Real-Time Data Analysis Using Spark and Hadoop

Khadija AZIZ, Dounia ZAIDOUNI, Mostafa BELLAFKIH
Networks, informatics and mathematics department
National Institute of Posts and Telecommunications
Rabat-Morocco
{k.aziz,zaidouni,bellafkih}@inpt.ac.ma

Abstract— **Big Data is at use every single day, from the adoption of Internet through social networks, mobile devices, connected objects, videos, blogs and others. Big Data real-time processing have received a growing attention especially with the expansion of data in volume and complexity. Big data is created every day, from the use of the Internet through social networks, mobile devices, connected objects, videos, blogs and others. In order to ensure a reliable and a fast real-time information processing, powerful tools are essential for the analysis and processing of Big Data. Standards MapReduce frameworks such as Hadoop MapReduce face some limitations for processing real-time data of various formats. In this paper, we highlight the implementation of the de-facto standard Hadoop MapReduce and also the implementation of the framework Apache Spark. Thereafter, we conduct experimental simulations to analyze a real-time data stream using Spark and Hadoop. To further enforce our contribution, we introduce a comparison of the two implementations in terms of architecture and performance with a discussion to feature the results of simulations. The paper discusses also the drawbacks of using Hadoop for real-time processing.**

Keywords- Big Data, Real-time processing, Real-time analysis, Hadoop, MapReduce, Spark, Twitter, HDFS

I. INTRODUCTION

Big Data is defined to refer to the increased volume of data, which makes it really difficult to store, process and analysis using classical processing tools. Most definitions of Big Data defined big data as characterized by 3 Vs [1]: Volume, Variety, and Velocity. These terms were originally introduced by Gartner to describe the elements of big data challenges.

Data volume mainly refers to all types of the data which is generated from different sources and continuously expands over time. In today's generation, the storing and processing includes exabytes (10^{18} bytes) or even zettabytes (10^{21} bytes) whereas almost 10 years ago, only (10^{6} bytes) were stored on floppy disks. For example, the social network Twitter generated in January 2013, 7 terabytes of data every day. The volume of the tweets generated grew increasingly at high rates. In Twitter's short history, Twitter was processing 5,000 tweets per day in 2007 to 500 million tweets per day in 2013 [11].

The data Variety refers to the different types of data collected via sensors, smartphones, or social networks. Such data types include video, image, text, audio, and data logs, either structured, semi-structured, or unstructured format.

The Velocity refers to the speed of data transfer. The contents of data constantly change because of the absorption of complementary data collections, introduction of previously archived data or legacy collections, and streamed data arriving from multiple sources.

Back in 2004, Google invented the MapReduce [2] programming model for processing the massive data, In fact, this model has allowed parallel and distributed computed of a very large amount data contained in a cluster. Additionally, it consists of a processing paradigm of executing data with partitioning and aggregation of intermediate results. Its main goal is to process data in parallel, where, data splitting, distribution, synchronization and fault tolerance are handled automatically by the framework MapReduce [12].

Various MapReduce frameworks exist. Actually they help provide off-the-shelf high scalability and can significantly shorten the processing time for big data. For example, the open-source framework Hadoop has become widely used for processing big data. Hadoop has some limitations of supporting real-time updates. In fact, Hadoop most analytic updates and re-scores have are completed by long-run batch jobs, which significantly delays presenting the analytic results. For example, in some businesses revenue loss can be measured in milliseconds, like the notion of a batch job which has no bearing. Therefore, the need for innovative techniques and methodologies that will be able to provide real-time results Open source frameworks such as Apache Spark address these needs.

This paper is structured as follows: Section II reviews the MapReduce Hadoop implementation. While Section III details the Spark implementation. The experimental evaluation of real-time data analysis with Hadoop and Spark are discussed in Section IV. Section V includes the drawbacks of using Hadoop with additional comparison of its functioning mechanisms with Spark in real-time processing. Finally, Section VI entails the concluding remarks.

II. HADOOP MAPREDUCE IMPLEMENTATION

Hadoop is an open source framework that allows distributed processing of a large amount of data through a cluster using a simple programming model. It allows applications to work on multiple nodes for big data processing.

A. MapReduce

MapReduce is a programming model that allows parallel computing on a large amount of data using a large number of machines called Cluster or Grid (more data more machines).

