UPPSALA
UNIVERSITET

Large Datasets for Scientific Applications (1TD268)
Spring 2020

Project Report

Reddit Comments Analysis

Team 13

Basit, Abdul   Díaz Ubieta, Cristina   Palm, Elin
Sankara Velayutham, Girish Shanmugam   Shivakumara, Shreyas

June 7, 2020

1 Background

In this project, a dataset containing Reddit comments was used which is a subset of the Pushlift Reddit dataset. This is a ***large dataset*** with the size of approximately *680 GB* of data [1]. The Pushlift Reddit dataset was collected from 2015 and is still being updated today. It contains billions of comments and hundreds of million posts.

The dataset was created to make it more convenient to access and perform analysis on the data. The dataset gives the opportunity for researchers to save more time and, in an easy way, use Reddit comments in their projects since this dataset has made it easier to query, fetch old data, and additional functionalities compared to the Reddit API. These advantages come from allowing full-text searches against the comments and provide five times larger objects limit than Reddit [2].

The dataset has been used widely in research and has a large record in peer-reviewed publications. It has been used in many different subject studies for example in research about online extremism, online community governance, online disinformation, web science, big data science, health informatics, and robust intelligence [2]. There is also other social media data collection that has been used in further research in a similar way as the Pushlift Reddit dataset [2].

2 Data Format

The Reddit dataset used in our work is semistructured in JSON format. This format is very flexible as it does not have a predefined organization, which makes it ideal for data exchange between very different databases. Being semistructured is also helpful when it comes to browsing the data. Semistructured data is also called self-describing since the way that it is organized is through tags or markers [5].

The advantage of using a semi-structured dataset is that it can support lists of objects, which will avoid confusing translations from a list to a relational model. It also supports nested and hierarchical data, which will simplify data models that represent complicated relationships between the entities. Some disadvantages would be that we cannot use an already made query language like SQL. Because there is no separation of the schema and the data, it is more difficult to interpret the relationship between the data [5].

On the other hand, structured data has a higher level of organization and predictability. This will make it easier to be used by machine learning algorithms, as well as the average business user. The disadvantages are mainly related to the lack of flexibility. Because the data has a predefined purpose, this reduces the possibilities of things that can be done with the data, also doing scalability on this type of data is not as easy as with the other format [6].

A third existing format is the unstructured data, even more, flexible than semi-structured. This data does not have a pre-defined data model, thus it is not a good choice for mainstream relational databases [4].

The semistructured dataset was born from the necessity of creating new forms of data for the already conventional database technology. And because of this, all its qualities mentioned it is the best-suited format for this Reddit comments dataset [5].

3 System Architecture

Since we are working with a large dataset, it might not be practically possible to store the entire data in a single machine, as it might require a lot of storage and the hardware could be very costly. Also, we would be facing the risk of losing the entire data, since we are only storing it in a single location. Another reason is that if the data were to be corrupted, it would be hard to retrieve it. To overcome these shortcomings, an alternate method is to use multiple smaller machines and connect them to form a cluster of distributed storage. For this purpose, we chose the Hadoop framework which is an open-source distributed file system), so that our large dataset could be stored in a distributed manner. Hadoop uses data replication technique to ensure fault tolerance [3]. We loaded our large dataset into Hadoop's Distributed File System (HDFS) with a block replication factor of 2. We wanted to make use of the distributed setup for processing the data that is stored in HDFS. Spark is an open-source distributed cluster-computing framework which provides the support for distributed processing. The framework is originally written in Scala but it provides an API in python for analysis called PySpark [7].

The *large dataset* that we had was loaded and stored in a *distributed storage* system using ***Hadoop*** and ***PySpark*** was used for *distributed processing.*

In our cluster setup, we had 5 Virtual Machines installed with Hadoop (v2.7) and Spark (v2.11.12) in a master-slave architecture where one of them acted as master and other four as slaves. All the machines had the same configuration: 1 VCPU, 2GB RAM, and 20GB disk space.

