Clayton School of Information Technology
Monash University

Honours Research Proposal — Semester 2, 2014

# A pipeline for the preprocessing and storage of heterogeneous big data [WORKING TITLE]

Jonathan Poltak Samosir [2271 3603]

Supervisors: Dr Maria Indrawan-Santiago
Dr Pari Delir Haghighi

# Contents

# 1 Introduction

Currently, as a society, we are generating very large amounts of data from a large range of different sources. These sources include scientific experiments, such as the Australian Synchrotron [5] and The Large Hadron Collider [7], companies, such as Amazon [2], and also data generated by end users of products, such as social networks. The rate of data that is being generated is constantly increasing, presenting major challenges when it comes to the storage and processing of that data [11]. This is what is often referred to as "Big Data". Out of all of this data we are faced with, often only specific parts of the data are of use for given purposes. Hence rather than attempting to store all the new data that is being generating, often what is done, in both academia and industry associated with big data, is the realtime processing and analysis of incoming data streams.

There are currently numerous realtime data processing frameworks that are in development and in production use, both in industry and academia. Examples of these realtime data processing frameworks include the widely used Storm project [1] developed at BackType and Twitter, Inc., and also the up-and-coming Spark Streaming project [9] developed at UC Berkeley's AMPLab [3], both of which are open- source projects. While there are a growing number of these projects being developed, often these projects are designed with a particular type of data in mind, or to facilitate a particular type of data processing. For example, the before mentioned Spark Streaming project, along with its mother project, Spark [4], was designed for highly parallelised data with the use-case in mind of processing data in-memory using highly iterative machine learning algorithms related to data analytics [19].

Given these, occasionally "narrow", use-cases for existing data stream processing frameworks, challenges are faced in supporting the variations in both data types and processing requirements for data processing applications. In this research project, we aim to study the different characteristics of the data processing requirements based on the different characteristics of the data types. The knowledge found of these characteristics will be compared with the properties of existing solutions for big data processing.

What we propose in this document is an entire heterogeneous data processing pipeline that will facilitate the following tasks, in sequence:

1. Take in streams of data from various sources.
2. Aggregate similar types of data.
3. Process the data appropriately, depending on its type and the application.
4. Store the results of the data processing on an appropriate storage medium.

From this research project, we aim to produce a set of recommendations on choosing the various components of the pipeline, along with recommendations on how the components should be interconnected. To complement this, we also aim to produce a design template on the deployment of the pipeline in a cloud environment. This will be expanded on in further detail in §5.

This document will be structured as follows:
Discussion of the existing research and work done into this area will be touched on in §2. Our research questions, along with an outline of what we will be doing will be outlined in §3. The methodology used to achieve the deliverables of this project will be discussed in §4. Finally, we will conclude with an overview, in §5, of what the expected outcomes of this project will be, along with the greater contributions this project will give back, in the way of technological, disciplinary, and societal contributions.

# 2 Research Context

## 2.1 Big data

Big data, as explained previously, is becoming commonplace in both industry and academia. Everyday companies are finding that they are generating too much data and that their traditional database management system (DMBS) solutions cannot scale to the epic proportions needed to handle this data in an efficient and robust manner [20]. Hence, companies and academics alike have started looking at alternative solutions designed with the goal of handling these massive datasets.

The most popular solution for this problem, up until recently, has been the MapReduce model of programming along with some type of scalable distributed storage system [10]. The MapReduce model was started at Google, Inc. with their own proprietary implementation along with their proprietary distributed file system, known as the Google File System (GFS). Without going into the low-level details of MapReduce and GFS, the use of this solution at Google allowed the company to easily handle all the data that was coming into their servers, and perform the necessary processing operations that was needed at the time [14] [12].

## 2.2 Apache Hadoop

From the success of MapReduce usage combined with GFS at Google, the open-source community responded swiftly with the development of the Apache Hadoop framework. Hadoop originally offered an open-source implementation of MapReduce and their own open-source distributed file system known as the Hadoop Distributed File System (HDFS) [22].

Hadoop soon became the subject of mass-adoption in both industry and academia, being deployed at a fast rate. Development of the Hadoop framework also grew at a fast rate, with new applications related to HDFS and MapReduce being built on top of Hadoop, greatly benefiting the ecosystem as a whole. Some of these applications grew into widely adopted systems in their own right. For example, Hadoop applications such as Apache Pig [13] and Hive [24] allow for easy querying and manipulation of data stored on HDFS, both coming with the addition of their own "SQL-like" query languages [21].

