Clayton School of Information Technology
Monash University

Honours Literature Review — Semester 2, 2014

A study of the Hadoop ecosystem for pipelined realtime data stream processing

Jonathan Poltak Samosir

[2271 3603]

Supervisors: Dr Maria Indrawan-Santiago
Dr Pari Delir Haghighi

Contents

1 Introduction **1**

2 Data types and characteristics background **1**
2.1 Velocity, variety, volume, and veracity 1
2.2 Classification of data 2
2.2.1 Characteristics of data, from Mysore et al. 3
2.2.2 Classes of data, from Mysore et al. 4
2.2.3 Characteristics of data, from Chen et al. 5
2.2.4 Characteristics of data, from Géczy 8
2.2.5 Criticisms of presented classification models 9

3 Big data processing background **9**
3.1 Batch data processing 10
3.1.1 MapReduce and GFS 10
3.1.2 Hadoop MapReduce and HDFS 10
3.1.3 Pig and Hive 11
3.2 Realtime data processing 12
3.2.1 Hadoop YARN 12
3.2.2 Storm 13

4 Relationships between big data classes and big data processing **13**

5 Conclusion **13**

1 Introduction

The realtime processing of big data is of great importance to both academia and industry. Advancements and progress in modern society can be directly attributed back to data. The value of data has become more apparent, and data has become a sort of currency for the information economy [27]. Hence, those in society who realised the value of data early immense power over the entire economy and thus society overall [20]. From seemingly inconsequential gains at the macro level, such as the ability to more accurately predict the rise and fall of airline tickets [7], to those of utmost importance for society as a whole, such as predicting and tracking the spread of the Swine Flu Pandemic in 2009 more accurately that the United States Centers for Disease Control and Prevention could [25] [22]. It is example applications of big data processing like these that have been recognised by academics and organisations in industry alike, with the last decade seeing a major shift in research and development into new methods for the handling and processing of big data.

This paper will give a background on the types and classes of big data, as well as the various methods employed to process those given classes of data. We will more specifically focussing on the methods that are involved with the analysis and processing of realtime data streams, as opposed to the batch processing of big data. This paper will look into detail at previous work that has been done in the field of big data, specifically those works that have had a greater influence on the field as a whole. This includes both works looking specifically at the processing of streaming data, and works involving processed big data in batch mode, given that batch mode processing arguably led onto the current hot-topic of realtime stream processing.

This paper will be structured in two main sections. In §2, an overview of the different classes and types of big data will be presented. This includes an overview of the big data classes presented through others' findings as well as our own proposed classes for big data, based on the criticisms of those prior findings. In §3, an overview will be given of the major open-source big data processing systems. A special emphasis will be given on data stream processing systems (DSPSs), given that the main area of this research is focusing on realtime data processing, or data stream processing.

§4 will then give a discussion relating to future work we have planned to form data processing recommendations based on the classification of specific data classes. All of the sections will then be summarised in the conclusion in §5.

As a an outcome of this paper, we will identify a gap in previous research and development in the big data processing field, upon which our future work will attempt to work towards filling.

2 Data types and characteristics background

2.1 Velocity, variety, volume, and veracity

Data, and more specifically, big data, are often characterised into what is known as the "four V's" [34]. These can be thought of as different "dimensions" of big data, and can be summarised as follows [9]:

- *Velocity:* The rate at which data is being collected and made available to the data consumers.
- *Variety:* The heterogeneity of data. Big data often exhibits substantial variations in both the structural level and the instance level (representations of real-world entities). This is often highlighted by data systems that depend on acquiring of data from a number of non-conforming, and sometimes unrelated, data sources.

- *Volume:* The amount of data that is obtained by the data consumer from the data source/s.
- *Veracity:* The quality, in terms of accuracy, coverage, and timeliness, of data that is consumed from the data source/s. Veracity of data can widely differ between sources.

