Data Processing of Reddit Comments - A Scalable Architecture

Project in Data Engineering I  
Vilmer Blystad, Nils Uwe Gegenmantel, Rajib Datta, Linus Westerberg & Jonatan Löfgren  
  
*Word count: 2498*

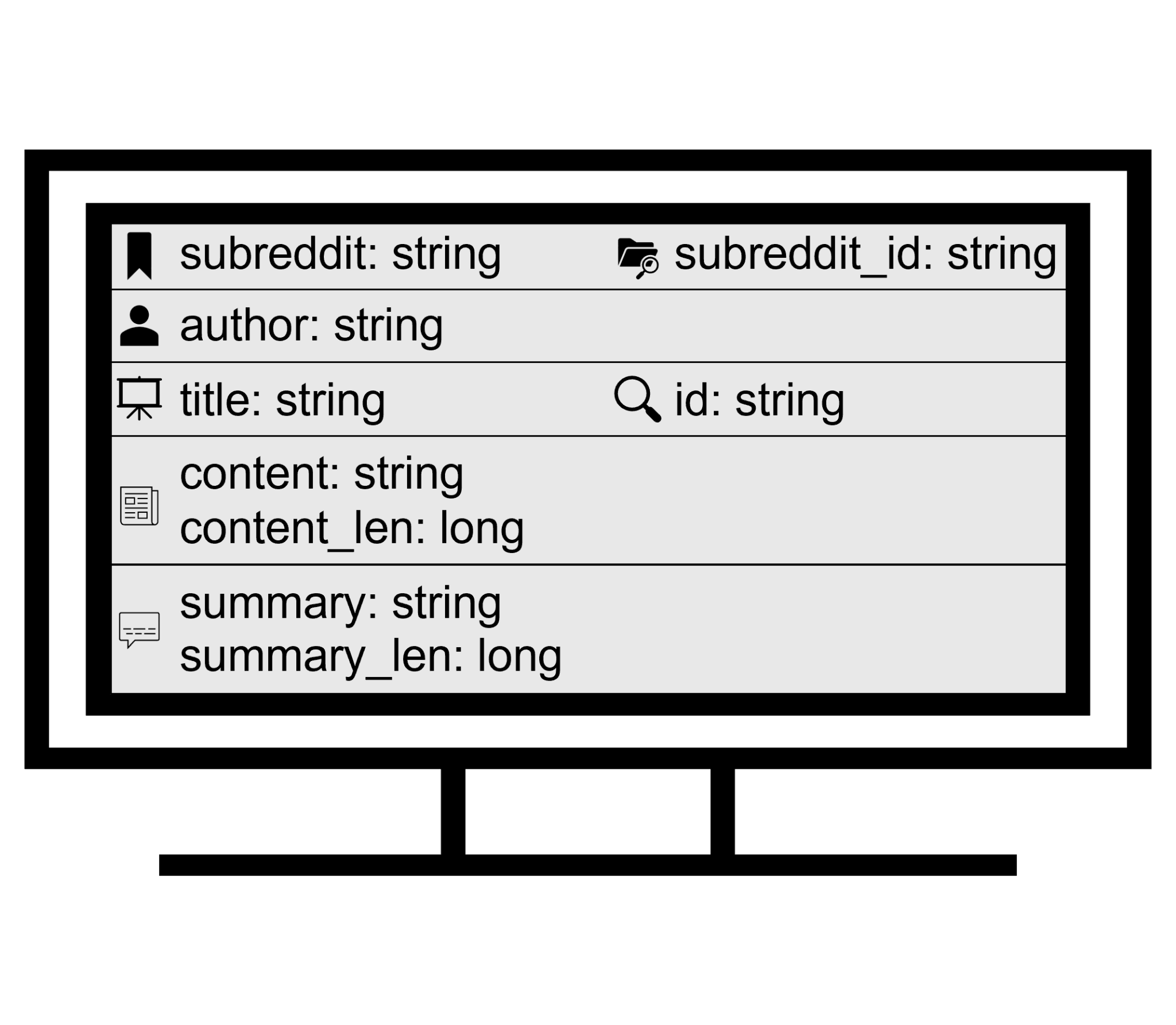
**Background**

The Webis-TLDR-17 Corpus dataset, created by Syed, Shahbaz; Völske, Michael; Potthast, Martin; Stein, Benno, consists of various pre-processed JSON objects (Syed et al., 2017). These represent posts and their summaries from the Reddit platform. This is also the reason for the name part "TLDR" of the data, which stands for "Too long did not read" (Knowles, 2023), and the category summary, which summarizes the content of the post. The authors of the dataset suggest using this summary category along with the content to train deep learning models (Völske et al., 2017). It is suggested because automatic summarization is very challenging (Völske et al., 2017) adapted from (Gambhir and Gupta, 2017). In the year of publication, the only other English datasets that allowed the training of summarization were news articles or newswire related, and therefore very biased and not user near (Völske et al., 2017). Combined, these sets only had about 4,201,120 training pairs. In relation to this, a new dataset with everyday topics and everyday language of nearly the same size with around 3.85 million key pairs was an impactful dataset for training summarization (Völske et al., 2017). To create this set, more than “286 million submissions and 1.6 billion comments posted to Reddit between 2006 and 2016” were used (Völske et al., 2017, p. 60). The source (Völske et al., 2017) describes the process of retrieving and processing the dataset and has been cited over 120 times. There are similar approaches such as the one by (Kim et al., 2019) with a smaller dataset which is of higher quality according to (Zhang et al., 2020). The further research in this area and the specific generation of further reddit-based TLDR datasets such as (Sotudeh et al., 2021) only confirms the importance of the present dataset. Until now, it has been used mostly for the intended analysis to train different models for summarizing text (Ouyang et al., 2022; Stiennon et al., 2020; Ziegler et al., 2019). The source (Björk & Svensson, 2018) also analyzes the dataset. Here, essential features such as the distribution of data across subreddits are analyzed.

Since the dataset has already been extensively utilized for its intended purpose and fundamental analyses have already been performed, the aim is now to use it for an entirely different application. As the dataset derives from a social media platform it can be used to provide an example of user behavior of such systems. A common phenomenon of social networks is the 1% Rule, also known as the 90-9-1 Principle. It means that 90% of social network users “observe and do not participate, 9% contribute sparingly, and 1% create the vast majority of new content” (Van Mierlo, 2014, p. 2). Whether this rule also applies to the excerpt from the reddit platform will be analyzed in the following. As the dataset only covers comments and top-level posts, statements on non-commenting users cannot be made. Instead it will be examined how comments/posts are distributed across users. This will be implemented as a type of batch processing job, where we aim to show horizontal scalability with the chosen processing solution.

