

# Research on Mapreduce Framework Based on Serverless Computing

Shuheng Zhou<sup>a</sup>

*School of Economics and Management, Harbin Institute of Technology, Weihai, Shandong, 264209, China*

**Keywords:** Big Data, Parallel Computing, MapReduce, Serverless Computing.

**Abstract:** As an important framework for handling big data, MapReduce has the ability to simplify complex computing tasks by means of parallel computing. However, the conventional MapReduce framework suffers from issues like high resource consumption and limited real-time capabilities. Fortunately, serverless computing has the advantage of automatic expansion and shrinking, fast power-up, low latency, and high efficiency of resource utilization, providing methods to improve such drawbacks. Therefore, this study tries to improve the behavior of the MapReduce framework through serverless computing. This study successfully implements the “word count” function of the adjusted framework through the ALiYun serverless computing platform and compares its performance with that of conventional MapReduce. In accordance with the experimental results, despite occupying more space, serverless computing evidently cuts function execution time, effectively improving the efficiency of the framework. Specifically, the adjusted framework works 19.42 times faster than conventional MapReduce. Research has shown that improving the MapReduce framework through serverless computing can significantly improve computing efficiency, especially in terms of processing speed, demonstrating the potential of serverless computing in big data processing.


## 1 INTRODUCTION

In contemporary times, big data featuring great volume, great variety, and great velocity has become a main focus of many research domains of data analysis work (Diwakar & Abdul, 2018).

To cope with massive amounts of data, the MapReduce programming framework is invented. Minocha&Singh (2016) pointed out that MapReduce can slice complex computing tasks into simple ones by parallel computing, thus improving efficiency. For example, the average time consumption of a distributed cloud-computing text classifier based on MapReduce is 88.55s, apparently lower than that of means (137.38s) and that of naive Bayes (130.23s) (Jiang, 2021). However, the object for conventional MapReduce tends to be offline and static, posing restrictions in terms of the real-time capability of the framework while consuming much time and space for maintenance (He, 2021). To overcome such challenges, serverless computing has provided an innovative strategy. Researchers have tried combining big data processing with serverless computing to cut down the cost and complexity of big data processing (Yang et al., 2022).

Besides, Cai et al. (2024) proposed SPSC, a stream processing framework based on serverless computing platforms. The SPSC framework has improved the problem of poor resource utilization and high latency through discretized processing and automatic adjustment of computing power (saving 10.8% of the cost on average, compared to Ali Flink). Therefore, if taking advantage of serverless computing's ability to scale up and down automatically, MapReduce can also perform more efficiently and flexibly in data processing tasks.

This study has implemented a word count based on Function Computing (FC) and Object Storage Service (OSS), slicing text data and counting the frequency of each word. Then, it made the comparison of time and space taken by each Map function and Reduce function, thus looking into how parallel computing based on serverless computing affects the efficiency of MapReduce.

<sup>a</sup> <https://orcid.org/0009-0007-5245-2593>

## 2 RESEARCH METHOD

### 2.1 Theoretical Basis

The theoretical basis of this study mainly consists of Serverless computing and MapReduce.

Serverless computing, a cloud computing paradigm abstracting server management, allows developers to focus on coding instead of the underlying infrastructure (Toosi et al., 2025). In the meantime, serverless computing has the advantage of pay-as-you-use, automatic control of scale, and ease of maintenance(Che,2022). By shortening working time and cutting down the cost, this change in the programming paradigm has brought about changes in methods of application development(Puliafito et al., 2024).

MapReduce is a programming model and a framework for processing massive data, which contains two main functions, Map and Reduce. After the whole data is put into the map function, some <key, value> pairs are generated for further processing. Then, the reduce function will capture <key, value> pairs with the same keys, combining their values and generating the final result (Minocha & Singh, 2016).

### 2.2 Experiment

MapReduce framework based on serverless computing platform (taking wordcount as an example)

#### 2.2.1 Choice of Data

The data for this experiment comes from English Novel Net (novel. tingroom. com). This website has collected 5663 common English novels and biographies, the content of which shows great authority and is of heavy value in terms of education and research.