While the four V's are often described in terms of big data, they can also apply to more traditional data warehousing and processing in general, albeit on a far smaller scale. In the domain of big data processing, data will exhibit signs of high velocity, variety, and volume [3], and hence the veracity of the data may also fluctuate. Meanwhile, in more traditional data processing, the scope may be limited, especially in terms of factors such as variety and, as a consequence, there is less need of an emphasis on veracity due to limited variety in data sources.

As will be made clear in the following sections, a lot of the identified classes and characteristics of data directly relate back to these four V's, whether or not it was intentional by the original authors. These can be considered the underlying features of many characteristics of data, both in the sense of big data and traditional data.

2.2 Classification of data

Data, in general, can be categorised into a number of different classes or types. In this paper, we will define the concept of a data class to mean the same as the terms of "data type", "data category", or "data format", as all terms were often used interchangeably in other literature.

Each class of data can be further defined and categorised via the characteristics they exhibit. Furthermore, these characteristics exhibited by data classes can be exploited and it is often possible to optimise the processing of each class of data by processing it using a specific method depending on those characteristics.To give an example of this, data that is expected to have highly iterative processing applied to it would benefit from a data processor that does not have to unnecessarily write to disk after every single iteration. The elimination of this I/O overhead is an example of the optimisations that could be applied to the overall process from correctly identifying the data class beforehand, and processing it accordingly.

Furthermore, particular classes of data are generally only found in particular applications or use cases of data processing.As this is the case, it narrows down the amount of classification needed, depending on the application that is being looked at. This will be elaborated on in later parts of this section.

There is no concrete, universally accepted standard for the classification of data. While the study of big data processing could arguably be considered still in its infancy (or at least temperamental toddler stage), data handling and processing in general is relatively mature. From preliminary research on looking at past work and literature in this area, it must be noted that there is a significant lack of research on the classification of data.

The literature that will be reviewed in this section is often not wholly focused on the idea of data classification, hence data classification is presented relative to whatever the overall topic of the literature is on. This is important to note, as one attempt at data classification may not be appropriate under a different context. This also explains the large variation in different classification attempts, although we will also highlight the recurring similarities between different data classification literature.

2.2.1 Characteristics of data, from Mysore et al.

The main piece of literature that this section sources is a white paper from IBM Architects Mysore, Khupat, and Jain, published by IBM in 2013 [1]. The white paper is targeted towards beginners in the area of big data processing; much like the set of recommendations that we intend to produce from this research project. The paper looks at identifying the different data classes, or "formats" as they were labelled in the paper, that are commonly encountered in big data. For each of these formats, what was identified was the underlying characteristics of the data, and it was noted that the type of processing needed would be dependent on those characteristics.

The characteristics of data, as put forward by Mysore et al., in [1], include the following:

2.2.1.1 Analysis type:

- Whether or not the data would be processed/analysed in realtime, or batched for later processing.
- Often this data class characteristic is dependent on the application of the data (*e.g.* The processing of social media data for the analysis of currently occurring events would want to be processed in realtime, regardless of the type of data that is involved).

2.2.1.2 Processing methodology:

- This characteristic involves the approach used when processing the data.
- Some examples of different processing methodologies include: predictive processing, analytical, ad-hoc queries, and reporting.
- Often the processing methodology for a particular class is determined by the business requirements or application of the data.
- Depending on the processing methodology used, many different combinations of big data technologies can be used.

2.2.1.3 Data frequency and size:

- The amount of data expected to arrive to the processing system, along with the speed and regularity of the incoming data.
- Knowing this characteristic beforehand can determine the methods for data storage and preprocessing, if needed.
- Examples of data frequency includes: on-demand data (social media), continuous/realtime (weather data, transactions), time-series (email).
- Considering the four V's, the characteristic of data frequency and size directly relates back to velocity and volume.

2.2.1.4 Content format:

- This characteristic relates back to the structure of the underlying data.
- Examples of data content format include: structured (JSON, XML), unstructured (human-readable literature), semi-structured (email).