**Data Format**

The dataset contains 3,848,330 posts (Syed et al., 2017). Each post is a JSON object with a specific schema displayed in Figure 1. The JSON format has many advantages: JSON is flexible for implementations, as it can be used with both NoSQL and relational database systems when integrated (Lv et al., 2019). This is a crucial advantage for the purpose of the dataset being used for training summarization, as it can be applied more widely and easily. The alternative to JSON, which is a rigid structured data format, cannot provide this flexibility. On the other hand a rigid structured format would ensure consistency. Another advantage of JSON is that it provides key-value pairs, allowing users to easily access specific keys with their respective values (Lv et al., 2019). This can be directly applied to easily obtain the summary and content values and train the summarization. All other information can be accessed if needed, but doesn't have to be.



*Figure 1: Structure of a reddit post of the dataset*

For the title and an id, one could argue that a number format would improve efficiencies such as data storage. On the other hand, the string allows for more id options, which makes sense since running out of post IDs would be a disaster. For the author a number format could have also been used, but a string allows for much more variety and is more user friendly. The number format would improve efficiency and compliance, since no user can choose a discriminating name.

# Computational Experiments

## Architecture of solution

### Data storage layer - Hadoop Distributed File System

The Hadoop Distributed File System is an industry-standard data storage solution for distributed systems. By dividing stored information into data blocks, and distributing the blocks across multiple disks called DataNodes, the amount of system resources can easily be scaled horizontally with additional DataNodes (Borthakur, 2013).

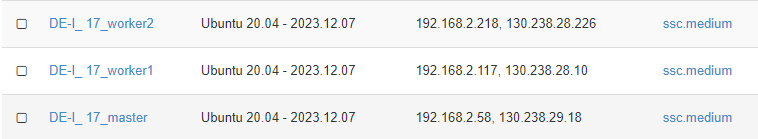
### Data processing layer - Apache Spark

Apache Spark moves data processing operations from disk to memory (Mavridis & Karatza, 2017). When connecting a Spark Cluster to HDFS, this means that Spark will perform a read operation moving data from disk to memory, perform the necessary operations on that data in-memory, and then write the results back to the cluster. In comparison, the MapReduce paradigm performs multiple read / write operations directly on disk for intermediate results, which increases the computing time. While memory typically is a more expensive resource than disk-capacity, Apache Spark can reduce operating costs by 40% while performing operations 4x quicker than MapReduce (Mavridis & Karatza, 2017).

