Eng Int Syst (2019) 4: 155–166

Engineering
Intelligent Systems

Computer Data Processing Mode in the Era of Big Data

Lian Jin*

School of Mathematics and Computer Science, Jianghan University, Wuhan, Hubei, 430056, China

The current big data cluster mining technique is based on the sampling of data, and the representative data is used to conduct the cluster analysis with point-to-face. For the processing of massive amounts of data, sample extraction probability is the most commonly-used method. However, this approach does not take into account the uneven distribution of the data. This research analyzes the data mining algorithms applied in the era of big data from the perspective of computer algorithms, and proposes an improved algorithm for big data processing. Moreover, this paper compares the performance of the proposed algorithm with that of the traditional algorithm. The research shows that the method proposed in this study is effective and applicable, and can provide a theoretical foundation for subsequent related research.

Keywords: big data; data mining; computer data; data processing; cloud computing

1. INTRODUCTION

In recent years, with the rapid development of the mobile Internet and the advancement of sensor technology, data generation is accelerating rapidly, and the scale of data is increasing enormously. This explosive growth of data requires a highly efficient data processing technology. Many research institutes and companies have developed tools for processing big data, such as Hadoop [1], Spark [2], Hive [3], and Flink [4], to name a few. For the average user, building a big data processing platform is cumbersome and expensive, so mature big data processing cloud services such as Alibaba Cloud E-MapReduce, Amazon EMR [5] and Microsoft Azure HDInsight [6] attract a large number of users.

Users often use performance metrics such as throughput and latency when analysing the performance of big data processing platforms. However, in the cloud environment, these performance indicators are not enough to evaluate big data processing platforms. In addition to assessing the system's traditional performance metrics, performance evaluation in the cloud environment comprises cloud-related metrics that users care about, such as scalability, resilience, fault tolerance, and reliability. The measurement of these indicators is more difficult than that used for traditional performance indicators. The cloud is a heterogeneous system environment. The load and application on the cloud are complex and changeable. Since there are many users in the cloud environment, the load scale performed by the users cannot be determined. These uncertainties exacerbate the difficulty of measurement [7]. Currently, there is no effective model for the measurement of the scalability of cloud services.

This paper proposes a scalability measurement model for providers and users of big data processing platforms to facilitate their analysis of the scalability of big data platforms. At the same time, this study provides a useful reference for community personnel wishing to optimize Hadoop and Spark. In addition, this paper offers guidance on effective performance optimization to users who are deploying and using big data processing platforms.

2. RELATED WORK

The genetic algorithm [8] is a randomization algorithm for global search proposed in the 1970s. It is intended to simulate the evolution process in nature involving the survival of the fittest and genetic selection. The ant colony algorithm [9] was

*Email: lianjin@jhun.edu.cn

proposed in the 1990s as a simulated evolutionary algorithm, intended to simulate the process of an ant search path. The immune algorithm mimics the process by which human immune cells produce antibodies to prevent disease, and is inspired by cell theory. The K-means algorithm [10] is one of the classic clustering algorithms and is invaluable to research. It is widely used in many fields such as text clustering and natural language processing. The advantages are its simple ideas, fast convergence, and easy implementation. The disadvantages are that it is sensitive to the initial clustering center [11], the local optimal solution is easy to form, and the running complexity is high, especially in the big data environment. In the initial study, Hu [12] defined the objective function as the sum of squares, proved the convergence of the sum of squares, and used it as a measure of clustering quality. In 1978, Shah [13] demonstrated that the best segmentation point can be found by using the probability convergence point. Bernasconi [14] validated Hartigan's conclusions in multidimensional space and derived a new standard for measuring clustering quality: under the premise of ensuring square sum convergence, it is necessary to ensure that all cluster centers also converge. There have been other corresponding clustering improvement studies in recent years both in China and abroad. Hou [15] proposed an algorithm based on initial weight and sequential search for initial cluster centers. Dat [16] proposed a density-based solution to select the initial clustering center. They are all optimized by K. The initial center of the means algorithm is selected to improve the accuracy of the cluster.

3. COMMON METRICS AND CLASSIFICATIONS FOR SCALABILITY

Scalability is a performance evaluation indicator of a system or service. It is used to define the performance of a system when it is facing a load by increasing the processor or improving the configuration of the machine. Throughput refers to the number of requests processed by the system within a particular unit of time. In big data processing, throughput can represent the speed of data processing, as shown in equation (1). Throughput T_P can be calculated from data volume D and task execution time T. When scalability is measured by throughput, this is often done by calculating the speedup of throughput.

$$T_P = \frac{D}{T} \tag{1}$$

Scalability can be divided into three categories: linear expansion, sublinear expansion, and superlinear expansion. As shown in Figure 1, top to bottom are linear expansion, superlinear expansion, and sublinear expansion, respectively. When the acceleration ratio shows a linear growth, it is a linear expansion; when it is lower than the linear expansion, it is a sublinear expansion; when it is higher than the linear expansion, it is a superlinear expansion.

As shown in equation (2), this paper defines an indicator P as the performance resource rate. In the formula, T represents the time spent on task execution, and its unit is second. C represents the average resource consumption (CPU, memory, network, etc.) of the task execution process. The resource consumption here refers to the average number of resources consumed during the execution of the task. The resource selected in this paper is the CPU.

$$P = \frac{1}{T * C} \tag{2}$$

$$C = R * U \tag{3}$$

C is calculated by equation (3), R represents the total amount of resources, and U represents the average resource usage rate during task execution. Equation (4) is a calculation method of scalability. There is a total of n loads. i represents the i-th load, S_i represents the size of the i-th load scalability, $D(i)$ and $D_a(i)$ represent the load scale of the i-th load before and after the platform expansion and its unit is GB, $P(i)$ and $P_a(i)$ represent the performance resource rate P before and after expansion, respectively. After calculation, the scalability can be obtained. When the result is close to 1 or greater than 1, the scalability is good. When the result is greater than 1, it is a superlinear extension, and when the result is equal to 1, it is a linear extension, and the closer the result is to 0, the worse the scalability.

$$S_i = \frac{D_a(i) * P_a(i)}{D(i) * P(i)} (i = 1, 2, 3, \cdots, n) \tag{4}$$

Since the scalability of different loads may vary, as shown in Equation (5), this paper calculates the scalability fluctuation of the big data processing platform when the load changes by calculating the standard deviation of the scalability S. In the formula, n means a total of n loads, and i represents the i-th load, ranging from 1 to n. S_i is the scalability of the i-th load. After calculation, it is found that the closer the value of V is to 0, the more stable is the scalability of the big data processing platform. The larger the value of V, the more unstable is the scalability of the processing platform, and the worse is the performance.

$$V = \frac{\sum \left(S_i - \frac{1}{n} \sum_{i=1}^{n} S_i \right)^2}{n} (i = 1, 2, 3, \cdots, n) \tag{5}$$

4. EXPERIMENTAL CONFIGURATION AND ANALYSIS

4.1 Experimental Configuration

The experiments in this study were conducted on the Hadoop platform of Alibaba Cloud E-MapReduce. Table 1 shows the configuration of the Alibaba Cloud Universal instance. Due to limited funding, the experiment uses a small type of configuration, the number of CPU cores is four, the CPU model is Intel Xeon E5, the system is Centos, the memory size is 16GB, the disk storage is four 80GB ordinary SAS hard disks, and the network is a classic 8MB network. Moreover, the version of Alibaba Cloud E-MapReduce is 3.4.3, the version corresponding to Hadoop is 2.7.2, and the version of Spark is 2.1.1.

