Scalability Analysis on Reddit data

Project in Large Datasets for Scientific Applications
Spring 2020

Group 18
Marina Mota Merlo, Mikaela Eriksson, Md Sayed Hossan Parvez, Dhanush Kumar Akunuri, Yosief Tesfamichael

Background

There are many scientific areas/problems which require big dataset to solve a problem or data-driven approaches to study it. Extracting knowledge from scientific literature is one of such a kind, which is the analytics challenges to process unstructured data found in publications and extracting scientific entities of interest(such as diseases or treatments) and establishing relationships between entities too. To list for some type of analysis made on the literature dataset are, on papers meta-data (on publications distribution, place of published) , text mining(Document processing , clustering, and per-processing).

Commented [1]: Things have to be included in the right part of the report and they should make sense and be correctly written

Commented [2]: Plus we already exceed the number of words, we can improve sections but not increase the length of the report

Commented [3]: I don't see how cosmology and galaxies makes sense in this context

We have chosen to work with the Reddit dataset in our project. Reddit is among the world's topmost visited sites (ranked #19), with more than 330 million sole users sharing news and discussing [1]. It is often referred to as the "front page of the Internet" [2]. In Reddit, registered users can post news and questions. Alternatively, they can share headlines, contents or external links, which are available to all the Reddit user community for voting and commenting. This commenting/discussion is the main feature of this community. For each submission, registered users can upvote and downvote the post. An upvote gives a positive +1 score and a downvote gives a negative -1 score. In this way, popular comments are filtered to be at the top. Most posts produce huge lengthy discussions, where this vast community of users engages with the content and produces a rich set of quality longitudinal data. These recognized longitudinal data can be used to do scientific research in various ways.

In this context, analysing the Reddit dataset can provide insight on complex socio-technical phenomena. Understanding such phenomena requires research on large datasets and, although Reddit is open to data acquisition compared to other social media like Twitter or Facebook, analysing its contents systematically is time consuming. One big-data storage and analytics project, containing a big set of reddit submissions and comments and used in this project, is Pushshift. Using Pushshift data instead of the Reddit API facilitates data analysis for researchers and enables the execution of queries on the whole dataset without downloading monthly dumps. This reduces the amount of storage which is required. It also includes a Slackbot to produce real-time visualizations of the dataset and discuss them with other researchers. To sum up, with the Pushshift Reddit dataset, researchers do not have to be concerned about data collection, cleaning, and storage [2].

Reddit data have been analysed in many different ways. For example, one common approach to study these data is predicting popularity. This issue has been considered in different social network systems. As it was mentioned, a post gets a popularity ranking based on user votes and stays on a page depending on said popularity. Early research found that in the Reddit dataset there are straight relations among post content's opening popularity and its later counts in terms of views and upvotes [3]. Tran et al. illustrate that the popularity of a post can be reasonably delayed, though post headlines and timing are important factors [4]. Another study found that more than 52% of the popular posts are overlooked after initial submission, so post popularity depends on some other factors [5]. In other studies, authors analysed the tree illustration of thread pyramids by examining the structure of comments in every single post [6].

Commented [4]: Is this a good example or should we look for something more related to our analysis?

Commented [5]: I don't know... It would be better to mention related examples, don't know why these examples were chosen... But I don't have so much more time to read articles before the deadline

Commented [6]: I think because we initially wanted to analyse popularity, so Sayed wrote this based on that idea

Because the Pushshift Reddit dataset is large (> 700 GB), scalable approaches are needed to study this relevant modern-day source of data and draw conclusions such as the ones presented above.

Project objective

The aim of this study is to design a scalable, fault-tolerant approach to study the Reddit dataset. A basic analysis will be run on Reddit data; in this case, we decided to calculate the most common words in the dataset. Then we will assess strong and weak scalability under our scalable data processing solution and we will discuss the performance of our proposed solution.

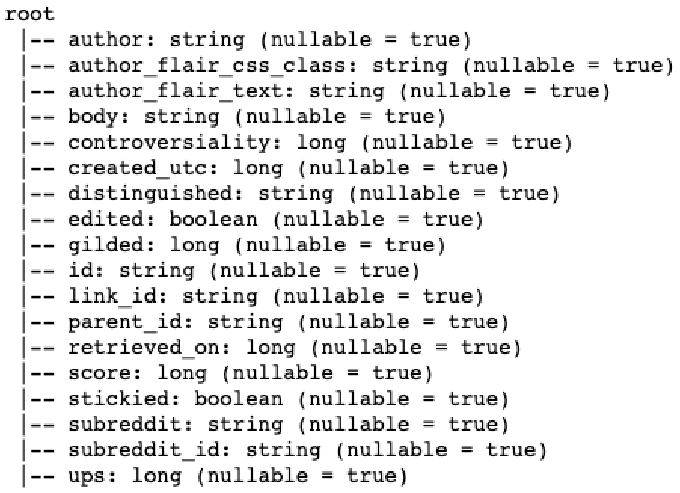
Commented [7]: • How clearly is the problem stated?

