Design of SPRINT Parallelization of Data Mining Algorithms Based on Cloud Computing

Lei Song, Huajie Zhang, Dongdong Feng

***Abstract*—For traditional data mining, all data shall be loaded into memory for analysis and calculation. It belongs to a stand-alone computing mode, which has low calculation efficiency, and a high mining failure rate during the work process. As the data storage and computer technology develop rapidly, how to store and process big data effectively has become an important problem to be solved. Cloud computing can quickly obtain resources from the computing resource pool, and implement parallel improvement of data mining algorithms, which can achieve an efficient combination of cloud computing platform and data mining, and effectively make up for the bottlenecks faced by traditional data mining processes. Therefore, based on the Hadoop cloud computing platform, this paper makes full use of the characteristics of the MapReduce programming framework, and proposes a parallel design of decision tree nodes, node attribute metrics, and Gini index ranking for the SPRINT decision tree algorithm. The performance of the parallelized SPRINT algorithm on classification accuracy, scalability, and speedup ratio is tested. The results indicate that the parallel design of the SPRINT algorithm can obtain good scalability and parallel speedup under the premise of ensuring classification accuracy, which verifies the feasibility of the parallel design of data mining algorithms on the basis of cloud computing.**

***Index Terms*—data mining, cloud computing, SPRINT algorithm, parallel design**

I. Introduction

With the continuous improvement of informationization and intelligence, we have gradually entered the era of information explosion, and the amount of data to be processed has increased significantly. How to discover and use valuable information in a timely and accurate manner has become the focus of current research. Data mining can obtain a lot of useful information from the database. The application of parallel cloud computing can provide great convenience for data mining, and it can dig out more valuable information in less time [1-10]. The promotion and application of cloud computing has created good conditions for large-scale data storage and processing, and the available data mining algorithms are more diverse [11-14]. For example, in business, IBM's cloud computing platform supports data clustering and regression prediction, and can automatically complete data analysis and processing [15]. SPSS's Clementine executes standardized data mining processes to provide users with business decisions [16]. Academia has also done a lot of research on its algorithms based on cloud computing. Literature [17] proposed an optimized clustering and access algorithm for cloud computing databases combined with chaotic probability analysis. The chaotic phase space reconstruction algorithm was used to perform phase space feature clustering of massive data sets in cloud computing databases. Based on this, massive data classification and feature extraction are performed to optimize data scheduling performance. However, the algorithm has poor convergence performance and the calculation process is too complicated. Literature [18] proposed a distributed access algorithm for cloud computing database retrieval based on information-based matching method. Through the single node data scheduling, the construction and access of the state characteristic model of the cloud computing database was realized, and the accuracy of the data was enhanced. But the algorithm has limitations in extracting prior knowledge, and the clustering performance of the data is poor. Literature [19] used the attribute weight allocation method of cloud computing database access channel for data mining, and conducted random sample query from the cloud computing database through the query interface. However, the algorithm reduces the classification accuracy and recall of massive data. Literature [20] improved the SPRINT decision tree classification algorithm to enable it to be deployed in parallel on the Hadoop cloud computing platform to reduce the duplication and unnecessary calculations in the data processing process. But the reliability is not ideal. Reference [21] improved the Bayesian classification (a classification algorithm that already exists in MLlib) of the machine learning library in Spark cloud computing platform and applied it to the Spark Streaming framework to achieve real-time classification of data. The directories should be introduced into the classification performance of metadata. They limit the development and use of applications. Literature [22] put forward a MapReduce parallel computing framework based on cloud computing platform Hadoop, and implemented an ID3 decision tree parallel classification algorithm for massive data sets. However, the calculation is so complex that the system response time performance is poor. Literature [23] suggested the SVM_KNN classification algorithm based on the advantages and disadvantages of KNN and SVM algorithm, and realized the parallelization of the algorithm on the

Manuscript received August 22, 2021; revised January 26, 2022.

L. Song is a lecturer at the Modern Education Center, Kaifeng Vocational College of Culture and Arts, Kaifeng 475004, China. (corresponding author to provide phone: +86-371-22115401, fax: +86-371-22115401, e-mail: songlei@kfwyxy.edu.cn).

H. J. Zhang is a lecturer at the Engineering Training Centre, Zhengzhou University of Technology, Zhengzhou 450044, China. (e-mail: 20146047@zzut.edu.cn).

D. D. Feng is a vice professor at the School of Software, Henan University, Kaifeng 475004, China. (e-mail: Postans1T@gmail.com).

Hadoop cloud computing platform. However, due to the lack of mutation process, the system is easy to fall into local optimization and cannot realize the optimal parallelization.