Apache spark allows for batch-, stream- and graph processing of data, with multiple machine-learning frameworks, making it a more versatile solution than HDFS with MapReduce (Mavridis & Karatza, 2017). While the analysis task performed in this project might have been achievable and appropriately scalable using only HDFS and MapReduce, creating a solution using Apache Spark opens up for the possibility of performing real-time time analytics and a bigger array of future possible applications. Such a real-time application on our dataset consisting of reddit comments could for example be the filtering of comments breaking the terms-and-conditions, or the detection of spam accounts.

### Architecture

The HDFS + spark configuration was deployed across three virtual machines in the SNIC cloud. One instance was started as the cluster master, and hosted both the apache spark master and the Hadoop files system. This created a simple point-of-contact for the user, with a single point of access for both the storage and computing clusters.



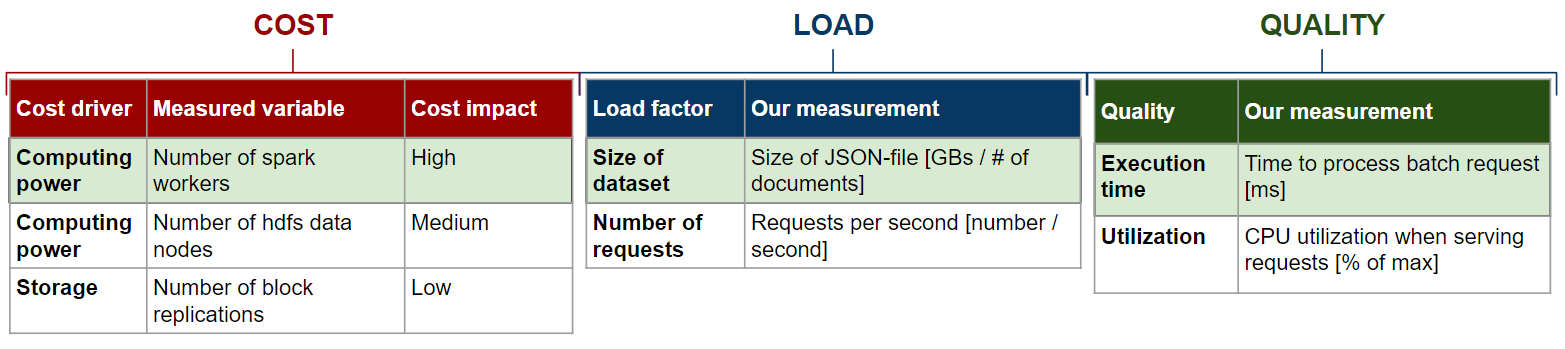
## Design of scalability experiments

### Analyzing the scalability of a distributed system

When performing scalability analysis of distributed systems, the overall cost of operating the system should be measured against relevant performance metrics for the application that the system runs (Duboc et al., 2007). This means that different performance and utilization variables might have different relevance depending on the project / application at hand. Here we perform the creation of a scalability evaluation framework according to the steps of Duboc et al. (2007):

1. *Defining the systems scalability goals*In a real-world, non-static application continually expanded with additional entries of summarized reddit comments in the form of JSON documents, scalability implies being able to cost-efficiently be able to scale the system to store and perform operations on the documents. In this case, this means performing filter and value aggregation operations over an increasingly large dataset in Apache Spark.
2. *Define relevant scaling dimensions*  
   The HDFS + Apache Spark architecture has two main scaling variables that increase the amount of resources in their corresponding cluster. These are defined as our main test “scaling variables'' that can be increased in response to an increasing size of the reddit comment dataset. The *dependent variables* are defined as the performance dimensions that we expect to be impacted by the adding of additional resources. For this application, that means the processing time of requests and the average utilization of resources when performing those requests. **Scaling variables:**Number of HDFS datanodes, number of Apache Spark cluster workers, number of JSON documents.   
   **Dependent variables:** Processing response time, memory usage (utilization), disk usage (utilization).   
   **Nuisance variables:** CPU clock speed, memory speed, bandwidth

Of these variables, the variable with the highest impact on cost is the number of apache spark workers that need both a CPU thread, and a suitable amount of memory to load the data into. We therefore chose the number of spark variables as our primary cost variable. The load variable available to us in this batch operation is the size of the data set, and the most relevant quality variable for batch jobs is the execution time.



