The Scalability of Spark Cluster

Haodong Zhao, Somiya Khurram, Halit Ege Ebiller

1. Background

Reddit is a social platform where everyone can post, comment or create content. For every post that has been created, every user can “vote” or “grade” the content. The purpose of this project is to develop a horizontally scalable data processing solution to do sentiment analysis : classify Reddit comments as “negative”, “neutral”, or “positive”, by using Natural Language Toolkit features. Pipeline has been divided into three steps as:

- Inputting Raw Data
- Processing & Classification
- Final results

and it will be thoroughly explained in the upcoming sections.

Two data sets have been used in this experiment, both containing user comments from 2011 and 2012 respectively. The data sets have been saved to HDFS since it performs better than local file systems, by saving the data into the chosen number of worker nodes as replications. While doing this experiment, it was not necessary to train a deep learning model so classifications have been made using NLTK. By the usage of MapReduce, the comments have been counted under their classes.

This report has been divided to four sections after the background section, these sections are:

- Data Format
- Computational Experiments
- Discussion and Conclusion
- References

Summarized process will be explained thoroughly in each relative section.

2. Data Format

Reddit [1]is a social news website, where the users, called "Redditors", post links to online content or create text postings themselves. This content often consists of news, but also of links to videos, images, blog posts and other material. For every submitted link, Reddit users can vote on how important or relevant they find the associated content. They can give positive(up-vote) and negative votes(down-vote). Based on the votes and using a metric, the postings are ranked. This ranking is used to determine the display position of the postings on Reddit front-page(s) in the default setting. All postings on Reddit are categorized by the poster into whatever category they feel the material belongs to and is called "subreddit".

We chose the Reddit Comment data which is available in **JSON** format and is **zipped.** Two datasets “RC_2011-07” and “RC_2012-12” from https://files.pushshift.io/reddit/comments/ are selected. These datasets contain the comments created by the users in December 2012 and July 2011. The “RC_2011-07” is about 0.95 GB in bz2 zip format. “RC_2012-12” is about 2.4GB in bz2 zip format. After unzipping the datasets, we obtained JSON objects.

JSON (JavaScript Object Notation) is an open standard file format for sharing data that uses human-readable text to store and transmit data. The following enabling features were considered while choosing the datasets with JSON format.

- Simple classification is possible using NLTK(Natural Language Toolkit).
- JSON requires less formatting.
- Large file size
- Integrates with APIs easily and allows scalability
- It is easy for machines to parse and generate.

Pros of using JSON dataset format

- Fast and efficient
- Responsive
- Key/Value pair approach
- Data sharing
- Extensive browser support

Cons of using JSON dataset format

- Less secure
- No date data type
- Verbose

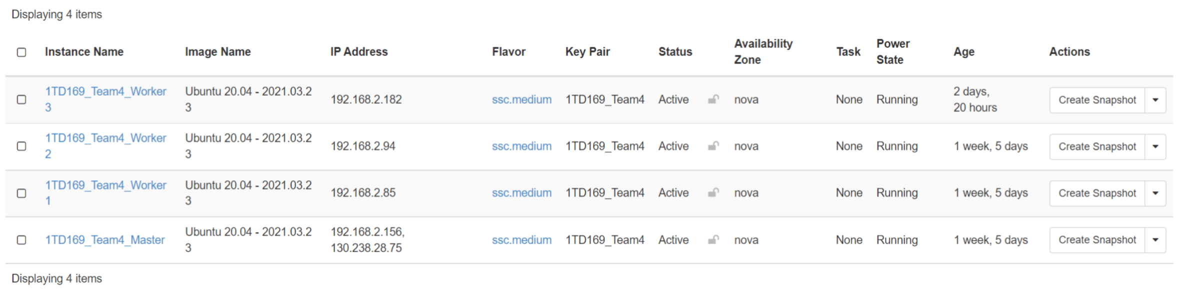
3. Computational Experiments

In this section, we describe the structure of our cluster where we do the experiments. After that, we discuss the experiments designed by us to illustrate scalability. At the end of this part, we show the results we get in the experiments.