This study has downloaded two novels as different datasets (*Jane Eyre* for dataset1 and *The Adventures of Tom Sawyer* for dataset2), and the data amount of dataset1 (1051kB) is far greater than that of dataset2 (479kB).

#### 2.2.2 Design

##### A. Overall Design

Fig. 1 illustrates the architecture of MapReduce. Algorithm. 1 is the pseudo-code for the Map function of word count based on MapReduce. Through FC and OSS provided by Aliyun, this study develops web functions in a serverless environment, implementing MapReduce distributed computing to text data previously stored in the OSS database. Moreover, this study employs the idea of serial simulation of parallelism, inferring the optimal time cost of parallel computing (In a case where the overload brought by communication is not taken into account) by the maximum time consumption of the workers.

Algorithm 1: Pseudocode for wordcount based on MapReduce (Tang et al.,2024).

Algorithm 1

Map Function

Input: document

Map Function Process:

for each word in document:

emit (word, 1)//Emit each word along with the number 1 as a key-value pair

Reduce Function Input: list of key-value pairs (word, count)

Reduce Function Process:

sum=0

for each count in list of counts:

sum+=count//Sum all the counts for each word

emit (word, sum) //Emit the word and its total count

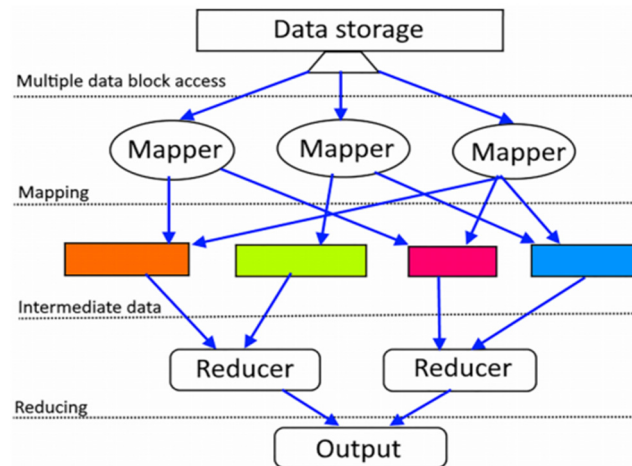


Figure 1: Architecture of MapReduce (Shukla, Alim, 2018).

### B. Map Design

In the map part, workers carry out tasks as below.

1. Using the SDK of Aliyun OSS and provided parameters (Access Key ID, Secret, Endpoint, and Bucketname), extract target text data for processing from the designated storage bucket.
2. Split the input data into a list of words and count the number of occurrences for each word.
3. Create a text file for each word. Name the file as the word itself, and the content of the file is the corresponding number of occurrences.
4. Write these to the specified bucket. If a file with the same name already exists in the bucket, the content of the txt file (word occurrence counts) will be summed.

### C. Reduce Design

In the reduced part, workers carry out tasks as below.

1. extract data produced by map functions from the designated storage bucket. And use a dictionary to record these <key, value> pairs.
2. Transform the dictionary into json file and write it into the bucket.

By adjusting the “Prefix” parameter, each reducer processes a designated part of the data. For instance, some workers can especially handle words starting

with “A”. In this way, it implements parallel computing. This part also compares the efficiency of parallel computing performed by 26 workers and that of serial computing performed by 1 worker.

## 3 RESULTS

In terms of validity, parallel computing and serial computing perform equally. Both parallel computing and serial computing can generate correct <key, value> pairs in the map part and write the proper json file into the designated bucket.

However, parallel computing and serial computing vary greatly in time and space overload. Time overload is labeled by execution time, in other words, time consumed for execution of the business code of a function, not including time taken for preparation of function computing platform and code packages.

Space overload means the maximum of the space occupied in the executive process. Table 1 and Table 2 compare performances of different computing methods in dataset1 and dataset2, respectively.

Table 1: Performances of different computing methods in dataset1

Framework	Computing Method	Execution Time	Space overload
Map	Parallel	26s763ms	1960. 42MB
	Serial	8min25s307ms	208. 25MB
Reduce	Parallel	21s294ms	1200. 68MB
	Serial	6min4s311ms	60. 19MB

The performances of functions show the same tendency in both datasets. Parallel computing always consumes less time than serial computing. While in terms of space overload, parallel computing consumes more.

