. Because of this the system is a fairly easy one to learn and use, programmers without much knowledge or experience of large distributed systems can still utilise all of the functions it offers.



**Apache Spark**

According to the Apache Spark website “Apache Spark™ is a fast and general engine for large-scale data processing.” Apache Spark was released on 30th May 2014 and was created at the University of California, Berkley before being donated to the Apache Software Foundation. Spark supports various programming languages, such as Java, Python and Scala.

Apache Spark was chosen for use in this comparison for a number of reasons. The first reason is that Apache Spark is a relative newcomer in the Big Data Analysis world but has quickly become one of the fastest and most reliable pieces of software. To investigate why this is due to performance data will help to show the reasons behind the recent increase in its popularity.

One of the benefits of using Apache Spark is its ease of use and wide range of functionalities. Included in Apache Spark is MLlib, Spark’s own machine learning library. MLlib supports various different languages and features a variety of regression and classification algorithms such as K-means, Decision Trees, Linear Regression and Naïve Bayes. MLlib was developed as part of the MLbase project in 2012 and was packaged and released with Spark in 2013, it now has over 140 contributors from over 50 organisations. (MLlib: Machine Learning in Apache Spark-Xiangrui Meng et al). Previous testing has been done on the performance of algorithms when implemented in Apache Spark’s MLlib compared with Apache Mahout running on Hadoop’s MapReduce by Xiangrui Meng et al in 2014. This found that the MLlib implementations were much faster, finding that ‘*MapReduce’s scheduling overhead and lack of support for iterative computation substantially slow down its performance on moderately sized datasets*.’ As this performance test was done on version 1.4 of Apache Spark, this may not fully reflect the current performance of the MLlib functions, therefore this project will aim to continue this research using the current version of Apache Spark which is 1.6.

**Synthesis**

**Required Resources**

In order to implement the required tests a number of resources will be needed.

One consideration for this project was on what machine or environment to run the performance tests. The first option considered for this was to install the software on real machines with identical specifications. This was found to be impractical as it was difficult to access machines that it would be possible to install all of the software on, and also had the required identical specifications.

Another possibility, and the one that was the most viable for this project, was to set up a number of identical virtual machines on one real machine. These would be used to install all of the software onto and then to run each of the algorithms in their environments. In order to do this software such as VirtualBox, VMware, or Parallels would be needed. As VirtualBox offers a large number of features and is free to use this will most likely be the software used to implement these VM’s. It is important that these VM’s all have the same technical specifications and Operating Systems installed on them, this is so to avoid interference from the machines themselves as having slower or faster machines will greatly impact on the final results.

It was decided to use Ubuntu 14.0 as the Operating System for each of the Virtual Machines; this was because a Linux machine was required for all of the software needed, and due to previous knowledge of Linux commands, and familiarity with this operating system.

**Datasets Used**

In order to test the scalability of both implementations three datasets of varying sizes will be required. The first dataset chosen was the smallest dataset, this is the dataset provided by Spark to test a Naïve Bayes algorithm on. This dataset consists of 12 rows and 4 columns of integers which can be classified into 5 different categories. The data is separated by commas which are stripped from the dataset before splitting each column into individual integers.

The middle sized dataset that will be used is the ‘Pima Indians Diabetes dataset’ (UCI Machine Learning Repository, 1990). This dataset is 768 rows and has 8 attributes. Each row of data in the set can be categorised as either 0 or 1, representing whether or not the patient in question shows signs of diabetes. The distribution of the categories in the dataset are 500 members of the ‘0’ category and 268 members of the ‘1’ category. This dataset is also separated by commas which must be stripped before appending each individual value to an array for use. This dataset consists of a combination of both integers and floats so this must be taken into consideration when handling the data.

The largest dataset to be used is the ‘Wine Quality Dataset’ (UCI Machine Learning Repository, 2009). This data set consists of 4898 rows and has 11 attributes. Each row represents features of the wines which allow the data to be classified into a quality category ranging from 0 to 10. The attributes in this dataset are separated by semi-colons, this makes it slightly different from the two previous datasets, however it will be handled in a similar manner treating semi colons as the delimiter. This dataset also features a range of integers and floats and so must be handled accordingly.