3.1 The Introduction to the Cluster

3.1.1 The Structure of the Cluster

There are 4 virtual machines in our cluster, and we make one node as the master node, the other 3 nodes as worker nodes. We create snapshot images for our nodes, so it's easy for us to add more nodes into our cluster. The VMs chosen by us are all "ssc.medium" flavor which has 2 VCPUs, 4GB RAM, and 20GB disk size, as shown in the screenshot below.

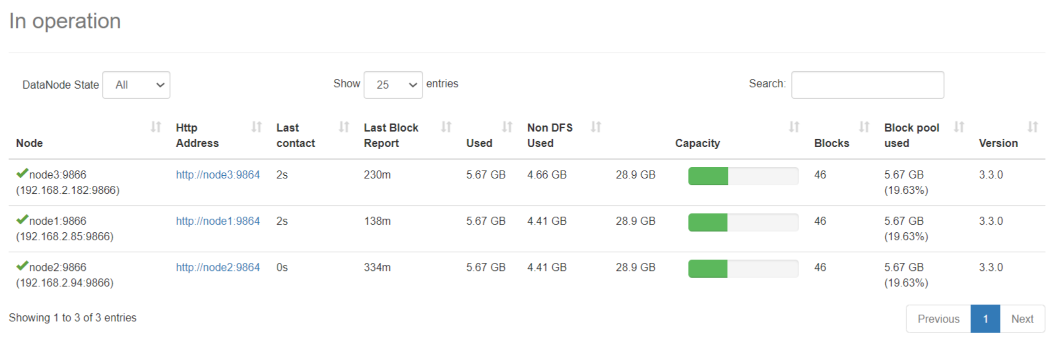


Displaying 4 items

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
1TD169_Team4_Worker 3	Ubuntu 20.04 - 2021.03.2 3	192.168.2.182	ssc.medium	1TD169_Team4	Active	nova	None	Running	2 days, 20 hours	Create Snapshot
1TD169_Team4_Worker 2	Ubuntu 20.04 - 2021.03.2 3	192.168.2.94	ssc.medium	1TD169_Team4	Active	nova	None	Running	1 week, 5 days	Create Snapshot
1TD169_Team4_Worker 1	Ubuntu 20.04 - 2021.03.2 3	192.168.2.85	ssc.medium	1TD169_Team4	Active	nova	None	Running	1 week, 5 days	Create Snapshot
1TD169_Team4_Master	Ubuntu 20.04 - 2021.03.2 3	192.168.2.156, 130.238.28.75	ssc.medium	1TD169_Team4	Active	nova	None	Running	1 week, 5 days	Create Snapshot

Displaying 4 items

We save the Reddit Comment dataset in HDFS(Hadoop Distributed File System). HDFS provides high throughput access to our application data. Unlike simple local file systems, HDFS chunks the data file into blocks and saves the chunks into different DataNodes and it also replicates data into different data nodes, therefore, it's highly fault-tolerant(3 replicas for data by default) and reliable. In HDFS, as shown in the screenshot below, we just save data into the 3 worker nodes, since we have to download the huge dataset and put the data into HDFS on the master node. These operations on the master node need a large amount of storage space.



In operation

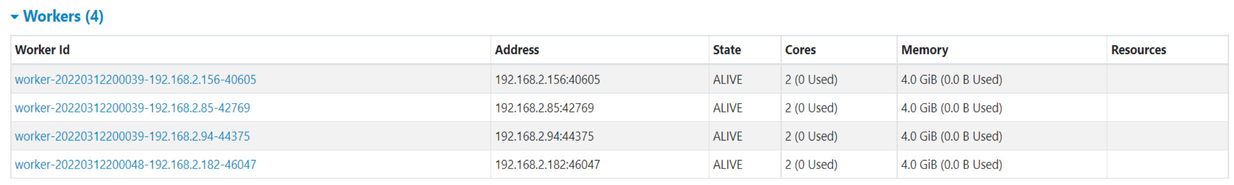
DataNode State All · Show 25 entries · Search:

