

Combining MapReduce and Serverless Computing for Efficient Word Frequency Statistics

Linxu Dai^a

Department of Control Science and Engineering, Tongji University, China

Keywords: Serverless Computing, Mapreduce, Word Frequency Statistics, Machine Learning.

Abstract: Mapreduce is an important data processing model that can process massive data effectively and conveniently. Serverless computing is a new type of cloud computing technology with huge development potential, which aims to alleviate the limitations of centralized model training in traditional machine learning algorithms and can significantly protect data security and personal privacy. Inspired by the advantages of the aforementioned two technologies, this paper considers combining these two technologies together for efficient word frequency statistics. Specifically, this article first describes the basic principles and advantages of the MapReduce model and serverless computing technology, as well as their research and development direction and practical application in recent years. Then, this article details the proposed program framework for frequent statistical tasks based on the MapReduce model and serverless computing technology. Extensive experiments are conducted to demonstrate the effectiveness of proposed method by analyzing the results of the program operation and operation time. Finally, this article discusses ideas and directions that optimize the performance of MapReduce models based on serverless computing technology.


1 INTRODUCTION

MapReduce is a programming model to process and generate large-scale data sets in various practical tasks (Dean, 2018). Users can perform computing tasks through the Map and Reduce functions. At this time, the system can use large-scale parallel computing between large-scale computer clusters, processing machine errors, and scheduling between machine communication to make resources the most effectively used (Dean, 2018).

Serverless is a concept in the field of cloud computing. It usually refers to a function that is a function, that is, the function of serving the FaaS (Function as a Service), which usually includes the back-end of the FAAS service BaaS (Software as a Service) (Yang, 2022). It is characterized by program developers without managing the server or back-end infrastructure, but can focus on completing the setting of the front-end program. Developers only need to deploy the finished front-end program on the platform without server computing providers, and the remaining tasks are responsible for the server-free computing provider (Omar, 2018).

For the model trained with a large amount of data using the MapReduce model, most of their work is usually only composed of repeated parallel operations. So it can naturally be built with serverless calculations, and can greatly use the flexible advantages of serverless computing, which can more efficiently support assignments with different resources during execution (Jonas, 2019). On the other hand, since serverless computing relaxes the burden of training data requirements by allowing each sub server to independently train models locally, more massive training data can be utilized while the data privacy can also be protected. All of these aspects show that combining the MapReduce model and serverless computing together has a promising future.

In this paper, a MapReduce model based on a serverless computing technology is designed to solve the word frequency statistics task. Specifically, the Map and Reduce sections as functions are deployed on the Alibaba Cloud Computing Platform, and a client is set up to trigger Map and Reduce functions in parallel. Finally, the frequency of words in the file are counted based on the proposed model. In the

^a <https://orcid.org/0009-0005-6912-0152>

experiment, the program also counts the total running time of the program, the communication time between the Map and Reduce parts, the calculation time, and their proportion to study the factors that affect the running speed of the program. The rest of this article is structured as follows: Section 2 introduces the existing research results in the fields of MapReduce and serverless computing; The Section 3 introduces the program architecture of this experiment; The Section 4 presents the results obtained from the experiment; The Section 5 is an analysis and discussion of the advantages and disadvantages of the experimental program framework, while proposing areas for further optimization.

2 RELATED WORK

MapReduce model was first proposed by Google in 2004 for parallel computing in large-scale data processing (Dean, 2018). It can effectively reduce the difficulty of parallel programming, improve programming efficiency, and also has the characteristics of cost reduction, good scalability, high efficiency and wide applicability (Li, 2011). Therefore, scholars at home and abroad have conducted research on various aspects of the MapReduce model, mainly focusing on the following aspects: (1) Programming model optimization: The initial MapReduce model had many areas for improvement, and typical achievements in this area include Barrier less MapReduce (Abhishek, 2013), MapReduce Merge (Yang, 2007), MaRCO (Faraz, 2013), etc. However, most of these improvements are only optimized for specific directions and do not have universality. (2) Job scheduling optimization: before executing the Map and Reduce functions, the program needs to allocate tasks to appropriate computing nodes based on various factors such as the job environment and task requirements. For example, MapReduce job scheduling in mixed storage mode (Yang, 2018), optimizing scheduler to process heterogeneous data (Kalia, 2022), etc. (3) Practical application: MapReduce model has been widely used in many aspects, such as digital image processing (Tian, 2017), big data processing platform and big data analysis algorithm (Song, 2014), large-scale network fast connectivity domain detection method (Bhat, 2024), etc.