• How well is the problem contextualized?
• How well does it cover related, relevant work?
Aim at approximately 4-5 references. You may of course cite papers from the literature seminar if they are relevant.

Data format

The dataset is in JSON format. A JSON object is a collection of unordered pairs of name/value, where the value can be an object, an array, a string, a number, true, false or null. This format is neither traditionally structured nor unstructured but semi-structured data, also known as self-describing structure, which is a form of structured data. It does not conform with the formal structure of data tables in, for example, relational databases, but does still contain markers to separate semantic elements and enforce hierarchies of fields and records within the data [7]. JSON is one of the most frequently used formats for data storage and, when loaded into a HDFS data store, it can easily be accessed and analysed with the processing engine Apache Spark (see section Spark cluster). We therefore decided to directly load JSON data into Spark. By default, JSON data is loaded with the dataframe API. The data frame consists of 18 columns containing information about each post, such as author name, id, score, if edited or not and so on. In the analysis, we focused mainly on the key 'Body', containing the text of the actual post in string format. We thereafter converted the data frame into RDD in order to analyse this property.

Commented [8]: Describe the data format(s) used in the dataset. Put them in context: why were the specific formats chosen, and would there be alternatives? What are the pros and cons of the formats used?



```
root
 |-- author: string (nullable = true)
 |-- author_flair_css_class: string (nullable = true)
 |-- author_flair_text: string (nullable = true)
 |-- body: string (nullable = true)
 |-- controversiality: long (nullable = true)
 |-- created_utc: long (nullable = true)
 |-- distinguished: string (nullable = true)
 |-- edited: boolean (nullable = true)
 |-- gilded: long (nullable = true)
 |-- id: string (nullable = true)
 |-- link_id: string (nullable = true)
 |-- parent_id: string (nullable = true)
 |-- retrieved_on: long (nullable = true)
 |-- score: long (nullable = true)
 |-- stickied: boolean (nullable = true)
 |-- subreddit: string (nullable = true)
 |-- subreddit_id: string (nullable = true)
 |-- ups: long (nullable = true)
```

Figure 1. Keys with corresponding format present in the dataset.

Architecture

Commented [9]: This is the main section of the report. Describe and motivate the choice of tools and the distributed system that you designed. Describe how you have designed your scalability experiments, and present the results. Think carefully about suitable ways to illustrate the scalability of your solution.

Spark cluster

Apache Spark is an open source processing engine for analyzing large data sets. With a wide range of libraries and language integrated APIs, it is one of the most widely used solutions [8]. To be able to handle large amounts of data and to do analyses on the Reddit dataset, a Spark cluster has been set up. The cluster consists of a Spark master and four workers, depending on this master. Spark was installed to both master and worker nodes. The analyses run on Spark were written in Python code, by using the Python API PySpark provided by Spark.

Data Storage in HDFS

HDFS is a distributed file system that runs on top of the local file system. One of its most remarkable attributes is its fault-tolerance, which is relevant when managing files in a computer cluster [9]. In this case, we used HDFS to store the Reddit dataset. First, we loaded a subset of the dataset in the master VM, and switched from Hadoop standalone mode to pseudo-distributed mode in that VM. Connecting to port 9000 from the master's IP, we were able to access these data with the python interface Jupyter Notebook. However, because that was only a small subset of data, we explored the possibility of expanding the storage capacity of HDFS. To do so, we attached a new volume to the master VM and created a mount point for that volume. We then changed the Hadoop datanode directory to the mount point. What we did was increasing the capacity of a existing datanode, but another possible way to approach this problem is adding new datanodes. We managed to increase the capacity from

20GB (the initial VM capacity) to around 1000GB (the capacity of the attached volume). This allowed us to download a great portion of the dataset; however, due to limited computational resources, we only analysed files ranging from 100-1000MB.