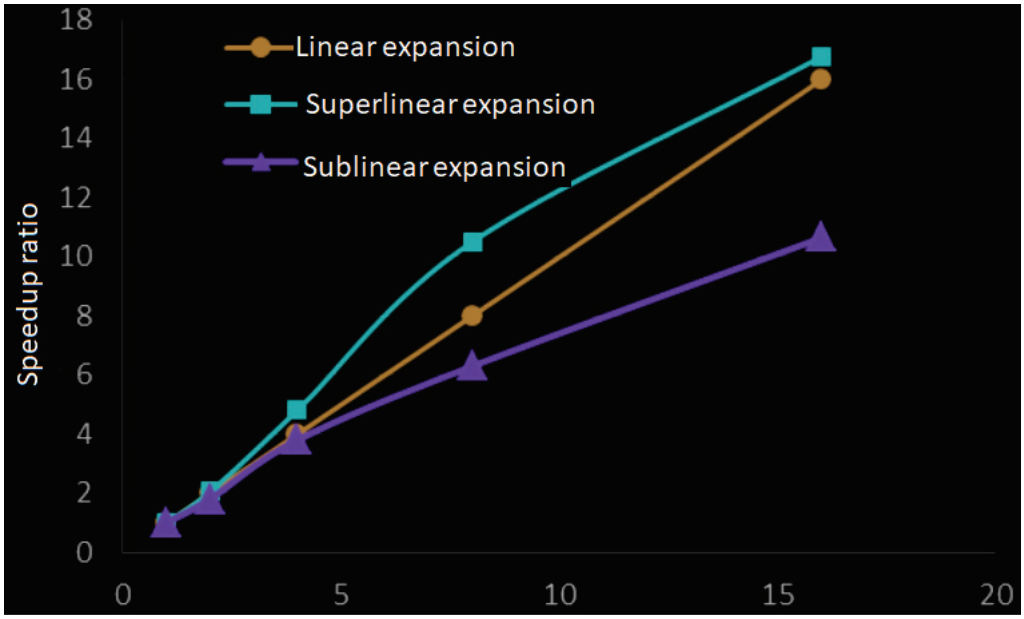

Figure 1 Figure 1Scalability classification.

Table 1 Alibaba Cloud Instance Type.

Type	CPU virtual core	Memory size (GB)	Storage size (GB)
Small	4	16	4*80
Medium	8	32	4*80
Large	16	64	4*80

To measure the scalability of Hadoop in a cloud environment, this studyselects four different loads: TeraSort, Sort, WordCount, WordMean. The first two are I/O intensive and the last two are CPU intensive. Table 2 shows the type of evaluation load, the size of the data used, and generation methodof the data set.

4.2 Experiment Analysis

The vertical axes of Figures 2 and 3 show the execution time and average CPU usage of the I/O intensive load TeraSort and Sort and the CPU-intensive load WordCount and WordMean on the Hadoop platform, respectively, and the horizontal axis represents the number of data nodes.

As shown in Figure 2, the execution time of the four loads decreases as the node size increases. When processing data of the same size, the execution speed of Sort load is faster than that of TeraSort. The main reason is that the data format is different. The object of TeraSort load sorting is text data, which comprises Big Data Bench based on random words generated by Wikipedia. However, the sorting object of the Sort load is a random binary sequence file.

As can be seen from Figure 3, in terms of average CPU usage, both the TeraSort and Sort loads increase first and then shrink. When the data node is two, the execution time of the load is long, and the average CPU usage of TeraSort and Sort is relatively low. When the number of data nodes is four, eight, or 12, the CPU usage is increasing while performance is improving. After increasing to 16 data nodes, the performance improvement has slowed down compared to 12 data nodes, and CPU usage has also decreased. In addition, the average CPU usage of the Sort load is lower than that of TeraSort.

Figure 4 and Figure 5 show the execution time and average CPU usage when running different data-scale loads on a cluster of eightdata nodes. The phenomenon is that as the data scales up, the execution time increases, the average CPU usage increases, and the CPU usage is lower when the load size is small.

5. EXPERIMENT ANALYSIS

5.1 Comparison Between HDFS and OSS

Figure 6 and Figure 7 show the execution time when the WordCount load uses the local file system, HDFS, and the remote storage service, OSS. On the Spark platform, the execution speed of HDFS is faster. On the 16 nodes, the speed difference between the two is not large, and the difference is within 1%.

Figure 8 and Figure 9 show the task execution time when Hadoop performs Grep load using HDFS and OSS. On the Spark platform, the execution speed of the HDFS is faster than the OSS regardless of the size of the cluster and data setOn the Hadoop platform, when there are twodata nodes and fourdata nodes, the execution speed of the OSS is faster than that of the HDFS. If the cluster continues to expand, the HDFS execution speed is dominant.

Table 2 Dataset size and generation method.

Evaluation load	Type	Data set size	Generation method of the data
TeraSort	I/O intensive	128GB	TeraGen
Sort	I/O intensive	128GB	RandomWriter
WordCount	CPU intensive	128GB	RandomTextWriter
WordMean	CPU intensive	128GB	RandomTextWriter

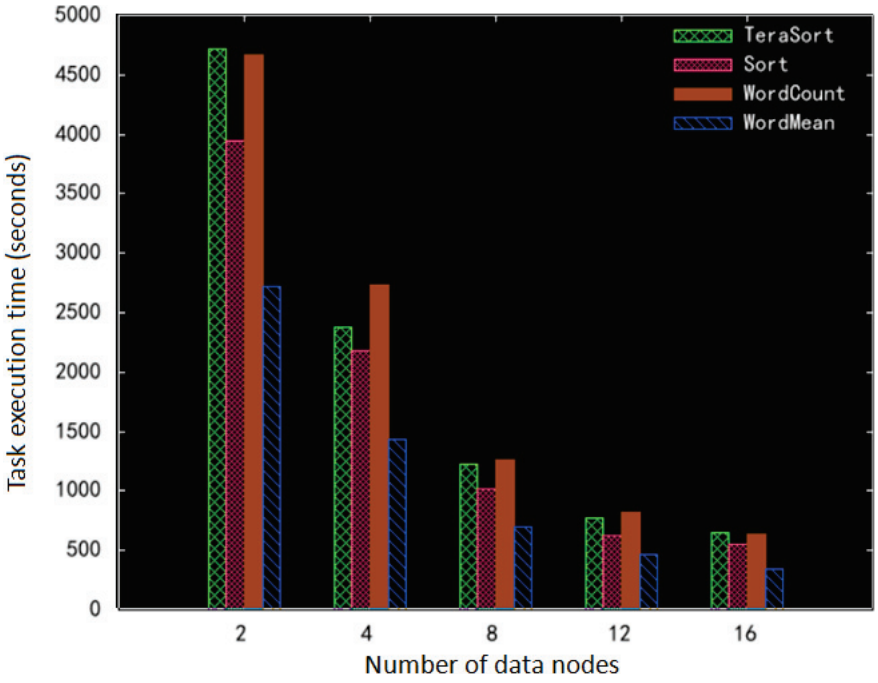

Figure 2 Execution time of load.