4 Computational Experiments

To test the scalability of our distributed solution proposed in Section 3, we planned to conduct two experiments based on a few analyses on the Reddit comments dataset. To get more accurate results and obtain solid conclusions, every experiment was carried out three times and the average time has been reported and used for analysis.

4.1 Analysis

Two kinds of analyses were designed to test the scalability of our solution in PySpark and HDFS, which are as follows:

- ***Top 10 popular words used in the comments:*** The most common and popular words used in the comments are listed using MapReduce implemented in PySpark. In the `map()` function, each line the JSON is parsed to get the comment, split it into words and generate a <key,value> pair in the form of <word,1>. The function to add two values is passed on as an argument to the reduce function `reduceByKey()` which sums up the key-value pairs based on the key.

- ***Top 10 active users:*** The users with a lot of activity (number of comments posted) are considered as active users. Instead of writing our own custom MapReduce functions in PySpark, we wanted to perform analysis using PySpark's dataframe. Hence, the JSON dataset was loaded as a dataframe and operations like filter, `groupBy()`, aggregations(`count()`) and sorting(`orderBy()`) were performed on the dataframe in this analysis.

4.2 Strong Scalability Test

To test whether our solution would scale when the number of cores is added in the future, we performed the strong scalability test on the two analyses mentioned in Section 4.1. The objective of this test is to observe the variation in execution time with an increasing number of cores for fixed data size. With a fixed data size of *991.71 MB* which had around *1787877* comments, we performed the experiments starting with a single core and, gradually, adding one core at a time, up to four cores. To vary the number of cores in our four core cluster setup, `spark.cores.max` property in Spark's configuration was used while initializing the Spark session. The experiments were carried out on both of the analyses increasing the cores, and the results are reported in Tables 1 and 2. A plot showing the ideal speedup that we expected and the actual speedup achieved by our solution is shown in Figure 1.