Table 2: Performances of different computing methods in dataset2

Framework	Computing Method	Execution Time	Space overload
Map	Parallel	15s676ms	1744. 05MB
	Serial	3min3s860ms	106. 25MB
Reduce	Parallel	6s241ms	1573. 60MB
	Serial	1min40s903ms	51. 94MB

## 4 DISCUSSION

This study expatiates a frame of MapReduce based on a serverless platform and uses it to count word occurrences in 2 datasets. Taking advantage of cloud computing and cloud storage, the study addresses the problem that MapReduce is a local operation instead of an online operation, allowing MapReduce to process online data and improving its real-time capability.

Besides, this study compares performances of parallel computing and serial computing based on the MapReduce framework, successfully confirming that parallel computing contributes to carrying out computing tasks that are massive and complex.

The shortcomings are as follows. First, the number of objects in a bucket is far more than that one line of code can extract (1000). Therefore, in the serial computing task, repeated traversal are made, bringing extra time and space overload. Second, Grag (2021) proposes that to use MapReduce, programmers' manual work to divide an algorithm into Maps and Reductions is unavoidable. This problem has not been addressed in this study. Third, Grag (2021) also notes that MapReduce is weak in processing large sets of graphs. In future works, we will try to optimize the framework of Pregel proposed by Google.

## 5 CONCLUSIONS

This study successfully implements the word count application of MapReduce based on a serverless platform. Besides, the efficiencies of parallel computing and serial computing are compared and analyzed in this study. The result indicates that MapReduce based on serverless computing can evidently cut down time overloads. Parallel trades off space for time, performing efficiently in data processing.

However, the study also shows that MapReduce does not do well when taking convenience into account. To implement a set of MapReduce jobs, complicated scripts are always unavoidable. Thus,

while MapReduce remains a powerful tool for big data processing, its adoption may be hindered by the complexity involved in setting up and managing MapReduce jobs, suggesting a need for further research into simplifying these processes or exploring alternative technologies that offer a better balance between performance and convenience.

## REFERENCES

- Adel, N.T., Javadi, B., Iosup, A., Smirni, E. and Dustdar, S., 2025. Serverless computing for next-generation application development. *Future Generation Computer Systems*, 107573-107573.
- Che, Y., 2022. Serverless computing. *Computer and Network*, 1, pp. 36–37.
- Garg, U., 2021. Data analytic models that redress the limitations of MapReduce. *International Journal of Web-Based Learning and Teaching Technologies (IJWLTT)*, 6, pp. 1–15.
- He, B., 2021. Big data computation analysis based on MapReduce. *Computer Programming Techniques and Maintenance*, 12, pp. 97–100.
- Jiang, Q., 2021. Distributed cloud computing data mining methods based on MapReduce. *Journal of Jingdezhen College*, 6, pp. 106–108+128.
- Puliafito, C., Rana, O., Bittencourt, F.L., et al., 2024. Serverless computing in the cloud-to-edge continuum. *Future Generation Computer Systems*, 161514-517.
- Minocha, S. and Singh, H., 2016. MapReduce technique: Review and SWOT analysis. *International Journal of Engineering Research*, 6, pp. 531–533.
- Shukla, D. and Alim, A., 2018. A review on big data: Views, categories and aspects. *International Journal of Computer Applications*, 18, pp. 34–42.
- Tang, J., Du, W. and Zhou, Y., 2024. Application of the MapReduce model in large-scale data parallel mining. *Intelligent IoT Technology*, 2, pp. 38–42.
- Yang, B., Zhao, S. and Liu, F., 2022. Research on serverless computing technology: A review. *Computer Engineering and Science*, 4, pp. 611–619.
- Zinuo, C., Zebin, C., Xinglei, C., Ruhui, M., Haibing, G. and Rajkumar, B., 2024. SPSC: Stream processing framework atop serverless computing for industrial big data. *IEEE Transactions on Cybernetics*.