


Word Frequency Statistics Based on Serverless Computing

Zhaoxin Jia ^a

School of Intelligent Systems Engineering, Sun Yat-sen University, Guangzhou, China

Keywords: Serverless Computing, Word Frequency Statistics, Mapreduce, Cloud Computing.

Abstract: Thanks to the continuous updating of server architectures, serverless computing has gradually become a research hotspot in the current cloud computing field in recent years because of its agile, scalable, and cost-effective features. This paper proposes a word frequency statistics method based on the serverless computing technology and MapReduce framework. Specifically, author add a step of preprocessing based on the basic method. Then, author adaptively assign tasks to each thread according to the total number of words in each file, which helps to reduce time waste. The filegroup will be split in the map stage. During the reduce stage, author further count the word frequency in each thread and store it in a temporary file. The project mainly achieved multi-threaded parallel task completion. Extensive experimental results successfully demonstrated the superiority of multi-threaded parallel processing efficiency in dealing with large amounts of data, which author suppose can bring more new insight for developing serverless computing.


1 INTRODUCTION

With the development of science and technology, research in the field of computer science gradually shows a trend: the speed of software development far exceeds the speed of hardware development. To meet the increasing computational complexity of software, most solutions related to hardware involve increasing the number of hardware devices to meet the high requirements posted by software. The accompanying problem is that the hardware is still in short supply. Serverless computing technology has emerged.

Serverless computing belongs to cloud service technology and is an emerging paradigm of cloud computing used to develop various software application processes (Wen, 2022). Serverless computing is based on cloud computing infrastructure and service providers allocate computing resources. The accuracy and real-time nature of this process mean that the resources allocated do almost exactly what the given code does. There is no need to worry about the cost of resource invocation and the underlying details of the computing platform. Serverless computing is like public transportation. To use it, you only need to pay the corresponding fee for the resources used, and there is no need to pay for idle virtual machines or containers (Kumar, 2019). This

feature greatly facilitates developers. The client no longer directly uses cloud infrastructure, but only uses application process logic without server provided process execution (Werner, 2018; Shafiei, 2019). However, at the same time, serverless computing also has some problems that cannot be ignored, such as the delay caused by platform function cold start and insufficient resource utilization (Li, 2021).

MapReduce was originally proposed by Google as a data parallel programming model for machine clusters, used to process and generate large datasets to solve various real-world problems and tasks. It is also an efficient task scheduling model (Sarkar, 2015). At the beginning of its research, it aimed at parallel processing of web page data in Google search engines. The MapReduce model consists of two functions, which are usually named "map" and "reduce". Firstly, the "map" function receives data and processes it into blocks. Each node in the cluster executes a set of Map tasks in parallel, without sharing any data with other nodes. Next, the data is partitioned across all nodes in the cluster. Finally, each node executes a set of "reduce" tasks in parallel on the partition it receives (Sarkar, 2015). The "reduce" function is responsible for accepting and reducing the intermediate results of the partitioned

^a <https://orcid.org/0009-0006-0265-3357>

data obtained by the "map" function to obtain the final result. The main advantages of the MapReduce model are: first, it can process massive amounts of data in parallel, greatly improving computational speed. Second, mask the intricate details of the underlying implementation, which effectively simplifies the difficulty of writing parallel structured programs and improves program efficiency. This allows developers to focus on the program itself, while issues such as resource allocation and fault tolerance can be handled by the platform (Czech, 2017). However, MapReduce also has some shortcomings. Since MapReduce improves the fault tolerance of long-term analysis through frequent checkpoints of completed tasks and data replication, the frequent input and output required will reduce efficiency. The goal of a parallel database management system is to improve efficiency rather than fault tolerance. It may lead to potential hazards, that is, in the event of a malfunction, a large number of operations need to be redone, greatly increasing the complexity (Lee, 2011).