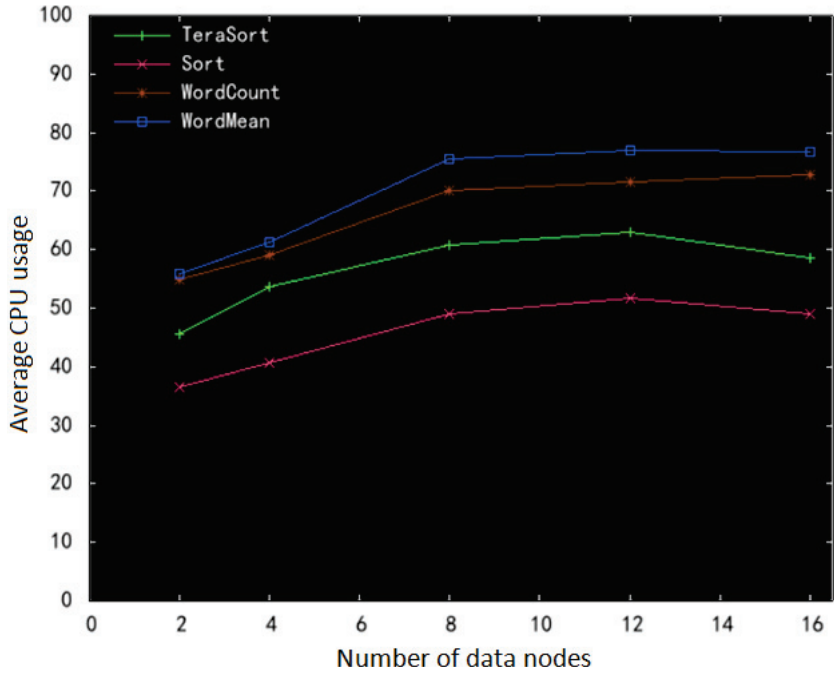

Figure 3 Average CPU usage of the load.

To analyze why Spark uses HDFS better than OSS, this chapter selects two samples and shows the stages of performing a 40GB WordCount load and a 60GB Grep load on four data nodes. The Spark task is called Job. AJob can be divided into multiple Stages, and each Stage is divided into multiple Tasks. Since Spark also includes the startup phase and the final cleanup phase during execution, the analysis process here also includes both. Equation (6) shows the

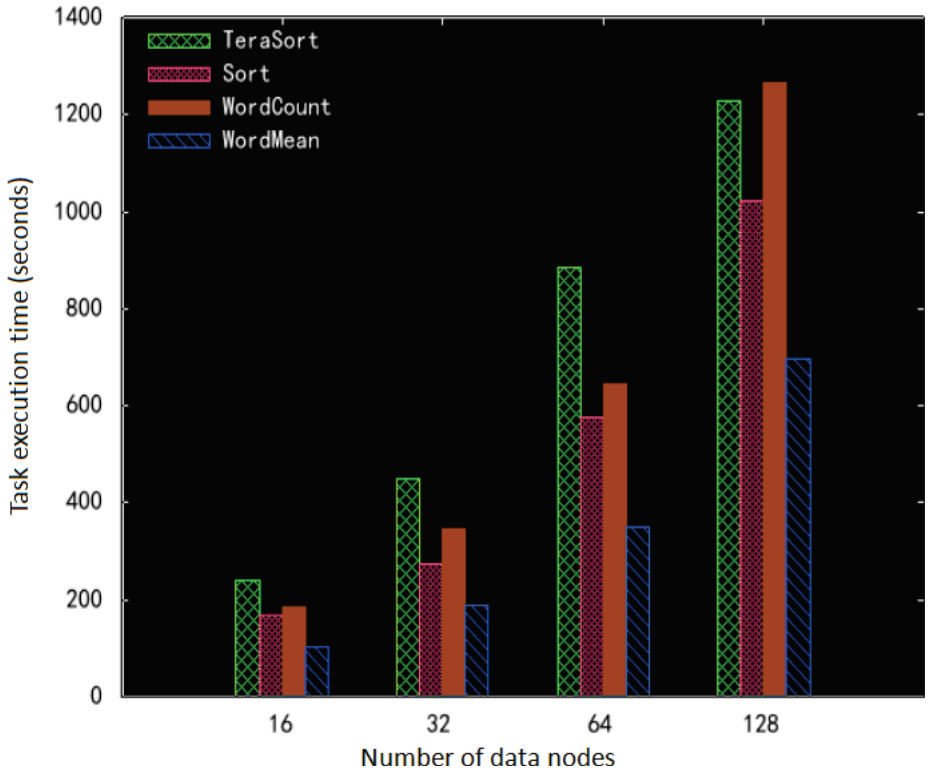

Figure 4 Execution time of different data size loads.

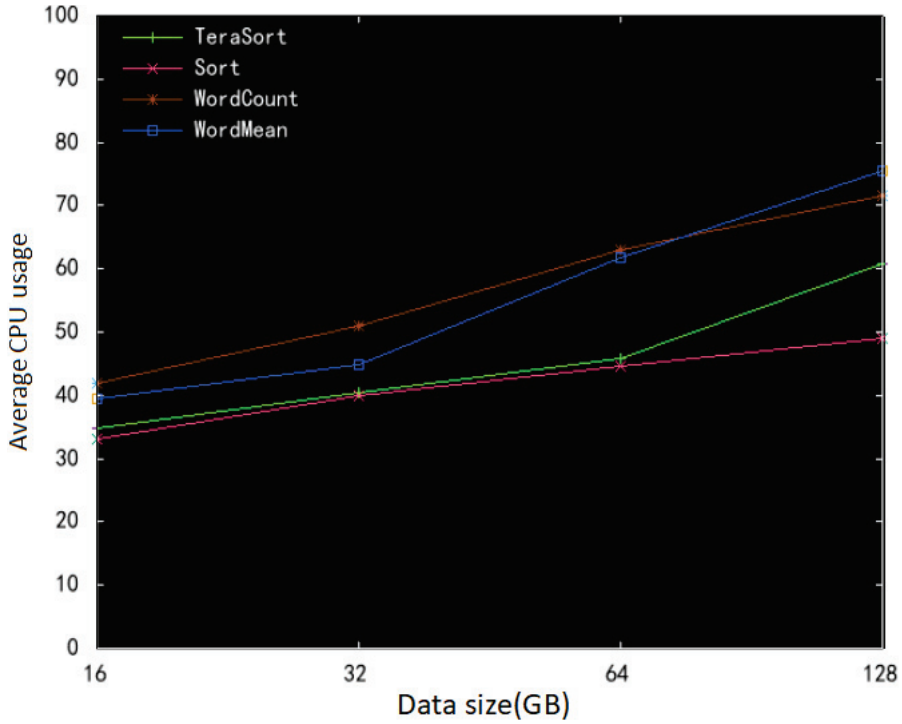

Figure 5 Average CPU usage for different data size loads.

composition of Spark's execution time. J represents the total execution time of the job, n is the total number of stages included in a job, S is the start time of the task, $S_t(i)$ is the run time of the i th stage, and i starts at 1 and ends at n.C is the time spent in the Spark task cleanup phase.

$$J = S + \sum_{i=1}^{n} S_t(i) + C \tag{6}$$

Figure 10 records the total value of network inbound and outbound packets for the nodes of the cluster during WordCount load execution. It can be seen that WordCount has a lot of network transmission when performing tasks on the OSS system, but it is rare on HDFS. This phenomenon confirms that network transmission can affect OSS performance.