*Figure 1: Cost, load, and quality variables. Chosen variables highlighted in green. From our project presentation.*

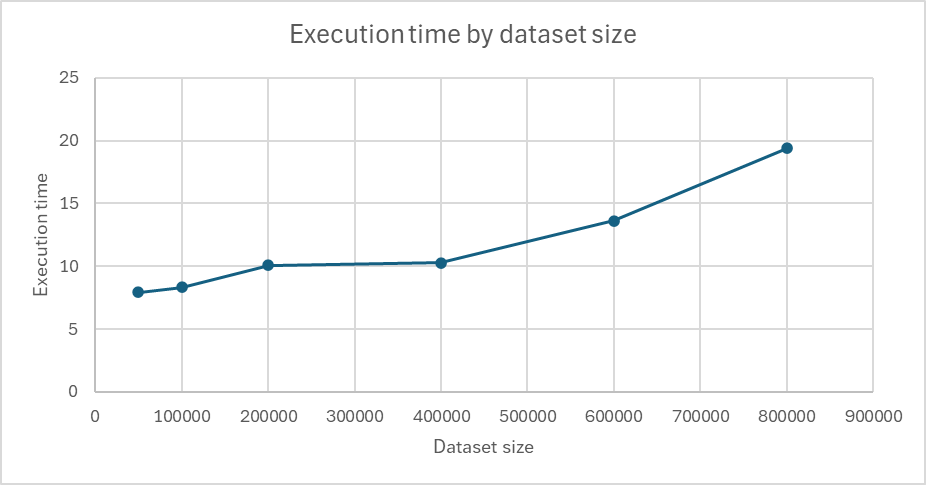
The experiments were then designed to run relevant PySpark operations used in our data analysis task (groupBy, filter, orderBy, and count) across the dimensions:

**Number of Apache Spark cores:** 1-4

**Size of dataset:** 0.1 - 4 GB

### Basic scaling behavior

We carried through a first test looking at the behavior of the application as the size of the problem increased, tracking the execution time for four cores. The hypothesis is that the relationship between execution time and problem size is directly proportional, with a doubling of problem size resulting in twice the execution time.



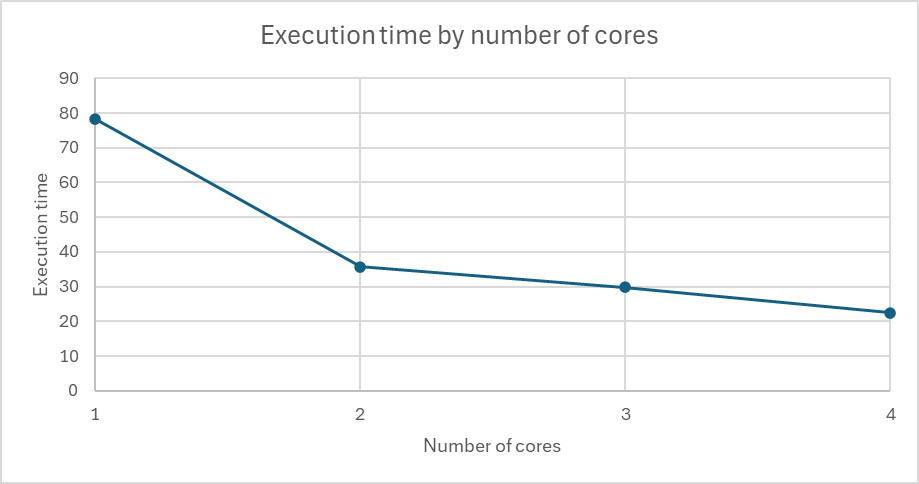
| **Dataset size** | **Execution time** |
| --- | --- |
| 800,000 | 19.4156 |
| 600,000 | 13.6415 |
| 400,000 | 10.298 |
| 200,000 | 10.0789 |
| 100,000 | 8.345 |
| 50,000 | 7.9496 |

Our experiments show that the relationship isn’t that simple, with a plateau in execution time from around 50,000-400,000 where the execution time stays around 10 seconds, only to increase as the number of documents increases to size 600k, and 800k.

