University of Hertfordshire
Computer Science Department

Master Project in Computer Science

What are some of the common problems encountered by organisations when analysing and managing Big Data, and can these be helped by using Distributed Computing?

By B. M. F. Scabbia

Web Version 3.2

https://www.linkedin.com/in/benjaminscabbia/
https://benjaminscabbia.co.uk

Submitted on: 30/05/2016

Supervisor: Dr. Thiago Matos Pinto

Abstract

The Internet of Things is growing rapidly and contributing approximately 10% of the world's Big Data. This, combined with advances in technology that have seen more sophisticated data-gathering sensors being inserted in a range of equipment, as well as social media, GPS tracking and more, has resulted in more Big Data being created today than ever before. Big Data (characterised by its large volume, variety of formats and the fast speed at which it is gathered) can be difficult for organisations to manage and analyse, but none more so than the organisation restricted by limited budgets. In fact, there is a widely disseminated sentiment that Big Data is only accessible to those operating within a big budget. This is because Big Data is difficult to manage and analyse due to a variety of interconnected key factors; it is costly, it is large (and so storage is problematic), it comes in a variety of formats (and so knowing what software to use to analyse it is difficult) and the tools currently available to service the market are incredibly complex (such as Hadoop). In addition, data of this scale commands a powerful system to be able to manage and analyse it. Centralized systems are expensive to scale, and limited in their capacity to scale-up to meet the required need, however distributed systems makes use of commodity hardware to scale outward indefinitely. For this reason, this project puts forward a middleware application, aimed at a small to medium sized organisation on a limited budget, built entirely using open-source software. It introduces a working prototype of an application capable of performing a basic analysis on a non-local data set via distributed computing. It uses existing technology to provide users with Big Data management and analysis tools in a low cost manner.

Ethical Issues Addressed

According to the Ethics Approval protocol at the University of Hertfordshire, this project does not raise any ethical issues.

However, there are issues with the collection of data field itself, which has in recent years been subject to privacy and security issues. My project makes use of only test data, and so does not put anyone at risk.

Acknowledgements

I would like to thank my project supervisor, Dr Thiago Matos Pinto for the invaluable support and guidance throughout this project. I would also like to thank my friends and family for the various discussions on the field of Big Data. I'd like to thank Margot Brace and Vittorio Scabbia for their help and support, without which none of this would have been possible. Finally, I'd like to especially thank my girlfriend, Harriet Burgham, for providing me with

financial and moral support throughout my studies to complete this project, and my Masters in Computer Science.

Table of Contents

Introduction

In his 1982 book ‘Megatrends’, author John Naisbitt penned the famous quote, ‘we are drowning in information but starved for knowledge’ (1). Today, this phenomenon is truer than ever. We live in a world that generates an abundance of data: we spend our days sending emails, and our spare time using social media to update our friends via our smart phones. We use search engines to answer our questions and internet-shopping to deliver our food. All the while the companies who made our way of life possible, are gathering data on our every move. Amazon tracks what we’ve bought, Google tracks our search history and Microsoft monitor our system usage. The digital footprint we create is growing at unprecedented speeds, yet we are only analysing 1% of the data we have in the world today (2). Figures this low can only lead one to assume that the management and analysis of Big Data is problematic. This paper aims to identify the problematic factors, and look at whether distributed computing is a methodology that could be used to navigate around these factors.

The objectives of this paper are to:

- **Core**
    - Identify the relationship between the Internet of Things and Big Data.
    - Identify what Big Data is, and where it comes from.
    - Assess common problems with the management and analysis of Big Data.
    - Identify the limitations of centralized computing when managing Big Data, and evaluate how distributed computing may solve these problems.
    - Discuss the current software available to support the management of Big Data.
    - Develop and evaluate an application that uses distributed computing to manage and analyse Big Data.
- **Advanced**
    - Develop a high level overview of the Hadoop framework and the problems it solves.
    - Integrate Apache Spark into the software.
    - Secure the software connection between the server and the client by establishing an encrypted connection via the Secure-socket Layer (SSL) protocol.
    - Enable the server dashboard to send simultaneous queries to the client.

Chapter 1

The rise of the Internet of Things

We are in the midst of 'a new revolution in computing and communication' (3); a revolution enabled by developments in new technologies, that has enabled 'billions of every day objects' (4) to remotely communicate with one another via the internet. We are living in an age of the Internet of Things (IoT).

The IoT is a network of smart, interconnected devices. A 'smart device' is embedded with electronics, sensors and network components (5) that enable data to be gathered and transmitted. These devices can include anything from wearables, smartphones, and sensors in smart-vehicles, to smart-buildings and even smart-cities (6).

The IoT is growing rapidly. It is estimated that there are 5.5 million devices connecting every day, with an estimated total of 6.4 billion 'things' to be connected by the end of 2016. This is a 30 percent increase from 2015, which saw approximately 4.9 billion devices connected (7). Gartner estimate a sharp surge in the number of connected devices over the next few years, hitting around 21 billion by 2020. Research carried out by the UK Government in 2014 estimated that the number of connected devices could even rise to as many as 100 billion by 2020 (4). McLellan attributes the advent of 'low-cost, low-power sensor technology, widespread wireless connectivity', as well as improved compute power and 'ample internet addresses courtesy of IPv6 protocol', as responsible for the explosion of the IoT (8).

As the IoT grows, more and more data is being generated and collected. Simultaneously, the technology that enables the IoT to work continues to develop, and with it, so do the sensors that collect the data (8). As the sensors become more diverse, so does the format of the data that they collect. This means that data is not only being collected at very fast speeds, but is also being gathered in a variety of formats; a concoction that some fear, standard processing means are unable to process (9). Data such as this, can be referred to as Big Data.

1.1 What will happen to the current data analysis and management landscape if the IoT continues to grow?

It seems clear from market predictions that the IoT is set to continue to grow and generate more complex and more diverse data. This being the case, the field of 'data management services and applications' has been forced to evolve to be able to service the market (e.g. Apache's open-source Hadoop is one response to the need for suitable software to tackle the problem – see section 3.3)(8). However, the current data-analysis landscape is far from being completely equipped to manage the ever-growing influx of data (10–12). If this market doesn't continue to grow, or one could argue 'catch-up' (11), with the influx of Big Data, business decisions will suffer (13,14), we could open ourselves up to potentially avoidable security disasters (15), the rate of innovation may slow (16), or potentially life-saving information (17,18) will be locked away within data sets that we don't have the tools to mine. As Atul Butte of Stanford School of Medicine famously once said, 'hiding within those

mounds of data is knowledge that could change the life of a patient, or change the world' (19). We just need to ensure that we have the tools to be able to get to it.

The next section will identify what Big Data is, where it comes from and will assess some of the problems commonly associated with managing and analysing Big Data.

Chapter 2

Big Data

'Data was getting "big" even before the Internet of Things (IoT)' gained traction, and 'generate[d] massive amounts of data' (8). IBM found that we produce 2.5m Terabytes of data per day, and estimate that as much as '90% of the data in the world today has been created in the last two years alone' (20)As much as 10% of this data is collected by connected devices from the IoT alone (21). Other data gathering methods are via social media, PCs, smartphones, GPS devices and more (22).

This rapid growth of data is only set to continue; Cisco predicts that 'more than 500 zettabytes will be generated by all people, machines and things by 2019, up from 135 zettabytes in 2014' (23). Much of the data generated is so large, varied and complex that it cannot be managed by traditional processing means (9,24,25), (i.e. storing and querying data in a Relational Database Management System, see section 3.1)(8). The following section will explore what constitutes 'Big Data', and the reasons why it can be problematic to manage and analyse.

2.1 What is Big Data?

Analytics experts SAS, describe Big Data to be 'when the volume, velocity, variability and variety of data exceed an organization's storage or compute capacity for accurate and timely decision making.' (26). In keeping with this description, there are three characteristics commonly associated with Big Data, known as 'the three Vs' (20,27,28). The first V refers to volume, or in other words the size of the data – which is always 'large'. The second V refers to variety, which means the type and structure of data, which can vary greatly. The final V stands for Velocity, which refers to the fast frequency at which the data is produced (20,29). In 2013, Zikopoulos/IBM followed up their 2012 study by adding a further two Vs: veracity and value. Veracity measures the accuracy of the data (a factor that can be 'difficult to ensure' due to its massive size and velocity (30)), and value refers to the potential-value of the data to the organisation (31,32). While the concept of 'the three Vs' is widely accepted, the addition of further V's varies from organisation to organisation.

2.1.2 The formats of Big Data

Big Data can be created in a variety of formats known as structured (e.g. data entered in a pre-defined data model), unstructured (e.g. data that does not have a pre-defined data model) and semi-structured (a cross between the two, i.e. structured data that lacks the strict data model structure)(33).

Structured data is normalized, meaning the structure is rigid and pre-planned (34). It is generally stored using more 'traditional' methods such as relational database management systems (RDMS), and is the easiest to analyse and query, which can be done by using the Structured Query Language (SQL).

Semi-structured data is schema-less, meaning the data is flexible and self-describing (34). Extensible Markup Language (XML) and Javascript Object Notation (JSON) are two

common semi-structured data types. No SQL databases are capable of storing semi-structured data (35), although unlike structured data that can be directly inserted into a database, some semi-structured data will need to be serialized before this is possible (36). This makes the management of semi-structured Big Data a little more complicated than that of structured Big Data.

The third form that Big Data can take is unstructured. Although unstructured data makes up the largest segment of an organisation's collected data, it is the most problematic to manage and analyse due to its highly varied nature. Unstructured data is often non-numeric (26), and is information that is not stored in a database. It can be textual as well as non-textual (22), and includes text files, PDFs, images etc. It is estimated that as much as 80% of data generated by a single organisation is 'unstructured' (37). This high volume is concerning, due to the difficulty in extracting value from it (22). Due to its extremely varied nature, Bacchelli explains that it is difficult to analyse because it is 'hard for researchers and practitioners to determine the appropriate technique(s) to deal with the problem at hand'(38). Worryingly, Hitachi and IBM have both estimated that quantities of unstructured data are set to grow at twice the rate of structured data (39,40). This means that reliable tools to deal with this type of data are more vital than ever.

2.2 The problematic nature of managing and analysing Big Data

Effective analysis of Big Data has proven its worth: it is a driving force in the fight against cancer (41), can account for the difference between the market share of two competing companies (42) and has even helped one city to reduce crime by as much as 42% in just four years (43). Yet despite the many successful outcomes, in 2012, data analysis experts EMC estimated that only 1% of the world's data was currently being analysed (2). This could be because managing and analysing Big Data can be problematic due to a variety of factors, ranging from its extreme size, cost and variety, to the challenge of maintaining data integrity whilst keeping the data secure. So often many of these factors overlap. In the following section I will look at some basic procedural activities that make-up the management and analysis of Big Data, such as storage, data analysis and setting up infrastructure, and will examine how the problematic factors of Big Data effect the task in hand.

2.2.1 Problematic factors affecting the storage of Big Data: volume, cost, maintaining integrity and security

Big Data can often be as large as hundreds of petabytes or more (44). In order for an organization to obtain the highest value from their Big Data, it needs to be stored somewhere that is secure, accessible and affordable (45). Achieving all three of these factors can be challenging however, particularly when taking into account the characteristically high-volume of Big Data. For example, state-of-the-art, encrypted (e.g. AES-256), accessible storage, that will be more likely to preserve the integrity of the data and less susceptible to data attack, can come at a high price (46). That said, with regard to hardware-based storage, the past 35 years have seen a steady decrease in the associated cost; in 1980, 1 gigabyte (GB) of standard storage cost $193,000, but in 2014 it cost just $0.03 (47). It is worth noting that the cheaper hardware storage options available (i.e. HDDs) will sacrifice speed (therefore limiting accessibility). More expensive solid state drives (SDDs) on the other hand, are generally considered faster (48), however in 2015, Mearian found them to be roughly 8 times more expensive, at $0.24 per GB (49).

Identifying suitable hardware to store the data can also be difficult and costly. A study carried out by Backblaze in 2013 found that as many as '22% of drives fail in their first four years' (50). This high-fail rate highlights a need for companies to invest in better quality drives as well as numerous back-ups, all of which are likely to mount up to a sizable expense. Should a hard drive fail and the organisation not have a back-up in place, costs to recover the corrupt data will be infinitely more expensive (51,52).

A data storage method that takes the risk of using hardware away from the user, is cloud storage. Companies such as Microsoft, Google and Amazon offer cloud storage services, at approximately $0.03 per gigabyte (53). Cloud computing is generally considered to be a cost effective solution to Big Data storage for businesses (54,55). Cloud storage offers a variety of benefits, from making the files remotely accessible, to generally including disaster recovery as standard, as well as often offering optional further disaster-preventative measures. Microsoft, for example, offer a service that will back up files in multiple datacentres spread across geographic locations (56). Nevertheless, there are associated problems with cloud storage. There are limits to the accessibility of the data – for example in the event of a loss of connectivity. Users also often find themselves dependent on, and locked into, cloud vendors (57). Lastly, and arguably the most complex issue, is security. The nature of the cloud computing model, subjects cloud services to cyber-attacks (57,58). By moving to the cloud, the organisation no longer has full-control of the storage, as they do 'not manage or control the core cloud infrastructure,' which unfortunately opens the data up to 'security repercussions in the form of hacking' (57,59,60).

If we look to the future, the size of Big Data is predicted to keep expanding, which could lead to storage complications. Some scientists fear that the hardware used to store data is not progressing fast enough to keep up with Big Data's growth (61).This could have catastrophic repercussions for the management of Big Data – especially for those who are reluctant to join the cloud as they are dealing with very sensitive data (60). Data storage solution provider NetApp's Einstein, who estimates that 'the capacity of hard drives isn't increasing fast enough', illustrates that a '50-fold increase in global data [is expected] by 2020' but at the same time hard drives are estimated to only grow by 'a factor of 15'(62). This could see a forced rise in the number of users of, the arguably less secure, cloud storage.

2.2.2 Problematic factors affecting the allocation of staff to Big Data projects: cost

Actually managing Big Data (i.e. the organization, administration and governance (63)) can also be expensive. Due to the extreme complexity and variety of Big Data, specialist "hardware, networking, software and human skill" are often required to adequately manage it (8,64). Many Computer Scientists list not having the correct skill-force to manage projects as one of the most common reasons for the failure of Big Data projects (65,66). In addition, having a qualified team of staff to facilitate the analysis of the data is costly. McLellan lists two types of professionals that are in high demand: 'business analysts' capable of querying the data and presenting the 'results to decision makers,' and 'data scientists' capable of utilising 'analytical tools and curate[ing] the veracity of the data'(8). Both of these roles demand a healthy salary: Payscale lists the average wage of a data scientist to be around £40,000 PA (67). This is 51% more than the UK national average[1] (68).