To solve the problems of the above traditional algorithms, this article has fully analyzed the Hadoop cloud computing platform, using the related features of MapReduce programming framework, implemented a parallel design of the SPRINT decision tree algorithm, and determined the feasibility of the design through relevant performance tests. Simulation analysis further verifies the effectiveness of this method, which obtains good scalability and parallel speedup under the premise of ensuring the classification accuracy. Compared with traditional data mining method, it has more outstanding performance advantages.

The remainder of this paper is structured as follows: In Section 2, the background of the cloud computing platform and features of Hadoop will be explained. In Section 3, the detailed design of parallelization of data mining algorithms on the basis of cloud computing will be elaborated. Section 4 describes the performance evaluation results. Finally, the conclusions and future work will be presented in Section 5.

II. Related Works

A. Cloud Computing Platform and Hadoop

In the face of very large-scale data sets and high-dimensional data types, traditional stand-alone processing is increasingly unable to meet actual needs in terms of computing power and time efficiency. Cloud computing follows the "give on demand" service model and uses technologies such as virtualization to connect computers into a computing platform with high fault tolerance, high storage, and high scalability. Due to the popularity of cloud computing technology, many algorithms can be parallelized and applied to cloud platforms [24]. This makes processing large-scale data no longer a problem. By parallelizing the algorithm, it can be deployed on a computing cluster composed of relatively inexpensive consumer-grade computers, and obtain computing power and efficiency comparable to expensive supercomputers [25, 26].

As shown in Table 1, compared with traditional data processing methods, cloud computing platforms have significant advantages.

TABLE I
COMPARISON OF TRADITIONAL AND CLOUD COMPUTING SERVICES

Item	Traditional mode	Cloud computing
Man-machine mode	Internet / LAN	Using SaaS with the Internet
Implementation mode	Equipment development system	External service
Technology model	Single user	Flexible, scalable, dynamic users
Business model	Pay for equipment and labor	What you use is what you pay

Hadoop is derived from the open source project of the Apache Foundation and belongs to one of many cloud computing platforms, which provides great convenience for data processing [27]. The Hadoop platform is mainly composed of MapReduce, Pig, HDFS, Hive and HBase [28]. In this platform, users can store large-scale data in HDFS systems and apply MapReduce model to process data in HDFS.

The Hadoop platform mainly includes the following features:

(1) During the parallel design process, the increase of nodes can expand the storage capacity and computing power of the Hadoop cluster.

(2) Since the cluster of the Hadoop platform can be built on a general commercial computer, the cost is relatively low.

(3) By using the MapReduce framework for task distribution, and implementing parallel processing and protocol summarization of different parts of the data, the desired results can be obtained, and there will be a positive correlation between the computing speed and the number of computing nodes.

(4) The cluster itself can complete fault detection and processing, which is more conducive to the successful completion of tasks.

B. Definition and Process of Data Mining

Data mining is to find useful information from a database. This process is closely related to many fields such as databases, statistics, artificial intelligence, and so on. From a functional perspective, data mining includes cluster analysis, classification and prediction, association analysis, and outlier detection [29]. Generally, the basic process of data mining should include the following steps:

(1) Data cleaning. Data mining algorithms cannot be applied on unqualified data sets. Thus, the data should be cleaned before data mining, and the data quality should be improved by eliminating data anomalies and filling in missing data, so that it can better meet the specifications of mining algorithms.

(2) Data integration. This process can implement combined processing on multiple data sources, merge them together, make data storage consistent, and eliminate possible data redundancy through appropriate processing.

(3) Data protocol. This process can compress and process related data through various methods such as data aggregation and deletion of redundancy, extract task-related data, and reduce the time complexity of data mining.

(4) Data transformation. Aggregation, data generalization, smooth processing and other methods are used to transform data. For data of real type, it can be transformed by using the concept of data discretization and layering.

(5) Knowledge discovery. It is the core process of data mining to analyze the data set using data mining algorithms to find and extract useful data.

(6) Model evaluation. Evaluate the mined data according to certain evaluation criteria to determine its reliability.

(7) Knowledge representation. The mined data is displayed visually, and the analysis results can also be stored for other applications to call.

The first four steps of data mining can be summarized as data preprocessing. The overall steps are shown in Figure 1.

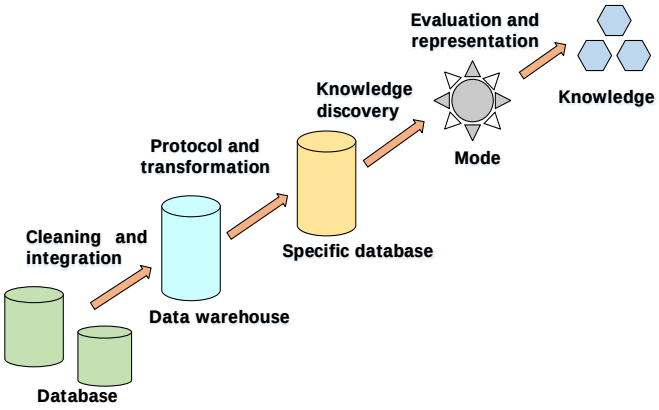

Fig. 1. Data pre-processing diagram.