The advantages of serverless computing and MapReduce show great complementarity in functionality, which inspires us to combine them. This study focuses on applying MapReduce to perform word frequency statistics on words in multiple texts, and adds a preprocessing step of counting the number of words in each text before the model, which is an optimization in the general approach. By combining the MapReduce model, this study provides an effective solution to the problem of word frequency statistics for large amounts of data and multiple texts, and provides direction for the further development and optimization of MapReduce models.

2 METHOD

2.1 MapReduce

MapReduce is an efficient programming model proposed by Google that combines task scheduling and parallel computing. Its main application direction and one of its obvious advantages is processing large-scale data. The reason why it is more efficient than other models is its core idea: divide and conquer. The "map" section decomposes complex tasks, which are massive amounts of data that require computation, into several tasks. The "reduce" section receives intermediate results from the Map section and summarizes them. The user specifies a map function that processes a key-value pair to generate a set of intermediate key-value pairs, and specifies a reduce

function that merges all intermediate values associated with the same intermediate key (Dean, 2004).

2.2 Alibaba Cloud OSS and FC

Object Storage Service (OSS) is a cloud storage service provided by Alibaba Cloud that has high reliability, security, and service availability. The stored files can be called in different ways, such as HTTP, RESTful, API, etc. It has a comprehensive permission control system. Function Calculate (FC) is a fully managed computing service provided by the Alibaba Cloud platform. It does not require purchasing or managing server computing infrastructure, and only requires uploading code. Regarding computing resources, service providers will prepare computing resources and run tasks flexibly. This design fully meets the requirements of serverless computing technology: it is based on cloud computing infrastructure, and service providers allocate computing resources.

2.3 Alibaba Cloud OSS and FC

The main calling function of this project, denoted as final-Copy1, is deployed on the Jupyter Notebook and is responsible for calling the dispatch, mapper, and reducer functions deployed on the Alibaba Cloud platform, which are responsible for evenly allocating computational tasks, decomposing computational tasks, and aggregating results. The dispatch function allocates computational tasks to each thread based on the number of words in the retrieved file; The mapper and reducer functions implement parallel execution of multiple mappers and reducers using a multi-threaded approach, once again improving computational efficiency and resource and time waste. These three functions directly access data files stored in Alibaba Cloud OSS in the form of HTTP and Alibaba Cloud secret keys. By applying this feature, experimenter have achieved the requirement of MapReduce method: to maintain high efficiency when processing massive data.

2.4 Alibaba Cloud OSS and FC

In this project, a basic requirement for the parallel execution of MapReduce was achieved by using a multi-threaded approach. Multi-threading refers to the generation of multiple tasks within the same process, namely threads, which can be executed in parallel. Next, experimenter will compare the advantages of multi-threading methods with serial

execution. Serialization refers to the execution of multiple tasks by a single thread in a certain order. In this method, the previous task must be completed before proceeding to the next task's process. All tasks cannot overlap in time. For multi-threaded parallel execution, it is the simultaneous opening of multiple threads, which strictly means the same occurrence at the same time, and multiple tasks overlap in time. Obviously, the efficiency and speed of parallel execution achieved by multi-threading are significantly better than serial execution. In addition, multi-threaded methods are also a convenient and fast way to achieve parallel task execution at present.

In addition to the multi-threaded method, another two points of this project are the dispatch function deployed on Alibaba Cloud FC. As mentioned in section 2.3, the dispatch function processes HTTP requests in an HTTP mode. For the list of files retrieved from the Alibaba Cloud OSS bucket, they will be sorted based on the number of words in each file and distributed to different threads to ensure that the total number of words allocated to each thread is as close as possible. This can reduce the time waste and improve the efficiency and resource utilization.

3 EXPERIMENT

3.1 Experimental Settings

This project aims to solve the problem of word frequency statistics for multiple files using the MapReduce method. The platform is the OSS and FC services provided by Alibaba Cloud. All the codes are implemented using the Python language and the Jupyter Notebook. The dataset used is 50 randomly collected English essays from the experimenter's middle school period, with a total word count of no less than 5000 words. It is worth noting that according to the file name number, the number of words in files 1-20 is relatively small, while in the following 30 files, the number of words in each file is obviously higher than in the first 20. The evaluation indicators mainly include the time it takes to sequentially count the word frequency in 10, 20, 30, 40, and 50 files, and compare it with the time it takes for a single-threaded serial execution of the same task, in order to study the efficiency of the MapReduce method combined with multi-threaded parallel operations.