This could be explained by fixed “time costs” in executing the processing job, where the time to initiate computations and start all relevant processes exceeds the time it takes to do the actual spark operations for the smaller datasets. Another factor that could explain this behavior is that the smaller problem sizes mean that the parallel processing potential of 4 cores is not being utilized.

### Strong scaling experiments

The second test used to evaluate the scalability of the cluster is a strong scaling experiment, where the size of the data set is kept constant while the number of workers is increased. We used the entirety of the loaded reddit comment dataset in the HDFS, totalling ~960,000 documents. Here we expect the opposite result to the test in task 1, with the execution time approximately halving with each doubling of number of cores.



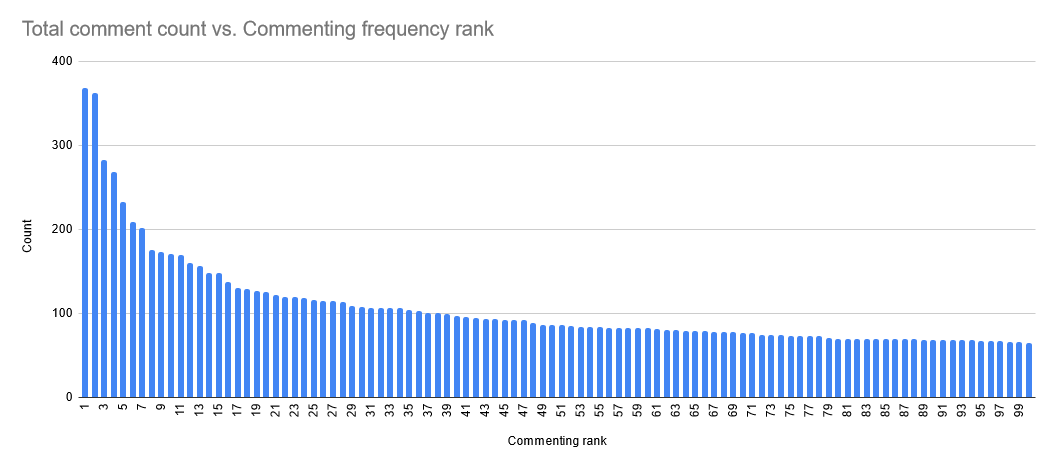
| **Cores** | **Execution time** | **Factor improv.** |
| --- | --- | --- |
| 4 | 22.406 | 3.49 |
| 3 | 29.739 | 2.63 |
| 2 | 35.652 | 2.14 |
| 1 | 78.281 | 1 |

In practice we see that while the improvement from 1 to 2 cores is 2,63 in execution time, this starts to plateau somewhat for 4 cores, with an improvement of only 3,49x compared to the expected 4x. This could potentially be due to the behavior seen in the “basic scaling experiment”, with core utilization or fixed time costs skewing the results for lower counts of documents / executors. Because of the limited size of the dataset, these results make it difficult to assess the weak scaling properties of the solution.