III. Parallel Design of Data Mining Algorithm Based on Cloud Computing

A. SPRINT Decision Tree Algorithm

When making a decision tree, it is necessary to calculate the attribute selection metric of each node, and then determine the node split. Most decision tree algorithms currently in use need to keep all or most of the data in memory at all times. Therefore, when applying these algorithms to deal with large-scale information, it is easy to cause memory overflow, which greatly limits the knowledge acquisition ability of data mining. With the increasing scale of data processing, the requirements for data mining and data processing capabilities have increased significantly. In this context, people have discovered a variety of decision tree algorithms such as RainFORST [30] and SPRINT [31], which can process large data sets.

1) Decision Tree Algorithm

This tree algorithm is more commonly used in the process of program design. Tree-shaped rules can be deduced from irregular and unordered training samples. It is the most basic and widely used algorithm model in machine learning algorithms. It consists of a decision map and possible results (costs and risks included). According to these, a plan can be created to reach the goal. The core idea is to filter out the final desired result through continuous decision making.

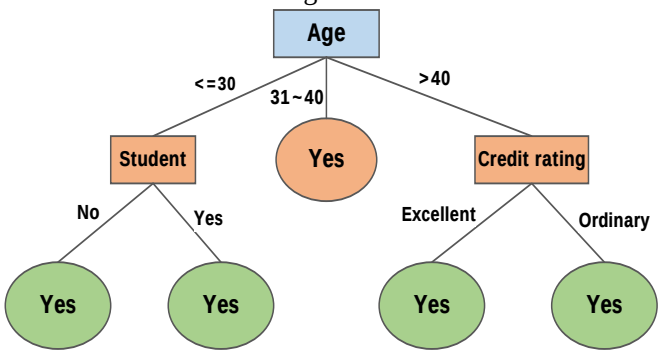

Fig. 2. Typical decision tree structure.

Decision tree uses a tree-like graph or decision model, including random event results, resource costs, and practicality. Leaf nodes are class labels, and the topmost level of the decision tree is the root node. A typical model is displayed in Figure 2.

2) SPRINT Algorithm and Basic Steps

The SPRINT algorithm has important value in processing large-scale data sets, and has good scalability. Compared with most memory-resident algorithms, the SPRINT algorithm can describe the characteristics of the data set by using attribute lists and other methods [32]. When constructing a decision tree, attribute selection metrics are often used to evaluate the classification effect of the attributes of the tree nodes. The following three are more commonly used attribute selection metrics to help determine the best segmentation attributes.

(1) Information gain. It is more common in the ID3 algorithm, and the information size can be measured by the uncertainty of the information.

(2) The calculation formula of the expected information of the tuple classification in *DS* is

$$Info(DS) = -\sum_{i=1}^{m} p_i \log_2(p_i) \tag{1}$$

Divide the tuples in *DS* according to attribute *A*. In order to obtain a more accurate classification, the calculation formula of the amount of information that needs to be obtained is

$$Info_A(DS) = -\sum_{i=1}^{m} \frac{|DS_j|}{|DS|} \times Info(DS_j) \tag{2}$$

Therefore, the calculation formula of the information gain is

$$Gain(A) = Info(DS) - Info_A(DS) \tag{3}$$

(2) Information gain rate. It is a new metric function that mainly penalizes multi-valued attributes by adding a split information item. Its calculation formulas are

$$SplitInfo_A(DS) = -\sum_{j=1}^{v} \frac{|DS_j|}{|DS|} \times \log_2\left(\frac{|DS_j|}{|DS|}\right) \tag{4}$$

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)} \tag{5}$$

(3) Gini indicator. The calculation of this indicator usually needs to be divided into two cases for analysis. The first case is if the attribute list is divided, as shown in the following formula.

$$Gini(DS) = \sum_{i=1}^{m} p_i(1-p_i) = 1 - \sum_{i=1}^{w} p_i^2 \tag{6}$$

The second case is if the data set *DS* contains m pieces of data and is divided into two parts $DS1$ and $DS2$. When $m1$ and $m2$ data are included in these two parts, the Gini index calculation formula is

$$Gini_{split}(DS) = \frac{m_1}{m} Gini(DS_1) + \frac{m_2}{m} Gini(DS_2) \tag{7}$$

The SPRINT algorithm decision tree generation process is:

① First split the dataset by column to form a list of attributes.

② Build the root node. After the build is complete, all the attribute lists should be near the node.

③ Calculate the Gini indicator and determine the split point.