Additionally, as further non-MapReduce model applications became of interest to the Hadoop community, Hadoop soon developed a further abstraction on top of the underlying resources (in most cases, HDFS). The goal of this was to facilitate the development and deployment of many different applications, varying in use-case, which could be run on the Hadoop ecosystem, without forcing developers to fit their application into the MapReduce model. This development was known as Apache Hadoop YARN: Yet Another Resource Negotiator, which can be thought of as an operating system sitting atop of the available Hadoop resources [26]. The abstraction YARN provides facilitated the development of much more advanced, and non-MapReduce technologies which have since become widely used parts of the Hadoop ecosystem [15].

## 2.3 Realtime data processing

One of the major limitation of Hadoop, and the MapReduce model in general, soon became obvious: MapReduce was designed with the goal of being able to process batches of data, hence, given Hadoop's dominance, batched data processing was the focal point of the entire distributed data processing domain [17]. Essentially, batched data processing is where data gets collected first into large enough batches before being processed all-at-once. The point of processing in such a way is so there would be less overheads than attempting to process each individual datum as it arrives. For a lot of use-cases this was, and still is, fine as there were no other drawbacks apart from a high level of latency between the stages of when the data arrives and when it gets processed. However, for other applications, such as stock trading, sensor monitoring, and web traffic processing, a more low-latency, realtime solution was needed [17].

Soon, many solutions, with different use-cases and design goals, were developed in the area of distributed stream processing systems (DSPS). Given the Hadoop ecosystem that was already widely adopted, most of these DSPSs were built upon the still new YARN layer, ensuring overall compatibility with the Hadoop ecosystem, and the underlying HDFS. Some examples of such projects include the beforementioned Apache Storm, currently being used at Twitter, Inc. [25], among many other companies. Also up-and-coming projects, such as Apache Samza which is a recently open-sourced project, currently being used in production at LinkedIn Corporation [8].

### 2.4 The future

While the overall area of big data processing has definitely been moving at a very fast pace in the last decade, the area focused on realtime big data processing is still relatively young and shows much potential for further growth in the way of research. As of writing, there are a considerable amount DSPS technologies available and in active development, and are fast gaining adoption in industry.

While many of the DSPS technologies available offer much needed realtime data processing functionality, it is rather confusing for the end-user to differentiate between which of the technologies would be best given their specific use-case and the class of data they are working with. There has recently been an attempt at introducing a DSPS-agnostic architecture, known as the Lambda architecture [20], that is currently gaining traction in the academic community [16] [19].

We propose looking into the Lambda architecture, and similar attempts, as inspirations for our system that will allow for processing recommendations for given data classes and dynamic construction of a heterogeneous data processing pipeline.

## 3 Objectives

### 3.1 Research questions

The following research questions will be the main focus of our project:

1. What classification methods can we best use to classify arbitrary data?
2. How can we formulate processing recommendations based upon the specific class of data identified for a particular dataset?
3. How can existing data stream processing solutions be used or be altered for use within this pipeline?

Of course, from these preliminary questions, we will probably have to decompose them into a number of workable units.

Additionally, after answering each of these preliminary research questions, we will want to properly implement the theoretical discoveries from each stage. We do this with the goal of achieving some deployable pipeline that can be then be used in the testing and overall evaluation stages.

### 3.2 Research aims

The main aim or goal of this research project is to develop a set of recommendations that can lead to the creation of this data stream processing pipeline. The purpose of these recommendations is to define exactly how specific classes of data coming into our pipeline should be processed in realtime. While we do not aim to have a fully usable, production-ready piece of software as a deliverable, we want to at least have a proof-of-concept working in the National eResearch Collaboration Tools and Resources (NeCTAR) cloud [6].

From these recommendations, we are planning to be able to formulate some NeCTAR compatible scripts that will enable NeCTAR end-users to be able to deploy a specific variation of the pipeline on their NeCTAR instances. The scripts, and the pipelines they produce, will vary in which technologies make up the pipeline for their given use-case. The use-case being the specific class of data that they are attempting to process.

Further detail on the specifics behind these recommendations, NeCTAR scripts, and data classification can be found in §5.

# 4 Research Design

## 4.1 Methodology

One of the key parts of our research methodology will be the applied use of the National eResearch Collaboration Tools and Resources cloud (NeCTAR) [6]. Access to this cloud will facilitate the majority of applied work that happens in this project, including the testing of existing big data stream processing technologies along with the evaluation of our project's technical outcomes. It will also serve as the target platform for the implementation and deployment of our pipeline.

To give a brief overview of NeCTAR, NeCTAR is a $47 million (AUD) project funded by the Australian government to facilitate Australian eResearch through providing shared Cloud infrastructures, among other facilities [23].

As of writing, we have requested two NeCTAR Cloud instances for this project's use. The instances having 16 shared cores, 6400 hours of processing time, and one terabyte of shared volume storage. Going by initial estimates, these requested resources should suffice for the scope of this research project. These resources are further touched on in §4.5.