Hadoop also has network transmission and network latency when using OSS, but it is faster to use OSS than HDFS. To

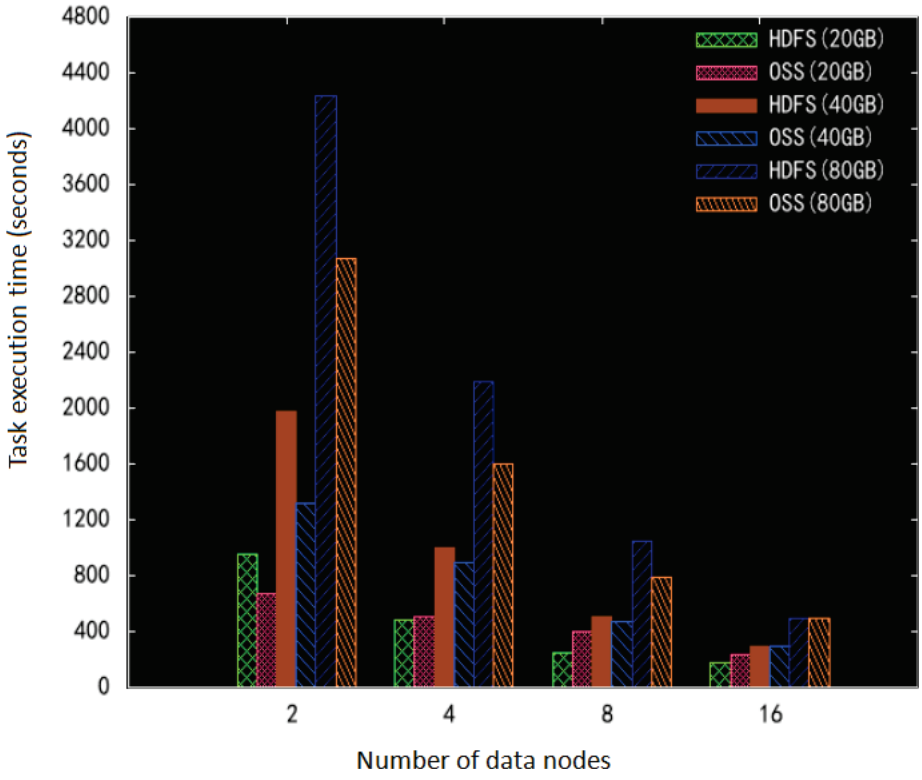

Figure 6 Execution time spent by Hadoop executing WordCount load.

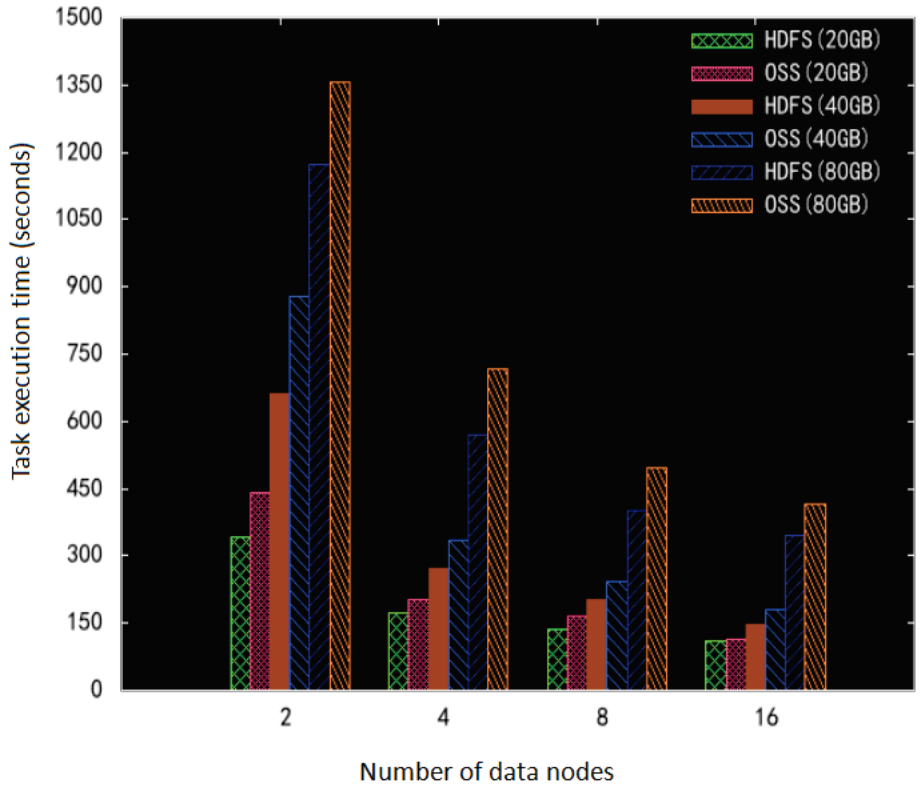

Figure 7 Execution time spent by Spark executing WordCount load.

explain the reason for this phenomenon, this study observed the difference between WordCount and Grep when processing 20GB of data. The difference is that when the algorithm uses OSS to perform tasks, the number of Maps generated is 40. However, when the algorithm uses HDFS to perform tasks, the number of Maps generated is 160. The default number of Maps is determined by size of the HDFS file block. As shown in equation (7), M_n represents the number of Maps, T_f represents the size of the input file, and B_s represents the size of the HDFS file block.