④ Compare the Gini index of each attribute list, select the appropriate segmentation attribute, and split it into two parts $(N1, N2)$.

⑤ Generate $N1$ and $N2$ decision trees until all datasets belong to the same class of targets.

B. MapReduce Programming Framework

MapReduce [33] is a distributed programming model proposed by Google, which is suitable for processing massive data offline. Its main idea is to evolve from functional programming, hide the specific implementation details of parallel programming, and distribute large-scale operations of data sets to each node in the cluster for parallel processing. Then the calculation results of each node are merged together to realize the calculation task. The MapReduce framework was first born in the search field. The reason is that people cannot input lots of data in a short time when there is more data to be processed. Therefore, these calculations must be distributed on lots of machines to meet the expected requirements. How to perform data distribution, parallel calculations, and error handling is also extremely important. On the Hadoop platform, MapReduce framework can be applied for data processing. By mastering the relevant content of the MapReduce framework, users can write program code on the Hadoop platform to complete data mining.

The programming model of the MapReduce framework first requires input data. After the parsing and iterative processing, the input <FName, FContent> pairs are mapped, and the data is reduced in units of groups, and the resulting values are saved. For example, to calculate the number of each word occurrences in a voluminous document, the prepared pseudo code is as follows.

```
map(String FName, String FContent):
//FName: file name
//FContent: file contents
For each word WD in FContent:
EmitIntermediate(WD,"1"):
Reduce(String FName，Iterator FContents):
//FName:a word
//FContents:a list of counts
Int result=0;
for each FC in FContents
Emit(AsString(result))
result+ =ParseInt(FC)
```

The Map and Reduce function can output each word in the document and accumulate the count of specific words. Using the MapReduce programming model, data mining algorithms can be designed and run on the Hadoop platform.

C. Implementation of SPRINT Algorithm Based on MapReduce Framework

By analyzing the SPRINT algorithm, we need to design according to the MapReduce framework when designing a parallelization scheme. It mainly includes three parts: decision tree node, node attribute measurement and parallel design of Gini index ranking.

1) Parallelization of Decision Tree Nodes

In a decision tree composed of a topological structure, the tree is generated through the continuous splitting of each tree node. Practically, the data set is split. After the tree nodes are continuously split, the decision tree can be generated, so the split of the data set is the root. The SPRINT algorithm needs to first use the Gini index to determine the best split point. The sub-attribute list after the tree node splits will be split twice in the new child node. The splitting mode in Figure 3 (a) belongs to the traditional SPRINT algorithm that was more commonly used in the past. MapReduce-based SPRINT algorithm is shown in Figure 3 (b). As shown below, since the algorithm of each layer of the spanning tree is the same, when processing it, the loop can be performed in order to complete the attribute list split and construct the node. During the execution of the algorithm, the attribute list marks its own tree node information.

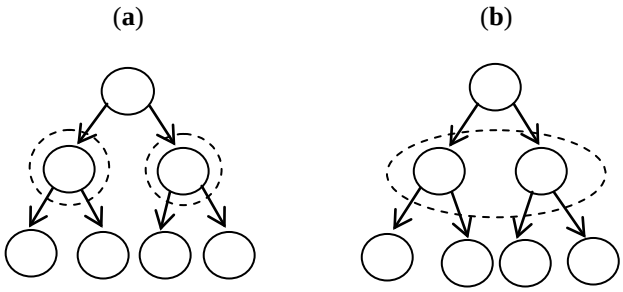

Fig. 3. Parallel design of tree nodes.
(a) Traditional SPRINT algorithm; (b) MapReduce-based SPRINT algorithm

2) Node Attribute Metrics in Parallel

In the SPRINT algorithm, Gini mainly performs a parallelizable process in the node attribute measurement. The calculation of the Gini indicator is an extremely important step [34, 35]. By calculating the Gini value of each attribute on the node, the best splitting attribute can be determined, so the calculation should be paid great attention. In the parallel design of SPRINT, since the Gini indexes of the attributes do not affect each other and are independent of each other, the parallel design of node attribute measurement can be performed according to the process shown in Figure 4.

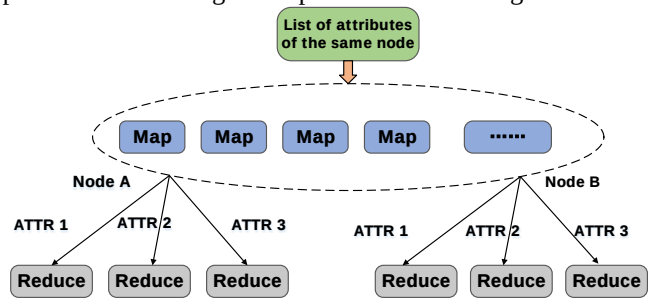

Fig. 4. Node attribute metrics in parallel.