Node	Http Address	Last contact	Last Block Report	Used	Non DFS Used	Capacity	Blocks	Block pool used	Version
✓node3:9866 (192.168.2.182:9866)	http://node3:9864	2s	230m	5.67 GB	4.66 GB	28.9 GB	46	5.67 GB (19.63%)	3.3.0
✓node1:9866 (192.168.2.85:9866)	http://node1:9864	2s	138m	5.67 GB	4.41 GB	28.9 GB	46	5.67 GB (19.63%)	3.3.0
✓node2:9866 (192.168.2.94:9866)	http://node2:9864	0s	334m	5.67 GB	4.41 GB	28.9 GB	46	5.67 GB (19.63%)	3.3.0

Showing 1 to 3 of 3 entries

Previous 1 Next

Apache Spark is built on the top of the Hadoop MapReduce module and it’s a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters. It was originally developed at UC Berkeley's AMPLab, later donated to the Apache. We make all the nodes able to do the jobs together, not just the worker nodes, so we have 4 workers. To sum up, as shown in the screenshot below, the Spark cluster has 8 cores and 16GB of Memory in total.



▾ Workers (4)

Worker Id	Address	State	Cores	Memory	Resources
worker-20220312200039-192.168.2.156-40605	192.168.2.156:40605	ALIVE	2 (0 Used)	4.0 GiB (0.0 B Used)	
worker-20220312200039-192.168.2.85-42769	192.168.2.85:42769	ALIVE	2 (0 Used)	4.0 GiB (0.0 B Used)	
worker-20220312200039-192.168.2.94-44375	192.168.2.94:44375	ALIVE	2 (0 Used)	4.0 GiB (0.0 B Used)	
worker-20220312200048-192.168.2.182-46047	192.168.2.182:46047	ALIVE	2 (0 Used)	4.0 GiB (0.0 B Used)	

Overall, the Hadoop cluster serves as the data source, we save the data distributedly in the cluster with the help of HDFS. And Spark is going to read data from HDFS, do some computations on the worker nodes, save or show the results at last.

3.1.2 Motivation

Unlike Hadoop, Spark provides memory-based computing, which can dramatically boost iteration efficiency. RDDs, which stand for Resilient Distributed Dataset, are the critical blocks of Spark applications. Spark not only provides a MapReduce algorithm, but it also extends the MapReduce model and provides more types of efficient and low-latency computations. Apart from that, the unified engine also supports SQL-like queries, streaming data, machine learning (ML), and graph processing. Hence, the cluster built by us is suitable to do the scalability experiments.