MapReduce has two main functions:

978-1-5386-4225-2/18/$31.00 ©2018 IEEE

- Mapper: consists of splitting out and distributing the different data partitions. It is realized by a node that distributes the data to other nodes. Each node in reception applies processing on received data.
- Reducer: (The inverse of the Mapper function). It consists of collecting the results computed in parallel and merge them into a single global result. Each distributed data partition consists of a couple (key/value). The key is used during the merge to group the values (which have the same key). All the values associated with a key are therefore collected at the end of the Reduce.

B. *MapReduce tasks in Hadoop MapReduce*

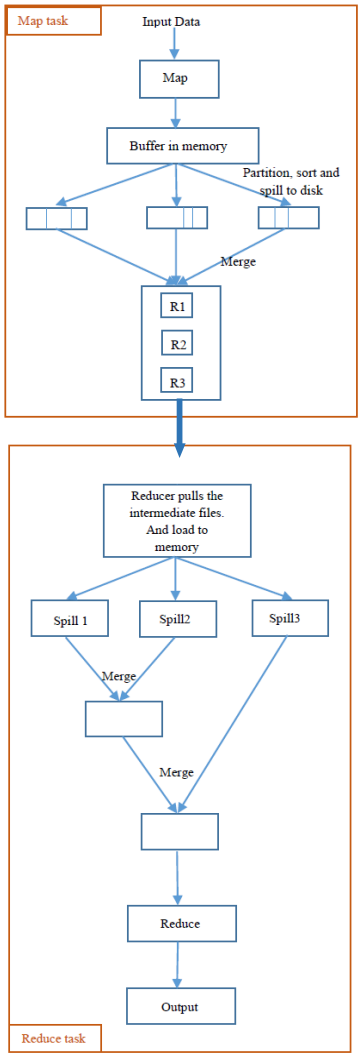

Figure 1. Map&Reduce tasks in Hadoop

- During the Map phase keys are created with an associated value in the form of (Key, Value) pairs.
- Storing the output in buffer in memory
- When the memory is full, the data is partitioned, sorted, and spilled onto the disk.
- Combining All data are into one large file according to Reducers
- Data or intermediate files created by the Map phase are sorted by Reducers and loaded into the memory.
- When the buffer is filled, it is spilled on the disk.
- Spilling data in buffer is merged into a larger file.
- During the Reduce phase, processing is performed on the merged data.

For Hadoop, the HDFS uses the MapReduce algorithm for data processing. After each processing operation, MapReduce saves a backup of data in a physical server, for data recovery in case of failure.

C. *HDFS (Hadoop Distributed File System)*

HDFS [3] is a distributed file system that runs on commodity hardware and provides high-performance access to data distributed in Hadoop clusters. This file system is designed to be extremely fault tolerant [4], facilitating the rapid transfer of data between the nodes of the system. This fact help actually keep Hadoop systems operational if a node breaks down. Additionally, this mechanism reduces the risk of a critical failure, even if the failure involves several nodes. HDFS is based on the Master/Slave architecture, the large data is divided into blocks and sub-blocks until it is ready to be processed by the different nodes of the Cluster.

III. APACHE SPARK IMPLEMENTATION

Apache Spark [6] is an open source processing engine that is highly concerned as fast and reliable [15]. It was essentially designed for fast computing. This system provides APIs (Application Programming Interface) in different programming languages such as Scala, Java and Python. This system was developed at UC Berkeley in 2009 [5]. Apache Spark supports in-memory computing across DAG (Directed Acyclic Graph).

Spark is designed to function in a Hadoop environment and to meet the limitations of MapReduce, it is the alternative of batch map/reduce. Spark can be used to make real-time data processing and fast queries that finish in some seconds.

A. *MapReduce tasks in Spark*

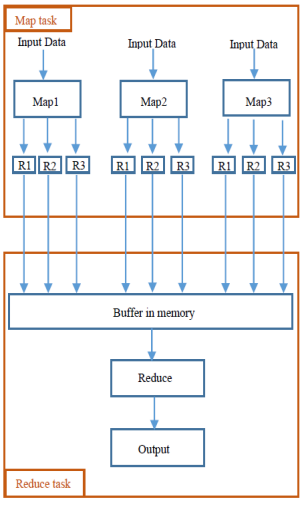

Figure 2. Map&Reduce tasks in Spark

- Writing the output of the Map phase in the OS buffer cache
- Deciding if the data will remain in buffer in memory or will be spilled on the disk
- Merging The output of Map coming from the same cores into a single file
- Generating several spill files as the number of Reducers
- Sending the data as intermediate files to the Reducers
- Writing the Data to the Reduce memory.
- Processing the data by the Reduce function.