2.2.1.5 Data source:

- This characteristic relates back to where the data originated from.
- As discussed previously in §2.1, the origin of data can have a great effect on whether or not that data is usable, as data often varies greatly, especially when many different sources are used which may or may not conform to a specific content format.
- Another thing that is dependent on the data source is whether or not the data can be trusted.
- Considering the four V's, the characteristic of data source directly relates back to veracity and variety.

2.2.2 Classes of data, from Mysore et al.

The following table highlights the different classes of data put forward by Mysore, et al., in [1]. The table organises each class, along with giving a brief explanation of the class. Furthermore, each class is related back to the previously explained characteristics in an attempt to show the connections between class and underlying characteristics.

Data class	Explanation	Characteristics
Machine generated data	• Data that is automatically generated as a by-product of some interaction with a machine. • While Mysore et al. present this as being a distinct class in itself, it could be argued that this class is an umbrella class which many other data classes presented in their paper fall under. This will be touched upon further in later sections.	• Structured data (JSON, XML). • Frequency of data varies depending on application.
Web and social data	• Data that is automatically generated through use of the Internet or social media, such as Facebook or Twitter.	• Unstructured text (long: blogs, short: microblogs, Facebook). • Miscellaneous multimedia (video, image, audio). • On-demand frequency. • Can be continuous feed of data in cases such as Twitter.
Transaction data	• Data that is automatically generated as a by-product of transactions, such as money transactions or otherwise.	• Structured text (JSON, XML, logs). • Continuous feed.
Human generated data	• Data that is solely produced by humans. • Examples of human generated data, as it is defined here, include such things as music, literature, recordings, and emails.	• Unstructured text (mail, literature). • Miscellaneous multimedia (audio, video, images). • Semi-structured text (email, online messaging services). • On-demand frequency.
Biometrics data	• Data that relates to human bioinformatics.	• Structured data. • On-demand frequency. • Continuous feeds of data in cases such as persistent health monitoring sensors (*i.e.* hospital patients).

The classes and characteristics of data presented by Mysore et al., in [1], are highly oriented towards industry and business users, coming from an IBM-published paper. While this is not an issue as such, as noted earlier in this section, these characteristics and data classes are defined within the domain relevant to this paper. As such, they may not be as relevant or appropriate for usage in other, non-business domains or even business domains with a different focus on data.

2.2.3 Characteristics of data, from Chen et al.

The second paper sourced is a paper from Chen, Chiang, and Storey, focusing on the impact of big data in the field of business intelligence and analytics [6]. Similarly to the paper looked at in §2.2.1, there is an emphasis on data classes and how they relate to the area of business and organisations. However, this paper has more of an explicit focus on business, being in published in the area of business intelligence and analytics (BI&A). BI&A in itself is a highly data driven field, where data is gathered and analysed to help make informed business decisions [35].

In the paper, Chen et al., discuss the evolution of the field of BI&A, which they categorise into three distinct stages. BI&A 1.0, being the first of the three, focuses on more traditional data processing and analysis. This includes highly structured and relational data. BI&A 2.0 involves more unstructured, web-based content with the rise of "Web 2.0" technologies, including social networks and opinion pieces, such as blogs. BI&A 3.0 looks at more mobile and sensor-based data. This data differentiates itself mostly to do with characteristics such as location-based data and data that is highly context dependent.

Chen et al., elaborate on these different stages of BI&A evolution through showing the major BI&A applications for the previously mentioned evolutionary stages. For each of the BI&A applications presented, they attempt to show the classes of data which are important for the particular application, and subsequently the characteristics associated which each class. The classes and characteristics of data, shown by Chen et al., in relation to BI&A will be presented here. They will be presented in terms of the BI&A application of which they are categorised under.

2.2.3.1 E-Commerce and Market Intelligence:

Types of data include:

- Website logs and analytics data.
- User activity logs for e-commerce websites.
- User transaction records.
- User-generated content, such as reviews, feedback.

Characteristics of the data include:

- Structured web-based data (transactions records, logs, network information).
- Unstructured user-generated content (reviews, feedback).