[1] Based on an average salary of £26,500, correct as at 24/05/2016

2.2.3 Problematic factors affecting the analysis of Big Data: variety, costs

Analysing Big Data is especially complicated due to the extreme variety of the data. Big Data expert Veeranjaneyulu writes that the value of Big Data is often compromised as 'it's hard to find a tool that deals [with the variety of] unstructured data' quickly and efficiently (22). This is an area of Big Data that has been the subject of a significant amount of research in recent years, resulting in many companies marketing models designed to help. For example, data analyst experts SAS combines machine-learning with in-memory analytics so as to offer 'immediate analytic insights' in real-time. In-memory analytics work to reduce the time it takes for an analysis of Big Data to be made. It does this by taking 'advantage of the hardware and RAM capabilities that have become cheaper' (69) over the years, in order to 'analyse data from system memory, (instead of from your hard disk drive)'(70). Simultaneously, machine learning works to combat the issue of the variety of the data. It 'allows a system to analyse hundreds of variables simultaneously, along with how they interconnect, to form patterns' (71). This process 'does extremely well with large volumes of unstructured data including images, text, audio, sensor data and more,' (71) because it is no longer limited by the explicit rules that are set by humans. A good example of machine learning combatting the variety and velocity of Big Data, is in Google's Google Translate (GT)(72). GT uses an algorithm that looks for patterns within a huge collection of human-translated texts, so that it can match an associated translation with the text you entered (73).

Of course, there are downsides to the technologies used in systems such as SAS's aptly named In-Memory Analytics. An analyst at IT research and analysis organisation Aberdeen Group Inc., wrote that in-memory appliances 'are almost exclusively meant for large enterprises and carry a price tag commensurate with the size of their intended customer[s]', potentially making in-memory a less viable option for small to medium sized corporations who may not be able to afford it (74). Aside from the cost, there are other drawbacks, such as in-memory's inability to scale and handle massive datasets (75). Castanedo explains that the 'R' programing language, for example, (which he labels the most 'popular statistical software in use… by data scientists' (75)), suggests using data structures that are 'no larger than 10-20% of a computer's available RAM.' This is partly because R itself, 'incurs significant memory overhead because they use temporary copies instead of referencing existing objects.' (75). With regard to machine learning, many feel that its downfall is it being too complex. Machine learning advocate Martin Hack writes that 'for machine learning to impact the world around us… we have to deliver Machine Learning in a smarter, more usable form,' a form that doesn't require users to be expensive 'data scientists [with] PhDs' to operate it (76).

2.2.4 Problematic factors affecting the creation of hardware and software infrastructure: data integrity, variety, cost

Setting up effective infrastructure for the management and analysis of Big Data can be difficult and costly, and is often an area of Big Data that is overlooked or underestimated by CEOs and company bosses (12,77). A recent survey found that only 16% of users in organisations working with Big Data were 'prepared to say their Big Data tools are of a high quality'(78). For a Big Data project to be successful, suitable infrastructure that works to maintain data integrity, provide suitable storage, and safe transfer processes must first be put in place. When suitable infrastructure is not adequately implemented, project failure is likely (79,80).

Project set-up, equipment and infrastructure also increases the cost of Big Data projects. Figure 1, developed by IT analyst, research and validation specialists The Enterprise Strategy Group in 2015, demonstrates the phenomenal price typically required for a medium-sized Big Data project. This only includes a three-year license, and features over 6-months of setup before the project even becomes operational.

Item	Cost	Notes
Servers	$708,561	Eighteen @ $39,365 each; enterprise-class, each with: two Intel Xeon 18-core processors, eight 16GB memory kits, twelve 8TB hard drives, network card, and dual power supplies
Networking	$37,500	Three Mellanox 36-port Infiniband switches at $10,500 each, plus $6,000 cabling
Rack enclosure	$5,000	Including cables, looms, patch panels, etc.
Hardware support (three years)	$112,659	@15% of list cost of above items
OS licenses	$66,042	Red Hat Enterprise Linux at $1,223 annually for each of 18 nodes for 3 years
Hadoop licensing	$388,800	Cloudera Enterprise Data Hub at $7,200 annually for each of 18 nodes for 3 years
Build Project Costs	***$1,318,562***	

Figure 1 - Big Data Project Cost (66)

It is often necessary for organisations to move Big Data around a network for either storage, access or analysis. When an organisation moves this data around their infrastructure, 'human error and technical equipment fail[ure] are real risks' to the data integrity (81). In addition, it is likely that during the data transfer process a large proportion of bandwidth is used, which will lead to network congestion. This will not only cause the organisation to have slower network speeds, but congested networks are a common cause of packet loss (82). Indeed, while an application protocol such as TCP would detect the loss and request a re-transmission of the lost data, this would result in latency issues. Furthermore, if the network scheduling policy specifies a maximum delay, this may cause data loss or, if unspecified, it may seriously reduce network performance.

It would, of course, be in the best interest of data preservation to refrain from moving the information (thereby negating the aforementioned loss/corruption-risk), and instead, set up an infrastructure that moves the required computation to the node with the data. This methodology is evident in distributed computing (see section 3.2), and is used by data-management and analysis tool, Hadoop (see section 3.3) (83). Hadoop not only reduces the computational load by moving the computation to the data, but duplicates data across multiple nodes, reducing the risk of data loss by eliminating the single point of failure (84). Despite the reliability the Hadoop framework can offer, it is often referred to as overly 'complex' and 'difficult' and so special training will need to be provided for the user (85,86).

From finding storage systems for large volumes of Big Data, to putting in place suitable infrastructure for managing the data, it can certainly be concluded that, in theory, Big Data has many potentially problematic elements that make it difficult to analyse and manage. In the following section, I will explore real-life examples to assess whether these theoretical issues make the management of Big Data difficult when in practice.

2.3 Big Data failures: case study

A survey carried out by Infochimps in 2013 found that over half of all Big Data projects are never completed (87). There are, of course, various reasons, both internal and external, for this high fail-rate, however, the problematic factors examined in the previous Chapters are often partly responsible. The following two case studies look at two Big Data projects, and why they failed.

The sheer volume and velocity at which Big Data is created is often troublesome for those tasked with analysing it quickly – especially when an analysis is required in real-time. In early 1986, the Space Shuttle *Challenger* broke apart too early during lift-off, tragically resulting in the deaths of all seven crew members. Although the term 'Big Data' had not yet been coined, the analysts at NASA were dealing with vast amounts of data – in fact, more data than they were equipped to manage. In Big Data consultant Marr's 2016 book *Big Data in Practice*, he explains how the 'mission controllers were passed an overwhelming amount of information from the technical staff monitoring the shuttle's vital systems'. The team were faced with such an extreme velocity, volume and variety of data that they were unequipped to analyse it quickly. Had they been able to analyse the data effectively, Marr argues the warning signs would have been evident, and thus the disaster may have been avoided (88).

Setting up a suitable infrastructure to enable the analysis of Big Data is no trivial task as my next example demonstrates. In 2015 Garner published a report discussing how an unnamed retail chain was unable to set up a system capable of analysing Big Data within the confines of their project boundaries (77). The IT department were tasked with implementing a recommendation engine, similar to Amazon's 'customers also bought' recommendations. The team, who were inexperienced within the Big Data environment, were given no additional manpower and just 6 months by the CEO to create it. The IT department soon realised the sheer challenge of the task, and although they attempted to implement a 'collaborative-filtering algorithm', they were unable to work with the Big Data. This was due to the 'data sparsity' (i.e. the vast number of attributes of the data made it difficult to comprehend and organise),and the 'scale of the huge datasets' (77). In order to appease their CEO, the company produced a 'fake engine' that always recommended bed sheets regardless of the item purchased. It took an additional two years and the employment of specially trained behavioural and operational experts and engineers however, before the recommendation engine was actually functional (77). This was because the data was simply too complex to be managed by the inexperienced IT department, who had no existing infrastructure to work from.

In both examples, the organisations attempted to manage Big Data without making use of specialised infrastructure, specially designed software or specially trained teams. In other words, the people in both examples encountered difficulties when attempting to manage Big Data using traditional processing means. This therefore highlights the need for specialised hardware or software that can enable organisations to be able to 'make the right data available in the right form to the right people at the right time' (89). Although there are tools on the market that can help manage Big Data (such as the aforementioned Hadoop or SAS's

In Memory Analytics), they are complex to use (i.e. Hadoop) (12,66,85), or costly (i.e. SAS's In Memory Analytics (54,90,91)).

2.4 Conclusion: Are big budgets required for Big Data?

As my paper has so far demonstrated, despite the difficulties associated with Big Data, it is generally considered that effective analysis of Big Data is difficult and expensive, but a worthwhile pursuit (42,44,92). In an article posted by Big Data news outlet 'Inside BIGDATA' that looks to dispel some common myths surrounding Big Data, they warn organizations to 'look beyond the promise of free and cheap' Big Data analysis tools (93). They explain that 'the cost of making open-source [software] enterprise-friendly' is always going to be high (93). They instead advise organisations to pursue their work with Big Data, but in order to see effective results, they urge them to increase their budget allocation (93). This sentiment, that accurate analysis demands big budgets, is emphasised in the findings of a 2015 market research survey in which 34% of respondents[2] attributed 'limited budgets' to be the factor preventing their IT department from providing 'high quality' Big Data analysis(92).

The concept of Big Data analysis requiring a big budget is developed in an article published by the Huffington Post in 2016, titled 'How to Do Big Data on a Budget?' (94) The article names open-source software such as Hadoop as a half-solution for keeping costs down. The article continues to explain the down-side of open-source software: 'it will take some time and technical skill to get free software set up and working the way you want'; 'time' and 'technical skill' one can translate into added cost (94). The relationship between Big Data, big budgets and complexity is an issue that I aim to address by creating a piece of software that can analyse Big Data, and be used by a competent IT department.

So as to identify the best methodology to use in my piece of software, the next section will examine methodologies that are used in the management and analysis of Big Data. I will then select the one best suited to making my software available for even those on smaller budgets.

[2] Respondents being 100 IT Decision Makers from organisations with 1000 employees or more

Chapter 3

Methodologies for managing and analysing Big Data

The advent of Big Data, along with the problematic factors associated with the management and analysis of it, have certainly impacted the data management and analysis landscape. From finding adequate storage solutions for the various data types, to identifying suitable systems that have the processing power to be able to cope with large data sets, the following section will look at solutions to dealing with Big Data so as to give me a better idea of the methodologies to use in my software.

3.1 How has Big Data impacted the data management and analysis landscape?

During the early 2000s, the standard method of storing data was via a Relational Database Management System (RDMS). The RDMS was a model capable of storing and analysing the structured data commonly generated at the time. A decade and a half later, in 2016, it is still the most utilised database architecture, proving to be an excellent model for storing and analysing large quantities of structured Big Data (95).

As Big Data began gaining traction, more data was being generated in an unstructured format, and so traditional RDMSs were unable to cope with its variety. RDMS' were 'costly, not terribly scalable, not as tolerant to failure as required, and possibly not as performant as desired' when faced with unstructured Big Data (96). Organisations that were dealing with the largest data sets (i.e. Google, Amazon etc.) were the first to experience such problems. Google and Amazon released papers in 2006 and 2007 respectively, titled *BigTable* and *Dynamo*, which put forward two highly scalable, fault tolerant, non-relational distributed storage models. Since their release, they have inspired the development and wide-spread adoption of NoSQL databases for storing and analysing semi- and unstructured Big Data (97–100). The NoSQL approach is focused on simplicity of design, flexibility and horizontal scaling, which is much more apt at handling the variety and velocity of Big Data.

As more diverse and complex Big Data was being generated, more efficient processing techniques were required in order to produce an accurate and timely analysis. In response to this need, Google developed a programming model specialised in 'processing and generating large data sets with a parallel, distributed algorithm on a cluster' (101) , known as MapReduce. Since then, MapReduce has seen a number of rivals, but perhaps none quite as strong a contender for 'general purpose' processing frameworks as Spark. Spark has been labelled a 'faster and simpler' competitor to MapReduce (102). Spark is an open source processing engine focused on analysing data quickly and easily (102,103).

Just as new storage techniques and new analysis methods were required to cope with the new formats of Big Data, more powerful processing hardware was also required. Centralized computing (i.e. performing all of the data processing via one machine) became unable to cope with Big Data's size and variety. Using a centralized system to manage and analyse large data

sets was slow, susceptible to bottlenecks and presented a dangerous single point of failure (104,105).

3.2 Centralized vs distributed computing

In order to make centralized systems more effective at working with Big Data, their processing power needs to be upgradable in-line with the growing complexity of the demand that is put upon it when attempting to analyse a Big Data set. While it is possible (although costly) to scale-up a centralized system, current technological limitations cap the processing potential. In the 1970s Moore's law predicted that 'processing power will double every two years' (106), which was accurate until around 2010. Since then, progress has slowed down so much so that we are no longer in line with the predicted trend. This is because since the 70s, transistors have been made physically smaller, and so more of them could fit in a processor therefore boosting processing speed. Now that transistors are so small that we are almost at atomic level, current technological capabilities are impeding the enhancement of processor speed. Moore had predicted this, stating that 'once transistors can be created as small as atomic particles, there will be no more room for growth in the CPU market where speed [is] concerned' (106) and Moore's law will end. While scaling-up can be a 'viable architectural decision for those whose growth needs fit Moore's law' (106–108), the limitations on its potential make it a far from ideal candidate for managing and analysing the growing complexities of Big Data.

As Big Data continues to grow, the limitations of scaling-up centralized systems has prompted organisations to revert to systems that scale-outwards. Scaling horizontally was historically out of favour due to the associated complexities (108). However, progresses in technology have caused fast and inexpensive processors and highly proficient computer networks to be readily available (109). The availability of commodity hardware and their integration with network components has enabled a 'computing paradigm called Distributed Computing' (109).

3.2.1 Distributed computing

Distributed computing, is the most popular methodology used to tackle Big Data due to its flexible, easily-scalable architecture. It creates a network of commodity computers – otherwise known as a cluster – that share workloads across multiple machines (110).The cluster is independent from the heterogeneity of a network, as the 'processing nodes, network topology, communication medium [and] operating system' (110) are irrelevant, which makes them a relatively low-cost, attractive model for most organisations.

One aspect of managing and analysing Big Data that distributed computing excels in is maintaining data integrity. This is because distributed computing works well with data replication. Although not used by all distributed computing systems due to its complexity, replication is a strategy 'for improving reliability, fault tolerance and availability' of data (111). This means that the data would be replicated across multiple nodes within the cluster, resulting in each piece of data being on at least one other machine. Duplication does not only protect against data loss, but also creates a more resilient process. For example, if a non-replicating cluster was performing an operation and a machine failed, it would have to restart the process which may already have been running for several days and would be costly. However, by duplicating the data, the more sophisticated frameworks are able to re-assign the

job that was being performed by the failing node to another machine, therefore, not-interrupting the analysis. However, implementing replication is a difficult and time-consuming task that is likely to add to the complexity of the application. It is only due to the time constraints of this project, that my software will not make use of data replication.

Distributed computing is so adept at handling Big Data, that the most popular Big Data analysis tool at the moment, Apache Hadoop, utilises this methodology. The following section will look at how Hadoop works, and will evaluate its strengths and weaknesses, so that my software can attempt to avoid the same pitfalls.