## 4.2 Proposed thesis chapter headings

The proposed structure of the thesis is as follows:

1. Introduction
   - 1.1. Overview
   - 1.2. Background
   - 1.3. Research problem
   - 1.4. Research questions
   - 1.5. Research scope
   - 1.6. Conclusion and thesis structure
2. Literature Review
   - 2.1. Introduction
   - 2.2. Definition of terms
   - 2.3. Big data in industry and academia
   - 2.4. Batch data processing
   - 2.5. Overview of batch data processing technologies

### 4.3 Timeline

The following is a preliminary estimated timeline of the proposed research project:

| **Task / Deliverable** | **Deadline** |
|---|---|
| First meeting with supervisors | 2014-07-29 |
| Scoping finalised | 2014-08-11 |
| Research proposal draft submission | 2014-09-01 |
| Research proposal final submission | 2014-09-05 |
| Preliminary research complete | 2014-10-16 |
| Literature review draft submission | 2014-10-31 |
| Interim presentation | 2014-11-03 |
| Literature review final submission | 2014-11-07 |
| Qualitative comparisons complete | 2014-12-07 |
| Data classification method complete | 2015-01-01 |
| Data processing recommendations due | 2015-01-28 |
| NeCTAR script implementation due | 2015-02-28 |
| Testing of pipeline on NeCTAR | 2015-03-15 |
| Evaluation of pipeline system | 2015-04-01 |
| Thesis draft submission | 2015-05-29 |
| Final presentation | 2015-06-08 |
| Thesis final submission | 2015-06-19 |

Note that the above is simply a proposed timeline, and it is highly probable that this will be subject to change as the project progresses.

### 4.4 Potential difficulties

While we believe that most components of this project are very much feasible given the time and resources we have been allocated so far, we have identified a small number of possible difficulties that may be encountered as the project progresses. The most obvious difficulty so far that we have identified is the need to acquire a substantial amount of data that we can use for both during the testing and the evaluation stages of the project. As this data will be used to test our data classification methods and evaluate our pipeline deployed in the cloud, it will need to be diverse. By diverse, what we mean is it must display heterogeneity in terms of its type and origin; data from many different sources would be ideal.

Currently we have no concrete leads on the acquisition of this data, although we will look into collaboration with other data-based research projects ongoing at Monash University. We also are yet to explore freely available data sets, such as the Enron corpus email dataset [18], although these may definitely be taken into consideration at a later stage in the project in the case that data acquisition proves infeasible.

### 4.5 Special facilities required

As we are aiming to deploy a proof-of-concept of this pipeline, the main special facility needed access to is a cloud solution that enables us to install and test our pipeline. For this, we have already requested access for what we are planning to do in this project on the National eResearch Collaboration Tools and Resources (NeCTAR) cloud [6]. This cloud is funded by the Australian Government and available to Australian researchers in many different disciplines.

With access to this cloud for the duration of this project, we will be able to install and perform qualitative comparisons between the numerous realtime data processing solutions available as of now. This will assist us in making our recommendations for the pipeline based on the classification of particular set of data.

This access to cloud resources at NeCTAR will also facilitate our later testing and evaluation stages of the project, where we will be hoping to test the pipeline with real heterogeneous data.

# 5 Expected Outcomes

The expected outcomes of this project include both technical contributions and theoretical contributions; the technical outcomes of the project essentially being implementations of the theoretical outcomes.

Our main theoretical contribution will be the realtime processing recommendations we produce for specific classes of data. These recommendations will recommend specific realtime data processing technologies for use within the pipeline to process the given data. Of course, to actually classify the data, we need to come up with some classification method. This will be another of our theoretical contribution outcomes of this project.

Looking at the project's outcomes in terms of technical contributions, they directly relate back to the theoretical contributions. The main technical contribution will be NeCTAR template scripts which enable the deployment of the pipeline on the NeCTAR cloud. These scripts will be constructed based upon the recommendations produced for specific dataset classes.

Relating back to the data classification method, we will also contribute an implementation of this so that given datasets can be fed into the system and be classified on-the-fly. What we aim to achieve from the implementation of this classification method is to have it as a one of the very initial stages of the pipeline, where heterogeneous data can be fed in, have its type classified, then have a NeCTAR script constructed based upon that data classes' recommended processing pipeline.

To summarise, the expected outcomes of this project include the following contributions:

- A classification method for classifying arbitrary datasets.
- An implementation of this classification method for use in the deployed pipelines.
- Recommendations for specific classes of data on how they should be processed in realtime.
- NeCTAR compatible scripts allowing NeCTAR users to deploy the recommended pipelines on the NeCTAR cloud.