Since the concept of serverless was proposed, cloud service providers around the world have launched public serverless services, such as AWS lambda, Google GCF, azure functions, IBM cloud

functions, aliyun FC, etc. Serverless computing technology has also derived many applications in many fields, such as big data processing combined with MapReduce model in this experiment, scientific computing (Spillner, 2017), machine learning (Wang, 2019), edge computing (Zhao, 2018), etc.

3 METHOD

3.1 Overall Framework

The structure of the program is shown in Figure 1. The basic idea of the MapReduce model can be simply summarized as separation and combination. When processing large-scale data sets, it first divides them into independent small data sets, and then allows multiple map functions to process these small data sets separately. The output of the map part will be used as the input of the reduce part, and finally summarized in the reduce part and merged into a result file.

In the MapReduce model, the Map and Reduce functions are used repeatedly. These two functions can be deployed on the cloud computing platform and implemented in a serverless architecture. Therefore, this work chooses to build MapReduce model with serverless architecture, using the way of local trigger cloud execution. Therefore, the main part of the program is to set up two functions of Map and Reduce on the serverless platform, and set up a local client to send request trigger function execution program to the cloud. The map part performs the step of counting the frequency of each word in the separate file, while the reduce part summarizes the intermediate results output by the map part, sorts the results in the order of the first letter of the word, and finally outputs a well-arranged result file.

In this experiment, the multithreading method is used to trigger multiple map functions to calculate the whole program in parallel, to count the number of words in multiple text files that have been allocated and stored in Object Storage Service (OSS), and to simulate the situation of multithreading counting multiple files.

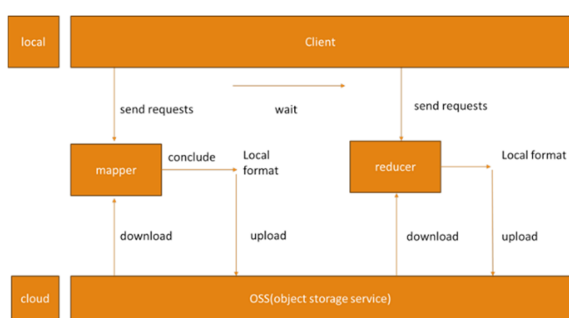


Figure 1: The structure of the proposed method (Photo/Picture credit: Original).

3.2 Basic Steps

First, the client sends a request to the cloud and passes the parameters. The thread class in Python's threading module is used to trigger multiple mapper functions in parallel. According to the different parameters passed in when sending the request, the mapper function will process different parts of the file in parallel.

The mapper function will complete four steps: downloading data from the cloud, counting word frequencies, local formatting files, and uploading intermediate result files to OSS. Download data: download the corresponding initial file according to the incoming parameters as the input file of this mapper function; Statistics of word frequency: the mapper function will traverse all the data in the input file. The counting results of different words will increase by one every time they appear until the last word. The statistical results are in the form of word frequency. The results of each word occupy one line, which is convenient for summarizing the results later; Local format: store the statistical results of each mapper function locally; Upload result: upload the intermediate result file stored locally to OSS.

After waiting for all the mapper functions to be executed, the client will send the request again. The thread class in the threading module is also used to trigger multiple reducer functions in parallel. According to the different parameters passed in when sending the request for the second time, different reducer functions will respectively count the frequency of words starting with different initials in the intermediate result file, and finally summarize them into the final result file in alphabetical order.

After that, the reducer function will also complete the four steps of downloading the file uploaded by the mapper from OSS, summarizing the results, formatting the local file, and uploading the final result file to OSS. Download data: download the intermediate result file; Summary result:, the reducer

function will traverse all intermediate result files counted by the mapper function, and count the frequency of the first letter word it is assigned according to the incoming parameters. When traversing the first word, the word and its frequency will be directly added to the result. When traversing the existing word, its frequency will be added to the frequency in the result, Until the last word of the last file, the final result of the part counted by the reducer function is obtained; Local format: summarize the statistical results of each reducer function into a final result file and store it locally; Upload result: upload the final result file stored locally to OSS.

When all the reducer functions to be executed, the results of word frequency statistics can be seen in OSS, and the word frequency of the corresponding words is arranged in alphabetical order. At the same time, this experiment also adds the step of recording time at the beginning and end of the program, before the client sent the request, after the OSS received the file, and at the beginning and end of the mapper and reducer functions, count the total time used in the process of running the program, as well as the communication time and calculation time of mapper and reducer functions.