3.3 Hadoop: the most popular Big Data framework

Hadoop is the open-source solution of choice for large-scale distributed Big Data processing (71,96,112). It is a framework consisting of dozens of compatible modules (84). The modules provide a vast range of functionality, from storing and organizing data, to the management of a cluster (96). These modules interact with the core of Hadoop, which can be split in two: Hadoop Distributed File System (HDFS) and MapReduce (96).

The role of the HDFS is storing the cluster's data. It is designed to enable high-performance, cross-platform compatibility, fault tolerance, and to move computation to the data (96,113). HDFS achieves this by breaking the files into a large block of data (generally 64-128 MB (84)), which by default, is then replicated three times and placed on different nodes in the cluster. The replicated data not only reduces the chance of data loss, but also helps with performance as the data can be accessed where it resides, and thus only the computation will require moving.

MapReduce, as previously mentioned in section 3.1, is a programming model created by Google for distributed computing. The programming model is based on a 'divide and conquer technique' that performs data analysis via two operations, Map and Reduce (33,113). The Hadoop software provides the Map function with a block of data to analyse, which then creates a Map key-value data model. For instance, if the user wanted to count the frequency of words, the user would define a map function and pass an input file from the HDFS, i.e. a text file containing 'University of Hertfordshire'. The mapper would then produce a set containing key-value pairs: ["University":1, "of":1, "Hertfordshire":1]. In the example, the key is the word, and value is the frequency of word reoccurrence. Once the mapping process is complete, Hadoop assigns a thread to each of the reduce functions (also user-defined) which will then sum the pairs and provide the overall aggregate result (96,113).

This simple use case demonstrates the core functionality of Hadoop, but in this example, the reducer is redundant. If the user performed another query in which the data was distributed, Hadoop would then instruct each node (i.e. a computer in the cluster) to perform the user-defined MapReduce function. It would firstly map the values from the relevant data blocks, and once complete, the reducer would combine all the information, sum the pairs and output the results.

3.3.1 Hadoop: The good

The Hadoop framework is open-source (and so free), highly scalable and is optimized to run on commodity hardware. One of the fundamental concepts of Hadoop is to solve the problem of diverse data, and so the Hadoop Distributed File System (HDFS) is able to manage

structured, semi-structured and unstructured data that is enormous in size. In addition, the open-source nature of Hadoop means that it is always being further developed by a vast community of developers. Many of the modules have been formalised and become projects of their own. These are: Core Technologies, Database and Data Management, Serialization, Management and Monitoring, Analytics Helpers, Data Transfer, Security and finally Cloud Computing and Virtualization (96). This means that Hadoop is able to cater for a much larger audience of users looking to Hadoop to help them with specific use cases. The modules are developed to be fully-compatible, so any project can make use of any module to help manage and analyse Big Data.

The first iteration of Hadoop, Hadoop 1.0, was only capable of analysing data via the MapReduce (MR) model. Hadoop 2.0 however, provides an additional tool called YARN that adds an additional layer between the HDFS and MapReduce. It was designed to provide a more generic alternative to MR, which could only perform queries that followed the MR programming framework. MR had a number of issues, but one of the most critical was described by, founder and CTO at cloud analysis provider Xplenty as being 'infamous for being very difficult to program'. The difficulty in recent times has led to a number of new modules to improve usability (such as Pig and Hive). YARN overcame some issues that Hadoop 1.0 encountered such as scalability, availability (Single Point of Failure), resource utilization and restrictiveness (limited to MapReduce) (114,115). Adrian, a Gartner analysist described pre-YARN Hadoop as a 'brute-force, batch blunt instrument for analytics' and post-YARN Hadoop as an 'interactive analytics tool [for] mixed workloads' (116).

3.3.2 Hadoop: The problems

Although the introduction of YARN solved many of the problems faced by Hadoop 1.0, Hadoop 2.0 is notoriously harder to use. One of the most prominent examples is the trouble Todd Papaioannou, once branded a Big Data 'industry veteran' by technology research firm GIGAOM (117), experienced during his tenure with Yahoo. Papaioannou was assigned the Chief Cloud Architect role, and was tasked with setting up 45,000 Hadoop servers (along with his 120 person team) (118). What should have been a stable platform that enabled Yahoo's engineers to develop and test their applications, was a product described by Papaioannou to be a 'bunch of redneck architecture' (118). This was fuelled by the multiple variations of libraries and tools that were being used by each of the teams. In a conference, Papaioannou summarized that 'Hadoop is hard' due to 'it's low-level infrastructure software, [that] most people… are not used to using' (118).

3.4 The best methodologies for managing and analysing Big Data

Distributed computing is the most effective processing methodology for managing and analysing Big Data. Data sets have grown so large and complex that centralized systems lack the processing power to be able to issue a timely analysis. Distributed computing boasts a range of advantages such as the ability to bring the processing to the data, however the most important factor making it the best fit for Big Data is the ability to scale outwards. As Big Data grows ever larger and more complex, distributed computing allows for an endless number of additional computers to join the network, therefore producing enough processing power to be able to tackle the data effectively.

Hadoop showcases the beneficial relationship between distributed computing and Big Data. Hadoop, an interface designed to manage and analyse Big Data, is open-source, highly scalable and proven capable, but is not without its flaws. Its popularity and growth may be the problem: as more developers are getting involved with the project, more modules are being developed and its overall complexity is increasing.

I have identified that there are competent tools that are capable of analysing data in a database (i.e. RDMS and NoSQL databases) and data that is not in a database (i.e. MapReduce, Spark). This has led me to the conclusion that I will develop a piece of middleware that acts as a panel from which these tools are available. In other words, I want to create a single interface from which a user can remotely command Spark to analyse unstructured data, but also remotely instruct a SQL or NoSQL database system to analyse data. By being middleware software, it will be easily integrated into existing architectures.

I will be using distributed system methodology in the creation of my middleware. Much like Hadoop, I plan on making it open-source and available for other software engineers to work on, but I would market the end-product at a less technical audience than Hadoop (which attracts a highly specialised data scientist audience (11)).

Chapter 4

A solution to the problematic nature of managing and analysing Big Data

The previous Chapters looked at what Big Data is, where it comes from and identified the characteristics of Big Data that make it difficult to manage and analyse. They also look at how the problematic factors of Big Data have impacted the data-storage and analysis landscape, and examine the most popular tool that is currently used to help organisations with this task, (Hadoop). I have also identified the reasons why distributed computing is an effective methodology for the management and analysis of Big Data. I have concluded that there are effective management and analysis tools available to largescale organisations with very large budgets. I have, however, identified a need for effective software that is available for organisations on smaller budgets. In response, I have created a piece of middleware that aims to do just that.

4.1 Introduction to my software

My middleware is aimed at small to medium sized organisations who have smaller budgets and limited resources, yet still need to manage and analyse Big Data. The technology used is all open-source and the middleware has been designed to function in an existing heterogeneous network, so as to cause the least amount of disruption and expense. It operates without the need for file reorganisation and is capable of running on commodity hardware, therefore not requiring specialized equipment. Also, the middleware has been designed to combat the complexity that surrounds the field of Big Data management and analysis (76) by being simple to use.

Middleware 'is the software that connects software components' (119).The middleware will enable an organisation to manage and analyse their data by sending user-inputted queries to a remote machine, without the need to move or download the data. The machine will then execute the queries, and return the results. Assuming the data is distributed among a number of machines (locally or remotely), the program is able to send and execute computational queries on each of these machines.

4.1.1 Overview of the Program

The research and insight demonstrated in the previous chapters were instrumental in the technical decisions I made when designing my middleware. My research helped me to understand how the software needed to help small organisations tackle the most prominent factors that make the management and analysis of Big Data problematic. Namely: complexity, variety, volume and cost.

The following sections will look at the technicalities of my software, and how they relate to the above factors in more detail. I will begin by discussing the software architecture, the core functionality and implementation.

4.1.2 Software Architecture – A technical overview

My software is a working prototype of a middleware application that can perform a basic analysis on a non-local data set via distributed computing. Namely it can search a text file, count the frequency of words in a text file and perform a simple user-specified filter query on a database located on a separate machine. It is envisaged that the software would be made open source, and that further versions of the software would incorporate additional functionality. The middleware software has been designed to easily integrate within existing infrastructure. There are two distinct parts to the middleware: MServer (i.e the server computer) and MClient (i.e. a remote machine that holds the data). They follow the typical Server-Client(s) architecture, and therefore their relationship is one-to-many (1..*); one server is capable of handling an arbitrary number of clients in a heterogeneous network. As it is heterogeneous, the server is not affected by the client's operating systems, processor or the type of data that may be present. The two components interact through a RESTful service, which utilises HTTP as the transport layer. As the MClient can be connected via the internet, they are not physically confined to a geographical location[3].

As Figure 2 demonstrates, the application sits above the local OS, the local data and the processing engine. The prototype has been integrated with a MySQL, MongoDB and Spark processing engine, thus enabling it to interact with a database (relational or non-relational) and with data not in a database (via Spark).

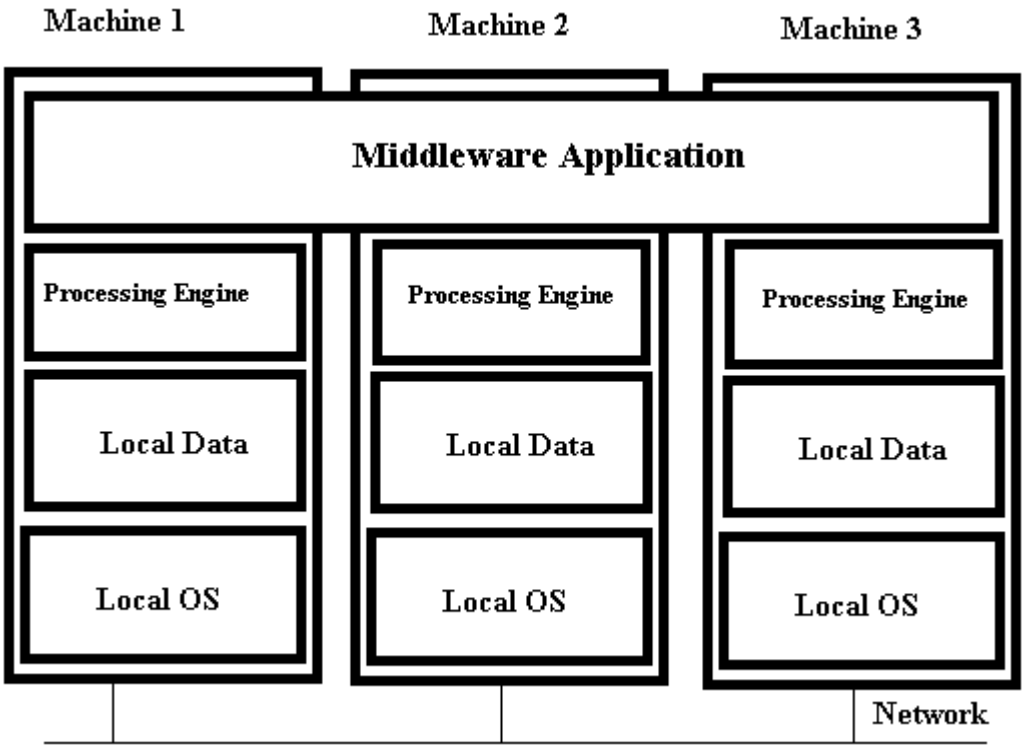


Figure 2 - System Architecture

[3] As long as the traffic is routed correctly

4.2 Software components[4]

4.2.1 MServer

The MServer enables the user to send instructions to the client(s), and is responsible for keeping track of the clients and sending out queries.

To view a class diagram for the MServer part of the application, please ref to appendix D.

4.2.1.1 MServer Front-end

The layout of the panel was developed with the user in mind, as it will be the only part of the application that a user will regularly interact with. I took care not to display too much information on the panel so as not to seem intimidating to the less-experienced user (i.e. I was attempting to reduce the notion that Big Data management and analysis tools require data scientists to operate them). I ensured that the design was clean, clear and the font was readable by selecting elements that were similar to those featured in user-friendly applications such as the WordPress admin panel.

As Figure 3 demonstrates, the panel lists all of the user-actions in the left menu, and the responses clearly in the central panel.

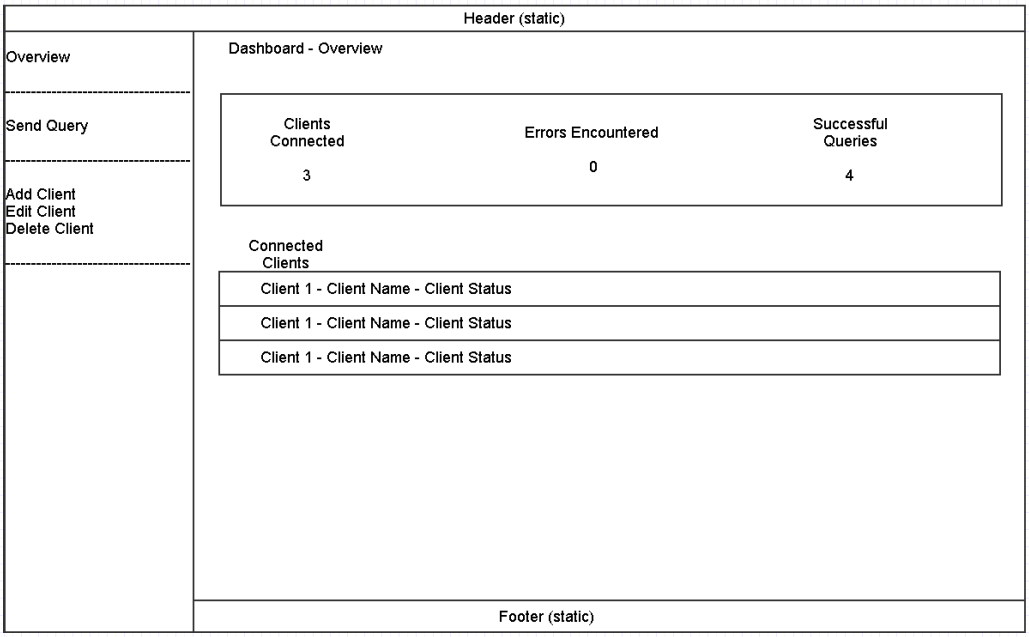


Figure 3 - Dashboard Design

The prototype provides two main commands that can be issued via the dashboard: 'Send Query' or 'Send Spark Query'. Between the two options, the user is able to query structured, semi-structured and unstructured data.

Java Spring projects generally take the form of Model-View-Controller (MVC) or Model-View-Presenter (MVP). Both architectures achieve the same goal of rendering an HTML

[4] For a full list of all the tools and software used to build the distributed application, please refer to appendix A and B.

page to the end user, but they follow different design patterns. The primary difference is that an MVC controller has the logic for returning more than one view, whereas MVP has a separate presenter for each of the views, and so the logic remains separated. Both architectures have their use cases, but generally, MVP is more complex. In Java Spring, MVP is implemented on-top of Spring MVC (utilising an extension called Spring-Roo (120)). Spring-Roo adds an extra layer to the MVC model, which would only be justifiable for large scale projects where separating logic from each of the views justifies the extra configuration. I decided to use MVC, as the additional benefits provided by MVP did not justify the extra configuration required. In addition, high-traffic MVC websites such as StackOverFlow, have proven MVC to be both reliable and scalable. This reassured me that my users will be able to connect a large number of clustered computers, without architectural performance issues.