2.2.3.2 E-Government and Politics 2.0:

Types of data include:

- Government information, such as statistics.
- Rules and regulations.
- Citizen-generated content, such as feedback, comments, and requests.

Characteristics of the data include:

- Fragmented data sources (think high data variety).
- Unstructured data (citizen-generated content).
- Rich textual content.

2.2.3.3 Science & Technology:

Types of data include:

- Machine-generated data from tools and instruments.
- Sensor data.
- Network data.

Types of characteristics include:

- High velocity data collection from instruments and tools and sensors.
- Structured data, often formatted in uncommon structures.

2.2.3.4 Smart Health and Wellbeing:

Types of data include:

- Genomics and sequence data (DNA sequences).
- Electronic health records.
- Health and patient social media.

Types of characteristics include:

- Varying, but interrelated, data.
- Data specific to individual patients.

2.2.3.5 Security and Public Safety:

Types of data include:

- Criminal record data.
- Statistical data (crime maps).
- Media content relating to crime (news articles).
- Cyber-crime data (computer viruses, botnet data).

Types of characteristics include:

- Highly sensitive information (identity data).
- Incomplete and deceptive content (speculative media content).
- Multilingual content.

As can be seen from the data classes and characteristics identified by Chen et al., in [6], there is a far greater variation to those previously presented by Mysore et al., in [4]. As explained earlier, this is mainly because of the more domain specific content of this piece of literature, while the paper from Mysore et al., while still having underlying tones of business and industry, had a less explicit focus on their particular domain.

2.2.4 Characteristics of data, from Géczy

Coming away from the business point-of-view, Géczy attempts to characterise data, and more specifically big data, in a more generic way [14]. He uses what he labels as "aspects" to determine what he believes to be the deciding characteristics of data, in terms of the way they should be processed and also simply their intrinsic traits.

Géczy uses the following aspects to determine the different intrinsic characteristics of data:

2.2.4.1 Sensitivity:

- Relates to whether or not given data contains sensitive information, *i.e.* personally identifiable information, confidential information, etc.
- The sentivity of the data determines the requirements relating to how it should be handled.
- Often it is either a legal requirement, or in the owners' interest, to keep protected the handled data deemed sensitive.

2.2.4.2 Diversity:

- Relates to the range of different data elements present within the data.
- The example given explains the ability of smart phones to produce highly diverse data; *e.g.* audio, video, location data, gyroscopic data, etc.
- Having high diversity in data can both be beneficial and detrimental; diversity can add factors of complexity, although also makes for a more rich dataset.
- Note that this data characteristic relates directly back to the *Variety* dimension, of the four V's.

2.2.4.3 Quality:

- Quality characteristics of data are defined to be features that affect data quality; *e.g.* completeness, accuracy, timeliness.
- Often the quality of data may be subject to the qualitative metrics of an organisation, or predefined standards.
- The quality of data relates back to the *Veracity* dimension, of the four V's.

2.2.4.4 Volume:

- Volume refers to the size of data in terms of its basic forms of measurement, bits and bytes.
- Volume is an important characteristic to take into consideration when it comes to determining the type of processing needed.
- Volume, as the name suggests, directly relates back to the Volume dimension of the four V's.

2.2.4.5 Speed:

- Data speed refers to the inflow and outflow speeds; inflow being the data that is being acquired, while outflow being the data leaving the system (often results of computations).
- Different classes of data often require different data speeds. *e.g.* audio is often streamed at a far lesser speed than video, due to the relatively low amount of data in audio when compared with video.

2.2.4.6 Structure:

- Structure relates to whether data are in structured or unstructured formats.
- Generally unstructured data is more suitable for human consumption, such as literature or music.
- Structured data is usually structured in such a way that it is easily able to be parsed by an algorithm, often automated by computers.
- The structure of data directly relates to the difficulty of processing that data, as unstructured data usually will need some pre-processing or artificial intelligence to process.

