# MapReduce Algorithm on a Serverless Platform

Hongpeng Zou*

School of Logistics Engineering, Chongqing Finance and Economic College, Chongqing, 401320, China

*Corresponding author: b20160902108@stu.ccsu.edu.cn

**Abstract.** Using the MapReduce (MR) programming technique, large-scale data processing tasks can be divided into tasks that are easier to manage and independent. The serverless implementation's performance is better than the non-serverless MR model when the model parameters are altered in the experiment. Optimizing the use of the MR model on the serverless platform can be achieved by taking into account the relationship between implementation efficiency and platform settings. Additionally, it can act as a source of inspiration for future serverless hardware support configurations. Also, the serverless platform demonstrates how it improves the effectiveness of resource utilization in machine learning training. The intended result can then be achieved by combining the results of these concurrently operating jobs on server clusters. This work adapts MR, a popular big data processing framework, to the serverless platform, emphasizing realization simulation principles and services, and then uses the results in a word count experiment. The experiment uses a word count of about eleven thousand words to evaluate how well MR is implemented on Alibaba Cloud. It is validated for execution time on the platform with varying Central Processing Unit (CPU) core counts, memory configurations, and worker counts. By testing several platform configurations, it is found that the memory configuration has very little impact on the model's execution time, while the size of the CPU core has a considerable impact on reaction time relative to the number of workers.

**Keywords:** programming, Model parameter, No server platform.

## 1. Introduction

In efforts to address the processing capacity bottleneck within the realm of big data, researchers have explored the establishment of distributed computing clusters as a solution for data handling. Among the prominent techniques utilized for this purpose is MapReduce (MR) [1, 2]. The fundamental concept of MR involves partitioning data sets into smaller segments and subsequently conducting multi-threaded concurrent processing to decrease the workload of individual tasks and enhance overall data processing efficiency [3].

The MR programming model enables the simultaneous processing of large-scale data collections [2]. By employing this paradigm, extensive data processing tasks can be subdivided into multiple smaller jobs, which are then executed concurrently within server clusters. The final results can then be generated by combining the computation results from these smaller tasks. Google came up with the idea of MR after it realized how to handle processing vast volumes of web page data in parallel for its search engine. Google redesigned the Web page index processing system for its search engine and later adapted it to a variety of large-scale data processing challenges using MR [3, 4]. The advent of MR is a significant advancement in big data processing in parallel that has altered how large-scale computing is organized. One might argue that MR is now the most efficient computing model because of its vast computational resources [5, 6]. One approach that prioritizes concepts and services is the serverless platform [7, 8]. Developers should concentrate on the real business logic of the software rather than worrying about the infrastructure. The term "serverless" in this context does not only relate to the lack of servers. Within the serverless platform, servers do exist, but they are not part of application development. The serverless platform has various benefits over the non-serverless MR implementation, including reduced costs, simpler equipment operation and maintenance, quicker development, and more effective resource allocation [9]. This new computing paradigm reduces costs,

eases the burden of server management, and allows developers to focus more on their innovative business ideas.

This research explores the widespread utilization of MR for tasks such as large-scale data sorting, pattern recognition, and various other applications. MR is valued for its simplicity and facilitation of parallel processing of extensive datasets. The study applies MR to word frequency statistics by implementing it on a serverless platform. Through experimentation, the study evaluates how different CPU core sizes, memory configurations, and worker counts impact the execution time of MR functions on the serverless platform. Numerous platform settings were adjusted and analyzed to investigate the correlation between implementation efficiency and platform configuration. Furthermore, in addition to showcasing the current state of resource optimization on serverless platforms during machine learning training, the study serves as a blueprint and reference point for optimizing MR function utilization on such platforms and examining the effects of increased hardware support for MR deployments.

## 2. Method

One kind of computing platform is a large data processing platform. Computing platforms, in general, refer to the software, hardware, and runtime libraries that make it possible to execute algorithms. As a result, platforms that can run big data processing algorithms are typically referred to as big data processing platforms. The MR platform supports a wide range of huge data processing techniques, including sorting, data purification, statistical analysis, et al. MR, which offers a distributed parallel operating environment and a programming style, can be used to construct the aforementioned algorithms [10]. An algorithm for big data processing receives large amounts of data as input, processes it while keeping in mind the limitations of its resources, and then determines the solution to a particular issue [11].