3.2 Scalability Experiments

3.2.1 Details of the Reddit Comments Datasets

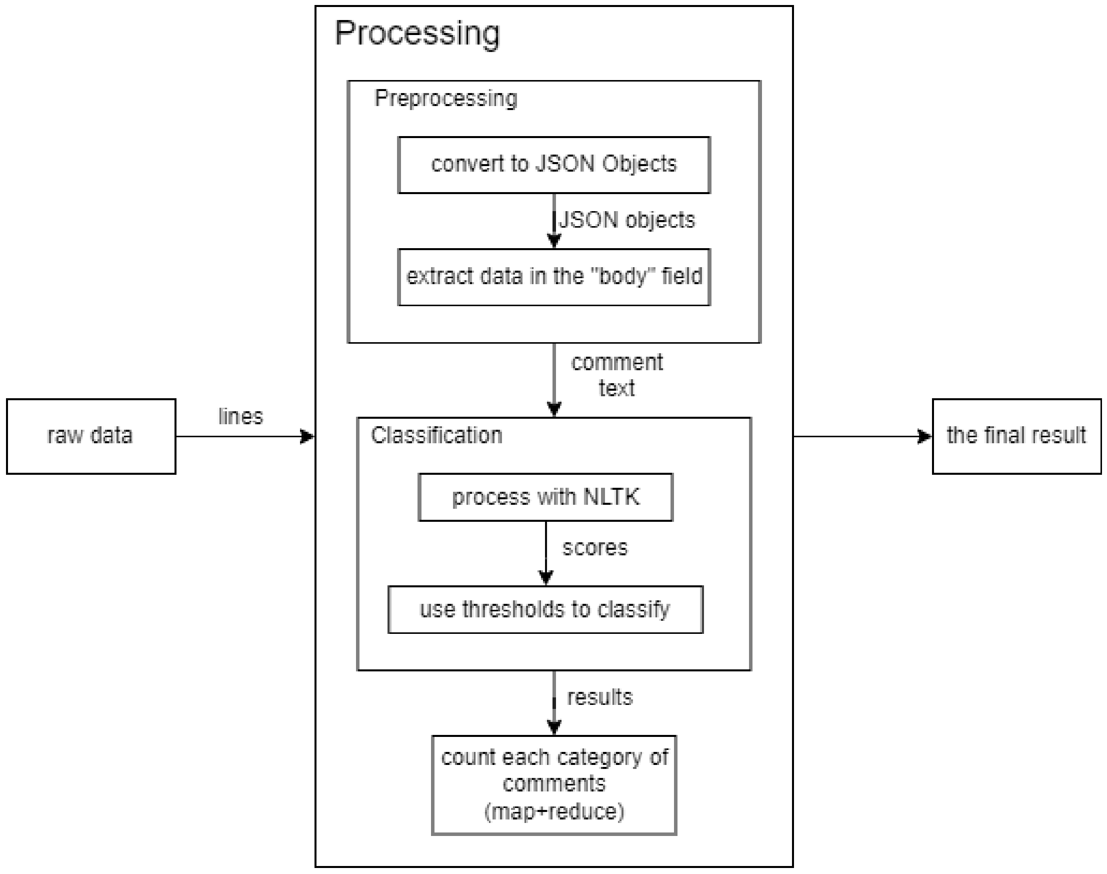
Dataset	Size (Before unzipping)	Size (After unzipping)	The Number of JSON Objects
RC_2011-07	0.95 GB	5.63 GB	10,557,466
RC_2012-12	2.4 GB	13.95 GB	26,080,276

The datasets used by us in the experiments are “RC_2011-07” and “RC_2012-12” from https://files.pushshift.io/reddit/comments/, which contains the comments created by the users of Reddit in December 2012 and July 2011.

“RC_2011-07” is about 0.95 GB in bz2 zip format. After unzipping, “RC_2011-07” is about 5.63 GB and has 26080276 JSON objects.
“RC_2012-12” is about 2.4GB in bz2 zip format. After unzipping, “RC_2012-12” is about 13.95 GB and has 26080276 JSON objects.
The datasets are huge enough for us to do the experiments. The details of the datasets are shown in the table above.

3.2.2 Sentiment Analysis Pipeline

We mainly use Pyspark to do the experiments. We design one task and try to solve it with different scales of resources. We would like to do some sentiment analysis on user comments which is classifying the comments as positive or negative, to explore scalability.



The pipeline to solve this task consists of 3 steps as shown in the figure above.

1. The preprocessing.
   Load the dataset, convert the data into JSON objects, and extract the “body” field;

2. Classify the comments.
   In this step, we do not train any Deep Learning models but just use NLTK(Natural Language Toolkit) to do some simple classifications, since the key point of this project is not about NLP.

   NLTK can make classifications and return a python dictionary of different scores, in the form like {'neg': 0.0, 'neu': 1.0 , 'pos': 0.705, 'compound': 0}. “Neg” represents negative; “neu” represents neutral; “pos” represents positive; “compound” represents compound. Each comment would have 4 score values. We can know the category of one comment from the value of “compound”.

   The compound score is computed by adding the scores of each word in the sentence. And then adjust some rules according to some specified heuristics, and finally normalize the value to make it between -1 and +1. -1 represents the most negative, and 1 represents the most positive.