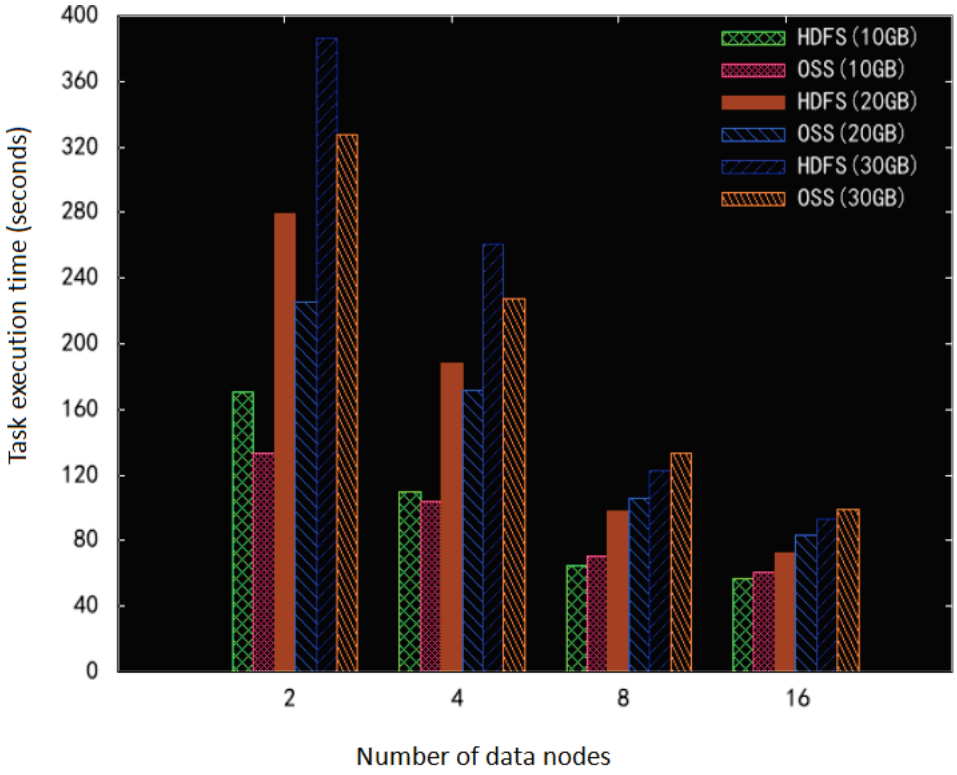

Figure 8 Execution time spent by Hadoop executing Grep load.

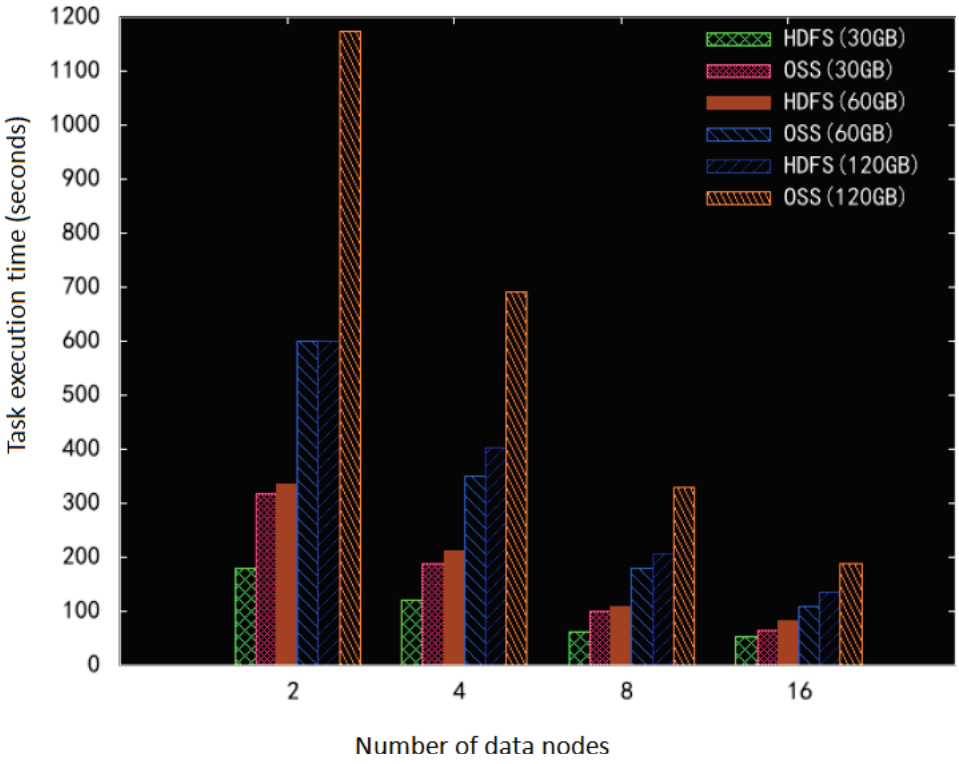

Figure 9 Execution time spent by Spark executing Grep load.

$$M_n = T_f / B_s \tag{7}$$

Typically, the file size set by the user is 64MB, while the default size of Alibaba Cloud is 128MB. Therefore, when Hadoop processes 20GB of data on Alibaba Cloud, if the file is placed on HDFS, the number of Maps is 120. The 20GB file in this article consists of 40 small files of 512MB; because OSS does not perform fragmentation, the number of Maps is 40 when using OSS. In order to verify that the number of Maps will affect the execution speed of the task, after changing the file block size to 512MB and restarting all components, two sets of experiments were conducted in this study. It was found that HDFS is faster than OSS when the number of Maps is consistent. The experimental results are shown in Figures 11 and 12.

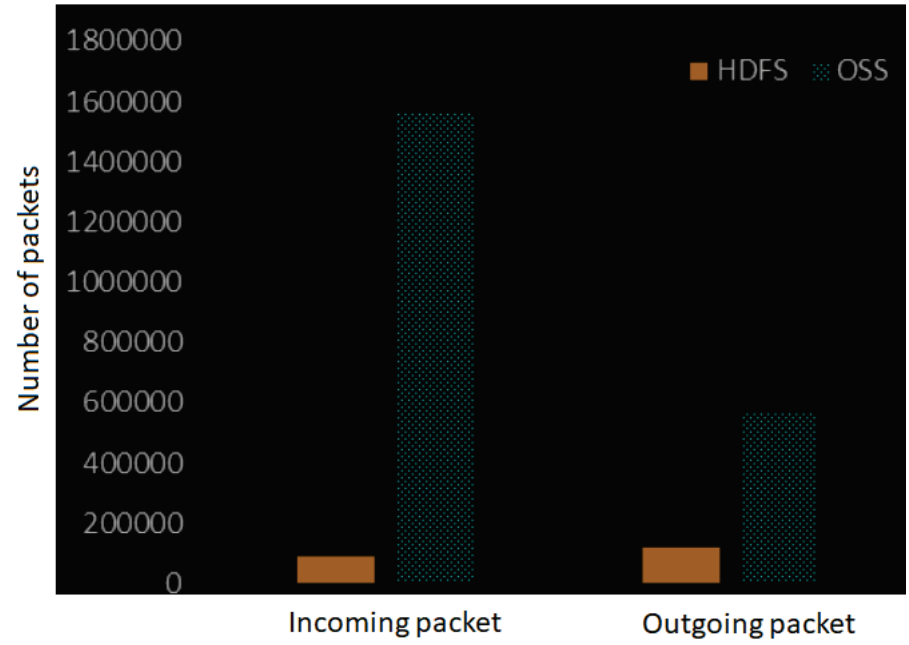

Figure 10 Network Packets when WordCount runs 40GB of data.