A. The workflow of the serverless platform involves conducting an MR task on the Alibaba Cloud platform. Alibaba Cloud, a leading global provider of cloud computing services, offers a serverless reference architecture for MR. Two primary tools utilized in this study are Alibaba Cloud Object Storage Service (OSS) and Function Compute (FC). Alibaba Cloud OSS provides secure and affordable cloud storage, accessible via a straightforward REST interface for uploading and downloading data from any Internet-connected device. Function Compute is an event-driven, fully managed serverless computing system, eliminating the need for users to manage servers or infrastructure. This experiment applies MR to word frequency statistics, requiring the creation of an OSS bucket with a globally unique name to store input data. The dataset consists of approximately 110,000 words from an English novel, divided into over 1,200 text files, each containing 300 words per line. Python functions for mapping and reducing are defined in Function Compute, with parameters dictating the processing range and dividing the workload into blocks. OSS is configured to access files and provide their content to the mapper and reducer for processing.

B. The fundamental concept of the Map algorithm involves applying a specific operation to each item in a list independently. In the word count experiment, the Map function processes data line by line from each block, saving lines in a dictionary where words serve as keys and their frequencies as values. If a word is encountered for the first time, it is added to the dictionary with a value of 1. The dictionary is then ordered alphabetically by word and retained in OSS for further analysis. The MR architecture transfers key-value pairs generated by the Map function to the Reduce function.

C. The Reduce algorithm arranges a list of elements, particularly helpful in scenarios requiring parallel processing. In the word count job, the Reduce function processes the value set received as key-value pairs, producing a result of 0 or 1 for each word. The Reduce function examines files downloaded from OSS and aggregates word frequencies in a two-dimensional dictionary, sorting words alphabetically by key and returning the dictionary as the output.

## 3. Result

A. Experimental details

In this serverless-based MR study, word frequency statistics serve as the experimental method. Three English books, selected randomly from the Internet and presented in text file format, are utilized for the experiment. The experimental data consists of a 110,000-word text document. Conducted on the Alibaba Cloud function computing service platform, the study adopts a controlled experimentation approach. To assess the impact of independent factors, the response times of the model's Map and Reduce functions are measured and analyzed using control variables [12]. The independent variables in this experiment include memory allocation, the number of workers, and the CPU core size in the function configuration, representing the quantity of Map or Reduce functions reacting simultaneously. Data preprocessing involves reading all three novels and consolidating the content into a text file named originalBook.txt. Next, specific punctuation marks should be replaced with spaces in the text document containing the summary data. The written content is then split 40 times, replacing the previous text with space marks. After a single classification based on the space sign, every 300 lines a new text document is generated. Ultimately, each newly processed text document is output by this task in the order of the word name (index). 1240 divided text documents are ultimately produced after completing the aforementioned procedures, and after that, they are output to the local files.

Procedure for the experiment:

Before deploying the ready Map and Reduce functions on Alibaba Cloud function computing service, a total of twelve hundred forty locally preprocessed text documents are uploaded to the files/folder in the Alibaba Cloud OSS object storage. Subsequently, the deployed function is tested, and the function log is checked to retrieve the HTTP request addresses of the Map and Reduce functions.

If the function configuration remains unchanged, parameters in Alibaba Cloud Function Compute's function configuration are adjusted accordingly.

A PyCharm script is created to instruct the cloud service to send an HTTP request to the deployed Map or Reduce function.

The memory usage of a single function response and the response times of various functions for each request are recorded in the function log of the cloud service deployment.

B. Analysis and outcomes of the experiment:

(1) Influence of workforce size: Utilizing the Alibaba Cloud serverless function computing platform, this study employs 1, 2, and 5 workers, respectively, where each worker represents a combination of a Map function and a Reduce function. Python scripts are utilized to implement HTTP response scripts for the Map and Reduce functions. Reaction times and memory usage of both Map and Reduce functions are documented. The function setup on the Alibaba Cloud platform forms the basis for the data presented in Figures 1 and 2. The specifications include a 0.5-core CPU, 512MB of RAM, and 512MB of storage. The number of workers serves as the independent variable, while the mean response time of ten request functions acts as the dependent variable (measured in seconds).