   The thresholds we use to classify sentences as either positive, neutral, or negative are:
   - positive: compound value >= 0.05;
   - neutral: -0.05 < compound value < 0.05;
   - negative: compound value <= -0.05;

3. Use the idea of MapReduce to count the numbers of the positive, negative, and neutral comments.

3.2.3 The Designs of the Computational Experiments

In this part, our designs of experiments are to explore Horizontal Scaling. We enhance the performance of the cluster by adding(scale-out) or removing(scale-in) instances of the cluster.
In order to show the scalability, we design the experiments in 2 different aspects, which are strong scaling and weak Scaling.

1. Strong Scaling
   In this part, firstly we select all the JSON objects from the “RC_2011-07” dataset. And then, we design 3 groups of experiments, the groups need to process 250 thousand, 500 thousand, and 1 million JSON Objects separately. In each experiment group, we use 1, 2, and 4 worker nodes with 3GB Memory per executor process to do computations on a fixed size of data. After that, we can get the different execution times, so that we can observe the changes in the execution times.

2. Weak Scaling

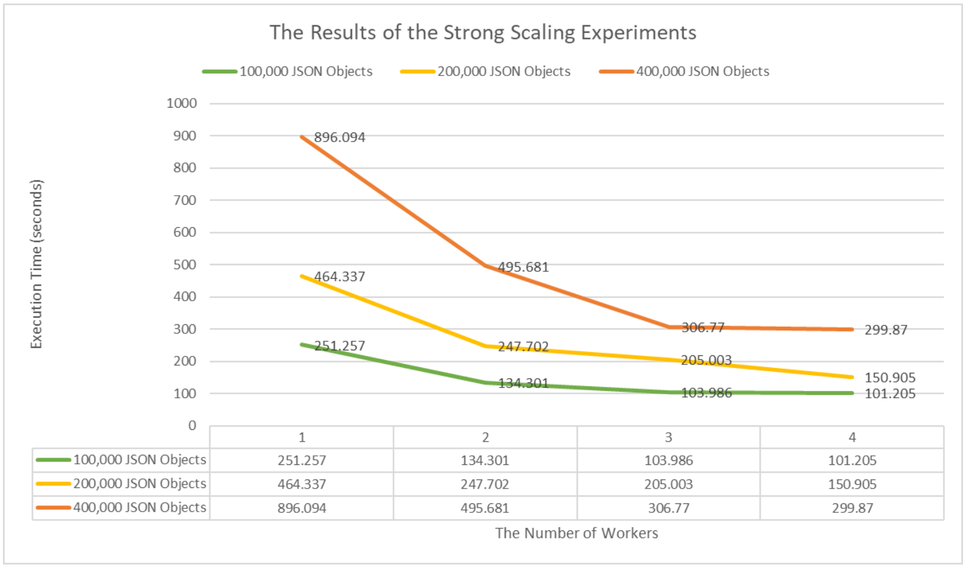
   In this part, we have an increased dataset size, but we set a fixed problem size per node. We use 1, 2, 3, and 4 worker nodes to process the selected data. After that, we can get the different execution times of 3 experiment groups with different sizes of data per processor, so that we can observe the changes in the execution times.

   We design 3 groups of experiments, each node in group 1 needs to take responsibility for 100 thousand JSON objects from the "RC_2012-12" dataset; each node needs to do computations on 200 thousand JSON objects; each node in group 3 needs to process 400 thousand JSON objects.