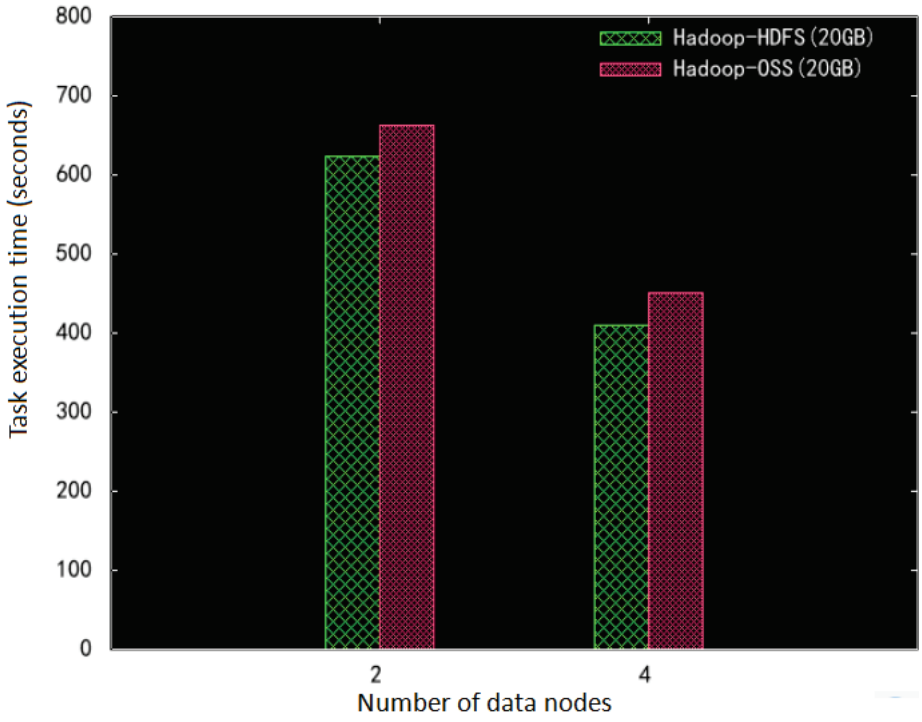

Figure 11 WordCount experiment after modifying the file block size.

If the file block is set too small, the number of Maps will increase, and when the cluster size is small, the pressure on each machine will increase, and the results of the Map phase will need to be pulled at the beginning of the Reduce phase. If the number of Maps is too large, the pull time will be longer and the performance will be degraded.

The difference between Sort and the first two loads is that Sort is an I/O intensive load and has a large amount of write data. This load can test the performance of HDFS and OSS in terms of write performance. Figure 13 and Figure 14 show the execution of the Sort load on Hadoop and Spark, respectively. The results show that the execution speed of OSS is slower than that of HDFS.

Network delay and HDFS data localization lead to OSS having a slower execution speed than that of HDFS, and the I/O-write performance of OSS is quite different from that of HDFS. This paper does not recommend the use of OSS for applications that write a greater amount ofdata. In addition, HDFS's file block settings will affect Hadoop's processing performance. For CPU-intensive applications, we can optimize the performance by increasing the file to an appropriate size and reducing the number of Map operations. For cloud service providers, the 'write' performance of OSS needs to be improved, which is one of the key factors affecting the customer experience.

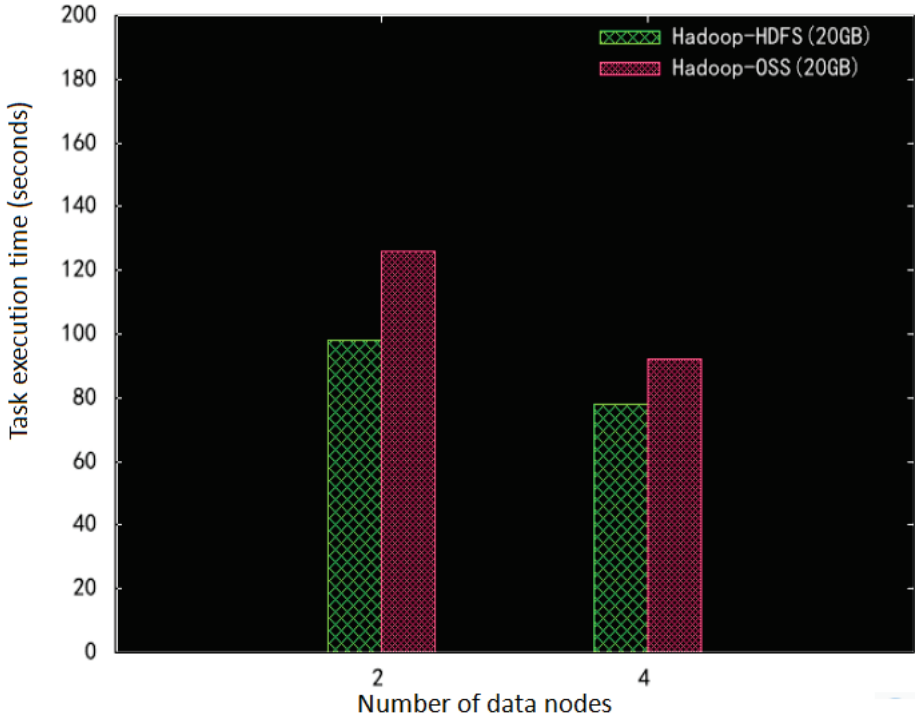

Figure 12 Grep experiment after modifying the file block size.

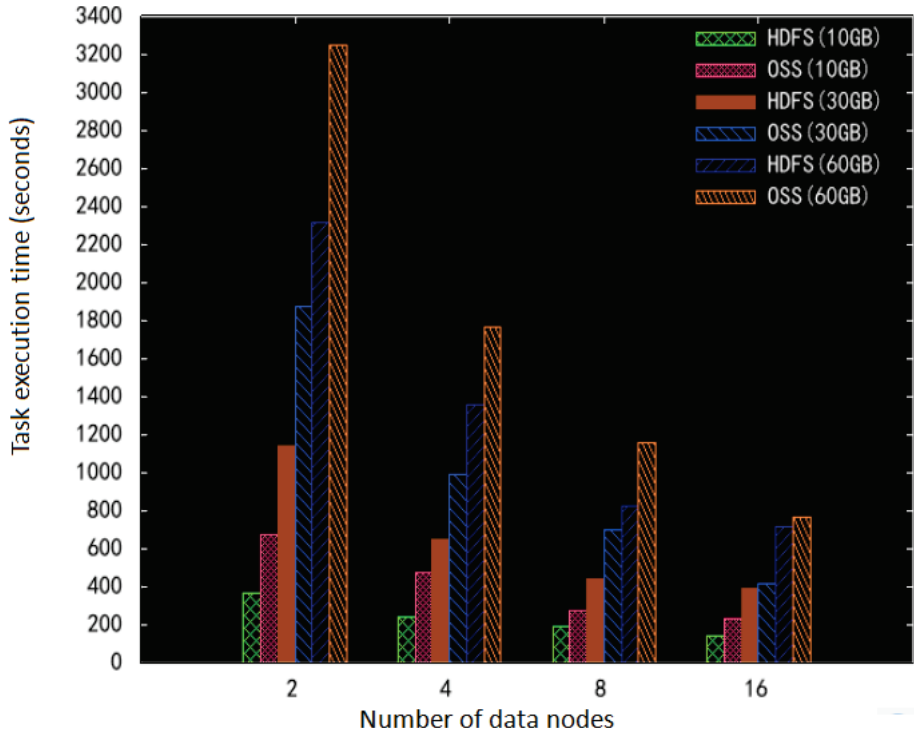

Figure 13 Execution time spent by Hadoop executing the Sort load.