Spark copies the data into RAM (processing in-memory). This type of processing reduces the time needed to interact with physical servers and this makes Spark faster than Hadoop MapReduce. For data recovery in case of a failure, Spark uses RRDs.

IV. Experimental Evaluation and Results

Twitter has rapidly gained worldwide popularity, with over 284 million registered users as of 2014, generating over 500 million tweets each day. Twitter's 140-character, plain text messages are relatively easily to process and store, and access to this stream of data (and related user account metadata) has been provided through Twitter's own application programming interfaces (APIs) and related third-party services.

In Our experimental study, we have retrieved, we retrieve the Twitter data (as twitter is a very important source of data) for analysis with Apache Spark and Hadoop MapReduce.

Both machines had the same characteristics. Table I and II show the characteristics of the nodes of each machine. Figure4 shows some information about Hadoop and Spark that are deployed in our study.

Table I. Information of Machines

Hostname	**hadoop**	**spark**
IP address	192.168.1.1/24	192.168.1.2/24
Memory	5GB	5GB
OS	Linux (Ubuntu)	Linux (Ubuntu)
Processors	1	1
Hard Disk	40GB	40GB

Table II. Software Configuration

Software name	**Version**
OS	Ubuntu 14.04 / 64 bit
Apache Hadoop	2.7.3
Apache Spark	2.0.1
JRE	Java(TM) SE Runtime Environment (build 1.8.0_131-b11)
SSH	OpenSSH_6.6.1p1
Virtualization platform	VMware Workstation Pro 12

A. Hadoop MapReduce

1) Proposed architecture

In this study, we developed an application in Scala language for streaming data from twitter in Apache spark. The architecture of this study as shown in figure 3.

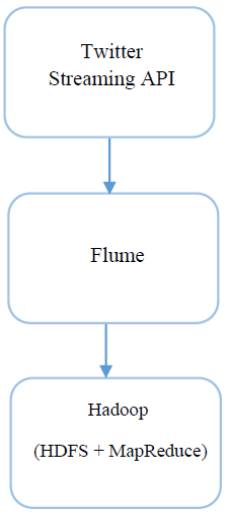

Figure 3. Architecture of processing data in Hadoop MapReduce

We have used the Twitter App and also: Consumer Key (API Key), Consumer Secret (API Secret), Access Token and Access Token Secret, as an arguments in our program in order to connect to Twitter and access the online tweets.

We have also employed the Apache Flume in order to retrieve data, after its recovery. This data was stored in HDFS, and then MapReduce processing was done on this data.

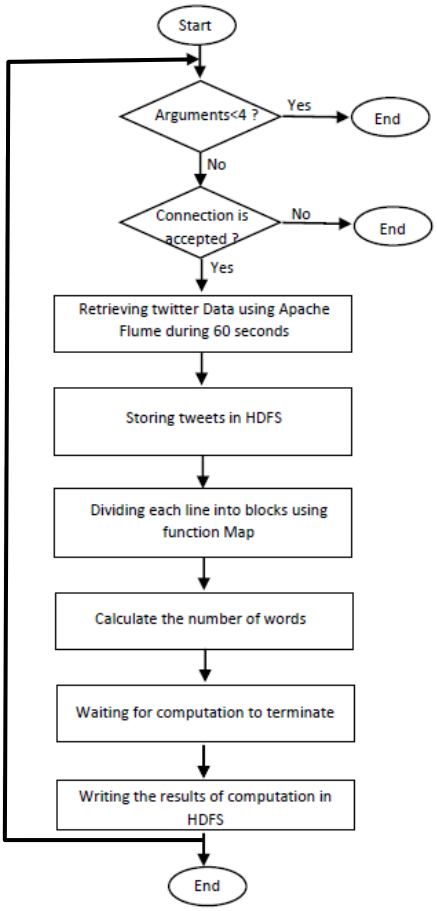

Figure 4. Workflow of processing data in Hadoop MapReduce

We have imported the Hadoop package to our Eclipse project. Then we have started calculating the tweets every 60 seconds as show in the following workflow (figure 4).