Computational experiments

Data analysis

In order to perform word count analysis on the dataset, some preprocessing had to be done. First, the data frame was converted to RDD, and deleted posts without any content were filtered out. Then all lines of sentences were converted into a list of lower case, distinct words without special characters and numbers with re (regular expression) Python library. In order to filter out words without meaning, so called stop words, the nltk.corpus library was imported. By using this library, stopwords were added to a list, used in a function to filter out words in the list. This was done in order to obtain a more meaningful word count, where it makes sense to exclude common words like ‘and’, ‘or’, ‘that’ and so on. Thereafter, the most common words were counted using a MapReduce approach.

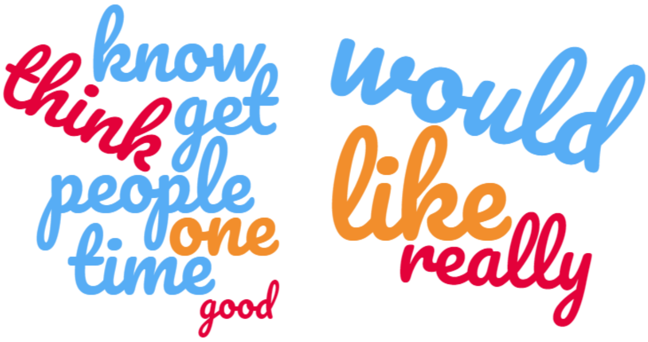


Figure 2. Summary of the most common words in the Reddit dataset.

A list of words, ordered by their count, was obtained after performing this analysis. Because it was a general approach, the most common words were, as expected, widely used in the English language. This approach can be further refined, but example, by choosing a set of subreddits and applying a word count on them. The word count would likely reflect the topic of the subreddit.

Scalability experiments

Scalability was evaluated using two approaches, namely:

- Strong scaling: How the execution time decreases as a function of increased resources (fixed problem size)
- Weak scaling: How the execution time behave as a function of increased resources, for an increasing problem size (i.e. fixed amount of work per processor)

We tested both strong and weak scalability in the whole analysis, as well as in critical steps that took most of the analysis time: loading the file and performing the word count. When using a file size of around 900MB, these steps take several minutes when all nodes are fully operational, whereas filtering steps take only a few seconds.

To test strong scalability, we used a fixed data size. The data size corresponded to the size of the file that we used for these experiments. We used a data size of around 900MB for these analyses and we changed the number of nodes from one to four, which is the total number of workers in our cluster.

Whole analysis

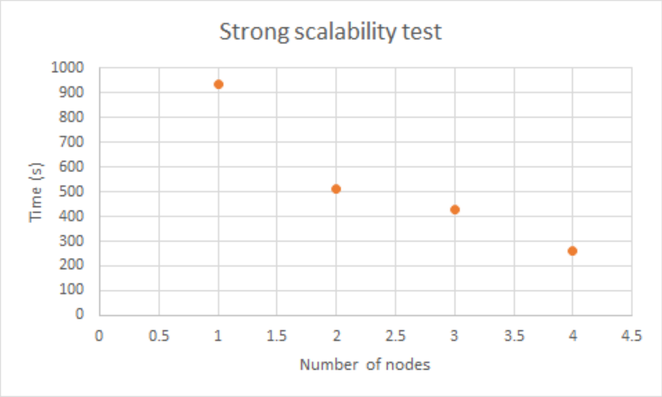


Figure 3. Testing strong scalability for the whole analysis

Wordcount

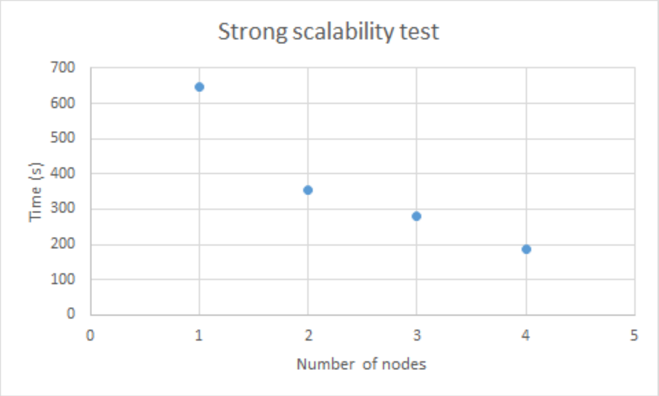


Figure 4. Testing strong scalability for the word count part

For both the whole analysis and the word count, we can see how the analysis time scales more or less linearly with the amount of nodes used; the more nodes, the less time it takes, with the complete analysis taking longer than the word count, as expected.