5.2 Performance Comparison Between Horizontal Expansion and Vertical Expansion

Figures 15, 16, and 17 show the performance of Hadoop on different scales of input data using different extensions when running WordCount, Grep, and Sort loads, respectively. The experimental results of the WordCount and Grep load show that the performance of the vertical expansion mode is better than the horizontal expansion mode, but the result of the Sort load is just the opposite.

The reason for this is that when Hadoop runs CPU-intensive workloads such as WordCount and Grep, data shuffling takes more time. During the load operation, although the overall configuration of the horizontal expansion mode and the vertical expansion mode are the same, the performance of the vertical expansion mode is superior to the horizontal expansion mode, and the task execution time is also less. The

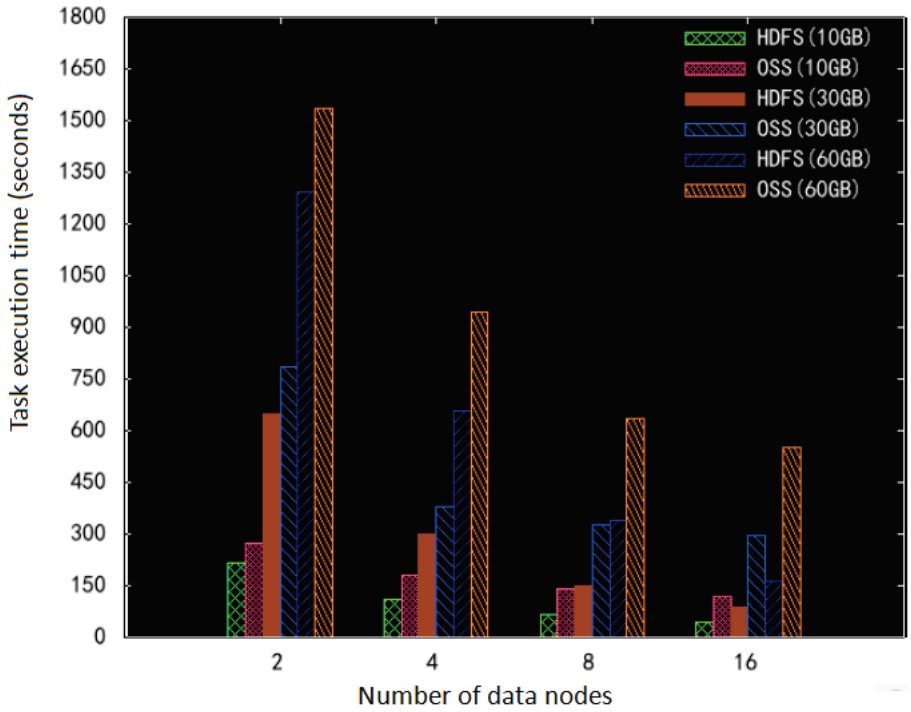

Figure 14 Execution time spent by Spark executing the Sort load.

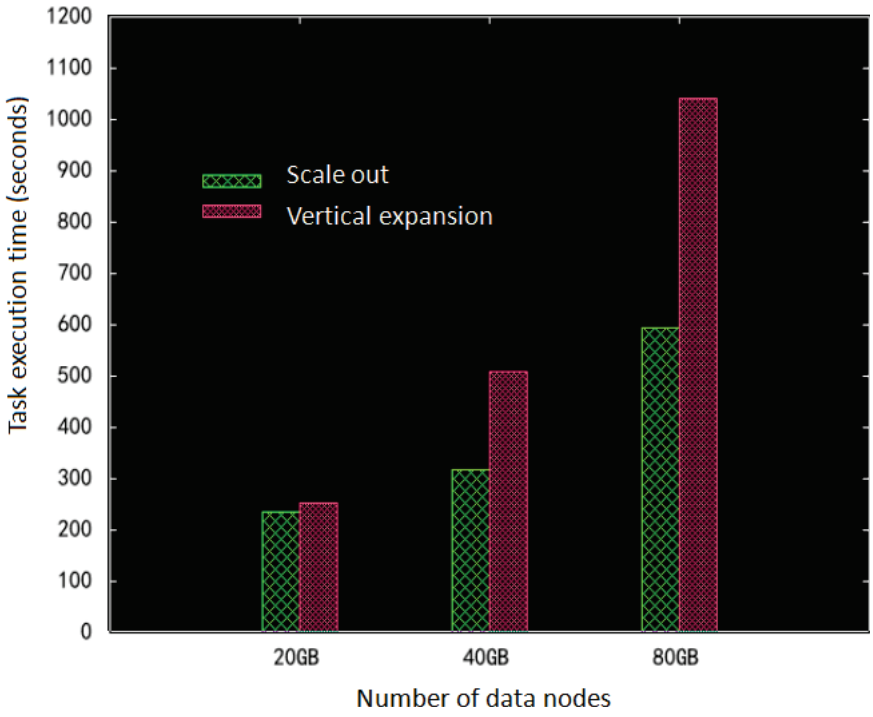

Figure 15 Execution time of vertical and horizontal expansion when Hadoop runs WordCount.

reason is that the intermediate result of the data shuffling process is to be transmitted across nodes and the transfer between multiple data nodes takes more time than does a small number of data nodes. For I/O intensive loads such as Sort, the I/O operation takes a lot of time, and the I/O performance of the extended mode determines the execution performance of the task. In the experiment of this paper, the horizontal expansion method performs I/O operation on a disk of 8 nodes, and the vertical expansion mode performs I/O operation only on a small number of disks of 2 nodes. The I/O operation on 8 nodes is less interfered by other I/O operations, so the performance of vertical expansion is better.

6. CONCLUSION

The development of big data technology is advancing in parallel with the rapid development of the information industry. Big data platforms such as Hadoop and Spark have

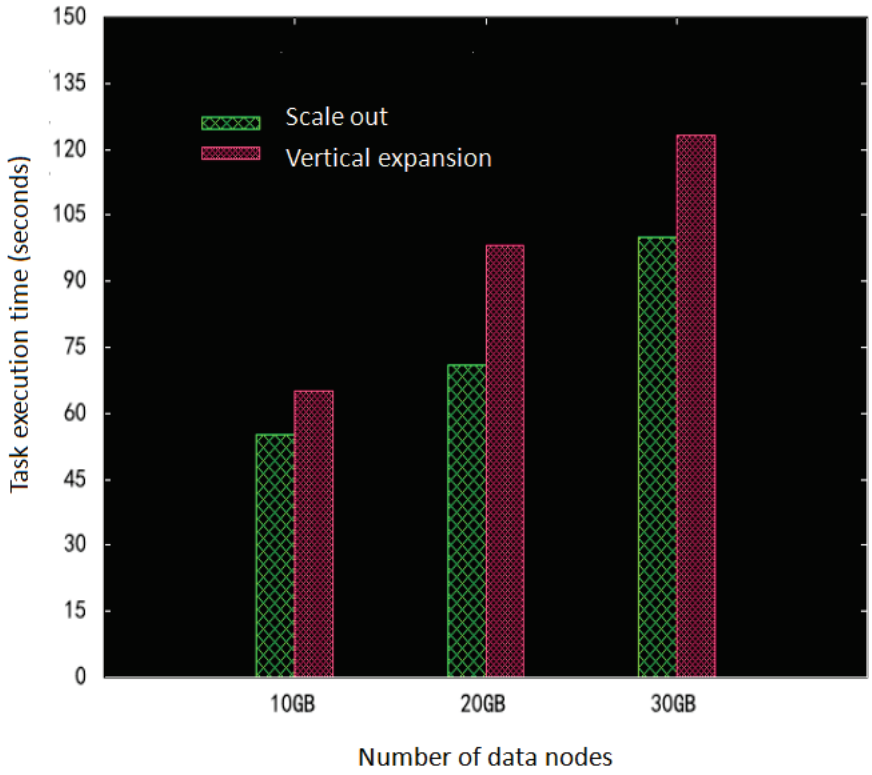

Figure 16 Execution time of vertical and horizontal expansion when Hadoop runs Grep.