2) *Experimental setup*

The following figure shows the MapReduce treatments on the Twitter data and the time elapsed during this processing.

```
17/09/15 15:55:12 INFO mapreduce.Job:  map 100% reduce 100%
17/09/15 15:55:12 INFO mapreduce.Job: Job job_local351169806_0001 completed successfully
17/09/15 15:55:12 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=5741280
                FILE: Number of bytes written=7741960
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=60073402
                HDFS: Number of bytes written=573994
                HDFS: Number of read operations=13
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=4
        Map-Reduce Framework
                Map input records=618887
                Map output records=5090840
                Map output bytes=49579466
                Map output materialized bytes=1433692
                Input split bytes=104
                Combine input records=5090840
                Combine output records=98636
                Reduce input groups=49318
                Reduce shuffle bytes=1433692
                Reduce input records=98636
                Reduce output records=49318
                Spilled Records=295908
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=141
                Total committed heap usage (bytes)=274866176
```

Figure 5. Processing Twitter Data with Hadoop MapReduce

B. *Apache Spark*

The following experimental shows the high quality of Apache Spark in processing data within a lapse of time.

3) *Proposed architecture*

In this study, we developed an application in Scala language for streaming data from twitter in Apache spark. The architecture of this study as shown in figure 6.

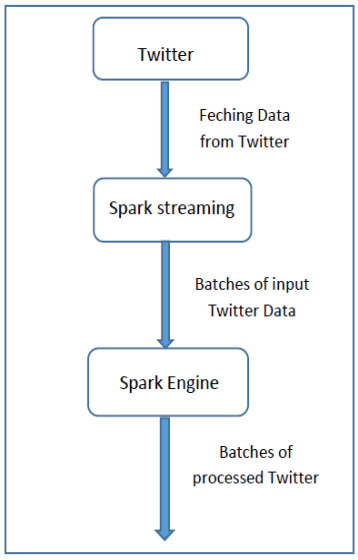

Figure 6. Architecture of processing data in Apache Spark

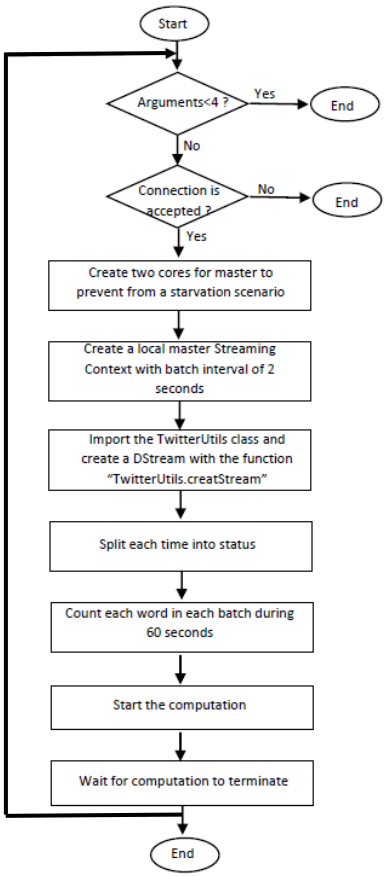

Figure 7. Workflow of processing data in Spark

First, we have imported Spark dependency then we have imported Spark Streaming classes and some implicit conversions from Streaming Context into our working environment. The main goal was to add useful methods to other classes.

On the other side, we have created Twitter App and then we have used: Consumer Key (API Key), Consumer Secret (API Secret), Access Token and Access Token Secret. This have helped to establish the connection to Twitter and the access to the online tweets.

Finally, the program had started the streaming and had calculated the tweets every 60 seconds as shown in figure 7.

4) *Experimental setup*

In this experiment, we developed a program in Scala language to show the reception rate of the twitter data and also the processing time of batches. And then, we show how Spark processes Twitter Data in some seconds. Apache Spark provides a web interface that gives an overview of running application to monitor, view, and analyze the operating state of Apache Spark.

