Comput Syst Sci & Eng (2020) 6: 523–532

International Journal of
Computer Systems
Science & Engineering

A Data-Aware Remote Procedure Call Method for Big Data Systems

Jin Wang[1,2], Yaqiong Yang[1], Jingyu Zhang[1,3*], Xiaofeng Yu[4], Osama Alfarraj[5] and Amr Tolba[5,6]

[1]*School of Computer & Communication Engineering, Changsha University of Science & Technology, Changsha, 410000, China*

[2]*School of Information Science and Engineering, Fujian University of Technology, Fujian, 350000, China*

[3]*School of Systems Engineering, National University of Defense Technology, Changsha, 410000, China*

[4]*School of Business, Nanjing University, Nanjing, 210093, China*

[5]*Computer Science Department, Community College, King Saud University, Riyadh, 11437, Saudi Arabia*

[6]*Mathematics and Computer Science Department, Faculty of Science, Menoufia University, Shebin-El-kom, 32511, Egypt*

In recent years, big data has been one of the hottest development directions in the information field. With the development of artificial intelligence technology, mobile smart terminals and high-bandwidth wireless Internet, various types of data are increasing exponentially. Huge amounts of data contain a lot of potential value, therefore how to effectively store and process data efficiently becomes very important. Hadoop Distributed File System (HDFS) has emerged as a typical representative of data-intensive distributed big data file systems, and it has features such as high fault tolerance, high throughput, and can be deployed on low-cost hardwares. HDFS nodes communicate with each other to make the big data systems work properly, using the Remote Procedure Call (RPC) mechanism. However, the RPC in HDFS is still not good enough to work better in terms of network throughput and abnormal response. This paper presents an optimization method to improve the performance of HDFS. The proposed method dynamically adjusts the RPC configurations between NameNode and DataNodes by sensing the data characters that stored in DataNodes. This method can effectively reduce the NameNode processing pressure, and improve the network throughput generated by the information transmission between NameNode and DataNodes. It can also reduce the abnormal response time of the whole system. Finally, the extensive experiments show the effectiveness and efficiency of our proposed method.

Keywords: Distributed file systems; HDFS; information interaction; adaptive RPC mechanism

1. INTRODUCTION

With the continuous development of the big data era, the data available to access in recent years has been increasing exponentially [1-3]. Due to the huge data scale, data processing cannot be realized on a single computer, therefore the distributed data processing system architecture has been adopted [4-5]. Figure 1 shows the current typical big data processing architecture. In the figure, the research on distributed file storage systems in the whole big data framework is becoming more and more important, due to the increasing demand for massive data sets storage with different data types [6-8]. Distributed file systems can effectively solve the problem of large-scale data storage and management [9]. Google File System (GFS) is an extensible distributed file system for large, distributed, data-intensive applications [10]. Hadoop Distribute File System (HDFS) [11], which is widely used at present. HDFS is a part of Apache Hadoop core project, and its design ideas refer to the GFS system [12]. As an underlying distributed file system framework widely used in recent big data services, HDFS can help to improve the quality of service, the system performance and reliability of data intensive applications [13]. It has received more and more attention from academia and industry.

* Jingyu Zhang. Email: zhangzhang@csust.edu.cn

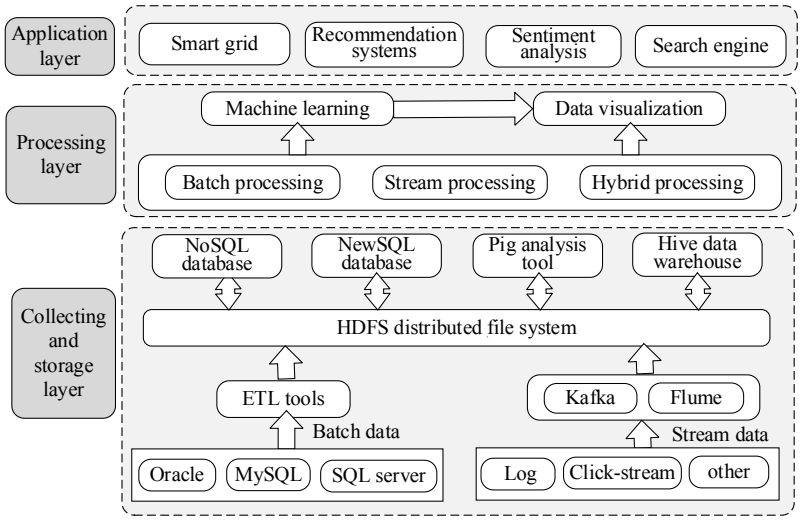


Figure 1 Big data processing architecture.