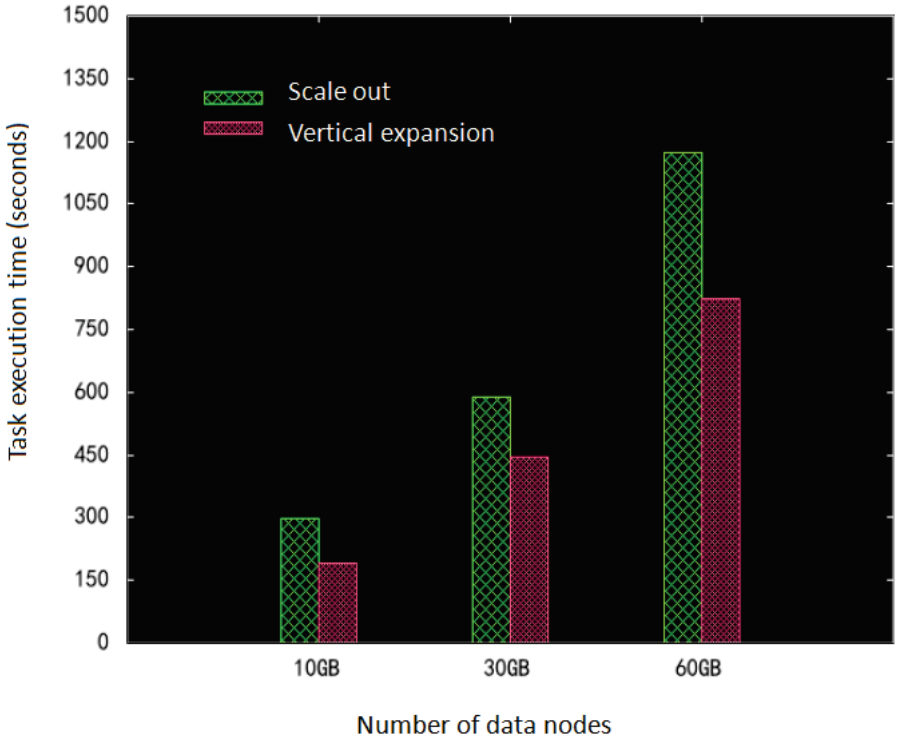

Figure 17 Execution time of vertical and horizontal expansion when Hadoop runs Sort.

been widely used in industry, and many network data items are used as rewards for everyone to calculate and learn. At the same time, the increase in the amount of data also makes it difficult for people to obtain information of interest. The cloud is a heterogeneous system environment. The load and application on the cloud are complex and variable. Moreover, since there are many users in the cloud environment, the scale of the load performed by the user cannot be determined. These uncertainties increase the difficulty of measurement. Currently, there is no effective measurement model for the scalability measure of cloud services. The work of this paper proposes a scalable measurement model for providers and users of big data processing platforms to facilitate the scalability analysis of big data platforms. Moreover, the

paper provides a reference for community personnel who wish to optimize Hadoop and Spark. In addition, users can obtaineffective performance optimization guidance from this paperwhen deploying and using big data processing platforms.

REFERENCES

1. Zhu L, Li H, Feng Y. Research on big data mining based on improved parallel collaborative filtering algorithm[J]. Cluster Computing, 2018(5):1–10.
2. Sethi K, Ramesh D. HFIM: a Spark-based hybrid frequent itemset mining algorithm for big data processing[J]. The Journal of Supercomputing, 2017.
3. Liu J W . Using Big Data Database to Construct New Gfuzzy Text Mining and Decision Algorithm for Targeting and Classifying Customers[J]. Computers & Industrial Engineering, 2018.
4. Liu Y, Ma C, Xu L, et al. MapReduce-based parallel GEP algorithm for efficient function mining in big data applications[J]. Concurrency and Computation: Practice and Experience, 2017:e4379.
5. Njah H, Jamoussi S, Mahdi W. Deep Bayesian network architecture for Big Data mining[J]. Concurrency and Computation Practice and Experience, 2018(1):e4418.
6. Dhyaram L P, Vishnuvardhan B. Classification Performance Improvement Using Random Subset Feature Selection Algorithm for Data Mining[J]. Big Data Research, 2018:S221457961730179X.
7. Alves W, Martins D, Bezerra U, et al. A Hybrid Approach for Big Data Outlier Detection from Electric Power SCADA System[J]. IEEE Latin America Transactions, 2017, 15(1):57–64.
8. Wei Z, Li X, Li X, et al. Medium- and long-term electric power demand forecasting based on the big data of smart city[J]. Journal of Physics Conference Series, 2017, 887(1):012025.
9. Yun W, Zhiquan H, Hao L, et al. A Fast Projection-Based Algorithm for Clustering Big Data[J]. Interdisciplinary Sciences: Computational Life Sciences, 2018.
10. Dawei L, Guangren S. Optimization of common data mining algorithms for petroleum exploration and development[J]. Acta PetroleiSinica, 2018.
11. Jiang D, Luo X, Xuan J, et al. Sentiment Computing for the News Event Based on the Big Social Media Data[J]. IEEE Access, 2016, PP(99):2373–2382.
12. Hu L, Ni Q, Yuan F. Big Data Oriented Novel Background Subtraction Algorithm for Urban Surveillance Systems[J]. Big Data Mining and Analytics, 2018, 1(02):57–65.
13. Shah Z, Mahmood AN, Barlow M, et al. Computing Hierarchical Summary from Two-Dimensional Big Data Streams[J]. IEEE Transactions on Parallel and Distributed Systems, 2018, 29(4):803–818.
14. Bernasconi S, Comini A, Corbella A, et al. Semi-automated De-identification of German Content Sensitive Reports for Big Data Analytics[J]. RöFo - Fortschritte auf dem Gebiet der Röntgenstrahlen und der bildgebendenVerfahren, 2017, 189(07):661–671.
15. Hou Y, Guo H, Nevin N. Research and Prospect of Multimedia Information Data Mining[J]. Recent Patents on Computer Science, 2017.
16. Dat N D, Phu V N, Tran V T N, et al. STING Algorithm Used English Sentiment Classification in a Parallel Environment[J]. International Journal of Pattern Recognition and Artificial Intelligence, 2017, 31(7).