# Results of data analysis

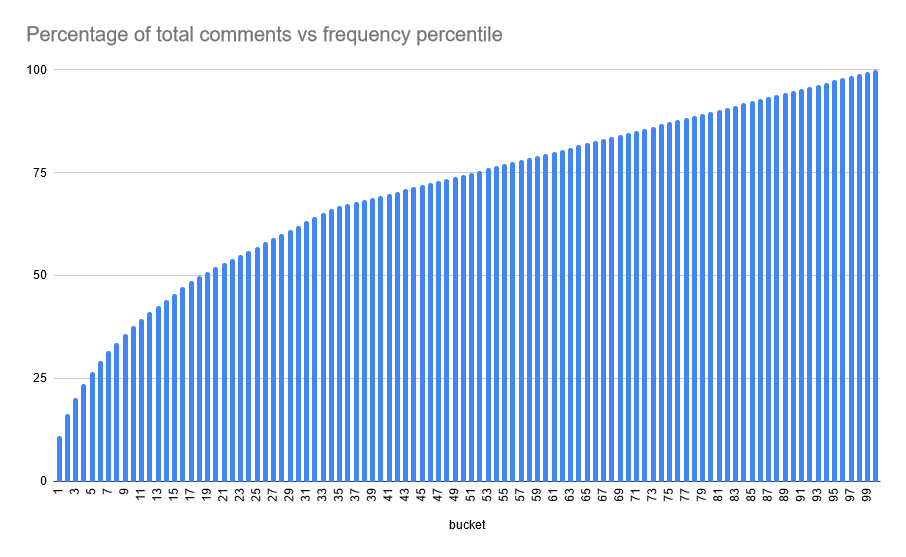
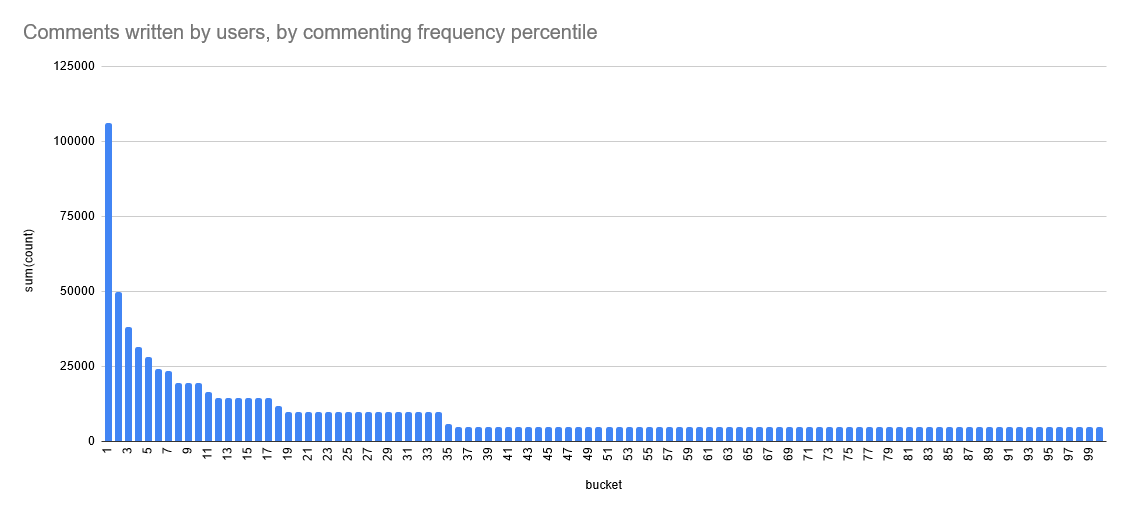
### Participation across Reddit

To investigate participation across all subreddits, the comments were aggregated by username. Taking a look at the top 100 users reveals a sharp dropoff from the top even in these very unusual users:



The top user has 368 total comments, while the 100th has just 65; less than 1/10 of the top commenter.

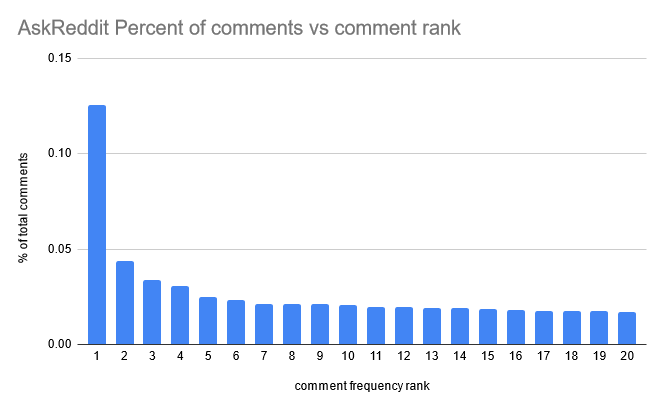
Next, the users were divided into essentially evenly sized buckets (differing by at most 1 user) of ~4820 users, with the top 0-1% most frequent commenters in bucket 1, the top 1-2% most frequent commenters in bucket 2 and so on. Aggregating all comments from each bucket revealed strong participation inequality.