Gini index is based on the parallel mechanism of tree nodes. Since the attribute list needs to be attached to the tree node, when designing for parallelization, the attribute list marked by different tree nodes needs to be parallelized. And it is necessary to determine related rules so that the attribute list of the same tree node and attribute tag can be assigned to the same Reduce.

3) Gini Indicator Sorting in Parallel

In the SPRINT algorithm, if the data belongs to a

continuous attribute list, it is necessary to find the split point when calculating the Gini index. Therefore, the continuous value attribute list needs to be sorted. Gini index sorting is based on the sorting technique used by the MapReduce framework, which can analyze and process the data in the data set more efficiently. If the data set is a continuous value attribute, the Gini index and the clear split point need to be calculated. To realize the calculation of Gini index, the continuous value attribute list must be sorted first. Cbelow and Cabove in the continuous value attribute list indicate respectively the attribute list information that has not been scanned or scanned.

When sequential scanning is performed on continuous value attributes, each scan record is taken as a candidate split point, and the Gini index is calculated, and the optimal split point is determined through repeated operations.

After completely splitting the data set into the attribute list and sorting it, in the case of large data size, we can use the MapReduce framework to deal with this problem. In this framework, the keys of <FName, FContent> pairs can be sorted by the <FName, FContent> pairs of Reduce through the Map process. By combining the above methods, the parallelization of the SPRINT algorithm can be realized using the MapReduce framework.

IV. Experiment and Analysis of Algorithm Parallelization

A. Experimental Environment Preparation

(1) Hardware environment. First install the VMware virtualization software on the server, and then build a virtual Hadoop platform. The server parameters are 32G memory, Intel (R) Core i7 6700K @ 4.00GHz processor.

(2) Software environment. The test is completed in a Linux environment and the software configuration of the test data cluster includes CentOS6.5, hadoop-2.5.0-cdh5.3.6, etc. Then the host name and network IP of the node machine are divided.

(3) Hadoop cluster building process. Modify the mapping between host and IP address of each node in Linux, turn off the firewall of each node, and set the node time synchronization. Set up SSH passwordless login, install jdk and Hadoop. After starting the zookeeper cluster, start *journalnode* on the corresponding node, format and start the HDFS file system, and perform Hadoop performance tuning.

B. Classification Accuracy Test

Select four data sets *DS1*, *DS2*, *DS3*, and *DS4* from UCI Machine Learning Repository. *DS1* has 300 sample points, *DS2* and *DS3* have 5000 and 48888 sample points, and *DS4* has 100,000 sample points. The key point of the improved algorithm is to reduce the sample data set that needs to be compared when searching for neighbors. As long as the limit neighbor distance of the algorithm is set properly, there will be no loss of neighbor points. That is, in theory, for the same data point to be tested, the neighboring sample points found by the parallelized SPRINT algorithm and the sample points found by the original algorithm should be the same.

Through the analysis of the experimental results in Figure 5, it is found that the classification accuracy of the MapReduce model for the four data sets *DS1*, *DS2*, *DS3*, and *DS4* is not significantly different from that of the serial model. It can be seen that although the SPRINT algorithm has not been significantly improved in classification accuracy after parallel design, it has not declined. It can ensure that the classification accuracy of the algorithm after parallelization meets the requirements.

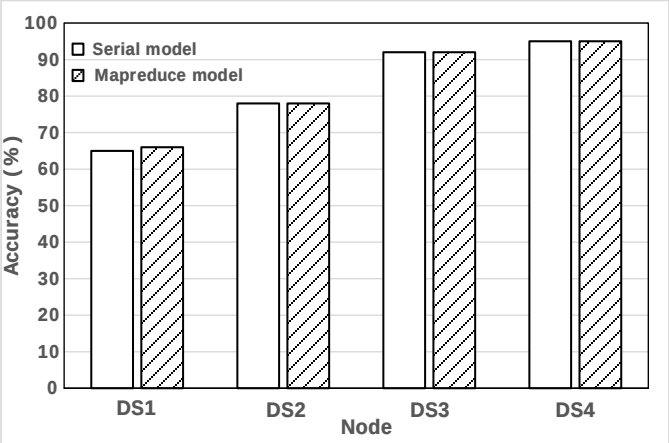

Fig. 5. Classification results of SPRINT algorithm.

C. Scalability Test

When testing the algorithm after parallel design, scalability is an important indicator that cannot be ignored. The calculation formula is

$$E = \frac{S}{P} \tag{8}$$

Where E represents scalability, S represents the speed ratio of parallelization of the cluster, and P represents the number of nodes of parallel computing. In UCI, *DS1*, *DS2*, *DS3*, and *DS4* are selected as four experimental sample sets of different sizes. The Hadoop platform runs on different numbers of clusters for data processing tasks. The nodes are slaver nodes, and the number of nodes is 01 to 07. After passing the test analysis, the scalability test results obtained are shown in Figure 6.