All of these contributions will hopefully be assembled together to make a complete pipeline generating system for any arbitrary type of data.

## References


[1] Storm, distributed and fault-tolerant realtime computation. `http://storm-project.net`, November 2013.

[2] Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more. `http://www.amazon.com`, August 2014.

[3] Amplab - UC Berkeley — Algorithms, Machines and People Lab. `https://amplab.cs.berkeley.edu`, 2014.

[4] Apache Spark™– Lightning-Fast Cluster Computing. `https://spark.apache.org`, 2014.

[5] Australian Synchrotron. `https://www.synchrotron.org.au`, August 2014.

[6] home — NeCTAR. `http://www.nectar.org.au`, August 2014.

[7] The Large Hadron Collider — CERN. `http://home.web.cern.ch/topics/large-hadron-collider`, August 2014.

[8] Samza. `http://samza.incubator.apache.org`, 2014.

[9] Spark Streaming — Apache Spark. `https://spark.apache.org/streaming/`, 2014.

[10] Bifet, A. Mining big data in real time. *Informatica 37*, 1 (Mar. 2013), 15+.

[11] Bohlouli, M., Schulz, F., Angelis, L., Pahor, D., Brandic, I., Atlan, D., and Tate, R. Towards an integrated platform for big data analysis. In *Integration of Practice-Oriented Knowledge Technology: Trends and Prospectives.* Springer, 2013, pp. 47–56.

[12] Dean, J., and Ghemawat, S. MapReduce: Simplified data processing on large clusters. *Commun. ACM 51*, 1 (Jan. 2008), 107–113.

[13] Gates, A. F., Natkovich, O., Chopra, S., Kamath, P., Narayanamurthy, S. M., Olston, C., Reed, B., Srinivasan, S., and Srivastava, U. Building a high-level dataflow system on top of map-reduce: The pig experience. *Proc. VLDB Endow. 2*, 2 (Aug. 2009), 1414–1425.

[14] Ghemawat, S., Gobioff, H., and Leung, S.-T. The google file system. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles* (New York, NY, USA, 2003), SOSP '03, ACM, pp. 29–43.

[15] Harrison, G. Hadoop's next-generation YARN. *Database Trends and Applications 26*, 4 (Dec. 2012), 39.

[16] Islam, M. Z. *A Cloud Based Platform for Big Data Science.* 2014. With the advent of cloud computing, resizable scalable infrastructures for data processing is now available to everyone. Software platforms and frameworks that support data intensive distributed ap ...

[17] Kamburugamuve, S., Fox, G., Leake, D., and Qiu, J. Survey of distributed stream processing for large stream sources.

[18] Klimt, B., and Yang, Y. Introducing the enron corpus. In *CEAS* (2004).

[19] Liu, X., Iftikhar, N., and Xie, X. Survey of real-time processing systems for big data. In *Proceedings of the 18th International Database Engineering & Applications Symposium* (New York, NY, USA, 2014), IDEAS '14, ACM, pp. 356–361.

[20] Marz, N. *Big data : principles and best practices of scalable realtime data systems.* O'Reilly Media, [S.l.], 2013.

[21] Olston, C., Reed, B., Srivastava, U., Kumar, R., and Tomkins, A. Pig latin: A not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2008), SIGMOD '08, ACM, pp. 1099–1110.

[22] Shvachko, K., Kuang, H., Radia, S., and Chansler, R. The hadoop distributed file system. In *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)* (May 2010), pp. 1–10.

[23] Sinnott, R. O., Galang, G., Tomko, M., and Stimson, R. Towards an e-infrastructure for urban research across australia. In *E-Science (e-Science), 2011 IEEE 7th International Conference on* (2011), IEEE, pp. 295–302.

[24] Thusoo, A., Sarma, J., Jain, N., Shao, Z., Chakka, P., Zhang, N., Antony, S., Liu, H., and Murthy, R. Hive - a petabyte scale data warehouse using hadoop. In *2010 IEEE 26th International Conference on Data Engineering (ICDE)* (Mar. 2010), pp. 996–1005.

[25] Toshniwal, A., Taneja, S., Shukla, A., Ramasamy, K., Patel, J. M., Kulkarni, S., Jackson, J., Gade, K., Fu, M., Donham, J., Bhagat, N., Mittal, S., and Ryaboy, D. Storm@Twitter. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data* (New York, NY, USA, 2014), SIGMOD '14, ACM, p. 147–156.

[26] Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., Saha, B., Curino, C., O'Malley, O., Radia, S., Reed, B., and Baldeschwieler, E. Apache hadoop YARN: Yet another resource negotiator. In *Proceedings of the 4th Annual Symposium on Cloud Computing* (New York, NY, USA, 2013), SOCC '13, ACM, pp. 5:1–5:16.