3.2 Experimental Environment

There are many implementation methods for the MapReduce model. Considering the specific research

questions and equipment conditions of this project, as mentioned above, the project applies the FC and OSS services provided by Alibaba Cloud for word frequency statistics, and the experimenter's laptop serves as the commander of this project. Below are the parameters of each part of the equipment.

(1)As the commander, the computer is equipped with the traditional 64 bit processor based x64 Windows operating system and the 12th generation Intel i7 central processor.

(2)The network connecting the computer to the Alibaba Cloud platform and Jumper Notebook is a regular home LAN, with a speed of approximately 8.5MB/s.

(3)Jupyter Notebook, formerly known as IPython Notebook, is an interactive notebook commonly used in popular areas in the computer field such as data simulation, statistical modeling, and machine learning.

3.3 Experimental Process

Firstly, deploy the dispatch, mapper, and reducer functions in the Alibaba Cloud, and deploy the client function as the main calling function on the Jupyter Notebook. The client function issues instructions and calls the cloud function to execute the task. After the dispatch function is executed, the files to be counted are divided into several groups based on the number of threads and the number of words in the files assigned. As introduced earlier, the dispatch function provides a solution for assigning files to each thread that will perform word frequency statistics tasks. However, it is worth noting that the dispatch function only provides an allocation scheme for the mapper function below, and the data provided to the mapper function is only some file numbers. After receiving the allocation scheme provided by the dispatch function, the mapper function will group the obtained files according to the scheme and allocate them to various threads. Each reducer function is responsible for the word frequency statistics of a thread. Firstly, the number of words in each file will be output as key-value pairs, and each will be written into a temporary file in json format. Next, by reading the key-value pairs in these json files, summarizing them again to obtain a final result, and writing it into another file. At the same time, the time module of the Python language was introduced into the code of the Jupyter Notebook, which records the start time when calling three cloud functions, the end time after all execution is completed, and the final running time is the difference between the two times.

3.4 Analysis of Experimental Results

According to the experimental method and steps described above, the experimenter conducts the experiment and records the experimental results, namely the program running time and the json format key-value pairs obtained from each round of the experiment for word frequency statistics files. At the same time, to ensure the accuracy and rationality of the experiment, the experimenter conducted a sampling check on the correctness of word frequency in the key-value pair statistical file. The experimental results are shown in Table 1, and the following points can be observed:

(1) Under the conditions of this experiment, the duration is approximately 0.73-1.01 seconds.

(2) According to the experimental process described in section 3.3 and the special arrangement of this experiment on the dataset described in section 3.1, it is evident that there is a significant time difference between the second and third results. This is consistent with the arrangement of the experimenter when deploying the test dataset, further proving the correctness and rationality of this experiment.

Table 1: Processing time for different lengths of words.

Case	Category A	Category B
1	0.730	0.730
2	0.745	0.747
3	0.885	0.925
4	0.947	0.999
5	1.012	1.090

(3) The results of multi-threaded parallel processing are also compared with those of single-threaded serial processing in Figure 1. It can be observed that in the first few experiments, due to the small number of words that need to be counted, the time difference between serial and parallel processing is extremely small and the time used is basically the same. In the latter groups of experiments, as the number of words increased, the difference in time between the experiments conducted using serial and parallel processing methods became more pronounced, with serial processing time gradually increasing significantly compared to parallel processing time. From this, it can be seen that in this project, parallel processing is superior to serial processing in terms of operational efficiency, and its advantages will become more apparent as the amount of data provided increases. This also strongly demonstrates the efficiency of the Mapreduce method

combined with multi-threaded parallel processing in processing huge data.