For the weak scalability test, we varied both the number of nodes and the file size proportionally. For four nodes, we used a file with a size around 800MB; for three nodes, around 600MB; for two, around 400MB and for one, 200MB. Because we kept the data size per number of nodes consistent, we should not see any variations in time, regardless of file size.

Whole analysis

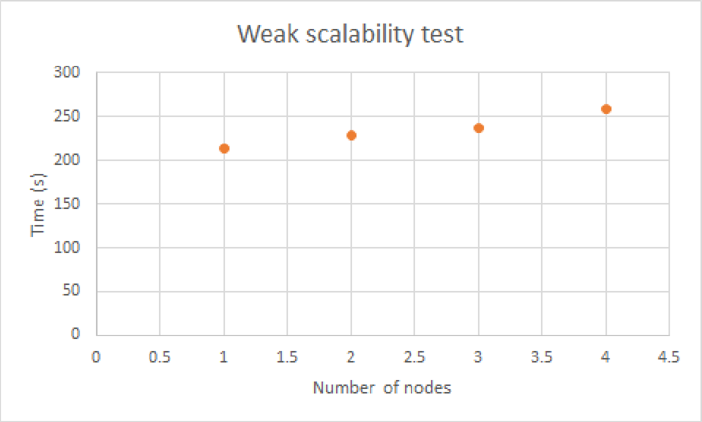


Figure 4. Testing weak scalability for the whole analysis

Wordcount

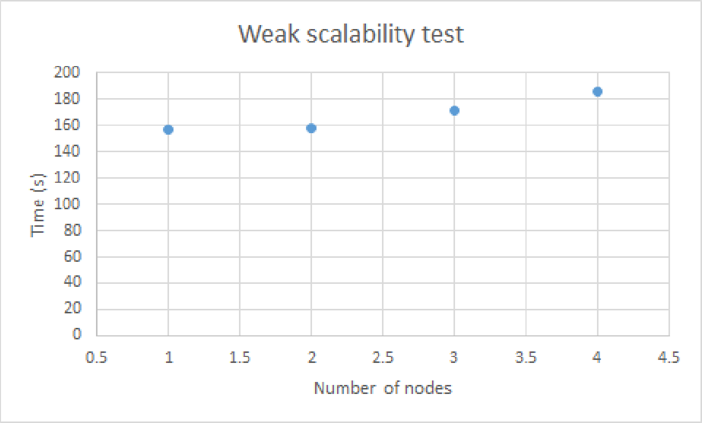
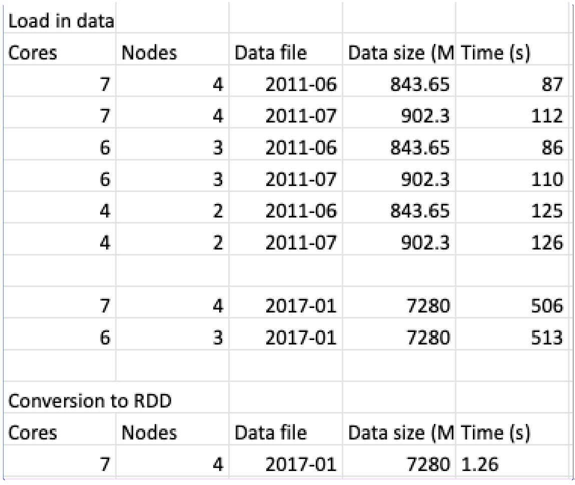


Figure 6. Testing weak scalability for the word count part

Although there are some variations, a slight increase in time when we use bigger file sizes, the result is close to being a constant value of time, unaffected by the amount of nodes or data size.

Scalability of loading data

The only step that took a significant amount of time other than the word count was loading data from the original JSON files. A strong scalability test was also performed for this step, where the time it took for different file sizes to be loaded was measured for different amounts of nodes. However, the amount of time did not vary proportionally to the number of nodes, unlike for the word count step. It was solely dependent on the size of the file.



Load in data				
Cores	Nodes	Data file	Data size (M	Time (s)
7	4	2011-06	843.65	87
7	4	2011-07	902.3	112
6	3	2011-06	843.65	86
6	3	2011-07	902.3	110
4	2	2011-06	843.65	125
4	2	2011-07	902.3	126
7	4	2017-01	7280	506
6	3	2017-01	7280	513
Conversion to RDD				
Cores	Nodes	Data file	Data size (M	Time (s)
7	4	2017-01	7280	1.26

Commented [10]: This would be better as an actual table rather than a screenshot, and it would be better not to include the conversion to RDD or include it separately

Commented [11]: I'll probably download the file to word document and then add tables when it's finished

Commented [12]: Ok!