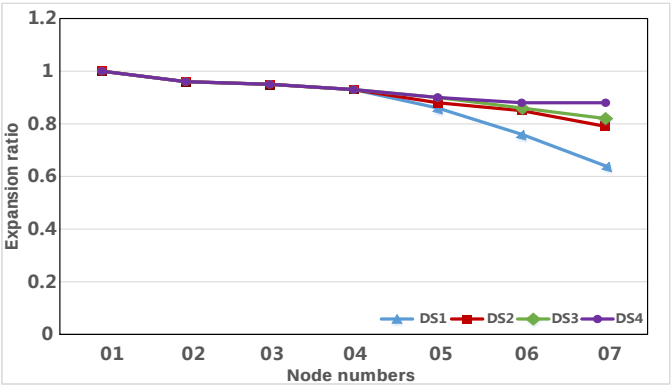

Fig. 6. Test results of SPRINT algorithm scalability.

According to the test results, the SPRINT algorithm has good scalability. It can increase the speed of large-scale data calculations. As the cluster nodes increase, the degree of parallelization and the corresponding data set size have been maximized, and cannot grow as the computing nodes in the cluster increase, and the scalability shows a downward trend. We believe that this trend may be related to factors such as the size of the corresponding data set and the maximum degree of parallelism. By analyzing the curve in Figure 6, it can be known that after the number of slave nodes reaches 5, the

scalability of the *DS1* dataset reaches a horizontal state, indicating that the *DS1* dataset only needs 5 computing nodes to maximize the degree of parallelization of the algorithm.

D. Speedup Ratio Test

The speedup ratio can reflect the parallel computing performance of the cluster. The calculation formula is

$$S = \frac{T_s}{T_p} \tag{9}$$

Among them, T_s is the calculation time of a single node, and T_p is the time of parallel calculation of P nodes with the same performance. When implementing the test, the larger the acceleration ratio, the better the cluster performance. Selected data for the speedup test is the same as the scalability test. The specific test results are shown in Figure 7.

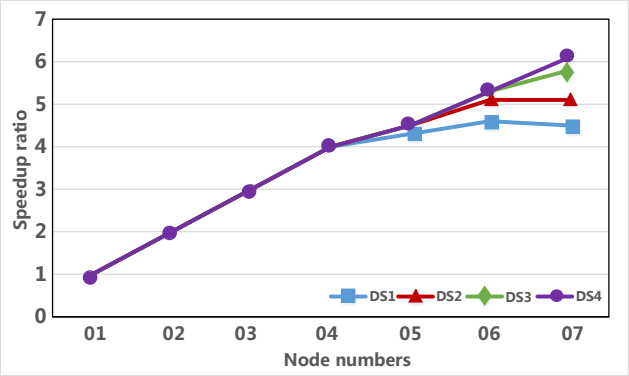

Fig. 7. Test results of SPRINT algorithm speedup ratio.

According to the test results, the SPRINT algorithm after parallel design has a good cluster acceleration ratio, and the calculation speed is good in each data set processing. The speedup ratio keeps stable or declines with the increase of cluster nodes, and the reason may be related to the maximum degree of parallelization. After the ratio is maximized, even if the number of cluster computing nodes increases, it cannot continue to grow. The reasons are as follows: (1) The more the number of cluster nodes, the higher the communication overhead between nodes; (2) The server that builds the experimental environment is a PC, and the computer's own hardware has limitations, which will reach the PC computing limit of the machine.

When the *DS1* data set reaches five slaver nodes, the speedup ratio remains at a level. The reason may be that there are 5 nodes in the *DS1* dataset with 23,000 sample points, which maximizes the degree of parallelization of the algorithm. For *DS3* and *DS4*, even if there are 7 computing nodes, the degree of parallelism has not reached the highest level.

V. Conclusions

Aiming at the problem that traditional data mining algorithms cannot effectively store and efficiently process data, this article uses the Hadoop platform to implement parallelization of the SPRINT algorithm to achieve efficient storage and processing of large-scale data sets. In terms of the analysis and processing of large-scale data sets, the parallel data mining algorithm on the basis of the cloud computing platform does not lose classification accuracy. Compared with the original algorithm, it has obvious advantages in computing speed and excellent parallel speedup. The following research work has mainly been done:

(1) The necessity and research background of parallel design of data mining algorithms is analyzed, and the characteristics of Hadoop platform and data mining are fully understood.

(2) The SPRINT decision tree algorithm is designed with parallelization of decision tree nodes, node attribute metrics, and Gini index ordering. Parallelization is achieved through the MapReduce programming framework.