I then decided on the server page technology that would render my View. Java Spring supports a number of open-source template engines including: Java Server Page (JSP), Thymeleaf, Velocity and Freemarker. As the most mature and commonly used engines, I narrowed the choice to Thymeleaf and JSP, ultimately deciding on Thymeleaf for its superior Expression Language (EL) and layout features. Thymeleaf, unlike JSP, offers a layout feature (121) which allows 'common page components like the header, footer and menu' to be reused across several pages. Although Thymeleaf requires additional configuration compared to the Spring ready-supported JSP, I felt that this was justifiable as it will reduce code duplication and enable encapsulation, which simplifies development and maintenance.

Perhaps the most important factor as to why I chose Thymeleaf, was because it helped me to easily create a 'wizard' that would guide the user through the query-creation process (thus making the software less complex).

I chose Bootstrap as the front-end framework as it is open-source, responsive, supported by most browsers (and so will make the software less problematic for the user) and provides many pre-designed templates. I used a template called 'SB ADMIN', that had a similar look and feel to my design.

Figure 4 shows the front-end file structure. It has a static folder that contains cascade style-sheets (CSS) and Javascript (JS) files.

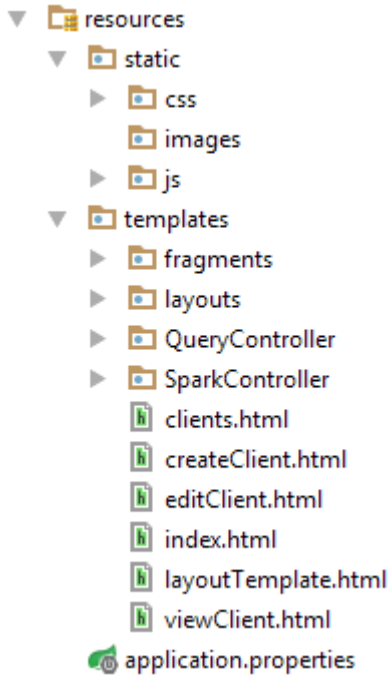


Figure 4 - Front-end file structure

The pages are logically separated, and so it is easy to navigate to find a particular file. In the templates folder, the first folder 'fragments' is deprecated. I originally set up the views to make use of outer-fragments (a layout feature) because I thought it would be best for providing reusable content, but this was before discovering that Spring supported hierarchical layout-dialects. The layouts folder contains a template file that appears to be an ordinary HTML page, however the page is actually a master file that is inherited by all other pages on my site. The master page explicitly specifies a layout-fragment, as shown in figure 5, which is the only area that a child page can override. This therefore minimises code maintenance and code duplication

```
<section layout:fragment="mainContent">
    <!-- Content replaced by each page's content fragment -->
    <p>This section will not render, overriden by child</p>
</section>
```

Figure 5 - Layout feature

The QueryController and SparkController folders contain controller related views, and the HTML pages located at the template root, act as the root pages for the site. To view all of the front-end pages developed, please see appendix C.

4.2.1.2 MServer Back-end

The MServer project back-end has four directories, as shown in figure 6. The folders have been structured using MVC conventions.

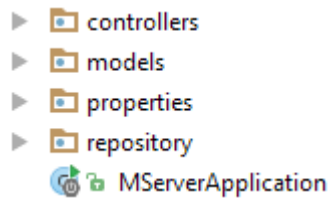


Figure 6 - MServer back-end file structure

4.2.1.2.1 Controllers

The 'controllers' folder contains controllers responsible for translating 'interactions with the view into actions to be performed by the model' (122). They are essentially classes with the application business logic which respond to actions performed by the user.

As previously mentioned, the project follows the MVC architecture. Each of the controllers, is responsible for processing the user actions and returning results requested by the user. The controllers have been separated according to their functionality (i.e. database queries are on the QueryCotroller and Spark operations are on the SparkController), and so it aims to be a cleaner implementation.

There are four controllers: home controller, client controller, Query controller and spark controller. The home controller, the simplest, is only responsible for returning the index page (the home page). The client controller is responsible for the management of the connected MClient. It enables the user to perform standard CRUD (create, read, update and delete) operations. The query controller is responsible for managing a user's query targeted at a database. The query controller is able to handle both a relational database (MySQL) and a non-relational database (MongoDB). The spark controller handles the users spark query. As my software is a working prototype, the controller currently only supports two spark operations on any given file: word search and count the frequency of every word. It is envisaged that a later version of the software would support richer spark queries with more parameters. It would provide additional spark operations such as performing data operations on a table, or running computing intensive tasks specified by the user.

4.2.1.2.2 Models

The models package contains classes that act as data models. Some of the models represent the entities from a table (such as Client), some are service classes providing behaviour to other classes (such as: URLService, QueryHandler and SparkHandler) and others act as temporary wrapper objects used during the submission of a form (such as: ClientWithInstructionWrapper). All of these models were carefully designed to ensure the system would remain loosely coupled. The advantage of a loosely coupled system is that classes are more independent, and so it simplifies testing and maintainability. In the models package there are also classes and methods that provide the communication, which will be discussed in section 4.2.3.

4.2.1.2.3 Properties

The properties package contains important configuration information for the application. For instance, it contains authentication settings such as determining which pages are accessible

by unauthenticated users. Furthermore, there are also a number of classes responsible for storing session information, including the number of successful queries, failed queries etc.

4.2.1.2.4 Repository

The repositories folder is used for database interaction. MServer connects to the database using a high-level interface provided by the Spring framework, called Spring Data JPA. Spring Data JPA is a high-level database access library, designed to remove the typical boiler plate required for database access, as well as giving support to a variety of SQL and NOSQL database systems. As it is only an interface, it does not have any implementation.

```
public interface ClientRepository extends JpaRepository<Client, Integer>{
    List<Client> findAll();

    Client findClientByClientID(int clientID);
}
```

Figure 7 - Spring Data JPA

Figure 7, demonstrates the simplicity of Spring Data JPA. The figure shows two methods that have been added; one to retrieve a list of clients (MClients), and another to retrieve an individual client by its ID. At runtime, Java Spring will automatically create a class which extends the interface and adds the implementation based on the methods signature.

4.2.2 MClient: backend

The MClient is a completely separate application that is independent from the MServer. To keep the software relatively simple to use, the MClient has been designed to be autonomous, and so will not require any user input. The MClient(s) await instruction from the server which will have been written by the user (and sent via the server). They will then be executed by the client. Once the query has been processed, the result is returned to the MServer.

The MClient follows an MVC architecture, but unlike the MServer that provides the user with a view, it returns the requested object. The MClient therefore has a Controller and a Model package. The controller package has two main classes; one that deals with database operations (SQL and NoSQL) and another that handles spark. The models package, like MServer, contains data models that have been carefully designed to perform a certain role. This includes classes which mirror those belonging to the MServer, such as Message, which is required for the deserialization of the object. The MClient also has a repository package that stores interfaces that deal with database queries. It also makes use of Spring Data JPA.

Generally the MClient simply waits for the MServer to make a request and send an object with instructions. Depending what mapping (URL) the MServer targets, it will specify the type of action required, which will be picked by one of the controllers. The controller will then deserialize the object and perform the instruction. The MClient will then write a response to the object, serialize it and return it back to the MServer.

4.2.3 Communication Layer

Having discussed the functionality of the front- and back-end, the next section will discuss the communication layer. The communication layer is important because it is how the user is able to retrieve information from the remote computer holding the data source. The communication is built with a REST architectural network, using HTTP as the protocol.

REST is the architecture used by the World Wide Web, which is proven to be scalable, performant and reliable; three elements that will assist my user in their analysis and management of Big Data. The MServer and MClient communicate via JSON REST requests, by using a Spring provided class called RestTemplate. I choose RestTemplate over the other viable solution, HttpClient, as it operated on a higher level of abstraction. This not only reduces some of the repetitive code associated with HttpClient, but also was a better fit for my project as RestTemplate provides automatic serialization to JSON.

Figure 8 shows one of the methods in the QueryHandler class on MServer. The method is used to check the status of a client, and return a Boolean. When the method is called, it constructs a URL using a custom-made service class called URLService, which encodes the URL. It then uses another static method, testResponse, to check that the client is connected (a response of 200 means it is online). Once it receives the response, it contacts the client via the object restTemplate. The restTemplate uses the getForObject() command to send the object to the client and demand a response. Depending on the client response, the appropriate Boolean is returned. This particular method is extremely useful to the application, as it is able to test if the client is connected, and so it will not attempt to perform a query if it returns false. This essentially speeds up the sending of a query considerably, as it will not wait for a never-occurring response. This will speed up the overall application, which will save the user time and provide quicker results.

```
private static RestTemplate restTemplate = new RestTemplate();


public static boolean healthHandLer(ClientWithInstruction clientWithInstruction){
    String url = URLService.urlEncoder(clientWithInstruction);
    url+= InstructionType.HEALTH.mapping();

    boolean connected = URLService.testResponse(url);

    if(connected){
        Health health = restTemplate.getForObject(url, Health.class);
        if(health != null && health.getStatus().equals("UP")){
            return true;
        }else{
            return false;
        }
    }else{
        return false;
    }

};
```

Figure 8 - Query Handler

The method shown in figure 8 is one of the simpler methods, but it shows the core functionality of the restTemplate. It creates a request using getForObject(), and waits for a response. It will then take the necessary steps required to determine the MClient's health status and return the appropriate Boolean.

4.3 Testing my middleware

I carried out a variety of tests on my software. These were functional, system, performance, boundary, acceptance, security and compatibility.

4.3.1 Functional testing

This uses use cases to test the functionality of the application.

Test 1 – Send a multi-database query to a client

Action	Input	Expected	Actual	Valid
Go to query	Query Client 1	queryIndex.html page loaded	queryIndex.html page loaded	Y
Select Queries to Perform (single Query)	QUERY_MYSQL select p from Person p where age > 34	4 results	4 results	Y
	QUERY_MYSQL select p from Person p where age > 34 [count]	4	4	Y
	QUERY_MONGO {firstName : 'Sarah', age: {$gte: 20}}	11 results	11 results	Y
	QUERY_MONGO {firstName : 'Sarah', age: {$gte: 20}} [count]	11	11	Y

Test 2 - Send a multi-spark query to a client

Action	Input	Expected	Actual	Valid
Go to query	Spark Client 1	sparkIndex.html page loaded	sparkIndex.html page loaded	Y
Select Queries to Perform (single Query)	WORDCOUNT d:\ben\desktop\test-small.txt sort: yes	{Key,value} list sorted by key	{Key,value} list sorted by key	Y
	WORDCOUNT d:\ben\desktop\test-small2.txt sort: no	{Key,value} list	{Key,value} list	Y
	WORDFIND d:\ben\desktop\test-small.txt [Manning]	Manning was found: 6 times	Manning was found: 6 times	Y
	WORDFIND d:\ben\desktop\test-small2.txt [Manning]	Manning was found: 0 times	Manning was found: 0 times	Y

Test 3 - Send a multi-client query to spark

Action	Input	Expected	Actual	Valid
Go to query	Spark Client 1	sparkIndex.html page loaded	sparkIndex.html page loaded	Y

Select Queries to Perform (single Query)	Machine 1 WORDCOUNT d:\ben\desktop\test-small.txt sort: yes	{Key,value} list sorted by key	{Key,value} list sorted by key	Y
	Machine 2 WORDSEARCH c:\harriet\desktop\test-file.txt [dog]	dog was found: 2 times	dog was found: 2 times	Y

4.3.2 System testing

This uses an end-to-end process to test the collective system.

Test 4 – Test all system features

Action	Input	Expected	Actual	Valid
Login	Login Server on Https.	Login.html page	Login.html page	Y
Authenticated	---	Index.html page	Index.html page	Y
Go to Client Management	Add a Client	createClient.html page loaded	sparkIndex.html page loaded	Y
Edit the Client	Edit	editClient.html page loaded with correct details	editClient.html page loaded with correct details	Y
Update the Client	Update Values	Redirect to clients.html with updated values	Redirected to clients.html with updated values	Y
Delete the Client	Delete	Redirect to clients.html with delete client	Redirected to clients.html with delete client	Y
Query Wizard	Repeat Test 1 (change input)	Test 1 Results (with changed input)	Test 1 Results	Y
Spark Wizard	Repeat Test 3 (change input)	Test 3 Results (with changed input)	Test 3 Results	Y
Query Wizard	Query Client: Health,Metrics, Shutdown	Connected, Metric-Data, Client has been shut down	Connected, Metric-Data, Client has been shut down	Y

4.3.3 Performance testing[5]

This assesses the application performance during normal working conditions, as well as testing the application boundaries by putting it under stress by telling it to execute lots of simultaneous queries.

Test 5 – Test Wordcount using range of file sizes

[5] Benchmarks set by: i7-3770K CPU @ 3.50GHz, 16GB RAM, WIN 7 (64x)

Action	Input	File Size	Execution Time mm:ss:ms
WORDCOUNT	d:\Ben\Desktop\test-small.txt	1.21 KB	00:00:54
	d:\Ben\Desktop\jungle-book28.txt	2.84 MB	00:00:80
	d:\Ben\Desktop\jungle-book94.txt	94 MB	00:04:99
	d:\Ben\Desktop\jungle-book188.txt	188 MB	00:09:53
	d:\Ben\Desktop\Test.txt	2.2GB	01:14:37
		Total Time	01:30:23

Test 6 – Test WordSearch using range of file sizes

Action	Input	File Size	Execution Time mm:ss:ms
WORDSEARCH	d:\Ben\Desktop\test-small.txt [Toomai]	1.21 KB	00:00:32
	d:\Ben\Desktop\jungle-book28.txt [Toomai]	2.84 MB	00:00:58
	d:\Ben\Desktop\jungle-book94.txt [Toomai]	94 MB	00:01:07
	d:\Ben\Desktop\jungle-book188.txt [Toomai]	188 MB	00:01:62
	d:\Ben\Desktop\Test.txt [Toomai]	2.2GB	00:07:14
		Total Time	00:10:73

Test 7 – Multi-spark word count on single client

Action	Input	File Size	Execution Time mm:ss:ms
SIMULTANEOUS WORDCOUNT	d:\Ben\Desktop\test-small.txt	1.21 KB	01:28:40
	d:\Ben\Desktop\jungle-book28.txt	2.84 MB	
	d:\Ben\Desktop\jungle-book94.txt	94 MB	
	d:\Ben\Desktop\jungle-book188.txt	188 MB	
	d:\Ben\Desktop\Test.txt	2.2GB	

Test 8 – Multi-spark search query on single client

Action	Input	File Size	Execution Time mm:ss:ms
SIMULTANEOUS WORDSEARCH	d:\Ben\Desktop\test-small.txt [Toomai]	1.21 KB	00:08:89
	d:\Ben\Desktop\jungle-book28.txt [Toomai]	2.84 MB	
	d:\Ben\Desktop\jungle-book94.txt [Toomai]	94 MB	
	d:\Ben\Desktop\jungle-book188.txt [Toomai]	188 MB	
	d:\Ben\Desktop\Test.txt [Toomai]	2.2GB	

Test 9 – Database queries on single client

Action	Input	Execution Time mm:ss:ms
QUERY_MYSQL	Select p From Person p where age > 34	00:00:52
QUERY_MONGO	{firstName : 'Sarah', age: {$gte: 20}}	00:00:49
SIMULTANEOUS QUERY_MYSQL QUERY_MONGO	Select p From Person p where age > 34 {firstName: 'Sarah', age: {$gte: 20}}	00:00:53

4.3.4 Compatibility testing

This tests the application by attempting to run it on a variety of OSs, and by accessing it via different web browsers.