**Requirements**

This project aims to compare the performances of two Naïve Bayes algorithms, one implemented in Apache Spark and one standalone implementation. Therefore the first requirement for this project is to implement a working Naïve Bayes algorithm in Apache Spark. This will also require a method to time the algorithm and calculate how long it takes for it to complete.

In order to compare the performance results a standalone version of the Naïve Bayes algorithm must also be implemented. This will be used to compare with the Apache Spark version to show the difference in performance between the two. Again, this will require a method to time the running of the algorithm.

Because these performance tests rely so heavily on the Naïve Bayes algorithms being identical an application must be created to compare the outputs of both of the algorithms. This will be written in python and will need to take the values outputted by each algorithm from the output text files and compare them to see if the predictions made by the algorithms are the same.

In order to ensure that both algorithms are given the same training data and testing data a separate application must be created to take each dataset, split it into individual attributes, shuffle the datasets to ensure the training data is representative of the whole dataset, and split the data into identical training and testing sets. This will then pass the training and testing sets to each algorithm which will be able to make identical predictions from this.

* Apache Spark implementation of Naïve Bayes.
* Standalone implementation of Naïve Bayes.
* Application to compare predictions of both algorithms.
* Application to shuffle and split dataset into appropriate training and testing data.

**Implementation of Standalone Naïve Bayes**

The standalone implementation of the Naïve Bayes Classifier was written in python, this was due to prior knowledge of this language and the fact that its Apache Spark equivalent would also be supported in python.

The implementation of this was based upon the Naïve Bayes formula which is: P(h|D) = P(D|h) \* P(h) / P(D). In this equation a number of values are required from the data in order to make a prediction. The first of these required values is the Prior Probability of the prediction being true, this is represented by P(h). To gather this value it was necessary to search through the training data and total up the number of times each category occurred. This was done like… (CODE GOES HERE)((MAYBE PSEUDO CODE))

The second value that is required is P(D), or the Prior Probability of attributes in the training data having occurred. This means how many times each attribute appears in a column of the training data, disregarding any of the other attributes or which category it is in. This can be calculated by counting how many times the attribute occurs, divided by the total number of rows in the training set. In the code this is found by (CODE, PSEUDO).

Also needed is the probability of h given D, this represents the probability of the category occurring based on the attribute in question. This can be found by counting the total number of a certain category occurring, and then counting how many times each attribute also occurs under that category. The total number of each attribute is then divided by the number of times that category occurs to give a value for P(D|h) . (CODE, PSEUDO)

The final calculation will work out how likely it is for the attributes in question to belong to each of the categories, from these outputs it will be seen which has the highest probability and therefore which category is predicted. This is called the posteriori probability of a hypothesis, h.

One problem with the Naïve Bayes algorithm occurs when an attribute never arises in a certain category, this is known as Zero Frequency. This is a problem as the Naïve Bayes model will then assign a 0 probability and this makes it unable to make a prediction. In order to counter act this, a value of ‘m’ can be added to the equation to ensure that a zero value never transpires. This is done by changing the algorithm to P(h|D) = P(D|h) \* P(h0 + m / P(D) + m. The value of m is arbitrary, however it must remain the same for all attributes in order for it to have no effect on the final prediction. In this scenario the value of m is set to 2 for each of the algorithms. (CODE, PSEUDO CODE)

**Implementation of Spark Naïve Bayes**

**Prediction and Accuracy Checker**

**Dataset Manipulator**

During the implementation of the accuracy and prediction checker it was noticed that the Apache Spark classifier and the standalone one both outputted different values, due to being given different training data and different testing sets. This made it difficult to ensure that both algorithms were doing the exact same thing with the data. In this scenario it was important to be able to confirm that both algorithms were identical as if this wasn’t the case then the final results and conclusions drawn could not be considered valid. In order to do this it was determined that both algorithms needed to be given not only the exact same datasets, but the exact same training and testing sets to work with. It was decided that the best way to ensure that this was happening was to create an external script that would split the data into the training and testing sets, while still shuffling the data beforehand to ensure an even spread of the categories and attributes.