3.3 Results

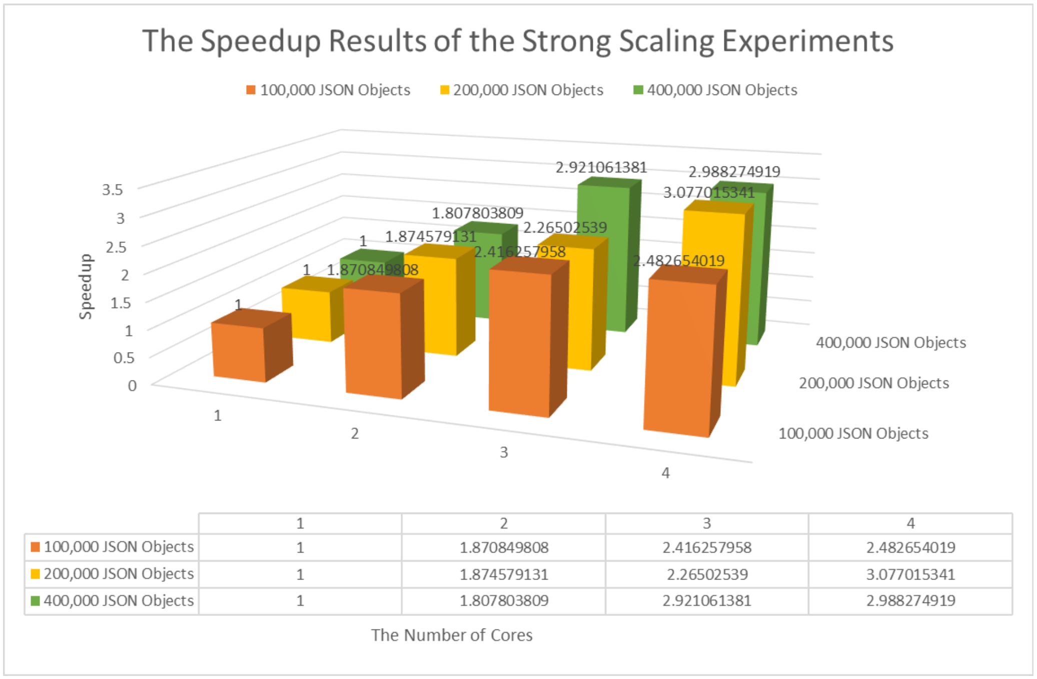
3.3.1 Strong Scaling

As is shown in the figures below, in each experiment group, by allocating 3GB of memory to each executor process.
In all the 3 groups, the runtimes for processing a fixed size of data all decrease dramatically as the number of workers(also cores) increases.



	1	2	3	4
100,000 JSON Objects	251.257	134.301	103.986	101.205
200,000 JSON Objects	464.337	247.702	205.003	150.905
400,000 JSON Objects	896.094	495.681	306.77	299.87

The speedup curves, as shown in the figure below, clearly illustrate the common trend.