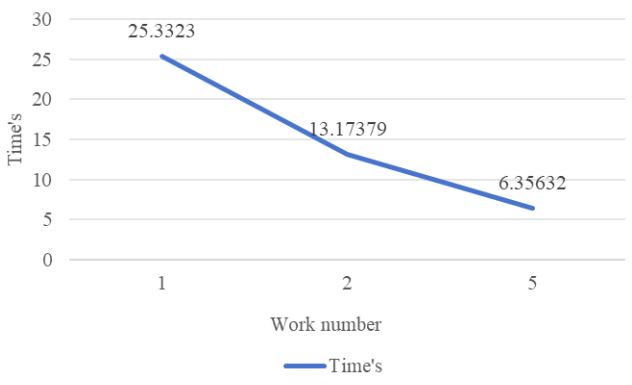


**Figure 1**. Average response time of map function vs. Number of workers [12].

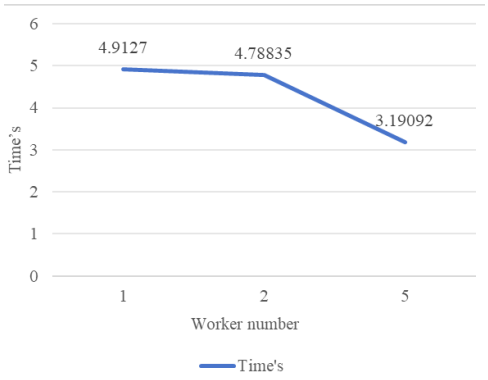


**Figure 2.** Average response time of reduced function vs. Number of workers [12].

(2) The function's response to memory usage was evaluated using Alibaba Cloud's serverless function computing platform, with memory configurations set at 512MB, 1024MB, and 2048MB for the experiment. Python scripts were developed to request HTTP responses for both Map and Reduce functions. Memory utilization under different memory configurations was recorded, generating data statistics and logging memory usage for seven function calls with varying memory size configurations. The function setup on the Alibaba Cloud platform, comprising one worker, 512MB storage, and a CPU with 0.5 cores, provides the foundation for data presented in Figures 3 and 4. The average memory consumption of the seven request functions serves as the dependent variable, while the memory size of the function configuration acts as the independent variable (measured in MB).

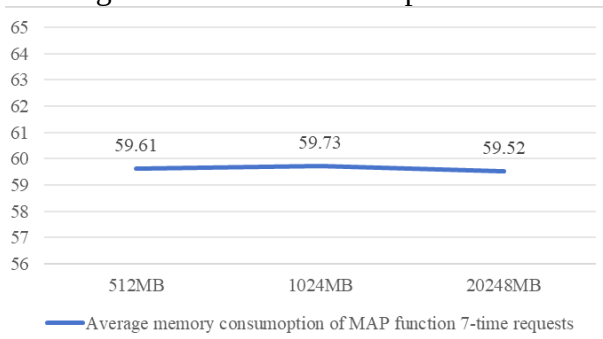


**Figure 3.** Average memory consumption of map function for 7 requests vs. Function configuration memory size [12].

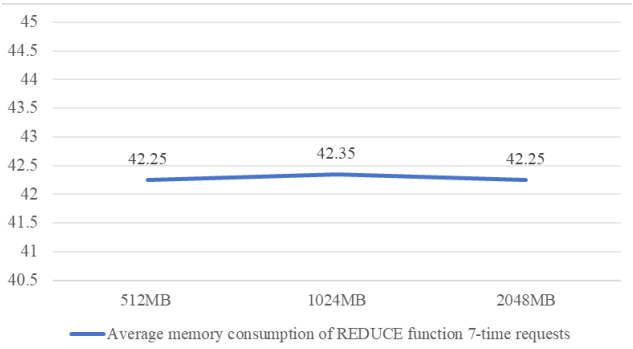


**Figure 4.** Average memory usage of reduce function for 7 requests vs. Function configuration memory size [12].

(3) The impact of function configuration memory size was examined using memory settings of 512MB, 1024MB, and 2048MB on Alibaba Cloud's serverless function computing platform. Python scripts were utilized to request HTTP responses for both Map and Reduce functions. This study recorded the average response time across various memory configurations, noting the average reaction times for ten Map functions and ten Reduce functions.