(PSEUDO CODE)

**Testing**

**Evaluation**

**Evaluation of work done**

**Evaluation of project process**

**Test Results**

**Conclusions and Suggestions for further research**

**References**

Big Data Analysis: Andrej Trnka, July 2014

The Age of Big Data, Steve Lohr

Big data: The next frontier for innovation, competition, and productivity, McKinsey Global Institute (James Manyika Michael Chui Brad Brown Jacques Bughin Richard Dobbs Charles Roxburgh Angela Hung Byers), June 2011

Data Mining: Practical Machine Learning Tools and Techniques, Second Edition: Ian D Witten, Eibe Frank

Data Warehousing- Shirley Becker 2002

J. Ross Quinlan - Programs for Machine Learning

Decision Tree Learning on Very Large Data Sets Lawrence O. Hall, Nitesh Chawla and Kevin W. Bowyer

Data Clustering: 50 Years Beyond K-Means Anil K. Jain

A k-nearest-neighbour classifier for assessing consumer credit risk – W.E Henley and D.J. Hand, 1995

Top 10 algorithms in data mining - XindongWu · Vipin Kumar · J. Ross Quinlan · Joydeep Ghosh · Qiang Yang · Hiroshi Motoda · Geoffrey J. McLachlan · Angus Ng · Bing Liu · Philip S. Yu ·Zhi-Hua Zhou · Michael Steinbach · David J. Hand · Dan Steinberg

Jorge López Puga, Martin Krzywinski & Naomi Altman - Points of Significance: Bayes' theorem

Naive Bayes classifiers - Kevin P. Murphy, 2006

Pro Hadoop – Jason Venner

Hadoop: The Definitive Guide – Tom White

– J Minnier, M Yuan et al

Feature Subset Evaluation and Classification using Naive Bayes Classifier - G. Keerthika , D. Saravana Priya

A Comparison of Event Models for Naive Bayes Text Classification - Andrew McCallumzy, Kamal Nigamy

MapReduce: Simplified Data Processing on Large Clusters: Jeffrey Dean and Sanjay Ghemawat

MapReduce advantages over parallel databases include storage-system independence and fine-grain fault tolerance for large jobs. by Jeffrey Dean and Sanjay Ghemawat

MLlib: Machine Learning in Apache Spark-Xiangrui Meng et al

**Bibliography**

**Appendixes**

**Appendix 1 – Terms of Reference**

**Background to Project**

The title of this final year project is ‘A comparison between Apache Spark and Map Reduce on Hadoop’. During the course of the project I intend to make an in depth comparison between the functionalities of both Apache Spark and MapReduce running on Hadoop. I intend to do this by implementing a Machine Learning algorithm with each of these frameworks and comparing their performance, ease of use, their methods of data processing, how failure tolerant each of these systems are, and any other areas that I feel will help me form a relevant and fair comparison between the two.

The problem domain I will be working in is the Internet of Things domain. The Internet of Things refers to the connection of physical, everyday objects to the Internet, allowing data to be gathered from systems embedded into these objects, such as sensors. In recent years the expansion of the Internet has grown exponentially and new ways to use the Internet to benefit society have also evolved. One example of this in which I am particularly interested is the creation of Smart Cities. A Smart City is designed to use this connection to the Internet to enhance the quality of public services, and allow the city to engage more effectively with it citizens. The building blocks of a Smart City come from the implementation of Smart Objects, these are the everyday objects in the city which have been equipped with an embedded system and are connected to the Internet of Things infrastructure. A Smart Object is able to gather data from its sensors, store that data, and then interpret the information from that data, allowing it to take action based on the information found. An example of this could be a lamppost equipped with a sensor allowing it to detect how light it is outside, it could then interpret that data and depending on the levels of brightness outside, turn itself on and off when needed. This particular example would help reduce the energy footprint of a city by ensuring that the street lights were only on when it was necessary for them to be, and help to increase safety as streets would never be left unlit in the dark.