The HDFS uses the Master/Slave system model. An HDFS cluster is usually composed of one NameNode and multiple DataNodes [14]. NameNode is mainly used to manage and maintain the namespace in the distributed file system, and is responsible for the mapping information and access operations between data files and DataNodes. The information on the NameNode is stored in the hard disk in the form of namespace mirror file and editing logs [15]. Figure 2 shows the system architecture of HDFS. At present, the remote communication and mutual calls between different nodes in the distributed system are based on Remote Procedure Call (RPC) [16]. RPC is a remote communication service in a distributed system. In a distributed file system, the heartbeat mechanism in the underlying RPC framework is very important for the information interaction between the entire system nodes, and our mainly focus is on the RPC heartbeat mechanism in this paper. In HDFS systems, the nodes call the RPC function periodically [17]. DataNodes regularly report node status and task running information to NameNode through fixed RPC heartbeat interval, and NameNode performs the whole system resource scheduling by obtaining resource usage and task running status information of each DataNode [18].

Currently, with the continuous expansion of the distributed system scale, the dependence of components in the system is also becoming more and more complicated, which makes the probability of system failures increasing. Therefore, fault-tolerant processing becomes a key issue in distributed systems [19-22]. Among the fault detection methods, RPC heartbeat detection algorithm is one of the most commonly used methods. In distributed systems, the main communication mechanism to transmit information between nodes is the RPC heartbeat mechanism. The heartbeat detection mechanism is usually used to probe the node failure, which is also one of the necessary ways to ensure the reliability and stability of the distributed systems [23]. Some related work is devoted to this area to improve the heartbeat mechanism. For instance, the paper [24] proposed two heartbeat mechanisms: adaptive interval and reputation-based detector. An adaptive interval can dynamically set the expiration time that is compatible with the job size. A reputation-based detector is used to evaluate the reputation of each worker. Once a worker's reputation is below the threshold, the worker will be considered a failed worker. Another work [25] proposed an energy saving scheme by converting heartbeat energy waste into useful data transmission, so as to realize energy saving and data fault tolerance in various application scenarios. In recent years, some other related researches have also been paid attention in other scenarios [26-34].

This paper proposes an adaptive RPC heartbeat interval method, which configures the heartbeat interval of the node in the running cycle according to the amount of data stored in the DataNodes, and realizes the dynamic adjustment of the node heartbeat sending mechanism. In our work, the network throughput generated by the information interaction between NameNode and DataNodes through the heartbeat mechanism can be reduced, and the abnormal response time of the whole system can be improved, so as to effectively improve the comprehensive performance of distributed systems.

In the following sections of this paper, section 2 describes how the distributed file system nodes interact with information. Section 3 introduces the proposed HDFS adaptive RPC heartbeat theoretical model; Section 4 introduces the design of adaptive HDFS heartbeat algorithm proposed in our work. Section 5 shows the experimental results and their qualitative and quantitative performance analysis. The last part summarizes our work in this paper.