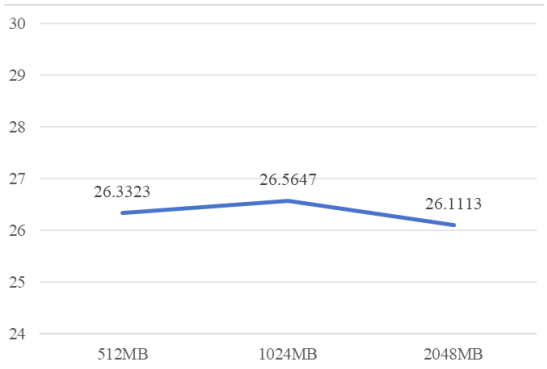


**Figure 5.** There is a correlation between the function-configured memory size and the average response time of 10 Map requests [12].

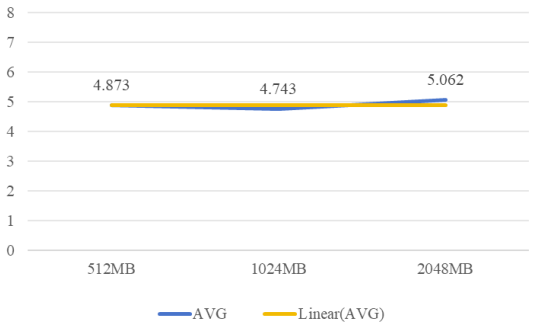


**Figure 6.** Relationship between function-configured memory capacity and average response time of 10 Reduces requests [12].

The data presented in Figures 5 and 6 is derived from the function setup on the Alibaba Cloud platform, which includes one worker, a 512MB storage device, and a CPU with 0.5 cores. In this setup, the size of the function configuration memory serves as the independent variable, while the average response time of 10 request functions acts as the dependent variable.

Regarding the effect of CPU architecture, the experiment utilized Alibaba Cloud's serverless function computing platform to configure function settings with CPU architectures of 0.2 cores, 0.35 cores, and 0.5 cores, respectively. Python scripts were then created to request HTTP responses for Map and Reduce functions using the same function setup. Ten recordings were made for each of the Map and Reduce functions' response times for the same request under varying CPU settings.

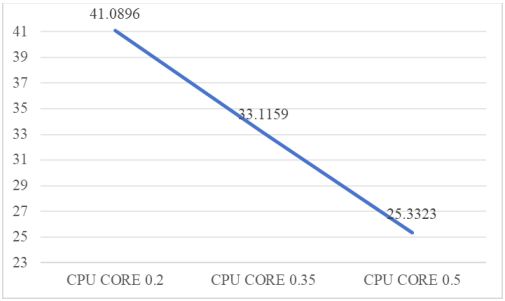


**Figure 7.** Comparison of CPU core size in function configuration and Average Response Time of Map function for ten queries each [12].

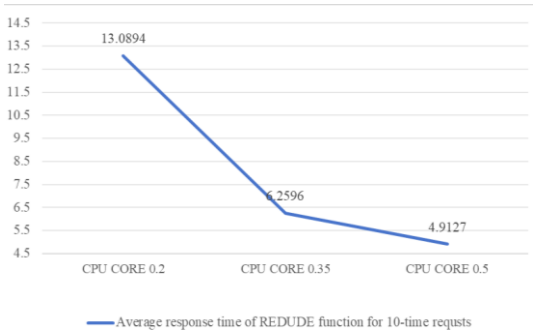


**Figure 8.** Comparison of CPU core size in function configuration and average response time of Reduce function for ten queries each. [12].

The data presented in Figures 7 and 8 is based on the function arrangement of the Alibaba Cloud platform, which includes one worker, 512 MB of RAM, and 512 MB of storage in the setups. The function configuration CPU core size serves as the independent variable, while the average response time of ten request functions acts as the dependent variable (unit: second).

Analytical testing reveals that the number of workers significantly influences the function's reaction time in similar scenarios. Increasing the number of workers with high concurrency theoretically lowers the function's response time.

Figures 3, 4, 5, and 6 demonstrate that, under identical conditions, the memory configuration size does not affect the function's reaction time. Moreover, different memory configurations for each function utilize approximately the same amount of memory. The function's memory usage should be determined by the size of the preprocessed file.

In similar scenarios, Figures 7 and 8 illustrate that the CPU core size has a notable impact on the function's response time. As the size of the CPU core in the function configuration increases, the function's reaction time decreases.

## 4. Discussion

This experiment investigates the impact of variations in CPU core size, memory capacity, and worker count on function response time and memory consumption.