4 EXPERIMENT

4.1 Original Data

The original data used in this experiment is an English article with 4244 words, which is manually allocated to seven text files. The data in each file is English words separated by spaces. They will be processed as input files in the mapper function.

4.2 Performance Analysis

In the result file, the words are sorted alphabetically, and the results are arranged in the form of word: frequency. A word is a line, which can clearly query the frequency of the words you want to query. In addition, the experiment also counts the running time of each part of the program. The following are the results of five consecutive runs of the program on the same file.

Table 1: Time cost of Map functions.

	Map communicati on time/s	Map calculatio n time/s	Map communicati on time ratio	Map calculation time ratio
1	1.2767	0.3029	80.82%	19.18%
	1.3529	0.3485	79.51%	20.49%
	1.3054	0.4471	74.49%	25.51%
2	0.5723	0.2308	71.26%	28.74%
	0.6352	0.1707	74.54%	25.46%
	0.5545	0.1038	57.86%	42.14%
3	0.5512	0.1707	75.00%	25.00%
	0.5904	0.1464	80.12%	19.88%
	0.6407	0.3001	68.10%	31.90%
4	0.6306	0.1502	80.76%	19.24%
	0.6183	0.1644	78.99%	21.01%
	0.6230	0.2995	67.53%	32.47%
5	0.5993	0.1682	78.08%	21.92%
	0.6671	0.4149	82.15%	17.85%
	0.6082	0.4159	66.03%	33.97%

Table 2: Time cost of Reduce functions.

	Reduce communicati on time/s	Reduce calculation time/s	Reduce communication time ratio	Reduce calculation time ratio
1	1.2455	0.4602	73.02%	26.98%
	1.2881	0.4155	75.61%	24.39%
	1.3015	0.4869	72.77%	27.23%
2	0.5605	0.4067	57.95%	42.05%
	0.4954	0.4788	50.85%	49.15%
	0.5373	0.5089	51.36%	48.64%
3	0.5471	0.3298	62.39%	37.61%
	0.5630	0.3437	62.09%	37.61%
	0.6640	0.4159	61.49%	38.51%
4	0.5262	0.3299	61.46%	38.53%
	0.5306	0.3383	61.27%	38.93%
	0.4842	0.4084	54.24%	45.75%
5	0.5352	0.3497	60.48%	39.52%
	0.5702	0.3391	62.71%	37.29%
	0.5147	0.4016	56.17%	43.82%

Table 3: Comparison of total time cost.

	1	2	3	4	5
Total time/s	3.5497	2.0116	2.0296	1.8220	1.8500

Analysis of Communication Time. As shown in Table 1 and Table 2, during each program run, the communication time exceeds the calculation time, accounting for more than 50% of the time. This is due to the insufficient amount of data in the statistical files. After increasing the amount of data in the original files, there is a situation where the communication time is less than the calculation time.

Analysis of Total Time. As shown in Table 3, the total time of each program run is different, but generally speaking, only the time of the first program run is significantly different from that of the next several programs, with a maximum difference of about 1.73 seconds and a minimum difference of about 1.52 seconds, while the total time of the second program run is not significantly different, with a maximum difference of about 0.21 seconds and a minimum difference of about 0.03 seconds. It can be seen that the first program run is slower than the next program run.

Analysis of Map and Reduce cost. Since both the Map and Reduce parts execute the next step after waiting for all threads of the previous step to execute, the total running time of each program is determined by the thread that takes the longest time in the run. From the data in Table 1 and 2, it can see that the maximum calculation time of the Map part is almost the same, with a maximum difference of about 0.22 seconds, and the maximum calculation time of the Reduce part is also about the same, with a maximum difference of about 0.18 seconds. There is a big gap between the communication time of Map and Reduce. The maximum communication time of the first time of Map is about 0.72 seconds different from that of the next few times. The minimum difference is about 0.69 seconds, and the maximum communication time of the first time of the Reduce part is about 0.77 seconds at most from the next few times. The minimum difference is about 0.63 seconds. It can be seen that the communication time is the main reason for the large difference between the total time of the first program and the total time of the subsequent programs in this experiment.

5 DISCUSSION

There are many valuable reflections and improvements in this experiment. There are many problems to be solved and many directions to be explored.

First of all, the Map and Reduce parts of this experiment only use three threads, count seven files, and the amount of data in each file is very small, so in fact, the time for the completion of the experiment is very short, and the biggest factor affecting the time is the impact of communication. Therefore, the running time of the program is not much different each time. Basically, only the difference between the communication time can be observed, and it is difficult to study the impact of other variables on the running time of the program.