Other possible bottlenecks, such as the conversion from dataframe to RDD, were considered, but this step goes fast even for the biggest file in HDFS of size 7280 MB. This means that this step, along with the rest of filtering steps before performing the word count, do not represent a bottleneck.

Fault tolerance

HDFS provides reliable storage of very large files across machines in a large cluster. Each file is stored as a sequence of blocks, all of the same size except from the last block. To increase the fault tolerance, the blocks of a file can be replicated, where the replication factor and block size can be configured per file or for all files in HDFS [9]. By default the replication factor is 3 and this can be changed in the hdfs-site.xml file in hadoop. However, we only used a replication factor of 1, since memory was a limiting factor before adding the extra volume. Increasing the replication factor increases fault tolerance, but this would also increase the memory usage on HDFS.

Limitations

The main limitation was the amount of resources. We could not deploy more VMs because the cloud reached the limit of resources. We therefore had to work with a small cluster, which prevented us from working with bigger files and from running the analysis on a bigger portion of the dataset or even on the whole dataset.

Discussion and conclusion

Commented [13]: Here you can discuss the outcome of the experiments and the experiences gained. Was your chosen approach suitable? What worked well and what could be improved?

Results from the strong scalability test, where the time decreased when the number of nodes was increased, suggests an effective scaling. This is further supported by the weak scalability tests, where time values remained constant. Small differences in this test could be due to the fact that the increase of file size was slightly bigger for 3 and 4 nodes. In addition, the last node to be added only has one core, whereas the rest have two, so, even though the increase in nodes is proportional to the increase in data size, this is not the case if we consider cores instead of nodes. This can also explain the small deviations from a completely flat plot.

Overall, these results indicate that the approach was suitable to study the dataset. In general terms, we could identify that the major bottleneck was running the MapReduce operation. Even though we ran a single MapReduce operation with our approach, they can be run consecutively as long as the cluster has sufficient resources (active nodes). It is important to control for MapReduce steps; however, running several filtering steps is not a problem. For example, filtering deleted posts or generating a list with all the words in the body of the messages. In addition, there are multiple Python libraries that simplify this part of the analysis.

Perhaps the greatest problem in the system was the fact that loading the file was an important time bottleneck, and, unlike the word count, it did not scale properly. Individual files can take more than 500 s (~8 min) to load. This is not such a big problem when dealing with individual files, but loading the whole dataset into Jupyter Notebook would be too time consuming. This means that the process of loading data should be optimized for the system to be used at larger scales.

Lastly, dealing with limited resources also restricted possible assays. For example, we could have devised an analysis consisting of several MapReduce steps, but taking into account the scalability of every step would have forced us to use smaller file sizes, since the size of our cluster was quite small and this single MapReduce step can take more than 10 min when we use less nodes to test strong scalability.

Link to Github repository with the code

https://github.com/mikaelaeriksson/LDSA_project.git

References

Commented [14]: Fix the format

[1] Wikipedia contributors. “Reddit” Wikipedia.org. https://en.wikipedia.org/wiki/Reddit (accessed Jun. 6, 2020)

[2] Baumgartner, J., et al. “The Pushshift Reddit dataset”, 2020. [Online]. Available: *arXiv preprint arXiv:2001.08435.*

[3] Szabo, G., Huberman, B.A., “Predicting the popularity of online content”, *Communications of the ACM,* vol. 53, no. 8, pp. 80–88, 2010.

[4] Tran, T and Ostendorf, M., “Characterizing the language of online communities and its relation to community reception”, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, p. 1030–1035, Nov. 2016.

[5] Gilbert, E., “Widespread underprovision on reddit, ”*CSCW ‘13: Proceedings of the 2013 conference on Computer supported cooperative work*, pp. 803–808, 2013.

[6] Gomez, V., Kaltenbrunner, A., and López, V., “Statistical analysis of the social network and discussion threads in slashdot” *Conference: Proceedings of the 17th international conference on World Wide Web,* pp. 645–654, 2008.

[7] Enterprise Big Data Framework, Jan. 2019. Available: https://www.bigdataframework.org/data-types-structured-vs-unstructured-data/] (accessed May. 26, 2020)

[8] Armbrust, M., Das, T., Davidson, A., et al., “Scaling spark in the real world: performance and usability” *Proceedings of the VLDB Endowment,* Aug. 2015. Available: https://doi-org.ezproxy.its.uu.se/10.14778/2824032.2824080

[9] Hadoop Documentation. Aug. 2019. Available: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html (accessed Jun. 4, 2020)