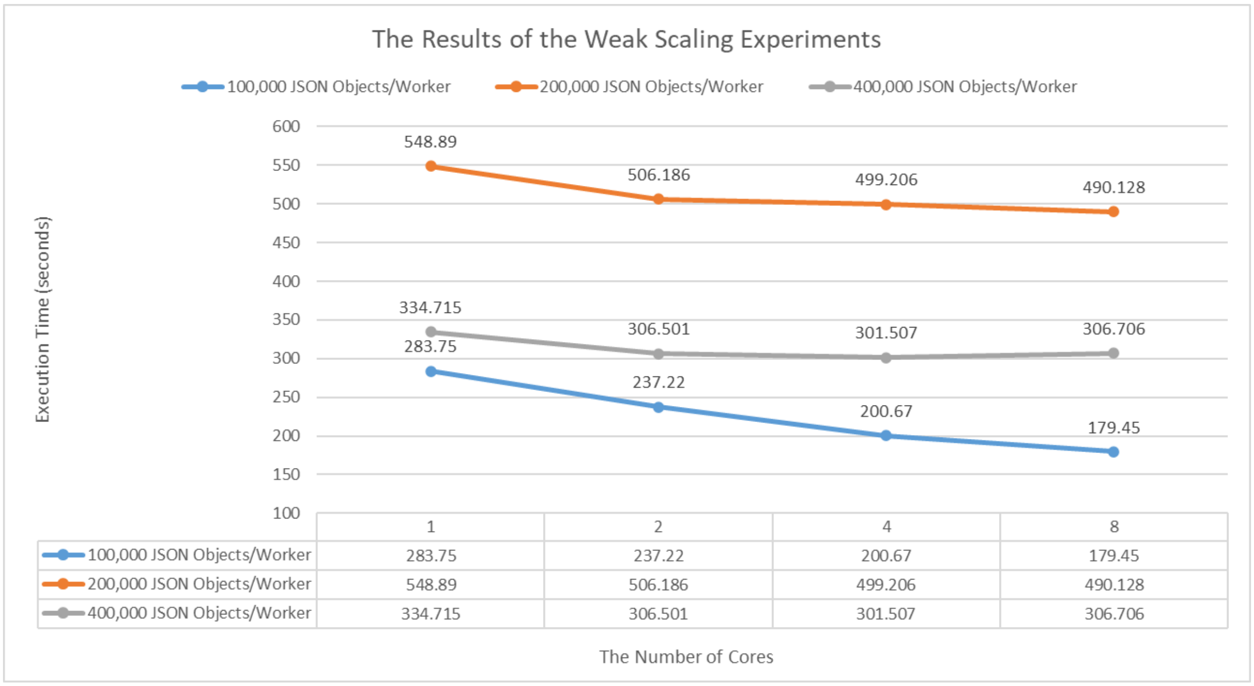


	1	2	3	4
100,000 JSON Objects	1	1.870849808	2.416257958	2.482654019
200,000 JSON Objects	1	1.874579131	2.26502539	3.077015341
400,000 JSON Objects	1	1.807803809	2.921061381	2.988274919

Based on the results we get in the experiments, we can make the conclusion that the strong scaling of our solution has been proved.

3.3.2 Weak Scaling

As illustrated in 3.2, each node needs to handle 100000, 200000, and 400000 JSON objects in 3 groups of experiments, and we allocate 3GB of memory that can be used by each core. The execution times are shown in the figure below.



	1	2	4	8
100,000 JSON Objects/Worker	283.75	237.22	200.67	179.45
200,000 JSON Objects/Worker	548.89	506.186	499.206	490.128
400,000 JSON Objects/Worker	334.715	306.501	301.507	306.706

The execution time slightly decreases as we increase the number of workers (also the processors), although the sizes of the data are the same on each processor. However, in general, it’s safe to say that the execution times remain stable in the 3 groups, which clearly shows the weak scalability.

4. Discussion and conclusion

In this project, we build up a Spark Cluster with 4 nodes to solve our designed problem and do some experiments related to scalability.

We work well in using virtual machines to build up a Spark cluster and do some scalability studies on the cluster. The problem we designed to solve is very close to our real life — we always need to do sentiment analysis on user comments to maintain the order of the online communities, or get the views of the majority, so that we can improve our products, services, etc. The performance of our solution is great, with 8 cores, our cluster can process more than 10 thousand JSON objects per second, which is a fast speed.

However, we still have some small flaws in our solution. Firstly, we do not use the models with high accuracy to do the sentiment analysis, so some comments may be misjudged. If we want to apply our solution to handle real-world problems, we need to improve the accuracy of the classification. Secondly, due to the limited resources, our Spark cluster is really small and we use only small parts of the datasets, the result would be more convincing if we do the experiments on a huge cluster with hundreds or even thousands of nodes and with enough disk space to store years of data.

In conclusion, we do a great job with the project. We successfully build up the Spark Cluster, do some processing on the data in the experiments, and demonstrate the weak scaling and strong scaling with the results we get from the experiments.

References

1. https://jgutman.github.io/assets/SNLP_writeup_gutman_nam.pdf
2. https://spark.apache.org/research.html
3. https://www.kth.se/blogs/pdc/2018/11/scalability-strong-and-weak-scaling/
4. https://storageconference.us/2010/Papers/MSST/Shvachko.pdf

Contribution

Our project repo is
https://github.com/hd-zhao-uu/1TD169_Project/

Haodong Zhao:

- Build up the cluster;
- Do the experiments and save the results;
- Part 3 of this report;

Somiya Khurram:

- Build up the cluster;
- Part 2 of this report;

Halit Ege Ebiller:

- Build up the cluster;
- Part 1 of this report;