With the implementation of Smart Cities comes a large amount of data, coming from each of the Smart Objects within the city. Each of these Smart Objects will produce huge amounts of data on a daily basis and this data must be stored and analysed in some way. The data that would be gathered by these Smart Objects would be highly varied, ranging from streams of numerical data from sensors, to images and videos. The amount of data gathered will also continually increase over time as more Smart Objects are added to the city and as more and more data is obtained and stored. This data is known as Big Data.

Big data is a term that is used to describe an extreme amount of data that can be either structured, semi-structured or unstructured and has the potential to be mined for information. It is characterised by three main V’s, Volume, Velocity and Variety. In this case, Volume represents the size of the data and covers the continuing growth of these data sets. Velocity represents the speed at which the data is being generated, and the speed at which the data must be handled. This characteristic is particularly prominent in situations where the data must be analysed at real time. The final main characteristic is Variety, this covers the fact that data today can come in many different types of formats, all which must be handled in different ways and create an extra problem when considering how to manage this data. Two additional V’s can also be considered when defining the characteristics of big data, these are Variability and Veracity. Variability considers inconsistent flows of data, such as peaks where significantly more data could be gathered than at other times, for example in a Smart City situation this could be seen during a morning rush hour when a large amount of vehicles move into the city and begin parking in spaces equipped with sensors. The final V to be considered is Veracity, this looks at the quality of the data captured and whether or not it is accurate. If the data being stored and analysed is not correct then it is both a waste of storage space, and the conclusions drawn from it may in fact be incorrect, therefore ruining the purpose of analysing the data in the first place.

The main problem with Big Data comes in finding an effective way to analyse it. Traditional databases such as Relational databases struggle with certain key elements of big data analysis, the main drawback being their inability to scale effectively. The costs of storing the massive amount of data required in a traditional relational database would be unfeasible, and in order to attempt to keep costs at a minimum older raw data would need to be deleted. This would lead to an incomplete store of data and can negatively affect the validity of the information gathered from it in the long term. This could cost a company tens of thousands of pounds per terabyte of data stored, whereas systems using software such as Hadoop could cost as little as hundreds per terabyte. Another important issue with relational databases is their incapability of dealing with unstructured data. Relational databases work well with structured data, however as Big Data can be unstructured this causes problems. Hadoop uses a distributed file system to store the data that effectively creates a map of where data is stored on the cluster, this makes it able to very quickly deal with large amounts of unstructured data in a way that a relational database could never achieve.

I am intending to undertake this particular project as Big Data is something being generated at an ever increasing pace, and analysing this data is increasingly important. I think completing this analysis will show the differences between these two different methods and may show which piece of software is the best to be used for different data types and in different situations.

**Proposed Work**

My proposed work for this project is to implement a machine learning algorithm into both Apache Spark and MapReduce which will both be hosted on virtual machines. This will help me to form a comparison between the two data analysing methods and see which is the most effective.

In order to decide which of these pieces of software I consider the best upon completion of the project I intend to look into four main categories for comparison, these are the overall performance of each piece of software, the ease of use of each, their individual methods of data processing and what this means in regards to what data is best analysed using this data, and how failure tolerant each of these are. On top of these four categories I will also be looking into any other areas of comparison that become evident as I work on this project.

In implementing the machine learning algorithm with Apache Spark I will be using the python API provided by the software, this means all the code I do for this part of the project will be in python. I will also be using python with MapReduce.

**Aims of the Project**

1. To compare the features of Apache Spark and Hadoop’s MapReduce and give an overview of each, while considering which is the most effective system for big data analysis.

**Objectives**

1. Investigate Apache Spark and Hadoop’s MapReduce in the form of a literature review, looking into pieces of academic writing focusing on the general area of big data analysis and the current functionality and uses of the two pieces of software.
2. Create design documentation to plan what will be implemented on each piece of software.
3. Use python to implement a machine learning algorithm on Apache Spark. Implement the same machine learning algorithm on Hadoop’s MapReduce.
4. Make a comparison between the functions of both of the systems, including ease of use, functionality, performance and fault tolerance.
5. Write up a report detailing the results of the comparison between the two pieces of software.
6. Write the chapters of the report.
7. Evaluate the project as a whole.