(3) The classification accuracy, scalability, and speedup performance of the SPRINT algorithm have been tested. The results indicate that the algorithm has good accuracy, speedup, and scalability.

In future, it is necessary to conduct further research in combination with the actual environment of massive data mining to enhance the mining ability of the system.

Author Contributions

Conceptualization, Lei Song; methodology, Huajie Zhang; software, Lei Song; validation, Huajie Zhang; formal analysis, Dongdong Feng; investigation, Lei Song; resources, Dongdong Feng; data curation, Huajie Zhang; writing, Lei Song & Huajie Zhang; supervision, Dongdong Feng.

Acknowledgments

The authors thank Mingzhe Song, Xiaojing Wang, Xi'an Zhang and Tong Song for their useful advice during this work.

References

[1] Y. Zang, T. Hu, T. Zhou, "An Automated Penetration Semantic Knowledge Mining Algorithm Based on Bayesian Inference," Comput. Mater. Contin., vol. 66, pp. 2573-2585, 2021.

[2] A. K. Arslan, C. Colak, M. E. Sarihan, "Different medical data mining approaches based prediction of ischemic stroke," Comput. Methods Programs Biomed., vol. 130, pp. 87–92, 2016.

[3] H. Wiemer, L. Drowatzky, S. Ihlenfeldt, "Data mining methodology for engineering applications (DMME)—A holistic extension to the CRISP-DM model," App. Sci., vol. 9, pp. 2407, 2019.

[4] Q. Yaseen, Y. Jararweh, B. Panda, Q. Althebyan, "An insider threat aware access control for cloud relational databases," Clust. Comput., vol. 20, no. 1, pp. 2669–2685, 2017.

[5] S. Huber, H. Wiemer, D. Schneider, S. Ihlenfeldt, "DMME: Data mining methodology for engineering applications—a holistic extension to the CRISP-DM model," Procedia CIRP 2019, vol. 79, pp. 403–408. In Proceedings of the 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 18–20 July, 2018.

[6] U. Fayyad, G. Piatetsky-Shapiro, P. Smith, "From Data Mining to Knowledge Discovery in Databases," Commun. ACM., vol. 39, pp. 37–54, 1996.

[7] G. B. Hima Bindu, K. Ramani, and C. Shoba Bindu, "Optimized Resource Scheduling using the Meta Heuristic Algorithm in Cloud Computing," IAENG International Journal of Computer Science, vol. 47, no.3, pp. 360-366, 2020.

[8] X. Y. Tian, Q.H. Zheng, and N. Jiang, "An Abnormal Behavior Detection Method Leveraging Multi-modal Data Fusion and Deep Mining," IAENG International Journal of Applied Mathematics, vol. 51, no.1, pp. 92-99, 2021.

[9] G. Galli , C. Patrone, A. C. Bellam, N. R. Annapareddy, and R. Revetria, "Improving process using digital twin: a methodology for the automatic creation of models," In Lecture Notes in Engineering and Computer Science: Proceedings of the World Congress on Engineering and Computer Science, pp. 22-24, 2019.

[10] M. Nijim, H. Albataineh, "En-Stor: Energy-Aware Hybrid Mobile Storage System using Predictive Prefetching and Data Mining Engine," Engineering Letters, vol. 26, no. 2, pp. 252-256, 2018.