Test 10 – Check browser compatibility

Page	Firefox (46.0.1)	Google Chrome	Internet Explorer (11.0.9600.18282)	Passed
Login.html	OK	OK	OK	Y
Clients.html	OK	OK	OK	Y
createClient.html	OK	OK	OK	Y
editClient.html	OK	OK	OK	Y
queryIndex.html	OK	OK	OK	Y
clientSender.html	OK	OK	OK	Y
clientSingleSender.html	OK	OK	OK	Y
clientResults.html	OK	OK	OK	Y
results.html (query)	OK	OK	OK	Y
sparkIndex.html	OK	OK	OK	Y
sparkSender.html	OK	OK	OK	Y
sparkSingleSender.html	OK	OK	OK	Y
results.html (spark)	OK	OK	OK	Y

Test 11 – Check for MClient compatibility

Test	Windows 7	Windows 10	Ubuntu 14.04	Passed
Boots up on OS	OK	OK	OK	Y
Queriable from MServer	OK	OK	OK	Y

4.3.5 Security/vulnerability testing

This will test the security of the Mserver, by attempting to bypass the login system. It will also test the security of the client by attempting to bypass the MServer and send queries directly to the MClient.

Test 12 – Test Authentication

Test	Expected	**Actual**	**Passed**
MServer requires authentication	Y	Y	Y
Cannot Access any page directly without authorization	Y	Y	Y
Incorrect password does not enable access	Y	Y	Y
Logout button logs user out	Y	Y	Y

Test 13 – SSL Connectivity

Test	Expected	**Actual**	**Passed**
MServer is accessed via an SSL connection	Y	Y	Y
Connection between MServer and MClient is encrypted	Y	N	N

MClient is only accessable via SSL connection	Y	N	N

Test 14 – Direct Access to MClient's properties and data

Test	Expected	**Actual**	**Passed**
Get Clients status via: http://clienturl/health	N	Y	N
Get Clients metrics via: http://clienturl/metric	N	Y	N
Shutdown client via: http://clienturl/shutdown	N	N	Y
Shutdown client via shell: curl –X POST clienturl/shutdown	N	Y	N

4.3.6 Acceptance testing

This will test whether the middleware is able to meet all of the users requirements, such as analysing structured, semi-structured and unstructured data.

Test 15 – Can user perform queries on structured, semi-structured and unstructured data

Data Format	Test	Expected	**Actual**	**Passed**
Structured	Query SQL Database (Limited Query)	Y	Y	Y
	Count results in a SQL Query	Y	Y	Y
Semi-Structured	Query NoSQL Database (Limited Query)	Y	Y	Y
	Count results in a NoSQL Query	Y	Y	Y
Unstructured	Count frequency of words in a text file	Y	Y	Y
	Search word (with number of reoccurences)	Y	Y	Y

Test 16 – Operations that a user can perform:

Action	Expected	**Actual**	**Passed**
Many Queries to Single Client (Database OR Spark)	Y	Y	Y
Single Queries to Many Clients (Database OR Spark)	Y	Y	Y
Many Queries to Single Client (Database AND Spark)	Y	N	N
Single Queries to Many Clients (Database AND Spark)	Y	N	N

4.4 Evaluation of my middleware

In order to evaluate my middleware, I decided to apply a number of different testing methodologies. These were: functional, system, performance, compatibility, security and acceptance testing.

4.4.1 What was my application good at

To determine whether my software was robust, I used the results derived from my functional and system tests. The application passed these tests because it makes heavy use of exception-handlers which ensure the application does not crash, and that it handles an exception appropriately.

The software is performant as it scales well and provides a timely result. I ran a number of successful performance tests with varying file sizes. The speeds were determined by calculating the difference in time from the user submitting the query, to the user receiving the

result. This application is a piece of middleware that links existing tools. These tools (i.e. MySQL, Mongo and Spark) process the analysis, and so performance is reliant on their processing speed. Nonetheless, the latency between sending an instruction from the MServer to the MClient is minimal, and this can be seen from comparing the results of tests {5,7} and {6,8}. The difference between executing each test individually and sending a simultaneous query is approximately 2 seconds, suggesting that the latency is around 0.2 seconds[6], which is insignificant.

During my tests, I tried to replicate a typical organisation's heterogeneous network, by running the MClient software on a number of different operating systems. As long as the Java version was compatible and it was connected to the network, the software did not experience any difficulty with operating on other hardware and operating systems, as shown in the compatibility test 11. Furthermore, the MServer software was accessed from a variety of browser, as shown in test 10. The dashboard did not have any browser interoperability issues. One of the key aims of the software was to provide a low-cost solution. The software is built entirely using open-source technologies. Open-source software enables users to fork[7] the project from a hosting service such as GitHub, and instantly begin configuring and running the software, incurring no additional costs or licensing fees. Furthermore, the initial setup and the interface is designed to be simple, as the user is guided via a series of 'wizards' and so minimal training is required.

The MServer has lots of control over the MClients. For this reason, I added an authentication page which will require a user to login before they are able to utilise or view the dashboard. If the user does not logout, after a certain time period, it will automatically request the user to re-enter their authentication details. Furthermore, the MServer has been setup to operate on a secure-socket layer. This can be seen by the MServers URL starting with HTTPS, which ensures the communication between the user and the MServer is encrypted. This is important as it prevents a hacker from intercepting the authentication details.

I performed an acceptance test to ensure that the software performed the necessary actions to make the middleware viable. The current version of my software is capable of querying all formats of Big Data: structured, semi-structured and unstructured data.

The software makes it extremely easy to add a new MClient to the network should the organisation's Big Data analysis and management requirements grow. If an additional data source was obtained, the organisation would need to install the MClient software onto the machine, connect to it via the MServer, and can begin analysing that data.

4.4.2 What was my application not good at

[6] 2 seconds was average difference between test {5,7} and {6,8}. 2 divided by the 5 tests showed a 0.4 latency from sending the query to receiving a response. This suggests an approximate 0.2ms latency from the MServer to the MClient

[7] Fork –a Git related term- refers to the action of taking a personal copy of a repository. These repositories are hosted on a Git repository hosting service, such as GitHub, which enable users to collaborate on open-source projects.

.

I had planned originally to adopt Test-Driven Development. I soon found that the rigidity of such a methodology was unsuited to my project. My lack of experience with a number of the tools (Gradle, Thymeleaf and NoSQL), frameworks (Spring Boot and Spark) and architectures (MVC and REST) meant that I encountered some unforeseen circumstances, issues and limitations that I did not expect during development. This made it extremely difficult to plan and design the classes and system architecture in the way required by Test-Driven Development. One example of this happened when I was unsure how to map the fields of a table that had many rows and columns to the correct object property. I was trying to edit the properties of a list of objects simultaneously. This is critical to my application if I wanted to enable a user to send queries to a number of clients simultaneously, and I originally thought it would be a simple operation.

Figure 9 demonstrates what I was trying to achieve. The empty objects were loaded into a table, the user would then add values to the properties, and then the edited objects (the bloated arrows) would be posted by the form and retrieved by the specified controller. This wasn't happening, however, because Thymeleaf does not natively support the mapping of a list of objects. This meant that when the user added a property and submitted the form, the retrieved list of objects was empty.

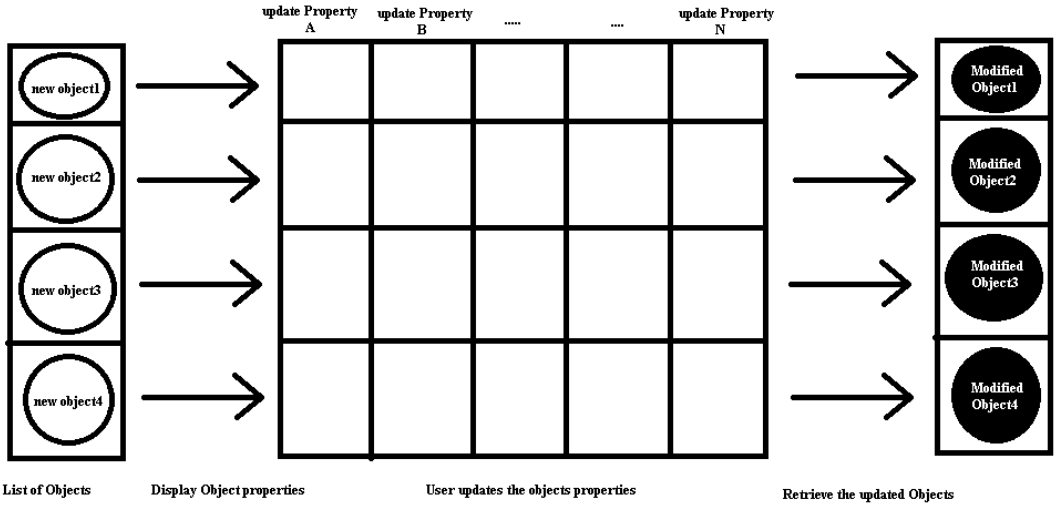


Figure 9 - Updating a list of objects

After extensive research, I discovered that I had to wrap the list in a wrapper object (123) as the wrapper object was required to encapsulate the list. Once the form was posted, the list would be saved inside the wrapper object which would then be picked up by the controller.

The problem above was one of many issues that I experienced during the development of my software. If I had tried to utilize the test-driven development methodology, I would have been forced to change the methods signature numerous times in order to cater for the additional objects required. For this reason, I decided not to use Test-Driven development.

Another issue that I found during the testing stage was directly related to the concept of distributed computing. Something I hadn't realised, was that the creation of software that acts as a bridge between other software is only viable if it provides the user with all the necessary

information to perform an action. When the user performs a Spark query, my software assumes the user knows the absolute path to the file. This may work in extremely simple contexts, but realistically, the user cannot be expected to know the full file structure in every case. In order for the software to become viable it would require a system that would provide information on the file structure, so they are able to directly select the file to query, rather than manually inputting the full file address. The issue of not providing the user with all the necessary information also applies to the users EJB queries. Here they are also expected to know the full database structure beforehand, and therefore some visual aid would be required to make the software viable. The software was only able to pass the acceptance test 15, by specifying that the query would be limited, and not take into consideration the user experience of performing the action.

Another issue with the software that forced me to amend the initial design, was the limitation of sending a query to numerous clients. Test 16 revealed that it failed a number of acceptance cases. The original design of the software envisaged the user sending just one query per client selected. A user may want to send numerous queries to a client, whether it was various database queries, retrieve the available memory of the system or analyse the various files using spark. For this reason, I was forced to design a new interface which would enable the user to send many queries to a single client.

Figure 10A demonstrates my original design, and 10B my extended implementation. Ideally, these two behaviours should be merged in a final version of the software, however, due to the complexity and time-constraints of this project, I was unable to construct such a model. A user would expect to mix their queries regardless of data formats and unfortunately, the current prototype cannot do this, causing the software to fail some of the acceptance tests.

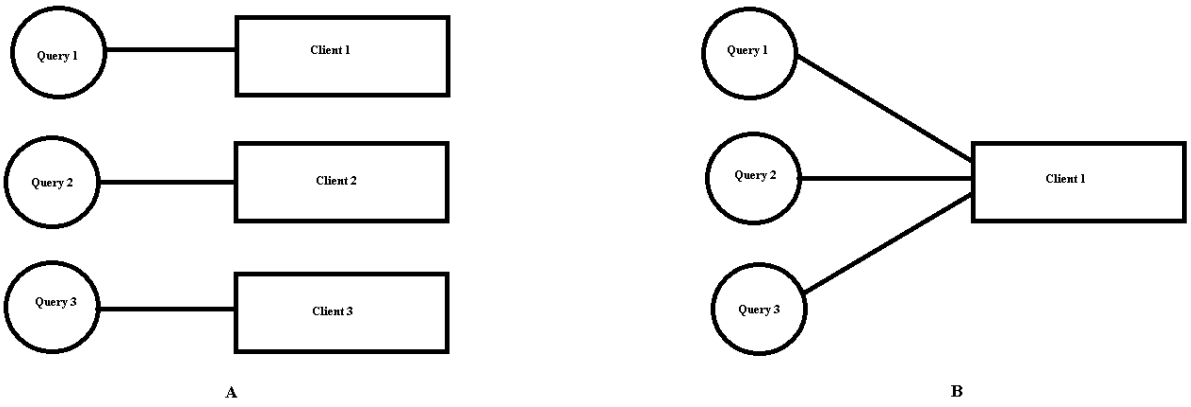


Figure 10 - Query limitations

The software also has a concerning security loophole which was uncovered during security tests 13 and 14. Although it has basic authentication (login system), and the connection to the MServer is encrypted using SSL communication, the data transmitted is unencrypted. This is not problematic in a private network, but if the communication travels via the internet this would be a major security concern. Furthermore, the software also failed the direct-accessibility test (test 14), that demonstrated how there is nothing stopping a user from

directly accessing the data. Although I was unable to configure curl[8] to access their data, I believe a more competent user would be able to view and download the data. Furthermore, I was also able to shutdown one of the MClients by issuing a simple curl POST command, which is a concern.

There are other limitations of the software that need to be addressed before the project is production ready. For instance, an external configuration file would remove the need to launch and configure the application through an IDE, which is impractical. Furthermore, as the user is generally entering instructions as plain text, errors and logic mistakes are likely. Ideally, the input would be validated before it is sent to the client to ensure minor mistakes are avoided, and thus saving improve the usability of the software.

[8] Curl is a tool to transfer data from or to a server

Conclusion

This paper set out to identify what the common problems are that are encountered by organisations analysing and managing Big Data, and to explore whether these problems could be helped by using distributed computing.

During my endeavour to answer this question, I have identified that the Internet of Things is rapidly growing, which, although not the largest contributor, is adding to the mass of Big Data being generated. My research identified what constitutes Big Data, and ascertained that the field of Big Data is also growing very quickly, with unstructured data being generated in the largest quantities, compared with structured and semi-structured. I discovered that Big Data has been a driving force in a variety of new discoveries, from finding cures for diseases, to reducing crime. This highlighted the importance of creating tools that are able to mine Big Data.