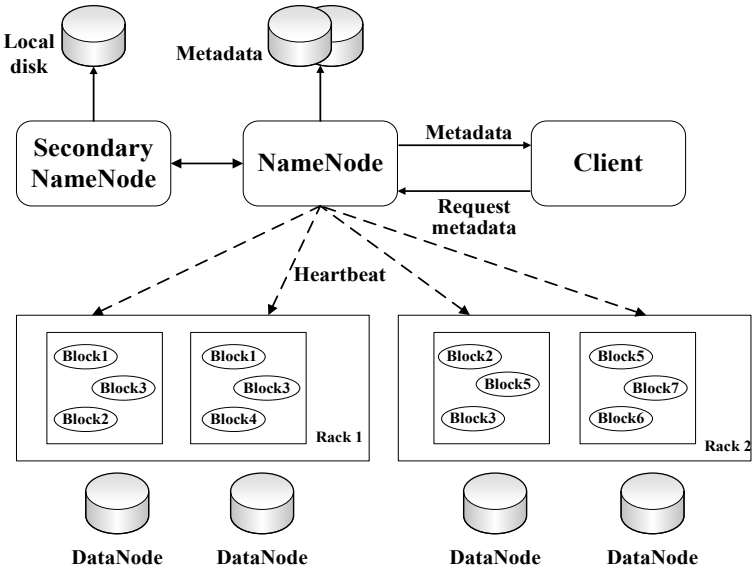


Figure 2 HDFS system architecture.

2. RELATED WORK

In this section, we review the related work. Distributed systems have many different general physical and logical computer resources. Since it is a software system built on the network, the computing nodes are distributed in various places, and they exchange the information through the computer network [35-36]. First, we introduce the related work about communication mode between NameNode and DataNode in an HDFS system. Then, we present some other finished work about the HDFS heartbeat triggering mechanism.

2.1 Communication Mode between NameNode and DataNode

In distributed file systems, the Master-Slave nodes usually interact with each other through the RPC heartbeat mechanism. The slave nodes (i.e., DataNodes) report the storage capacity and other state information to the master node (i.e., NameNode) through heartbeat packet transmission. The master node will aggregate all DataNode information. When the client sends file operation request to the system, NameNode will then send user request instruction to the slave nodes for corresponding file operations [37]. In general, there are two common heartbeat transmission modes in distributed systems: Pull and Push. The comparison of the heartbeat transmission modes is shown in Table 1.

Push mode: The master node actively sends heartbeat information to the slave nodes at a certain time interval. If the information contains a user operation request, the slave nodes start to execute the user instruction at the same time and respond to the master node with heartbeat information. If the master node does not receive the reply message from a slave node in more than a certain duration, it is judged that the node fails [38].

Pull mode: The slave nodes actively send heartbeat information to the master node at a certain time interval [39]. The information contains the status information of the slave nodes (such as the location information of the storage node, the current available disk space, etc.). After receiving the heartbeat information of the slave nodes, the master node updates the current system state [40]. If the master node does not receive the reply message from the slave nodes in more than a certain duration, it is judged that the node may fail. The HDFS system uses the Push model to complete the communications between different connected nodes.

Table 1 The comparison of the heartbeat transmission modes.

Heartbeat mechanism	Push	Pull
State preservation	Centralized; Focus on the master node	Distributed; Focus on the slave node
Real-time	Update in time	Depends on the heartbeat interval
Advantage	High timeliness; Low user requirements	System handling simplicity; Small amount of information transmission on a single node
Disadvantage	Low system fault tolerance; High storage pressure on a master node	Increased network load

2.2 Node Heartbeat Triggering Mechanism

The heartbeat mechanism can effectively complete the information interaction between master and slave nodes in distributed system. Generally, there are two driving modes to activate this mechanism: one is to send heartbeat information periodically, and the other is to send event-driven heartbeat information.

1) Periodic heartbeat mechanism: Our work is using this mechanism, and it is used to send heartbeat information periodically between master and slave nodes. The communication between master and slave nodes can only be carried out at fixed intervals, which is convenient for system management and maintenance [41]. In HDFS, the system usually implements the heartbeat transmission every 3 seconds.

2) Event-driven heartbeat mechanism: This method no longer adopts the periodic heartbeat mechanism. Instead, it determines whether a node sends a heartbeat based on whether an event occurs [42]. The events that can be used as drivers include: user task request, task starts, task waits, task completes, etc.

3. ADAPTIVE HEARTBEAT MODEL

Different distributed systems set the node information communication method according to their own requirements. For most distributed systems, the periodic heartbeat transmission mechanism is adopted to control the node interactions. In this section, we present the adaptive heartbeat model used in our proposed optimization method.