Géczy later goes on to talk about the aspects of data that relate to data processing, similar to what will be talked about later in §3.

Overall, Géczy looks at data characteristics, not from any particular perspective, but from one that attempts to capture the interests and be relevant to a number of disciplines. This impartiality is a nice refreshment from most other literature available on the topic, which have been shown to have been looking at data classification from a certain point-of-view. However, this should not be misinterpreted as a criticism of the previous literature. It is simply that the classes of data identified in other author's literature was more appropriate for the topic on which the rest of their research was focussed on. Hence, the way they treated data changed accordingly. The paper presented by Géczy, simply titled *BIG DATA CHARACTERISTICS*, was focussed on nothing other than characteristics of data, hence there was no reason to attempt to classify those characteristics based on any other domain-related biases. Additionally, Géczy's paper was published in a notable interdisciplinary journal, rather than one aimed at a particular discipline. This difference in terms of impartiality is the important difference to note with this paper.

2.2.5 Criticisms of presented classification models

From the literature presented previously in this section, there are a number of points to note. Firstly, they were all highly varied in the classifications and characteristics of data given. As previously stated, this can be attributed to the variety in sources for this literature; they were all published from quite different sources, and each of the authors from different fields with different intentions. Hence, it is not

3 Big data processing background

Much more work has been done in the area of data processing than the area related to classification of data; both in the areas of big data and traditional data processing. Unlike data classification, which was more aimed at the classifying of data in general, when looking at data processing, we are more interested in the relatively newer technologies which enable the processing of big data, both in batch mode and realtime. Note that in this paper, we will refer to realtime big data processing as just that; realtime data processing in the context of big data. Through use of this term, we encompass the meanings of "data stream processing", "realtime stream processing", and all other related terms that essentially have the meaning of:

3.1 Batch data processing

Over the last decade, the main "go-to" solution for any sort of processing needed on datasets falling under the umbrella of big data has been the MapReduce programming model on top of some sort of scalable distributed storage system [4]. From a very simplified functionality standpoint, the MapReduce programming model essentially combines the common **Map** and **Reduce** functions (among others), found in the standard libraries of many functional programming languages, such as Haskell [19] or even Java 8 [29], to apply a specified type of processing in a highly parallelised and distributed fashion [37].

The MapReduce data processing model specialises in batch mode processing. Batch data processing can be thought of where data needed to be processed is first queued up in batches before processing begins. Once ready, those batches get fed into the processing system and handled accordingly.

3.1.1 MapReduce and GFS

Dean and Ghemawat, in [8], originally presented MapReduce as a technology that had been developed internally at Google, Inc. to be an abstraction to simplify the various computations that engineers were trying to perform on their large datasets. The implementations of these computations, while not complicated functions themselves, were obscured by the fact of having to manually parallelise the computations, distribute the data, and handle faults all in an effective manner. The MapReduce model then enabled these computations to be expressed in a simple, high-level manner without the programmer needing to worry about optimising for available resources. Furthermore, the MapReduce abstraction provided high scalability to differently sized clusters.

As previously stated, the MapReduce programming model is generally used on top of some sort of distributed storage system. In the previous case at Google, Inc., in the original MapReduce implementation, it was implemented on top of their own proprietary distributed file system, known as Google File System (GFS). Ghemawat et al., in [11], define GFS to be a "scalable distributed file system for large distributed data-intensive applications", noting that can be run on "inexpensive commodity hardware". Note that GFS was designed and in-use at Google, Inc. years before they managed to develop their MapReduce abstraction, and the original paper on MapReduce from Dean and Ghemawat state that GFS was used to manage data and store data from MapReduce [8]. Furthermore, McKusick and Quinlan, in [23], state that, as of 2009, the majority of Google's data relating to their many web-oriented applications are rely on GFS.