First, the experiment compares the reaction times of the Map and Reduce functions when configured with 1, 2, and 5 workers. Both functions exhibit slower responses as the number of workers increases. Notably, the Map function's response time decreases significantly, while the Reduce function's response time changes only slightly. This difference is likely due to the Map function's more complex task of processing the source file compared to the Reduce function's simpler task of processing files in a uniform format. The study suggests that increasing the number of workers primarily reduces the overall task time by decreasing the Map function's reaction time.

Second, the experiment compares the function's average memory usage and reaction time under different memory configurations (512MB, 1024MB, and 2048MB). It concludes that memory capacity has little effect on average memory usage or reaction times, indicating that altering RAM configuration size is not beneficial. Memory consumption decreases with the number of blocks in the preprocessed file.

Lastly, the experiment examines the reaction times of the Map and Reduce functions at CPU core counts of 0.2, 0.35, and 0.5. It finds that CPU core size significantly impacts overall performance, with higher core sizes leading to faster responses. However, increasing core sizes also results in greater variability in Reduced function response times and increased experiment costs.

Despite its insights, the experiment has limitations. First, the function service's handling of Alibaba Cloud OSS storage calls may affect response time, but this is not evaluated in the experiment. Second, Alibaba Cloud Function Computing Service cannot handle concurrent requests for MR functions, potentially impacting response times.

To address these limitations, future work should measure the duration of communication between functions and OSS storage and consider alternative serverless function computing platforms like AWS. Additional experiments should compare data collection methods between platforms and discuss their impacts on model performance. Moreover, further serverless implementation experiments will support design guidelines.

## 5. Conclusion

This study investigates the utilization of serverless infrastructure for big data processing, using MR as a case study, where FaaS is denoted as a serverless MR model. The primary objective of the experimenters was to analyze the effects of different CPU core sizes, memory capacities, and worker counts on function response time and memory consumption. They also provided broad specifications for applications utilizing serverless data processing. The tests revealed that serverless infrastructure can decrease operational and infrastructure costs while maintaining the reliability of existing systems. Furthermore, experimental results indicated that server-related concerns do not require excessive developer attention, and the serverless implementation outperforms the non-serverless MR model in terms of performance and response time when model parameters are altered.

Application developers can leverage the Alibaba Cloud platform to deploy the MR programming model and adjust settings according to their needs. Multiple experiments involving parameter adjustments concluded that, under the same configuration, the number of workers and CPU core size significantly influence function response time, while the size of the memory configuration has minimal effect.

## References

[1] H. Karloff, S. Suri, S. Vassilvitskii, “A model of computation for MR”. In Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms, (2010), pp. 938-948.

[2] J. Dean, S. Ghemawat, Communications of the ACM, 51(1), 107-113, (2008).

[3] S. N. Khezr, N. J. Navimipour, Journal of Grid Computing, 15(3), 295-321, (2017).

[4] G. Yang, “The application of MR in the cloud computing”. In 2011 2nd International Symposium on Intelligence Information Processing and Trusted Computing, (2011), 154-156.

[5] S. Sakr, A. Liu, A. G. Fayoumi, ACM Computing Surveys (CSUR), 46(1), 1-44, (2013).

[6] C. Doulkeridis, K. Nørvåg, The VLDB Journal, 23(3),355-380, (2014).

[7] G. McGrath, P. R. Brenner, “Serverless computing: Design, implementation, and performance”. In 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), (2017), 405-410.

[8] I. Baldini, P. Castro, K. Chang, et al, In Research Advances in Cloud Computing, 1-20, (2017).

[9] V. Giménez-Alventosa, G. Moltó, M. Caballer, Future Generation Computer Systems, 97, 259-274, (2019).

[10] H. Yang, Z. Luan, W. Li, et al, Journal of Grid Computing, 10(2), 279-310, (2012).

[11] K. Kimura, Y. Nomura, Y. Tanaka, et al, “Runtime composition for extensible big data processing platforms”. In 2015 IEEE 8th International Conference on Cloud Computing, (2015), 1053-1057.

[12] J. Cai, K. Huang, Z. Liao, “Efficiency Assessment of MR Algorithm on a Serverless Platform,” In 2023 IEEE 3rd International Conference on Electronic Technology, Communication and Information (ICETCI), (Changchun, China, 2023), pp. 1201-1207.