I wanted to understand what the problems were that surround the field of Big Data management and analysis. My exploration of a variety of procedural activities associated with working with Big Data, identified a range of interconnected factors making Big Data difficult to manage and analyse. These were the variety of the formats in which Big Data is created, the large volume that it is created in, the speed at which new data is being generated, and the costly nature of hardware, software and skillsets capable of working with the data. I then backed up these findings by analysing real life examples of failed Big Data projects, which concreted my findings as there were correlations between the factors I had highlighted to be problematic, and the reasons why the projects failed. My research into the current Big Data management and analysis packages currently servicing the market revealed that there are plenty of effective tools accessible to organisations operating within large budgets, but revealed that there are limited tools for organisations operating on smaller budgets. The open source software that is available to organisations for free, such as Hadoop, proved too complex for a non-specialist team to use. My research uncovered an unnerving way of thinking surrounding the Big Data field; that Big Data analysis is only available to those with big budgets. The valuable insights that are potentially locked away within Big Data sets, highlight the importance of bringing an accessible piece of software to open Big Data management and analysis to everyone, and combat this way of thinking.

The problematic factors of Big Data have shaped the data analysis and management landscape, leading to the creation of more flexible databases, and specialist analysis tools. A review of the current hardware proved centralized systems to be unsuitable for working with Big Data, because they create a single point of failure, and are susceptible to bottlenecks. In addition, I discovered that centralized systems are subject to a cap in potential processing power subject to current technological limitations. Distributed computing on the other hand, negates these risks; it uses a network of systems to eliminate the possibility of a single point of failure, and is highly scalable, meaning that the processing power can be increased as required.

The problematic factors of Big Data that my study revealed, are alleviated by utilising distributed systems. Mainly this is accountable to the additional processing power that enables systems to work with the data despite its extreme volume and variety, but it also

helps to combat the costly nature of Big Data. Distributed systems are designed to run on commodity hardware, thereby negating the requirement to buy state-of-the-art machinery. In addition, (and although not all distributed systems utilise this method) the distributed system structure maintains data integrity by removing the need to move the data set across a network, by bringing the computation to the data.

Based on the gap in Big Data analysis and management tools suitable for small and medium sized organisations that my research unveiled, I decided to create a tool for this market. I knew that it needed to be relatively easy to use, and cost very little. My research uncovered open source storage and analysis tools, which enabled me to focus my application on creating a piece of middleware that harnesses the processing power of distributed computing, to offer an interface from which these tools are accessible. My middleware is a working prototype capable of performing simple analyses of Big Data on non-local machines.

My software solves some of the problems of Big Data analysis and management. It is open-source, it fits in a heterogeneous network and can query structured, semi-structured and unstructured data, but has some limitations. For example it has limited functionality, and security issues. However, with further development these are problems that I am confident can be overcome.

Perhaps the most concerning problem surrounding the management and analysis of Big Data, is that existing tools are only accessible to multi-million pound corporations. Until there are accessible analysis tools for everyone, Big Data will continue to hold its secrets. We could already be sitting on the data we need to bridge the way into the next age; the age of knowledge.

Table of Figures

Appendices

Appendix A – Software Used

		Name	Type	Link
Hardware	Computer	Desktop – Intel Core i7 3770k 16GB RAM	Personal	
	Computer	Laptop – Inter Core i3 4GB RAM	Personal	
Software	Operating Systems	Windows 7 Home Edition	Commercial	https://www.microsoft.com
		Windows 10 Home Edition*	Commercial	https://www.microsoft.com
		Linux Ubuntu 14.04*	Open-source	http://www.ubuntu.com/
	Tools	IntelliJ IDEA Ultimate Edition 15	Commercial	https://www.jetbrains.com/idea
		Gradle	Open-source	http://gradle.org/
		Spark	Open-source	http://spark.apache.org/
		MySQL	Open-source	https://www.mysql.com/
		MongoDB	Open-source	https://www.mongodb.com/
	Languages and Frameworks	Java JDK 1.8.0	Open-source	https://www.oracle.com/uk/java
		Spring 4	Open-source	http://spring.io/
		Spring Boot	Open-source	http://projects.spring.io/spring-boot/
		HTML 5	Open-source	https://www.w3.org/
		CSS	Open-source	https://www.w3.org/
		Bootstrap 3	Open-source	http://getbootstrap.com/
		jQuery 2.2	Open-source	https://jquery.com/
		Javascript	Open-source	https://www.javascript.com/
		EJB	Open-source	https://docs.oracle.com/cd/E12840_01/wls/docs103/ejb/EJB-QL.html

Appendix B – Brief introduction to some of the tools utilised to build the software

Gradle

My middleware uses an assembly tool called Gradle, which is a relatively new open-source build automation tool. Its purpose is to enable efficient project automation by automatically compiling the project, packaging the software and managing dependencies. It is similar to older build-tool Maven, but instead of XML configuration, Gradle uses a Groovy-based Domain Specific Language, which is considerably less verbose. Below is a sample of the Gradle configuration file:

```
apply plugin: 'java'
apply plugin: 'eclipse'
apply plugin: 'idea'
apply plugin: 'spring-boot'

jar {
    baseName = 'mserver'
    version = '1.0.0-SNAPSHOT'
}
sourceCompatibility = 1.8
targetCompatibility = 1.8

repositories {
    mavenCentral()
}


dependencies {
    compile('org.springframework.boot:spring-boot-starter-web')
    compile("org.springframework.boot:spring-boot-starter-thymeleaf")
    compile("org.springframework.boot:spring-boot-starter-data-jpa")
    compile("com.h2database:h2")
    compile('mysql:mysql-connector-java')

    compile group: 'org.apache.httpcomponents', name: 'httpclient', version: '4.5.2'

    //spring security
    compile("org.springframework.boot:spring-boot-starter-security")

    //dev tools dependency
    compile "org.springframework.boot:spring-boot-devtools"

    testCompile('org.springframework.boot:spring-boot-starter-test')
    testCompile("junit:junit")
}
```

I chose Gradle because it is open-source (and so would help to keep my software free) and I knew it would considerably speed up my development. Gradle has a number of useful features (such as automatic dependency management) that are a great aid in development.

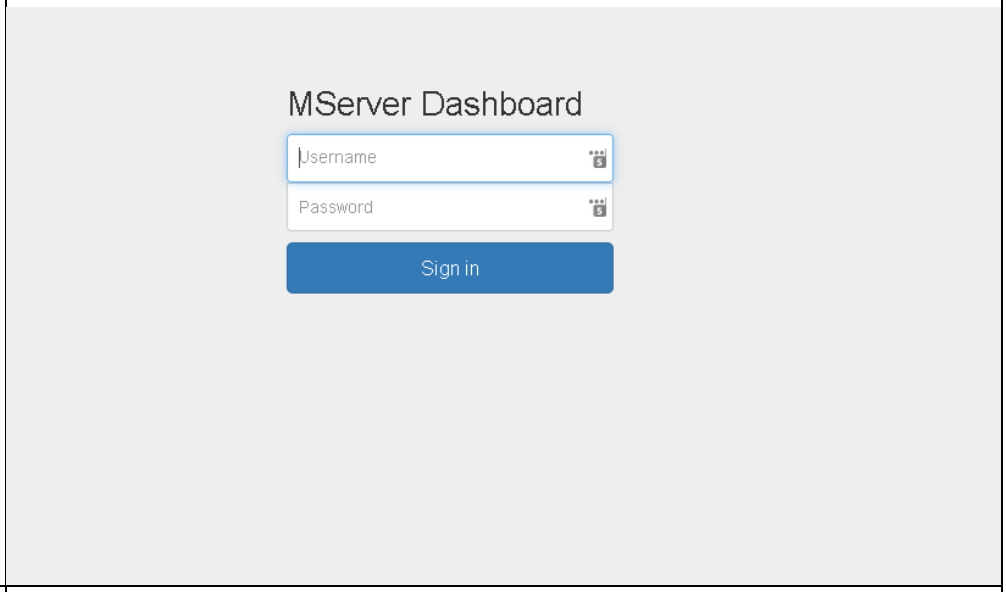
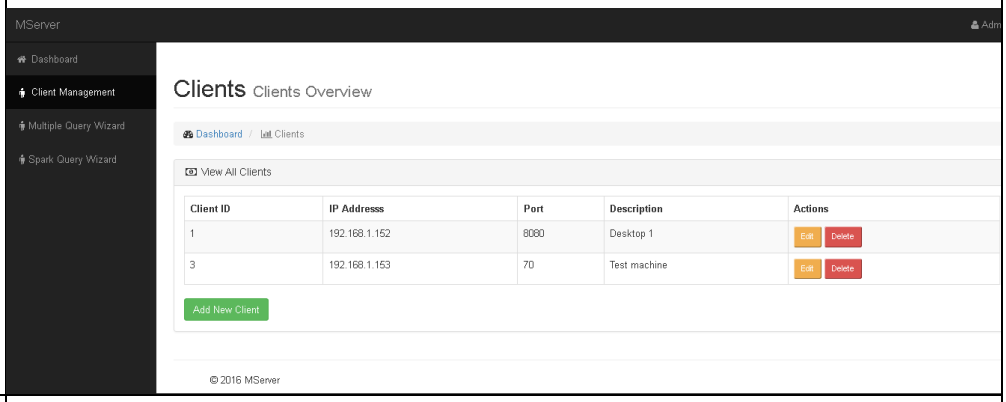
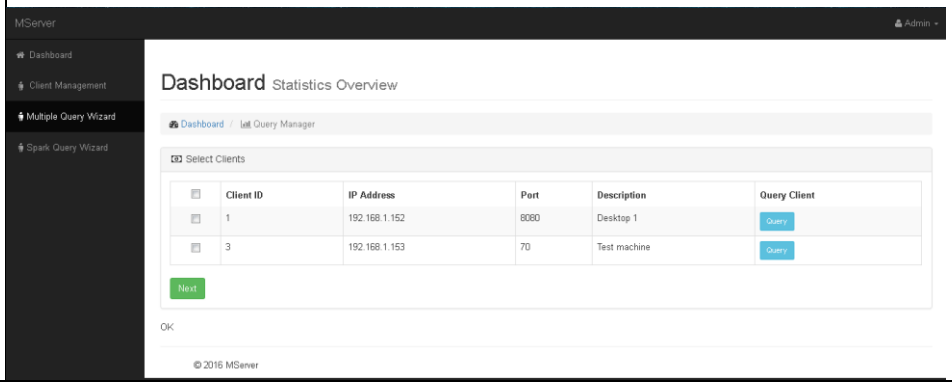
This will be useful when the software is made open-source and other developers begin modifying it.

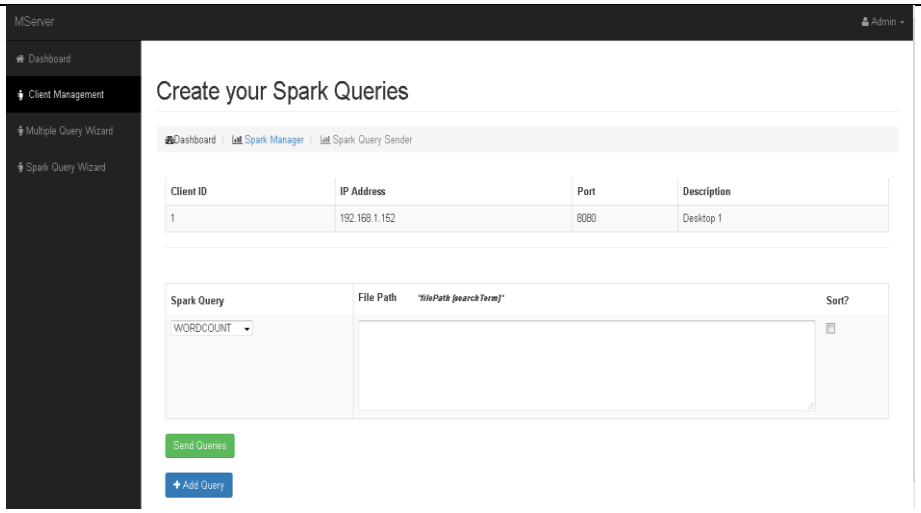
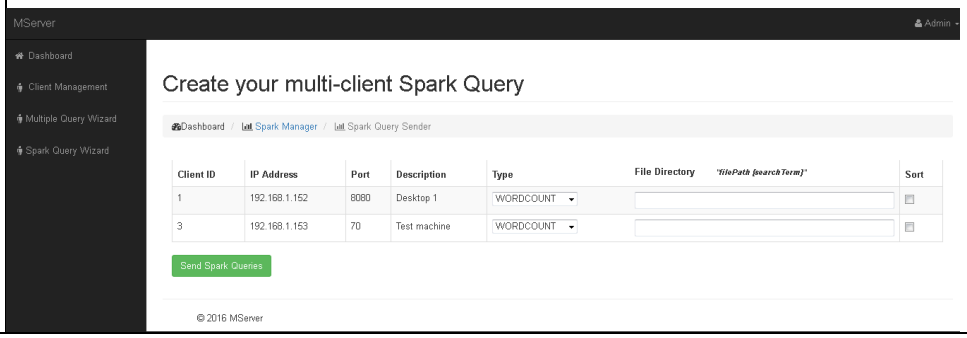
Spring Boot

The project has been built using a higher-level of the Spring framework, Spring Boot. The release of Spring Boot was in response to the steep learning curve associated with a Spring application (124). Boot aims to simplify the development process by taking an 'opinionated view of the Spring platform' (125). In doing so, it makes assumptions about the project by providing a range of non-functional features that are commonly used such as embedded servers and configuration files. Spring Boot is open-source, and so there will not be any additional costs to the user. It provides an excellent platform for a developer to quickly begin amending its native behaviour, so it may be beneficial to an organisation who aim to extend the software's functionality.

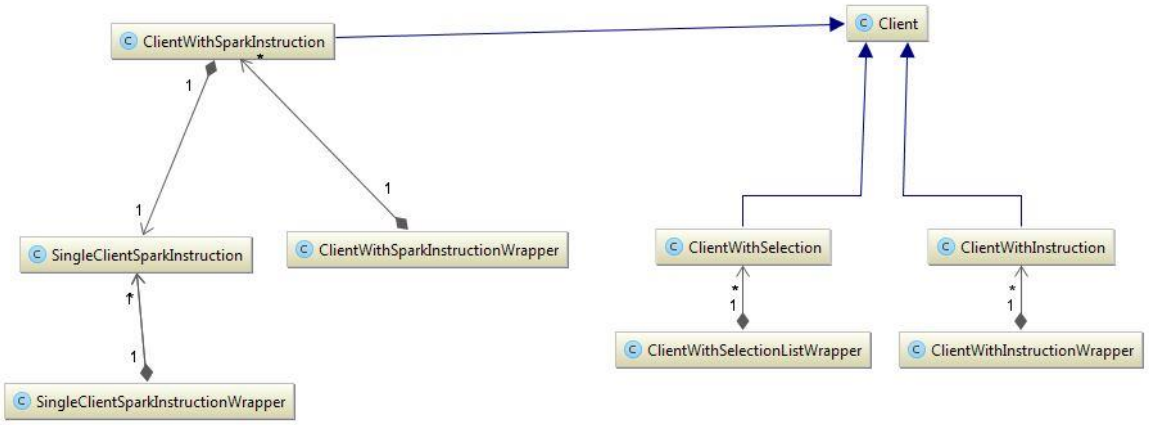
Spring Boot offers an auto-configuration annotation, which attempts to configure the application based on dependencies present in the project (126). Without this auto-configuration, it would require a number of explicit configuration files, which slows down and complicates development. The Spring team have constructed a number of pre-bundled packages that help extend the behaviour of the application. Among other functionality, the packages can add MVC functionality and database access via Spring data JPA. These packages contain various dependencies that will aid internal maintenance, should an IT company feel competent to extend the software's functionality.