3.1.2 Hadoop MapReduce and HDFS

While MapReduce paired with GFS proved to be very successful solution for big data processing at Google, Inc., and there was notable research published on the technology, it was proprietary in-house software unique to Google, and availability elsewhere was often not an option [12]. Hence, the open-source software community responded in turn with their own implementation of MapReduce and a distributed file system analogous to GFS, known as the Hadoop Distributed File System (HDFS). Both of these projects, along with others to date, make up the Apache Hadoop big data framework [1]. The Apache Hadoop framework, being a top level Apache Software Foundation open source project, has been developed by a number of joint contributors from organisations and institutions such as Yahoo!, Inc., Intel, IBM, UC Berkeley, among others [15].

While Hadoop's MapReduce implementation very much was designed to be a functional replacements for Google's MapReduce, HDFS is an entirely separate project in its own right. In the original paper from Yahoo! [26], Inc., Shvachko et al. present HDFS as "the file system component of Hadoop" with the intention of being similar to the UNIX file system, however they also state that "faithfulness to standards was sacrificed in favour of improved performance".

While HDFS was designed with replicating GFS' functionality in mind, several low-level architectural and design decisions were made that substantially differ to those documented in GFS. For example, in [5], Borthakur documents the method HDFS uses when it comes to file deletion. Borthakur talks about how when a file is deleted in HDFS, it essentially gets moved to a `/trash` directory, much like what happens in a lot of modern operating systems. This `/trash` directory is then purged after a configurable amount of time, the default of which being six hours. To contrast with this, GFS is documented to have more primitive way of managing deleted files. Ghemawat, et al., in [11], document GFS' garbage collection implementation. Instead of having a centralised `/trash` storage, deleted files get renamed to a hidden name. The GFS master then, during a regularly scheduled scan, will delete any of these hidden files that have remained deleted for a configurable amount of time, the default being three days. This is by far not the only difference between the two file systems, this is simply an example of a less low-level technical difference.

3.1.3 Pig and Hive

Given the popularity of Hadoop, there were several early attempts at building further abstractions on top of the MapReduce model, which were met with a high level of success. As highlighted earlier, MapReduce was originally designed to be a nice abstraction on top of the underlying hardware, however according to Thusoo et al., in [30], MapReduce was still too low level resulting in programmers writing programs that are "are hard to maintain and reuse". Thus, Thusoo et al. built the Hive abstraction on top of MapReduce. Hive allows programmers to write queries in a similarly declarative language to SQL — known affectionately as *HiveQL* — which then get compiled down into MapReduce jobs to run on Hadoop [31].

Another common abstraction that was developed prior to Hive was what is known simply as Pig. Like Hive, Pig attempts to be a further higher level abstraction on top of MapReduce, which ultimately compiles down into MapReduce jobs, although what differentiates it from Hive is that instead of being a solely declarative SQL-like language, it is more of a mix of procedural programming languages while allowing for SQL-like constraints to be specified on the data set to define the result [24]. Olston et al. describe Pig's language — known as *Pig Latin* — to be what they define as a "dataflow language", rather than a strictly procedural or declarative language.

Furthermore, note that Pig and Hive, being high level abstractions on top of MapReduce, also enable many of their own optimisations to be applied to the underlying MapReduce jobs during the compilation stage [10, 31] as well as having the benefit of being susceptible to manual query optimisations, familiar to programmers familiar with query optimisations from SQL [13].

[1] https://hadoop.apache.org

3.2 Realtime data processing

With HDFS being an open source project with a large range of users [36] and code contributors [15], it has grown as a project in the last few years for uses beyond what it was originally intended for; a backend storage system for Hadoop MapReduce. HDFS is now not only used with Hadoop's MapReduce but also with a variety of other technologies, a lot of which run as a part of the Hadoop ecosystem. Big data processing has moved on from the more "traditional" method of processing, involving MapReduce jobs, which were most suitable for batch processing of data, to those methods which specialise in the realtime processing of data. The main difference of which is that rather than waiting for all the data before processing can be started, in realtime data processing the data can be streamed into the processing system in realtime at any time in the whole process.