Secondly, because the amount of data in the original file is small, the original file is manually divided into several small files, and then the whole program is run. In practical applications, the MapReduce model usually deals with large-scale data sets, and manual allocation requires too much time, which is difficult to achieve. And because this program is always waiting for each thread to execute before executing the next step, you can also set a scheduler at the beginning to detect the size of the file to be counted before triggering the mapper function, and then reasonably allocate several files of similar size to each thread to execute the program, so that the program execution time of each thread is the same, reducing the waiting time.

In addition, based on the program framework of this experiment, we can use this program to study other problems, such as image recognition, by changing the original data file and changing the internal I module for word frequency statistics according to the needs. This reflects the high scalability of the serverless framework and the MapReduce model.

6 CONCLUSION

This article studies the application of the MapReduce model based on a serverless computing platform in the task of word frequency statistics. Specifically, the basic principles and advantages of MapReduce model and serverless computing technology is described, following by the development, current situation and practical application of these two technologies. A program framework is then designed based on the research content and a MapReduce model for text word frequency statistics is successfully built on the serverless platform based on this framework. The total running time of the program, the communication time and the calculation time of Map and Reduce parts are counted, and their proportion in the time of this part is calculated. The main factors affecting the running time and total time of each part of the program are explored. All the experiment results show the effectiveness of combining the MapReduce and serverless computing. To sum up, this article provides a useful reference for the application of the MapReduce model based on serverless computing platform in the task of word frequency statistics, and also provides new ideas and methods for the research in related fields.

REFERENCES

- Abhishek, V., Brian, Cho., Nicolas, Z., Indranil, G., Roy, H., 2013. Breaking the MapReduce stage barrier. *Campbell Cluster Computing*, 2013.
- Bhat, S., Y., Abulaish, M., 2024. A MapReduce-Based Approach for Fast Connected Components Detection from Large-Scale Networks. *Big Data*. 2024 Jan 29.
- Dean, J., Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 2008, 51(1): 107-113.
- Faraz, A., Seyong, L., Mithuna, T., Vijaykumar, T., N., 2013. MapReduce with communication overlap (MaRCO). *Journal of Parallel and Distributed Computing*, 2013(5).
- Jonas, E., Schleier-Smith, J., Sreekanti, V., 2019. Cloud programming simplified: A Berkeley view on serverless computing. *arxiv preprint arxiv:1902.03383*, 2019.
- Kalia, K., Dixit, S., Kumar, K., 2022. Improving MapReduce heterogeneous performance using KNN fair share scheduling. *Robotics and Autonomous Systems*, 2022, 157: 104228.
- Li, J., Cu, J., Wang, R., Yan, L., Huang, Y., 2011. Survey of MapReduce Parallel Programming Model. *Acta Electronica Sinica*, 2011, 39 (11): 2635-2642.
- Omar, A., Nur, S., 2018. Serverless Computing and Scheduling Tasks on Cloud: A Review. *American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)*.
- Spillner, J., Mateos, C., Monge, D., 2017. A faster, better, cheaper: the prospect of serverless scientific computing and HPC. *Proc of Latin American High Performance Computing Conference*, 2017:154-168.
- Song, J., Sun, Z., Mao, K., Bao, Y., Yu, G., 2017. Research Advance on MapReduce Based Big Data Processing Platforms and Algorithms. *Journal of Software*, 2017, 28(03):514-543.
- Tian, J., Yang, R., 2014. Research of digital image processing based on MapReduce. *Electronic Design Engineering*, 2014, 22(15): 93-95+100.
- Wang, H., Niu, D., Li, B., 2019. Distributed Machine Learning with a Serverless Architecture. *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, Paris, France, 2019, pp. 1288-1296.
- Yang, B., Zhao, S., Liu F., 2022. A survey on serverless computing. *Computer Engineering and Science*, 2022, 44(04):611-619.
- Yang, H., Dasdan, A., Hsiao, R., L., 2007. Map-reduce-merge: simplified relational data processing on large clusters. *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 2007: 1029-1040.
- Yang, Z., Niu, T., Ming, Lyu., 2023. MapReduce Job Scheduling in Hybrid Storage Modes. *Computer Systems and Applications*, 2023, 32(03):70-85.
- Zhao, Z., Liu, F., Cai, Z., Xiao, N., 2018. Edge Computing: Platforms, Applications and Challenges. *Journal of Computer Research and Development*, 2018(02).