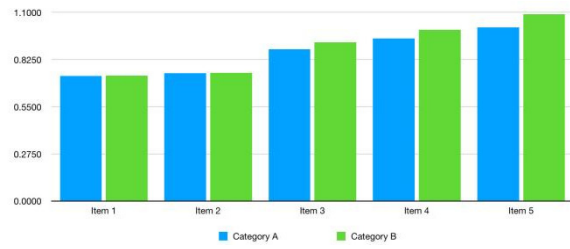


Figure 1: Performance comparison between multi-threaded and single-threaded processing (Photo/Picture credit :Original).

4 DISCUSSION

This experiment applied the MapReduce method to solve the problem of counting the frequency of words in multiple files, and added a data preprocessing step before the processing step to count the number of words in each file and distribute it evenly to each thread. However, there are still many directions for further development or in-depth research in this project.

(1) Combining with other algorithms. The proposed solution in this project can serve as the fundamental solution for the problem of word frequency statistics. The object of word frequency statistics should not be limited to word frequency statistics in multiple files, which may be too simplistic. The big language model represented by ChatGPT has attracted attention from all walks of life and has been imitated by major companies around the world. These powerful models have enormous potential in solving various problems, such as natural language processing (NLP) tasks and real-world cases, from natural language understanding (NLU) to generation tasks, and even paving the way for Artificial General Intelligence (AGI) (Yang, 2023). The recently proposed model Sora for video processing provides a development direction for this project: applying word frequency statistics to audio and even video. Speech recognition technology can be used to recognize and record the textual parts in recorded videos, and then perform word frequency statistics.

(2) Test with larger scale data. Due to the limited hardware equipment of the experimenter, the dataset did not form a large scale, and the total number of words in the test files has not yet reached the level of big data. In the future, equipment may be updated to support a larger amount of data for testing.

5 CONCLUSIONS

This project is based on the MapReduce method in serverless computing technology, and applies multi-threaded methods to achieve parallel processing. It combines the OSS and FC services provided by Alibaba Cloud, and adds a processing step of evenly distributing files based on the word size before entering the thread. Finally, the statistical problem of word frequency in multiple files was completed. In the experiment, a comparison is also made between single threading and multi threading, demonstrating the efficiency of multi threaded parallel processing.

REFERENCES

- Czech, Z. J., 2017. Introduction to Parallel Computing. Cambridge: Cambridge University Press.
- Dean, J., Ghemawat, S., 2004. MapReduce: Simplified Data Processing on Large Cluster. *International Journal of Research and Engineering*, 5, 399-403.
- Kumar, M., 2019. Serverless Architectures Review, Future Trend and the Solutions to Open Problems. *American Journal of Software Engineering*, 6(1), 1-10.
- Lee, K., H., Lee, Y., J., Choi, H., Chung, Y., D., Moon, B., 2011. Parallel data processing with MapReduce: A survey. *SIGMOD Record*, 40(4), 11-20.
- Li, Z., Guo, L., Cheng, J., Chen, Q., He, B., Guo, M., 2021. The Serverless Computing Survey: A Technical Primer for Design Architecture. *ACM Computing Surveys (CSUR)*, 54, 1-34.
- Sarkar, A., Ghosh, A., Nath, A., 2015. MapReduce: A Comprehensive Study on Applications, Scope and Challenges. *International Journal of Advance Research in Computer Science and Management*. 3. 256-272.
- Shafiei, H., Khonsari, A., Mousavi, P., 2019. Serverless Computing: A Survey of Opportunities, Challenges, and Applications. *ACM Computing Surveys*, 54, 1-32.
- Wen, J., Chen, Z., Liu, X., 2022. A Literature Review on Serverless Computing. *arxiv preprint ArXiv, abs/2206.12275*.
- Werner, S., Kuhlenkamp, J., Klems, M., Müller, J., Tai, S., 2018. Serverless Big Data Processing using Matrix Multiplication as Example. 2018 *IEEE International Conference on Big Data*, 358-365.
- Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., Yin, B., Hu, X., 2023. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. *ArXiv, abs/2304.13712*.