Table 1: Top 10 popular words

Cores	Execution Time(s)	Speed up
1	129	1
2	71	1.8
3	48.4	2.66
4	33.7	3.8

Table 2: Top 10 active users

Cores	Execution Time(s)	Speed up
1	101	1
2	51.03	1.97
3	40.2	2.51
4	34.76	2.9

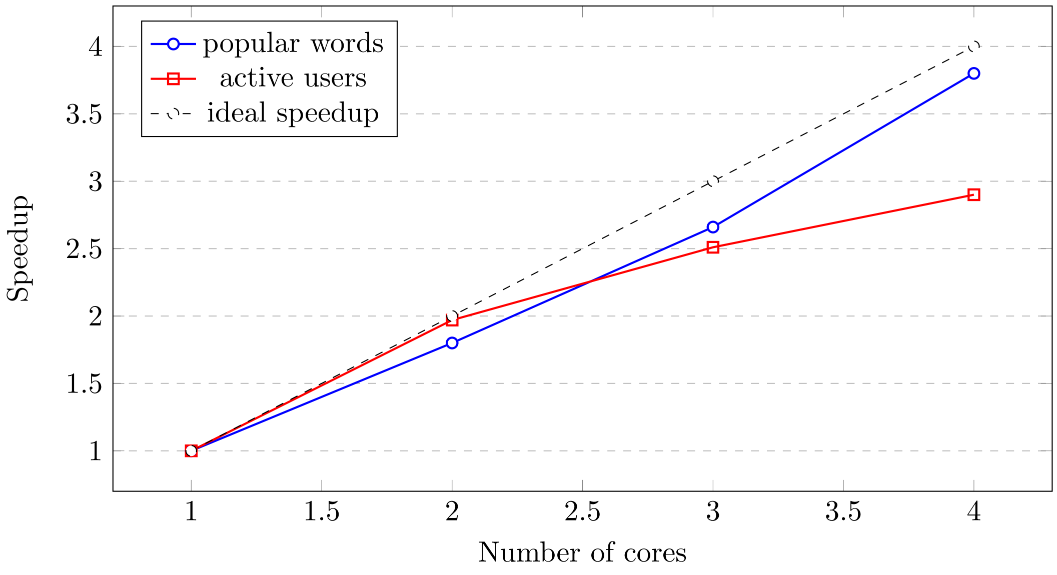

Figure 1: Strong scalability test

4.3 Weak Scalability Test

The objective of this test is to observe whether the execution time remains constant when each core handles the same amount of work. This is observed by increasing the number of cores and size of the data simultaneously, such that each core processes the same amount of data. The load on each core was fixed to have *595959* comments. Starting with a single core and *595959* comments, we tried running the experiments by adding one core and *595959* comments, simultaneously, during each step up to 4 cores. The results obtained for both of the analyses are reported in Table 3.

Table 3: Weak scalability test

Cores	Data Size	Comments	Time(s) for popular words	Time(s) for active users
1	346.63 MB	595959	47.3	44.2
2	661.14 MB	1191918	50.53	45.73
3	991.71 MB	1787877	49.7	45.6
4	1.29 GB	2383836	49.1	46.7

5 Discussion and Conclusion

5.1 Outcome of the Experiments

The results from our experiments show that our solution performed considerably well on both of the tests conducted. From Figure 1, it can be seen that the actual speedup that we get from our solution is very close to the ideal speedup during the strong scalability test. However, the performance seems to deteriorate as we keep on increasing the number of cores, but still, it would be better than the serial implementation. Our solution performed exactly as expected for the horizontal scalability or the weak scalability test as we get consistent execution times when we add more machines and data to be processed for both the analyses as shown in the last two columns in Table 3. Overall, our distributed solution for the large dataset is suitable and expected to work on a large scale setup.

However, there are a few things which could be improved in our experiments.

1. Due to constraints in resources, we restricted our tests to four cores. Running the experiments up to four cores might not be good enough to obtain robust conclusions for scalability studies. Though our solution seems to perform well in both the tests in our current setup, there might be cases where the performance might get poor when we increase the cores further.

2. For the weak scalability test, the same set of data was added during every step. This might lead to a skew in the dataset and affect the experiment results. Hence, it is ideal to avoid using duplicated data and instead use new data, or use both and compare them for the weak scalability studies.

5.2 Experiences gained

A couple of the members in the team had never set up clusters or run tests for scalability. So getting to do the entire process from start to finish, downloading the data, setting up the clusters, using Jupyter notebooks to write the scripts in PySpark, and calculating the scalability, made overall great learning experience for those of us who were new to the subject.

We got to design a distributed solution for large datasets, we set up a cluster by connecting several machines and used them for distributed storage and processing. We learned how to store a large dataset in a distributed setup in Hadoop, and also how to process a large dataset in a distributed setup using Spark. Finally, we learned how to check whether a distributed system is scalable.

6 Contributions

All members of our group have contributed equally for the project.

References

[1] J Baumgarten. "Reddit data dump". In: *URL https://files.pushshift.io/reddit* ().

[2] Jason Baumgartner et al. "The pushshift reddit dataset". In: *Proceedings of the International AAAI Conference on Web and Social Media*. Vol. 14. 2020, pp. 830–839.

[3] Dhruba Borthakur et al. "HDFS architecture guide". In: *Hadoop Apache Project* 53.1-13 (2008), p. 2.

[4] David Boulton and Martyn Hammersley. "Analysis of unstructured data". In: *Data collection and analysis* (1996), pp. 282–297.

[5] Peter Buneman. "Semistructured data". In: *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. 1997, pp. 117–121.

[6] David F Huynh, David R Karger, and Robert C Miller. “Exhibit: lightweight structured data publishing”. In: *Proceedings of the 16th international conference on World Wide Web*. 2007, pp. 737–746.

[7] Apache Spark. “Apache spark”. In: *Retrieved January* 17 (2018), p. 2018.