Appendix C – Front-end Pages

Login.html	
Index.html	
clientIndex.html	
queryIndex.html	

sparkSingleSender.html	
sparkSender.html	

Appendix D – MServer Class Diagram



Note: the diagram represents the main hierarchical relationship between the main classes. There are a number of service classes which have been removed for clarity.

Bibliography

1. Frenzel L. Drowning in Information But Starved for Knowledge [Internet]. 2011 [cited 2016 May 30]. Available from: http://electronicdesign.com/blog/drowning-information-starved-knowledge

2. EMC. EMC News: New Digital Universe Study Reveals Big Data Gap: Less Than 1% of World's Data is Analyzed [Internet]. 2012 [cited 2016 Apr 5]. Available from: http://www.emc.com/about/news/press/2012/20121211-01.htm

3. Sriram RD, Sheth A. Internet of Things Perspectives. IT Prof [Internet]. 2015;17(3):60–3. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7116432

4. Government Office for Science. The Internet of Things: making the most of the Second Digital Revolution. 2014;(1st):15, 40.

5. Kröhnert S. EPTC: Internet of Things. 2015 [cited 2016 Mar 21]; Available from: http://www.eptc-ieee.net/IOT_series_EPTC_2015.pdf

6. Bonomi F, Milito R, Zhu J, Addepalli S. Fog Computing and Its Role in the Internet of Things. Proc first Ed MCC Work Mob cloud Comput [Internet]. 2012;13–6. Available from: http://doi.acm.org/10.1145/2342509.2342513\npapers2://publication/doi/10.1145/2342509.2342513

7. Gartner. Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016, Up 30 Percent From 2015 [Internet]. 2015 [cited 2016 Feb 26]. Available from: http://www.gartner.com/newsroom/id/3165317

8. McLellan C. The Internet of Things and Big Data: Unlocking the Power [Internet]. 2015 [cited 2016 Apr 25]. Available from: http://www.zdnet.com/article/the-internet-of-things-and-big-data-unlocking-the-power/

9. Dabhade KR. Big Data An Overview. Int J Sci Technol Res [Internet]. 2014;3(10):255–7. Available from: http://www.ijstr.org/final-print/oct2014/Big-Data-An-Overview.pdf

10. Najafabadi MM, Villanustre F, Khoshgoftaar TM, Seliya N, Wald R, Muharemagic E. Deep learning applications and challenges in big data analytics. J Big Data [Internet]. 2015;2(1):1, 14–9. Available from: http://www.journalofbigdata.com/content/2/1/1

11. Zicari R V. Big Data: Challenges and Opportunities. Big Data Comput [Internet]. 2013;111–2, 117. Available from: http://www.odbms.org/wp-content/uploads/2013/07/Big-Data.Zicari.pdf

12. Buckley J. Why the Majority of Big Data Projects Fail [Internet]. 2015 [cited 2016 Apr 3]. Available from: https://www.qubole.com/blog/big-data/big-data-projects-fail/

13. Tavana M, Puranam K. Handbook of Research on Organizational Transformations through Big Data Analytics [Internet]. IGI Global; 2014 [cited 2016 Apr 29]. 561 p. Available from: https://books.google.com/books?id=iqGXBgAAQBAJ&pgis=1

14. Baker W, Kiewell D, Winkler G. Using big data to make better pricing decisions. 2014

[cited 2016 Apr 29];(McKinsey & Company). Available from: http://www.mckinsey.com/business-functions/marketing-and-sales/our-insights/using-big-data-to-make-better-pricing-decisions

15. Stevens M. Can Big Data Help Keep Us Safe? [Internet]. IBM. 2013 [cited 2016 Apr 29]. Available from: http://www.ibmbigdatahub.com/blog/can-big-data-help-keep-us-safe

16. Kayyali B, Knott D, Kuiken S Van. The big-data revolution in US health care : Accelerating value and innovation. McKinsey Co [Internet]. 2013;(April):1–6. Available from: https://digitalstrategy.nl/wp-content/uploads/E2-2013.04-The-big-data-revolution-in-US-health-care-Accelerating-value-and-innovation.pdf

17. Leonard J. How BroadReach is using analytics to improve healthcare in Africa [Internet]. Computing. 2014 [cited 2016 Apr 29]. Available from: http://www.computing.co.uk/ctg/news/2381463/how-broadreach-is-using-analytics-to-improve-healthcare-in-africa

18. BBC. The challenge of saving lives with “big data” [Internet]. 2016 [cited 2016 Apr 29]. Available from: http://www.bbc.co.uk/news/health-35491177

19. Stanford. What it means for out health and the future of medical research [Internet]. 2012 [cited 2016 May 4]. Available from: http://sm.stanford.edu/archive/stanmed/2012summer/article3.html

20. IBM. IBM - Bringing big data to the enterprise - What is big data? - Australia [Internet]. IBM Corporation; 2016 [cited 2016 Feb 26]. Available from: http://www-01.ibm.com/software/au/data/bigdata/

21. Skies B. Processing Distributed Internet of Things Data in Clouds. 2015;76–9. Available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7091808&tag=1

22. Veeranjaneyulu N, Bhat MN, Raghunath A. Approaches for Managing and Analyzing Unstructured Data. 2014;6(01):19–24. Available from: http://www.enggjournals.com/ijcse/issue.html?issue=20140601

23. Cisco. Cisco Global Cloud Index: Forecast and Methodology, 2014–2019 White Paper [Internet]. 2014 [cited 2016 Feb 26]. Available from: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html

24. Fortinet. Securing Big Data: Big Data Security impact, challenges and solutions. 2014;1–4. Available from: https://www.fortinet.com/sites/default/files/solutionbrief/SB-Securing-Big-Data.pdf

25. Stedman C. IoT data analytics spurred on by big data’s expansion [Internet]. 2015 [cited 2016 Mar 10]. Available from: http://searchbusinessanalytics.techtarget.com/feature/IoT-data-analytics-spurred-on-by-big-datas-expansion

26. SAS. Big Data Meets Big Data Analytics. Sas [Internet]. 2012;1–13. Available from: http://www.sas.com/content/dam/SAS/en_us/doc/whitepaper1/big-data-meets-big-data-analytics-105777.pdf

27. Normandeau K. Big data volume, variety, velocity and veracity [Internet]. 2013 [cited 2016 Apr 11]. Available from: http://insidebigdata.com/2013/09/12/beyond-volume-variety-velocity-issue-big-data-veracity/

28. Rouse M. 3Vs (volume, variety and velocity) [Internet]. 2013 [cited 2016 Apr 11]. Available from: http://whatis.techtarget.com/definition/3Vs

29. Zikopoulos P, Eaton C, DeRoos D. Understanding Big Data [book] [Internet]. New York et al: McGraw …. 2012. 166 p. Available from: http://www.lavoisier.fr/livre/notice.asp?ouvrage=2609842

30. O'Leary D. Big Data, The Internet Of Things and The Internet Of Signs [Internet]. 2013 [cited 2016 Feb 12]. p. 54. Available from: http://ud7ed2gm9k.search.serialssolutions.com/?ctx_ver=Z39.88-2004&ctx_enc=info:ofi/enc:UTF-8&rfr_id=info:sid/summon.serialssolutions.com&rft_val_fmt=info:ofi/fmt:kev:mtx:journal&rft.genre=article&rft.atitle='BIG+DATA',+THE+'INTERNET+OF+THIN

31. Zikopoulos P, DeRoos D, Parasuraman K, Deutsch T, Corrigan D, Giles J. Harness the Power of Big Data: The IBM Big Data Platform [book]. Journal of Chemical Information and Modeling. 2013. 1689-1699 p.

32. O'Leary D. Big Data, The Internet Of Things and The Internet Of Signs [Internet]. 2013 [cited 2016 Feb 12]. Available from: http://ud7ed2gm9k.search.serialssolutions.com/?ctx_ver=Z39.88-2004&ctx_enc=info:ofi/enc:UTF-8&rfr_id=info:sid/summon.serialssolutions.com&rft_val_fmt=info:ofi/fmt:kev:mtx:journal&rft.genre=article&rft.atitle='BIG+DATA',+THE+'INTERNET+OF+THIN

33. Bhatia RK, Bansal A. A Closer Looks Over Hadoop Framework. 2014;14(3):150–2. Available from: http://ud7ed2gm9k.search.serialssolutions.com/?ctx_ver=Z39.88-2004&ctx_enc=info:ofi/enc:UTF-8&rfr_id=info:sid/summon.serialssolutions.com&rft_val_fmt=info:ofi/fmt:kev:mtx:journal&rft.genre=article&rft.atitle=A+Closer+Looks+Over+Hadoop+Framework&rft.

34. Eltabakh M. Semi-Structured Data. 2012;(Xml):42–5.

35. MongoDB. NoSQL Databases Explained | MongoDB [Internet]. 2016 [cited 2016 Mar 29]. Available from: https://www.mongodb.com/nosql-explained

36. Microsoft. Pre-processing and serializing the data [Internet]. 2016 [cited 2016 Apr 8]. Available from: https://msdn.microsoft.com/en-gb/library/dn749786.aspx

37. Steward D. Big Content: The Unstructured Side of Big Data [Internet]. 2013 [cited 2016 Feb 26]. Available from: http://blogs.gartner.com/darin-stewart/2013/05/01/big-content-the-unstructured-side-of-big-data/

38. Bacchelli A, Bettenburg N, Guerrouj L. Workshop on mining unstructured data (MUD)... because "mining unstructured data is like fishing in muddy waters"! Proc - Work Conf Reverse Eng WCRE [Internet]. 2012;5–6. Available from: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6385096&tag=1

39. Hitachi. In the Age of Unstructured Data, Enterprise-Class Unified Storage Gives IT a Business Edge - enterprise-class-unified-storage-in-the-age-of-unstructured-data-

white-paper.pdf [Internet]. 2014 [cited 2016 Feb 26]. Available from: https://www.hds.com/assets/pdf/enterprise-class-unified-storage-in-the-age-of-unstructured-data-white-paper.pdf

40. IBM. IBM Analytics – Overview: Innovate with Analytics Tools – United States [Internet]. IBM Corporation; 2013 [cited 2016 Mar 31]. Available from: http://www.ibm.com/analytics/us/en/what-is-smarter-analytics/innovate-with-analytics-tools.html

41. Stickland E. Big Data Beats Cancer [Internet]. 2015 [cited 2016 May 16]. Available from: http://spectrum.ieee.org/biomedical/diagnostics/big-data-beats-cancer

42. Brown B, Chui M, Manyika J. Are you ready for the era of "big data"? [Internet]. 2011 [cited 2016 May 16]. Available from: http://www.mckinsey.com/business-functions/strategy-and-corporate-finance/our-insights/are-you-ready-for-the-era-of-big-data

43. IBM. ROI Case Study - IBM SPSS. 2012;(M153):3. Available from: http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=YTL03131USEN&appname=skmwww

44. Reilly TO, Winge S, Schneider S. Big Data: Release 2.0 Issue 2.0.11. O'Reilly Media, Inc. 2009;(2):4.

45. Clark J. Will Storage Kill Big Data? - The Data Center Journal [Internet]. 2013 [cited 2016 Apr 8]. Available from: http://www.datacenterjournal.com/storage-kill-big-data/

46. Bradbury D. Cutting edge security: Expensive kit won't save you [Internet]. 2016 [cited 2016 Apr 18]. Available from: http://www.theregister.co.uk/2016/04/13/cutting_edge_security_its_not_about_tools/

47. Komorowski M. A history of storage cost (update) [Internet]. 2014 [cited 2016 Apr 7]. Available from: http://www.mkomo.com/cost-per-gigabyte-update

48. Russom P. Managing Big Data. Tdwi Res. 2013;Fourth Qua:10, 19.

49. Mearian L. Consumer SSDs and hard drive prices are nearing parity [Internet]. 2015 [cited 2016 Apr 7]. Available from: http://www.computerworld.com/article/3010395/solid-state-drives/consumer-ssds-and-hard-drive-prices-are-nearing-parity.html

50. Backblaze. How long do disk drives last? [Internet]. 2013 [cited 2016 Apr 11]. Available from: https://www.backblaze.com/blog/how-long-do-disk-drives-last/

51. Smith S. The Real Cost of Data Loss And How To Prevent It [Internet]. 2015 [cited 2016 Apr 19]. Available from: http://www.windowsnetworking.com/articles-tutorials/netgeneral/real-cost-data-loss-and-how-prevent-it.html

52. Kovacs E. Downtime and Data Loss Cost Enterprises $1.7 Trillion Per Year: EMC [Internet]. 2015 [cited 2016 Apr 19]. Available from: http://www.securityweek.com/downtime-and-data-loss-cost-enterprises-17-trillion-year-emc

53. NT B. 5 advantages and disadvantages of Cloud Storage [Internet]. 2014 [cited 2016 Apr 18]. Available from: http://bigdata-madesimple.com/5-advantages-and-

disadvantages-of-cloud-storage/

54. Castelino C, Gandhi D, Narula HG, Chokshi NH. Integration of Big Data and Cloud Computing. 2014;16(2):100–2. Available from: http://www.ijettjournal.org/volume-16/number-2/IJETT-V16P220.pdf

55. Gholami A, Laure E. Security and Privacy of Sensitive Data in Cloud Computing : A Survey of Recent Developments. Comput Sci Inf Technol (CS IT) [Internet]. 2015;131–46. Available from: http://arxiv.org/abs/1601.01498

56. Microsoft. Pricing – Cloud Storage | Microsoft Azure [Internet]. 2016 [cited 2016 Apr 18]. Available from: https://azure.microsoft.com/en-gb/pricing/details/storage/

57. Tsagklis I. Advantages and Disadvantages of Cloud Computing - Cloud computing pros and cons [Internet]. 2013 [cited 2016 May 5]. Available from: https://www.javacodegeeks.com/2013/04/advantages-and-disadvantages-of-cloud-computing-cloud-computing-pros-and-cons.html

58. Barsoum A. Cloud storage promising, but security concerns remain - San Antonio Express-News [Internet]. 2015 [cited 2016 Apr 18]. Available from: http://www.mysanantonio.com/opinion/commentary/article/Cloud-storage-promising-but-security-concerns-6471971.php