The following figure shows some Tweets that we got in Console Window Eclipse.

```
USA (8 tweets)
jobs (6 tweets)
BAMA2017 (4 tweets)
BAMA2017DiamondEdition_Zayn (3 tweets)
Charleston (1 tweets)
job (1 tweets)
Presidentielle2017 (1 tweets)
U… (1 tweets)
cdnpoli (1 tweets)
Orlando (1 tweets)
```

Figure 8. Example of some Tweets processed in last 60s

Figure 9 shows the diffusion of the Streaming application. The input rate means that the application receives data at a rate of about 46 events per second. For the timeline, we note that there are some peaks in the curve, this means that the reception of data at these moments has increased compared to the average rate.

Figure 10 shows the Scheduling Delay, which defines the waiting time of a new batch processing for that the preceding batches are completed (to start another batch). This Scheduling Delay must be absolutely too small for the batch to be processed quickly (from its creation). For our example, the Scheduling Delay average is 0.003s. So this delay allows us to know that the streaming of our application is stable.

Figure 11 shows that the batches are processed within 222 ms on average processing, this average is smaller than the batch interval. This means that the treatment is in good state.

- *Timelines*

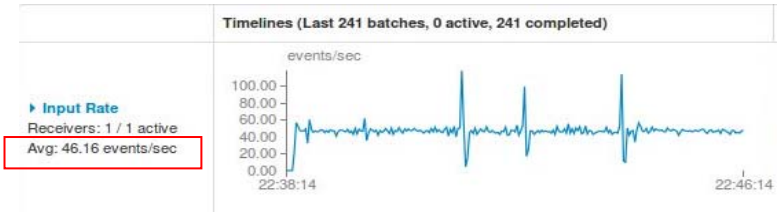

Figure 9. Input Rate

Figure 10. Scheduling Delay

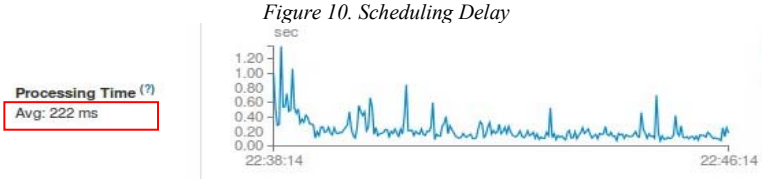

Figure 11. Processing Time

Figure 12 shows the completed batches, it is noted that some batches have taken a long time to finish. So to have more details on the batch and to know the reason of processing delay of these batches, it is enough just to click on the Batch Time which has long processing time as shown in figure 13. This figure shows the duration and the status as well as the jobs generated (in our case 2 jobs generated).

- *Spark jobs*

Input Size	Scheduling Delay (?)	Processing Time (?)	Total Delay (?)
73 events	1 ms	89 ms	90 ms
87 events	2 ms	65 ms	67 ms
60 events	1 ms	0.1 s	0.1 s
68 events	2 ms	84 ms	86 ms

Figure 12. Completed Batches

Batch Duration: 2 s
Input data size: 60 records
Scheduling delay: 1 ms
Processing time: 0.1 s
Total delay: 0.1 s

Output Op Id	Description	Duration	Status	Job Id	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total	Error
0	foreachRDD at PopularTags.scala:54 +details	0.1 s	Succeeded	1392	65 ms	2/2 (5 skipped)	3/3 (50 skipped)	
				1393	16 ms	1/1 (6 skipped)	2/2 (52 skipped)	

Figure 13. Details of Batch

V. DISCUSSION AND COMPARAISON OF SPARK AND HADOOP

Hadoop and Spark are two different distributed software frameworks. Hadoop is a MapReduce framework on which you may run jobs supporting the map() and reduce() functions. Spark is not a MapReduce framework, but can be easily used to support a MapReduce framework's functionality; it has the proper API to handle map() and reduce() functionality. Spark is not tied to a map phase and then a reduce phase. A Spark job can be an arbitrary DAG (directed acyclic graph) of map and/or reduce phases. Spark programs may run with or without Hadoop and Spark may use HDFS or other persistent storage for input/output.

Spark outperforms Hadoop for the following reasons:

Spark utilizes memory-based storage for RDDs and Hadoop MapReduce processes disk-based operations.

Spark workloads have a higher number of disk accesses per second than Hadoop's and has a better memory bandwidth utilization than Hadoop.

In Spark, task scheduling is based on an event-driven mode but Hadoop employs heartbeat to tracking tasks, which periodically causes a few seconds delays.