Comparing batched data processing to realtime data processing, it is useful to relate back to the four V's identified in §2.1. Velocity of data is often inconsistent with realtime processing, while in batch mode processing, where you are processing the data that has already arrived and is waiting in batches to be processed, the velocity can be considered consistent. Veracity of data is often not expected to be as consistent in realtime, as sometimes there might be times where data does not arrive or only certain parts of the data arrive at certain times. A realtime processing system, often called a data stream processing system (DSPS) in other literature, needs to be able to deal with these timeliness issues, while a batch data processing system may expect everything that needs to be there to be available.

3.2.1 Hadoop YARN

As previously looked at, the focus of the MapReduce model was performing distributed and highly parallel computations on distributed batches of data. This suited a lot of the big data audience, and hence Hadoop became the dominant method of big data processing [21]. However for some more specialised applications, such as the realtime monitoring of sensors, stock trading, and realtime web traffic analytics, the high latency between the data arriving and actual results being generated from the computations was not satisfactory [17].

A recent (2013) industry survey on European company use of big data technology by Bange, Grosser, and Janoschek, noted in [2], shows that over 70% of responders show a need for realtime processing. In that time, there has certainly been a response from the open-source software community, responding with extensions to more traditional batch systems, such as Hadoop, along with complete standalone DSPS solutions.

On the Hadoop front, the limitations of the MapReduce model were recognised, and a large effort was made in developing the "next generation" of Hadoop so that it could be extensible and used with other programming models, not locked into the rigidity of MapReduce. This became known officially known as YARN (Yet Another Resource Negotiator). According to the original developers of YARN, Vavilapalli et al. state that YARN enables Hadoop to become more modular, decoupling the resource management functionality of Hadoop from the programming model (traditionally, MapReduce) [33]. This decoupling essentially allowed for non-MapReduce technologies to be built on top of Hadoop, still interacting with the overall ecosystem, allowing for much more flexible applications of big data processing on top of the existing robust framework Hadoop provides.

Examples of such systems now built, or in some cases ported, to run on top of Hadoop, providing alternative processing applications and use cases include:

- Dryad, a general-purpose distributed execution system from Microsoft Research [16]. Dryad is aimed at being high level enough to make it "easy" for developers to write highly distributed and parallel applications.

- Spark, a data processing system, from researchers at UC Berkeley, that focuses on computations that reuse the same working data set over multiple parallel operations [38].
- HBase, a layer on top of a distributed file system (such as HDFS, or GFS) allowing for the storing of huge amounts of structured or semi-structured data [18].

These are just some of the more popular examples of applications built to interact with the Hadoop ecosystem via YARN.

3.2.2 Storm

One very notable DSPS technology developed independently of Hadoop, and that is gaining immense popularity and growth in user base, is the Storm project. Storm was originally developed by a team of engineers lead by Nathan Marz at BackType [2]. BackType has since been acquired by Twitter, Inc. where development has continued. Toshniwal et al. describe Storm, in the context of its use at Twitter, as "a realtime distributed stream data processing engine" that "powers the real-time stream data management tasks that are crucial to provide Twitter services" [32]. Since the project's inception, Storm has seen massive adoption in industry, including among some of the biggest names, such as Twitter, Yahoo!, Alibaba, and Baidu [28].

4 Relationships between big data classes and big data processing

5 Conclusion

[2] https://storm.apache.org/

References

[1] Big data architecture and patterns, part 1: Introduction to big data classification and architecture, Sept. 2013.

[2] Bange, C., Grosser, T., and Janoschek, N. Big data survey europe - usage, technology and budgets in european best-practice companies, 2013.

[3] Beyer, M. Gartner says solving 'big data'challenge involves more than just managing volumes of data. *Gartner http://web. archive. org/web/20110710043533/http://www. gartner. com/it/page. jsp* (2011).

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

[5] Borthakur, D. The hadoop distributed file system: Architecture and design.

[6] Chen, H., Chiang, R. H., and Storey, V. C. Business intelligence and analytics: From big data to big impact. *MIS quarterly 36*, 4 (2012), 1165–1188.