59. Aleem A, Sprott CR, Walterbusch M, Martens B, Teuteberg F. Let me in the Cloud: Analysis of the benefit and risk assessment of Cloud Platform. J Financ Crime Ind Manag & Data Syst Libr Hi Tech News Manag Res Rev Downloaded by Univ GREENWICH [Internet]. 2011;20(11):6–24. Available from: http://dx.doi.org/10.1108/13590791311287337\nhttp://\nhttp://dx.doi.org/10.1108/07419050911010741

60. Leopold G. Security Concerns Grow As Big Data Moves to Cloud [Internet]. datanami. 2016 [cited 2016 Apr 29]. Available from: http://www.datanami.com/2016/02/24/security-concerns-grow-as-big-data-moves-to-cloud/

61. Feinleib D. Big Data And Storage: Why You Can’t Save Everything Forever - Forbes [Internet]. 2012 [cited 2016 Apr 8]. Available from: http://www.forbes.com/sites/davefeinleib/2012/10/09/big-data-and-storage-why-you-cant-save-everything-forever/#376c37ed5f70

62. Einstein D. Big Data Needs Big Storage: Where to Keep Gigabytes, Terabytes and Petabytes of Data [Internet]. 2012 [cited 2016 Apr 11]. Available from: http://www.forbes.com/sites/netapp/2012/08/15/big-data-needs-big-storage-where-to-keep-gigabytes-terabytes-and-petabytes-of-data/#5ef416545550

63. Rouse M. What is big data management? [Internet]. 2013 [cited 2016 May 24]. Available from: http://searchdatamanagement.techtarget.com/definition/big-data-management

64. Rouda N. Getting Real About Big Data: Build Versus Buy. 2014;(July):16. Available from: http://www.oracle.com/us/corporate/analystreports/esg-getting-real-bigdata-2228170.pdf

65. Delgado R. Why Big Data Projects Fail [Internet]. 2015 [cited 2016 Apr 29]. Available

from: http://www.business.com/technology/5-reasons-most-companies-big-data-projects-will-fail/

66. Rounda N, DeMattia A. The Surprising Economics of Engineered Systems for Big Data. 2015;(December):3, 7. Available from: http://www.oracle.com/us/technologies/big-data/eng-systems-for-big-data-esg-wp-2852701.pdf

67. PayScale. Data Scientist, IT Salary (United Kingdom) [Internet]. 2016 [cited 2016 May 24]. Available from: http://www.payscale.com/research/UK/Job=Data_Scientist%2c_IT/Salary

68. ICalculator. UK Average earning for 2016 with salary illustration [Internet]. 2016 [cited 2016 May 24]. Available from: http://www.icalculator.info/news/UK_average_earnings_2014.html

69. Stackpole B. Match in-memory processing speed to big data analytics business need [Internet]. 2013 [cited 2016 Apr 25]. Available from: http://searchbusinessanalytics.techtarget.com/feature/Match-in-memory-processing-speed-to-big-data-analytics-business-need

70. SAS. In-Memory Analytics [Internet]. 2016 [cited 2016 Apr 25]. Available from: http://www.sas.com/en_us/software/in-memory-analytics.html

71. Beck J. Unstructured data: A big deal in big data. 2013;1–3. Available from: http://www.digitalreasoning.com/resources/Holistic-Analytics.pdf

72. Google. Machine Translation [Internet]. 2016 [cited 2016 May 8]. Available from: http://research.google.com/pubs/MachineTranslation.html

73. Grajales C. The statistics behind Google Translate [Internet]. 2015 [cited 2016 May 29]. Available from: http://www.statisticsviews.com/details/feature/8065581/The-statistics-behind-Google-Translate.html

74. Morrison T. In-memory technology pushes analytics boundaries, boosts BI speeds [Internet]. 2013 [cited 2016 Apr 25]. Available from: http://searchbusinessanalytics.techtarget.com/feature/In-memory-technology-pushes-analytics-boundaries-boosts-BI-speeds

75. Castanedo F. New scalable solutions for data analysis with R - O'Reilly Radar [Internet]. 2014 [cited 2016 Apr 25]. Available from: http://radar.oreilly.com/2014/07/new-scalable-solutions-for-data-analysis-with-r.html

76. Hack M. Use Data to Tell the Future: Understanding Machine Learningng Machine Learning | WIRED [Internet]. 2014 [cited 2016 Apr 27]. Available from: http://www.wired.com/insights/2014/03/use-data-tell-future-understanding-machine-learning/

77. Gartner. How to Prevent Big Data Analytics Failures [Internet]. 2015 [cited 2016 Apr 13]. Available from: http://www.gartner.com/smarterwithgartner/how-to-prevent-big-data-analytics-failures/

78. Vansonbourne. Big Data Infographic [Internet]. 2014 [cited 2016 May 4]. Available from: http://www.vansonbourne.com/research-insights/big-data/big-data-infographic/

79. Connolly B. Inadequate infrastructure halting big data projects [Internet]. [cited 2016 Apr 29]. Available from: http://www.cio.com.au/article/578636/inadequate-infrastructure-halting-big-data-projects/

80. Asay M. Big Data Failures Owe More To Business Culture Than Technology [Internet]. 2015 [cited 2016 May 9]. Available from: http://readwrite.com/2015/02/09/big-data-failure-blame-corporate-culture/

81. Senetas. The business end of data protection. 2013;SENE5635:2–7. Available from: http://www.senetas.com/_uploads/files/SENE5635_Business_End_of_Data_Protection_White-paper_2.pdf

82. Kaiser G. Understanding App Performance on the Network: Packet Loss [Internet]. 2014 [cited 2016 Apr 11]. Available from: http://apmblog.dynatrace.com/2014/07/03/understanding-application-performance-on-the-network-packet-loss/

83. Hadoop. HDFS Architecture Guide [Internet]. 2013 [cited 2016 Apr 19]. Available from: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

84. Karanth S. Mastering Hadoop [book] [Internet]. Packt Publishing; 2014. 12-14 p. Available from: https://www.packtpub.com/big-data-and-business-intelligence/mastering-hadoop

85. Latamore B. No easy answers to Hadoop's complexity, says Wikibon analyst | SiliconANGLE [Internet]. 2015 [cited 2016 Apr 6]. Available from: http://siliconangle.com/blog/2015/12/01/no-easy-answers-to-hadoops-complexity-says-wikibon-analyst/

86. Asay M. Hadoop complexity is part of the master plan, says Cloudera exec - TechRepublic [Internet]. 2015 [cited 2016 Apr 6]. Available from: http://www.techrepublic.com/article/hadoop-complexity-is-part-of-the-master-plan-says-cloudera-exec/

87. Infochimps. CIO & Big Data [Internet]. 2013 [cited 2016 Apr 3]. Available from: http://thumbnails-visually.netdna-ssl.com/CIOs_5102c7e4c65b6_w1500.png

88. Marr B, Wiley. Big Data in Practice (use Cases): How 45 Successful Companies Used Big Data Analytics to Deliver Extraordinary [Internet]. John Wiley & Sons; 2016 [cited 2016 Apr 18]. 320 p. Available from: https://books.google.com/books?id=wAOmCgAAQBAJ&pgis=1

89. Intel. Enabling Big Data Solutions with Centralized Data Management. 2013;(January):1–6.

90. Dinsmore TW. How to Buy SAS Visual Analytics [Internet]. 2015 [cited 2016 May 18]. Available from: https://thomaswdinsmore.com/2015/04/30/how-to-buy-sas-visual-analytics/

91. Savitz E. The Big Cost Of Big Data [Internet]. 2012 [cited 2016 May 18]. Available from: http://www.forbes.com/sites/ciocentral/2012/04/16/the-big-cost-of-big-data/#6112f4456a21

92. Vansonbourne. Big Data Infographic [Internet]. 2015 [cited 2016 May 16]. Available from: http://www.vansonbourne.com/files/6114/3453/7610/Big-Data-Infographic.pdf

93. InsideBIGDATA. Big Data: 5 Reminders to Keep It Real [Internet]. 2015 [cited 2016 May 16]. Available from: http://insidebigdata.com/2015/06/26/big-data-5-reminders-to-keep-it-real/

94. Marr B. How to Do Big Data on a Budget? [Internet]. 2016 [cited 2016 May 20]. Available from: http://www.huffingtonpost.com/bernard-marr/how-to-do-big-data-on-a-b_b_9200944.html

95. DB-Engines. DB-Engines Ranking - Trend Popularity [Internet]. 2016 [cited 2016 Mar 17]. Available from: http://db-engines.com/en/ranking_trend

96. Sitto K, Presser M. Field Guide to Hadoop: An Introduction to Hadoop, Its Ecosystem, and Aligned Technologies [book] [Internet]. 2015. 1-2 p. Available from: https://books.google.com/books?id=qEXkBgAAQBAJ&pgis=1

97. Brockmeier J. Amazon Takes Another Pass at NoSQL with DynamoDB [Internet]. 2012 [cited 2016 May 17]. Available from: http://readwrite.com/2012/01/18/amazon-enters-the-nosql-market/

98. Chang F, Dean J, Ghemawat S, Hsieh WC, Wallach DA, Burrows M, et al. Bigtable: A distributed storage system for structured data. 7th Symp Oper Syst Des Implement (OSDI '06), Novemb 6-8, Seattle, WA, USA [Internet]. 2006;1. Available from: http://research.google.com/archive/bigtable-osdi06.pdf

99. DeCandia G, Hastorun D, Jampani M, Kakulapati G, Lakshman A, Pilchin A, et al. Dynamo: Amazon's Highly Available Key-value Store. Proc Symp Oper Syst Princ [Internet]. 2007;1. Available from: http://dl.acm.org/citation.cfm?id=1323293.1294281

100. Ouyang A, Drummond D. Cassandra: Daughter of Dynamo and BigTable [Internet]. 2015 [cited 2016 May 20]. Available from: http://www.planetcassandra.org/blog/cassandra-daughter-of-dynamo-and-bigtable/

101. Dean J, Ghemawat S. MapReduce: Simplied Data Processing on Large Clusters. Proc 6th Symp Oper Syst Des Implement [Internet]. 2004;1. Available from: http://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf

102. Twentyman J. Spark versus MapReduce: which way for enterprise IT? [Internet]. 2015 [cited 2016 May 29]. Available from: http://www.computerweekly.com/feature/Spark-versus-MapReduce-which-way-for-enterprise-IT

103. Databricks. What is Apache Spark [Internet]. 2016 [cited 2016 May 29]. Available from: https://databricks.com/spark/about

104. Williams T, Scheutz M. A Framework for Resolving Open-World Referential Expressions in Distributed Heterogeneous Knowledge Bases. Aaai [Internet]. 2016; Available from: http://hrilab.tufts.edu/publications/williamsetal2016aaai.pdf

105. Chung Y, Zamanian E. Using RDMA for Lock Management. 2015;1–6. Available from: http://arxiv.org/abs/1507.03274

106. Moore G. Moore's Law [Internet]. 2016 [cited 2016 Apr 8]. Available from: http://www.mooreslaw.org/

107. Murgia M. End of Moore's Law? What's next could be more exciting [Internet]. [cited 2016 May 19]. Available from: http://www.telegraph.co.uk/technology/2016/02/25/end-of-moores-law-whats-next-could-be-more-exciting/

108. Bias R. Up, Out, Centralized, and Decentralized [Internet]. 2009 [cited 2016 May 17]. Available from: http://cloudscaling.com/blog/cloud-computing/up-out-centralized-and-decentralized/

109. Thampi SM. Introduction to Distributed Systems. 2009;1–17. Available from: http://arxiv.org/ftp/arxiv/papers/0911/0911.4395.pdf

110. Firoj Ali M, Zaman Khan R. Distributed Computing: An Overview. Int J [Internet]. 2015;2635:2630–5. Available from: http://www.ijana.in/papers/V7I-9.pdf

111. Gwalherkar S, Gehlod L. Integrity Improvement Using Controlled Data Replication in Distributed Environment. 2016;(1):402–4.

112. Bessis N, Dobre C. Big Data and Internet of Things: A Roadmap for Smart Environments [book] [Internet]. 2014. 71, 169-186 p. Available from: http://link.springer.com/10.1007/978-3-319-05029-4

113. Rao GSA, Pandey P. Big Data Problems : Understanding Hadoop Framework. Int J Sci Eng Technol [Internet]. 2014;42(3):40–2. Available from: http://ijset.com/ijset/publication/v3s1/IJSET_2014_109.pdf

114. DeZyre. Hadoop 2.0 and YARN - Advantages over Hadoop 2.0 [Internet]. 2014 [cited 2016 May 24]. Available from: https://www.dezyre.com/article/hadoop-2-0-yarn-framework-the-gateway-to-easier-programming-for-hadoop-users/84

115. Saphana. Hadoop 1.0 vs Hadoop 2.0 [Internet]. 2016 [cited 2016 May 24]. Available from: http://saphanatutorial.com/hadoop-1-0-vs-hadoop-2-0/

116. Gartner. Hadoop is a Recursive Acronym [Internet]. 2014. Available from: http://blogs.gartner.com/merv-adrian/2014/10/13/hadoop-is-a-recursive-acronym/

117. Gigaom. Data Collective adds Todd Papaioannou as EIR [Internet]. 2013 [cited 2016 May 26]. Available from: https://gigaom.com/2013/08/16/data-collective-adds-todd-papaioannou-as-eir/

118. Fitchard K. Hadoop: "It's damn hard to use" [Internet]. 2013 [cited 2016 Apr 6]. Available from: https://gigaom.com/2013/03/21/hadoop-its-damn-hard-to-use/

119. Oracle. Fusion Middleware Concepts Guide [Internet]. 2015 [cited 2016 May 24]. p. 4. Available from: http://docs.oracle.com/cd/E21764_01/core.1111/e10103/intro.htm#ASCON109

120. Spring. Spring Roo Project [Internet]. 2016 [cited 2016 May 29]. Available from: http://projects.spring.io/spring-roo/

121. Borowiec R. Thymeleaf Page Layouts [Internet]. 2016 [cited 2016 Apr 22]. Available from: http://www.thymeleaf.org/doc/articles/layouts.html

122. Kargupta H, Han J, Yu PS, Motwani R, Kumar V. Next Generation of Data Mining [Internet]. CRC Press; 2008 [cited 2016 May 29]. 191 p. Available from:

https://books.google.co.uk/books?id=YTlVP6OnfrcC&pg=PA191&lpg=PA191&dq=Controller+-+The+controller+translates+interactions+with+the+view+into+actions+to+be+performed+by+the+model.&source=bl&ots=HQtISxXN57&sig=TeOdFIgNgfnEbT1JRg7ZWPNL8bY&hl=en&sa=X&ved=0ah

123. StackOverFlow. How to bind an object list with thymeleaf? [Internet]. 2016 [cited 2016 May 28]. Available from: http://stackoverflow.com/questions/36500731/how-to-bind-an-object-list-with-thymeleaf/36522177

124. Atkison S. Why I hate Spring [Internet]. 2014 [cited 2016 May 29]. Available from: http://samatkinson.com/why-i-hate-spring/

125. Spring. Part I. Spring Boot Documentation [Internet]. 2016. Available from: http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-documentation

126. Spring. Part III. Using Spring Boot. 2016; Available from: http://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-auto-configuration.html