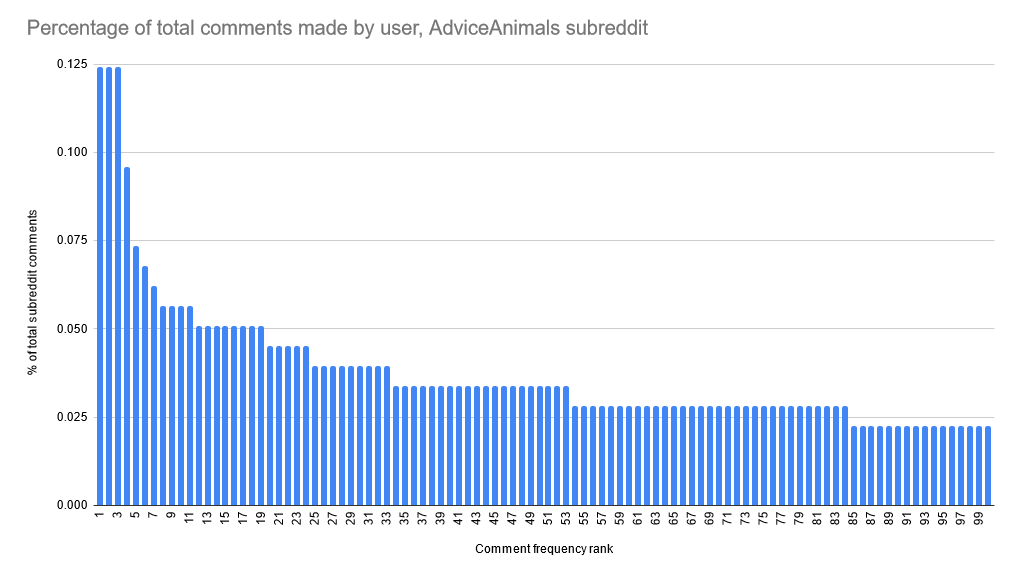
We see that the top 1% of commenters write 11% of comments, the top 10% write 37%, and about 65% of users only have a single comment. However, the 80/20 rule does not quite hold: the top 20% write only 52% of all comments. We could see this as an artifact of not being able to write fractional comments, which is also why the graph turns linear after a certain point. Nevertheless, it is clear that most comments on Reddit are from relatively few users, even among users that comment.

### Participation in subreddits

Looking at comment frequency in individual subreddits, we aggregate by subreddit and user.



In the AskReddit subreddit, the top user has ~0.13% of all comments at 277. The 20th most frequent commenter has ~0.02% at 37 comments.



In the relatively high-traffic AdviceAnimals and AskReddit subreddits, any particular user is not very influential. However, the last data we have shows that the top user in the Android subreddit had 0.9% of all 2534 comments in that subreddit at 0.9%, with the second most frequent commenter in that subreddit having 0.5%.The results are largely the same as for Reddit in general. We did find some strong differences between just how dominant the top users were depending on the subreddit, with the top user ranging from ~0.1% to ~0.9% of all comments.

# Discussion and conclusion

In this project a data processing solution with HDFS and Apache Spark was created. The spark cluster consisted of two workers, each with 2 cores. To test the solution a dataset containing ~1,000,000 reddit comments was loaded into HDFS, and timed the execution time for common spark methods over varying data sizes and compute amounts.

The scalability approach, although successful, could have been more extensive. Here one could have opted to use more worker instances with more cores and more memory, combined with a larger dataset in HDFS. This would then have enabled the analysis of a larger dataset and a more nuanced analysis of scalability, which now has some question marks surrounding if the dataset was appropriately large for all cores and workers to be utilized during the experiments.

Another takeaway from this project is the importance of setting up automated procedures for creating instances, forwarding ports and installing application requirements when working in larger groups of data engineers. As all of the steps in setting up the cluster were not documented and directly reproducible, setting up the cluster additional times would be equally time-consuming. Automated procedures also make sure that there is consistency and reproducibility in cluster behavior, which could potentially be a problem. Nevertheless, the proposed solution has been shown to be scalable for the task at hand, and the key objectives of the project have been achieved.

As for our actual data analysis task, we have seen that while the distribution of reddit commenters does not quite follow the “80/20” or “1% rule”, there is a large overrepresentation of comments made by super users.

# References

**Github repo** : The private repository has been shared with TA Tianru Zhang, if additional people need access please reach out to Vilmer Blystad at [vilmer.blystad.7412@student.uu.se](mailto:vilmer.blystad.7412@student.uu.se).

Borthakur, D. (2013). *HDFS Architecture Guide.* <https://hadoop.apache.org/docs/r1.2.1/hdfs_design.pdf>

Duboc, L., Rosenblum, D., & Wicks, T. (2007). A framework for characterization and analysis of software system scalability. *Foundations of Software Engineering: Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering; 03-07 Sept. 2007*, 375–384. <https://doi.org/10.1145/1287624.1287679>

Jogalekar, P., & Woodside, M. (2000). Evaluating the scalability of distributed systems. *IEEE Transactions on Parallel and Distributed Systems, 11*(6), 589–603. <https://doi.org/10.1109/71.862209>

Mavridis, I., & Karatza, H. (2017). Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark. *The Journal of Systems and Software, 125,* 133–151. https://doi.org/10.1016/j.jss.2016.11.037

Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010, May). The hadoop distributed file system. *In 2010 IEEE 26th symposium on mass storage systems and technologies (MSST)* (pp. 1-10). Ieee.