[7] Darlin, D. Airfares made easy (or easier). *The New York Times 1* (2006), B1.

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

[9] Dong, X. L., and Srivastava, D. Big data integration. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on* (2013), IEEE, pp. 1245–1248.

[10] 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. *Proceedings of the VLDB Endowment 2*, 2 (2009), 1414–1425.

[11] 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.

[12] Grossman, R. L., and Gu, Y. On the varieties of clouds for data intensive computing. *IEEE Data Eng. Bull. 32*, 1 (2009), 44–50.

[13] Gruenheid, A., Omiecinski, E., and Mark, L. Query optimization using column statistics in hive. In *Proceedings of the 15th Symposium on International Database Engineering & Applications* (2011), ACM, pp. 97–105.

[14] Géczy, P. BIG DATA CHARACTERISTICS.

[15] Hadoop.apache.org. Who we are, 2014.

[16] Isard, M., Budiu, M., Yu, Y., Birrell, A., and Fetterly, D. Dryad: distributed data-parallel programs from sequential building blocks. In *ACM SIGOPS Operating Systems Review* (2007), vol. 41, ACM, pp. 59–72.

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

[18] Khetrapal, A., and Ganesh, V. Hbase and hypertable for large scale distributed storage systems. *Dept. of Computer Science, Purdue University* (2006).

[19] Lämmel, R. Google's mapreduce programming model—revisited. *Science of computer programming 70*, 1 (2008), 1–30.

[20] Lievesley, D. Increasing the value of data. *BLRD REPORTS 6122* (1993), 205–205.

[21] 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.

[22] Mayer-Schönberger, V., and Cukier, K. *Big Data: A Revolution that Will Transform how We Live, Work, and Think*. An Eamon Dolan book. Houghton Mifflin Harcourt, 2013.

[23] McKusick, M. K., and Quinlan, S. Gfs: Evolution on fast-forward. *ACM Queue 7*, 7 (2009), 10.

[24] 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* (2008), ACM, pp. 1099–1110.

[25] Ritterman, J., Osborne, M., and Klein, E. Using prediction markets and twitter to predict a swine flu pandemic. In *1st international workshop on mining social media* (2009), vol. 9.

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

[27] St Amant, K., and Ulijn, J. M. Examining the information economy: Exploring the overlap between professional communication activities and information-management practices. *Professional Communication, IEEE Transactions on 52*, 3 (2009), 225–228.

[28] Storm.apache.org. Storm documentation, 2014.

[29] Su, X., Swart, G., Goetz, B., Oliver, B., and Sandoz, P. Changing engines in midstream: A java stream computational model for big data processing. *Proceedings of the VLDB Endowment 7*, 13 (2014).

[30] Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., and Murthy, R. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment 2*, 2 (2009), 1626–1629.

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

[32] Toshniwal, A., Taneja, S., Shukla, A., Ramasamy, K., Patel, J. M., Kulkarni, S., Jackson, J., Gade, K., Fu, M., Donham, J., et al. Storm@ twitter. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (2014), ACM, pp. 147–156.

[33] Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., et al. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing* (2013), ACM, p. 5.

[34] Wang, L., Zhan, J., Luo, C., Zhu, Y., Yang, Q., He, Y., Gao, W., Jia, Z., Shi, Y., Zhang, S., et al. Bigdatabench: A big data benchmark suite from internet services. *arXiv preprint arXiv:1401.1406* (2014).

[35] Watson, H. J. Tutorial: Business intelligence-past, present, and future. *Communications of the Association for Information Systems 25*, 1 (2009), 39.

[36] Wiki.apache.org. Poweredby - hadoop wiki, 2014.

[37] Yang, H.-c., Dasdan, A., Hsiao, R.-L., and Parker, D. S. Map-reduce-merge: simplified relational data processing on large clusters. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data* (2007), ACM, pp. 1029–1040.

[38] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing* (2010), pp. 10–10.