Apache Spark comes to solve the limits of Hadoop MapReduce especially when it comes to velocity. Spark can handle real-time processing and ML (machine learning) which Hadoop MapReduce can't do. Spark processes the data in memory only once, while Hadoop MapReduce uses batch processing, so it needs to analyze the data step by step. So, more we increase the amount of data, more Hadoop becomes slow.

Apache Hadoop is designed to process different types of data (structured, semi-structured, and unstructured data) in HDFS. Hadoop MapReduce can process a large volume of data on a distributed system in batch mode. Apache Spark can also process on different types of data, but it is designed especially for faster processing. To ensure faster processing, it uses several types of processing like batch, iterative operations and streaming.

Apache Hadoop uses the MapReduce algorithm for data processing, this algorithm reads and writes from disk which makes the processing speed of MapReduce slower. Apache Spark reduces the read/write cycle on disk and stores intermediate data in memory that is why it can process up to 100 x faster in memory and 10 x faster on disk [7].

Hadoop MapReduce is designed to perform batch processing on a large volume of data, so when it comes to streaming or processing data in real time, Hadoop MapReduce can do nothing. By contrast Apache Spark, it can process data in real time via Spark Streaming. Spark Streaming launches a set of micro-batch processing on RDDs, and it makes parallel processing across several nodes in a cluster [13].

Hadoop MapReduce is relatively slower, because it is designed to do batch processing on a large amount of data and varied formats. Apache Spark adopts micro-batch processing, that is why it is faster than Hadoop MapReduce. Apache Spark caches a large amount of the input data on memory by the RDDs and stores the intermediate data in the memory itself and then writes the data to the disk.

The disadvantage of Apache Spark is that it does not have a distributed storage system, so it is essential to use an additional system for data storage. For Hadoop MapReduce, it uses HDFS (Hadoop Distributed File System) to process an enormous amount of data on hard disks distributed in a cluster.

When it comes to structured data (eg in SQL databases), in this case Hadoop MapReduce is a wise choice, it can process data quickly and at less cost. On the other hand, Spark is the best solution when analyzing any type of unstructured data (videos, social networks ...).

VI. CONCLUSION

In this article we used Apache Spark for analysis Twitter data. As obtained in our experimental setup, Spark analyzed the tweets in a small period of time which is less than a second, this prove that Spark is a good tool for processing streaming data in real-time.

If the data is static and it is possible to wait for the end of batch processing, MapReduce is sufficient for processing. But if we want to analyze streaming data, it is absolutely necessary to use Spark. And as shown in our study, Spark was able to analyze data in few seconds. Spark is a high quality tool for memory processing that allows processing streaming data in real-time on a large amount of data. Apache Spark is much more advanced than MapReduce. It supports several requirements like real-time processing, batch and streaming.

Further research should focus of the optimization of Hadoop and Spark by tuning different default parameter configuration settings. In order to satisfy this optimization, we suggest applying this tuning to improve the performance when running on Hadoop or Spark respectively.

VII. REFERENCES

[1] Gunarathne, T. (2015). Hadoop MapReduce v2 Cookbook. Packt Publishing Ltd.

[2] Dean, J., & Ghemawat, S. (2010). MapReduce: a flexible data processing tool. Communications of the ACM, 53(1), 72-77.

[3] Hadoop, https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[4] Patel, A. B., Birla, M., & Nair, U. (2012, December). Addressing big data problem using Hadoop and Map Reduce. In 2012 Nirma University International Conference on Engineering (NUiCONE) (pp. 1-5). IEEE.

[5] Databricks, https://databricks.com/spark/about

[6] Spark, http://spark.apache.org/

[7] Karau, H., Konwinski, A., Wendell, P., & Zaharia, M. (2015). Learning spark: lightning-fast big data analysis. " O'Reilly Media, Inc.".

[8] Spark, https://spark.apache.org/docs/1.6.0/spark-standalone.html

[9] Spark, https://spark.apache.org/docs/1.6.0/running-on-yarn.html

[10] Spark http://databricks.github.io/simr/

[11] Tweets, http://www.internetlivestats.com/twitter-statistics/

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

[13] Liu, X., Iftikhar, N., & Xie, X. (2014, July). Survey of real-time processing systems for big data. In Proceedings of the 18th International Database Engineering & Applications Symposium (pp. 356-361). ACM.

[14] Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., ... & Ghodsi, A. (2016). Apache spark: a unified engine for big data processing. Communications of the ACM, 59(11), 56-65.