Björk, J., & Svensson, K. (2018). Abstractive Document Summarization Using Generative  
 Adversarial Networks. Department of Mathematical Sciences, Chalmers University of Technology.

Kim, B., Kim, H. & Kim, G. (2019). Abstractive Summarization of Reddit Posts with Multi-level Memory Networks. In J. Burstein, C. Doran & T. Solorio (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 2519–2531). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1260

Knowles, S. (2023). Shining through the smog: how to tell powerful, purposeful stories about sensory science in a world of digital overload. In *Digital Sensory Science* (pp. 271–283). Elsevier. https://doi.org/10.1016/B978-0-323-95225-5.00018-3

Lv, T., Yan, P. & He, W. (2019). On Massive JSON Data Model and Schema. *Journal of Physics: Conference Series*, *1302*(2), 022031. https://doi.org/10.1088/1742-6596/1302/2/022031

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L. E., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J. & Lowe, R. J. (2022). Training language models to follow instructions with human feedback. *ArXiv*, *abs/2203.02155*. https://api.semanticscholar.org/CorpusID:246426909

Sotudeh, S., Deilamsalehy, H., Dernoncourt, F. & Goharian, N. (2021). *TLDR9+: A Large Scale Resource for Extreme Summarization of Social Media Posts*. https://doi.org/10.48550/ARXIV.2110.01159

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D. M., Lowe, R. J., Voss, C., Radford, A., Amodei, D. & Christiano, P. (2020). Learning to summarize from human feedback. *ArXiv*, *abs/2009.01325*. https://api.semanticscholar.org/CorpusID:221665105

Syed, S., Völske, M., Potthast, M. & Stein, B. (2017). *Webis-TLDR-17 Corpus* [Data set]. Zenodo. <https://doi.org/10.5281/ZENODO.1043504>

Van Mierlo, T. (2014). The 1% Rule in Four Digital Health Social Networks: An Observational Study. *Journal of Medical Internet Research*, *16*(2), e33. https://doi.org/10.2196/jmir.2966

Völske, M., Potthast, M., Syed, S. & Stein, B. (2017). TL;DR: Mining Reddit to Learn Automatic Summarization. *Proceedings of the Workshop on New Frontiers in Summarization*, 59–63. https://doi.org/10.18653/v1/W17-4508

Zhang, J., Zhao, Y., Saleh, M. & Liu, P. J. (2020). *PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization* (arXiv:1912.08777). arXiv. http://arxiv.org/abs/1912.08777

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P. & Irving, G. (2019). *Fine-Tuning Language Models from Human Preferences*. <https://doi.org/10.48550/ARXIV.1909.08593>

# Contribution statement

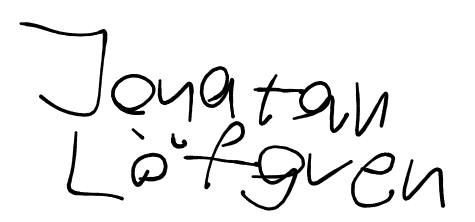
**Nils Uwe Gegenmantel**: Wrote the sections Background, Data Format and parts of the Discussion and Conclusion. Also helped initialize the spark cluster and set up and performed a first analysis of performance on the dataset. Also produced the corresponding parts for the presentation.

*Signature*

**

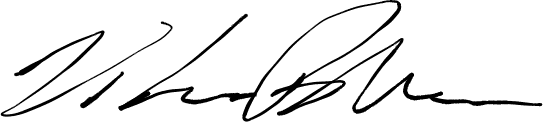
**Jonatan Löfgren**: Formulated and implemented the question and analysis of the question, sketched part of the Introduction (moved from Results section of report), wrote “Results of data analysis” section, wrote part of the Discussion and Conclusion. Helped set up group communication.

*Signature*

**

**Vilmer Blystad:** Wrote report section “Computational Experiments”, and parts of Discussion & Conclusion. Organized group meetings, created agendas, and to-dos for the group members.

*Signature*



**Rajib Datta:** Practical setup of spark cluster, wrote cluster setup instructions available in Github repo under “setup.md”.

*Signature*

**

**Linus Westerberg:** Practical setup of spark cluster

*Signature*