[11] M. Kalra, S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," Egypt. Inform. J., vol.16, pp. 275–295, 2015.
[12] M. Kumar, S. Sharma, A. Goel, S. Singh, "A comprehensive survey for scheduling techniques in cloud computing," J. Netw. Comput. Appl., vol. 143, 1–33, 2019.
[13] J. Tang, C. Huang, H. Liu, N. Al-Nabhan, " Cloud Storage Strategy of Blockchain Based on Genetic Prediction Dynamic Files," Electronics, vol. 9, pp. 398, 2020.
[14] J. M. Do, Y. J. Song, N. Park, "Attribute based proxy re-encryption for data confidentiality in cloud computing environments," In Proceedings of the 2011 First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering (CNSI) 2011, 23-25 May, 2011, pp. 248–251.
[15] M. N. Birje, P. S. Challagidad, R. H. Goudar, M. T. Tapale, "Cloud computing review: concepts, technology, challenges and security," IEEE Trans. on Cloud Comput., vol. 6, pp. 32-57, 2017. P. Lemenkova, "Numerical Data Modelling and Classification in Marine Geology by the SPSS Statistics," Int. J. Eng. Technol., vol. 5, pp. 90–99, 2019.
[16] S. Gharehpasha, M. Masdari, A. Jafarian, "Power efficient virtual machine placement in cloud data centers with a discrete and chaotic hybrid optimization algorithm," Cluster Comput., vol. 24, pp. 1293-1315, 2021.
[17] M. Tarahomi, M. Izadi, M. Ghobaei-Arani, "An efficient power-aware VM allocation mechanism in cloud data centers: a micro genetic-based approach," Cluster Comput., vol. 24, pp. 919-934, 2021.
[18] A. Khosravi, L. L. Andrew, R. Buyya, "Dynamic vm placement method for minimizing energy and carbon cost in geographically distributed cloud data centers," IEEE Trans. Sustain. Comput., vol. 2, pp. 183-196, 2017.
[19] S. Kikuchi, Y. Matsumoto, "Impact of live migration on multi-tier application performance in clouds," In Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, pp. 261-268, 2012.
[20] D. Jiang, P. Huang, P. Lin, J. Jiang, "Energy efficient VM placement heuristic algorithms comparison for cloud with multidimensional resources," In Proceedings of the International Conference on Information Computing and Applications, vol. 7473, pp. 413-420, 2012.
[21] F. N. Afrati, N. Stasinopoulos, J. D. Ullman, A. Vassilakopoulos, Sharesskew, "An algorithm to handle skew for joins in mapreduce," Inf. Syst., vol. 77, pp. 129-150, 2018.
[22] J. Pinto, P. Jain, T. Kumar, "Fault prediction for distributed computing Hadoop clusters using real-time higher order differential inputs to SVM: Zedacross," Int. J. Inf. Comput. Secur., vol. 12, pp. 181-198, 2020.
[23] G. Fylaktopoulos, M. Skolarikis, I. Padadopoulos, G. Goumas, A. Sotiropoulos, I. Maglogiannis, "A distributed modular platform for the development of cloud based applications," Future Gener. Comput. Syst., vol. 78, pp. 127–141, 2018.
[24] M. Jensen, J. Schwenk, N. Gruschka, L. L. Iacono, "On technical security issues in cloud computing," In Proceedings of the IEEE International Conference on In Cloud Computing CLOUD'09, pp. 109-116, 2009.
[25] G. Yang, M. Jiang, W. Ouyang, G. Ji, H. Xie, A. M. Rahmani, H. Tenhunen, "IoT-based remote pain monitoring system. From device to cloud platform," IEEE J. Biomed. Health. Inform., vol. 22, pp. 1711-1719, 2017.
[26] B. Qureshi, S. Alwehaibi, A. Koubaa, "On power consumption profiles for data intensive workloads in virtualized Hadoop clusters," In Proceedings of the 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 42-47, 2017.
[27] S. Ibrahim, "Governing energy consumption in Hadoop through CPU frequency scaling: An analysis," Future Gener. Comput. Syst., vol. 54, pp. 219-232, 2016.
[28] U. Shafique, H.A. Qaiser, "Comparative Study of Data Mining Process Models (KDD, CRISP-DM and SEMMA)," Int. J. Innov. Sci. Res., vol. 12, pp. 217-222, 2014.
[29] F. F. Malavelle, J. M. Haywood, L. M. Mercado, G. A. Folberth, N. Bellouin, S. Sitch, P. Artaxo, "Studying the impact of biomass burning aerosol radiative and climate effects on the Amazon rainforest productivity with an Earth system model," Atmos. Chem. Phys., vol 19, pp. 1301-1326, 2019.
[30] C. G. Derington, T. H. Gums, A. P. Bress, J. S. Herrick, T. H. Greene, A. E. Moran, J. J. Saseen, "Association of Total Medication Burden with Intensive and Standard Blood Pressure Control and Clinical Outcomes: A Secondary Analysis of SPRINT," Hypertension., vol. 74, pp. 267-275, 2019.
[31] Z. Guo, M. Liu, H. Qin, B. Li, "Mechanical Fault Diagnosis of a DC Motor Utilizing United Variational Mode Decomposition, SampEn, and Random Forest-SPRINT Algorithm Classifiers," Entropy., vol. 21, pp. 470, 2019.
[32] A. Bechini, F. Marcelloni, A. Segatori, "A MapReduce solution for associative classification of big data," Inf. Sci., vol. 332, pp. 33–55, 2016.
[33] L. Leydesdorff, C. S. Wagner, L. Bornmann, "Interdisciplinarity as diversity in citation patterns among journals: Rao-Stirling diversity, relative variety, and the Gini coefficient," J. Informetr., vol. 13, pp. 255–269, 2019.
[34] P. David, "Optimization of Gini Coefficient Affected by Imperfect Input Data," Eur. J. Bus. Sci. Technol., vol. 5, pp. 21-29, 2019.
[35] R. Valbuena, K. Eerikäinen, P. Packalen, M. Maltamo, "Gini coefficient predictions from airborne lidar remote sensing display the effect of management intensity on forest structure," Ecol. Indic., vol. 60, pp. 574-585, 2016.