**Skills Needed**

In order to complete this project to a high standard I will need a number of skills, some of which I already have, and some of which I will need to acquire as the project develops.

**Knowledge of python** – already obtained during placement year working on python scripts.

**Ability to use Linux command terminal –** Obtained during year on placement working on Fedora 14.

**Knowledge and Understanding of Machine Learning –** Will be acquired during the course of the project and cemented in semester two during which I will be undertaking the Machine Learning module. In order to gain knowledge of machine learning before the commencement of this module I have acquired the lecture slides that will be used during semester two and I will also conduct my own research and reading.

**Ability to use Apache Spark –** During the course of the project I aim to become familiar with ApacheSpark by undertaking the tutorials given for new users, and then adapting that knowledge to be able to implement my own algorithm. As this will be done in python I already have prior knowledge of the language.

**Ability to use MapReduce –** Again,I aim to learn this over the duration of the project by completing tutorials given for this piece of software and then using that understanding to implement other algorithms. This is also in python which I am already familiar with.

**Understanding of the theory of Relational Databases –** I gained a knowledge of relational databases during my first year of university while undertaking the Databases module. This has given me a basic understanding of how they work, which I will now develop through further reading and research.

**Sources of Information**

http://www.tandfonline.com/doi/abs/10.1080/10630732.2011.601117

http://www.sas.com/en\_us/insights/articles/big-data/big-data-and-iot-two-sides-of-the-same-coin.html

http://wolfweb.unr.edu/homepage/ania/NYTFeb12.pdf

http://www.intel.com/content/www/us/en/big-data/unstructured-data-analytics-paper.html

http://searchcloudcomputing.techtarget.com/definition/big-data-Big-Data

http://www.ibmbigdatahub.com/infographic/four-vs-big-data

**Resources**

For this project I will require the use of the open source software available for both Apache Spark and MapReduce. I will also require a Linux virtual machine for initial testing, and then multiple connected virtual machines to allow me to fully test both pieces of software.

**Report Structure**

**Abstract** The abstract will consist of a concise summary of the project. It will include some details on the subject area and the aims and conclusions drawn from the report as a whole.

**Introduction** The introduction will provide the background for the project. It will give the reasons for this work and introduce the project. Included will be the aims and objectives of the project. A brief summary of the work will also be encompassed into this section, as will the methods, scope and limitations of the project.

**Analysis**

**Background to the Project –** This chapter will clearly identify the problem being investigated and will include any necessary background material. This will include both technical knowledge and knowledge on the topic area.

**Literature Review -** The literature review will involve a critical review of some relevant literature which supports the formulation of this project. It will draw on the hypothesis of the project and use the literature to support it. It will look into literature on both Hadoop and Apache Spark, and investigate some regarding the wider internet of things scope and other related big data topics.

**Justification –** This will justify the method of comparing the two pieces of software, and will validate the tools used to do so.

**Synthesis**

The synthesis will include a description of the work undertaken over the duration of the project. It will cover the design, implementation and testing of the comparison. This will reference the literature review and help to support the hypothesis.

**Conclusions and Evaluation**

This will investigate the end results of the comparison between the two pieces of software. It will look into the strengths and weaknesses of both of the software, and of the method of comparison itself. I will also evaluate the project as a whole and consider what could be improved on to make a more in depth comparison, and to make it the most relevant it could be.

Finally I will compare with the initial aims and objectives of the project and determine whether these were fully met or not, and overall whether the project was a successful comparison.

**Marking Scheme**

**Project Type –** General Computing Project

**Project Report -**

|  |  |  |  |
| --- | --- | --- | --- |
| **Report Section** | **Marks** | **Chapters** | **Objectives** |
| **Abstract & Introduction** | 5 | 1, 2 | 1 |
| **Analysis** | 30 | 2, 3, 4 | 1, 2 |
| **Synthesis** | 30 | 5, 6 |  |
| **Evaluation & Conclusions** | 30 | 7, 8 | 5, 6 |
| **Presentation** | 5 | - |  |
| **Product** | 30 | - | 3, 4 |
| **Viva** | 10 | - |  |

**Project Plan**