When there are too many nodes in the system, sending heartbeats periodically may cause the master node to process too much information [43]. Since the information interaction between the system nodes using event-driven heartbeat mechanism requires the triggering of corresponding events, when the node fails, the fault node can not timely report the node situation to the system manager, which reduces the system performance. To this end, this paper proposes an adaptive heartbeat mechanism suitable for node information exchange in HDFS systems.

3.1 HDFS Heartbeat Model Based on Data Awareness

HDFS is designed to support the processing of large batch file data. HDFS divides user-uploaded files into multiple blocks of the same size as storage units. Due to the different upload locations of user data, the number of data blocks stored on different nodes in the system may be unevenly distributed over time [44]. Data imbalance may cause problems such as untimely information interaction between master and slave nodes. To this end, this paper proposes a HDFS heartbeat model based on data awareness. NameNode and DataNodes follow the mechanism to communicate. At first, the user client sends a request to the master node, and the master node replies with a list of slave nodes to the client. Then, the client begins to send data to the slave nodes. When a slave node executes a big data task, it reports the task execution status to the master node according to the dynamically adjusted heartbeat time interval. Figure 3 shows the processing flow of the proposed adaptive heartbeat interval setting method.

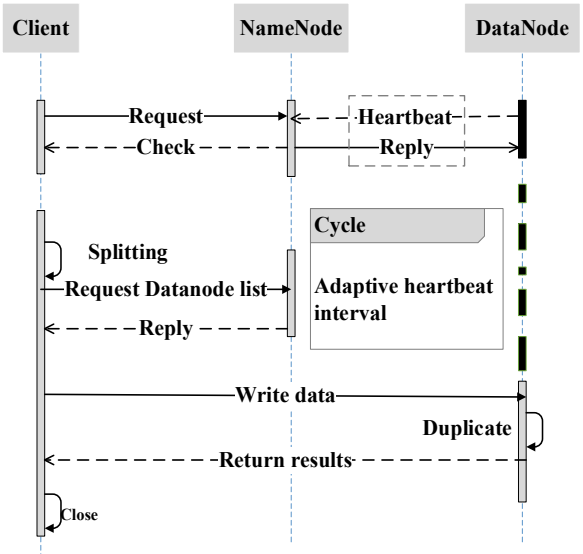


Figure 3 Big data processing architecture.

Assuming that the number of data blocks stored by node i in k cycles is relatively small, it is considered that the visiting popularity of this node may be low. Then the probability that the user will visit the node will also be reduced, and also the probability of the node's state change will also be decreased. At this time, the heartbeat sending interval of this node can be appropriately increased, which can reduce the network throughput of heartbeat transmission between master and slave nodes. On the contrary, the heartbeat sending interval can be appropriately reduced in the distributed systems, so that the node update

information of the master node can be obtained more quickly, and the real-time performance of data update information in the cluster can be further enhanced.

Here we present the mathematical adaptive heartbeat model, and the definition of relevant symbols is shown in Table 2. In this paper, a running cycle means a specific and fixed running time duration of the HDFS system. Node i counts the number of data blocks in a jth running cycle, $D_{ij}(i = 1,2,\cdots,N; j = 1,2,\cdots,k)$. We assume that D_{ij} will affect the heartbeat sending time interval t_{ij} of node i in the current cycle. The greater the value of D_{ij}, the shorter the heartbeat sending interval t_{ij} in jth cycle, and the faster the heartbeat sending frequency. Conversely, the smaller the D_{ij}, the longer the heartbeat sending interval t_{ij} in jth cycle, and the slower the heartbeat sending frequency. In our model, M refers to the expected mean number of data blocks stored by all nodes in the cluster derived from the historical records, and it has the following form:

$$M = E(D_{ij}) = \frac{1}{N}\sum_{i=1}^{N} D_{ij} \tag{1}$$

DataNode uses D_{ij} in jth cycle to calculate the heartbeat sending interval t_{ij} of the node. Using formula (1), the calculation method of the heartbeat interval for node i is:

$$t_{ij} = round\{t' \cdot \frac{\omega \cdot M}{D_{ij}}\} \tag{2}$$

In the model, t' refers to the initial system RPC heartbeat configuration. The heartbeat factor ω is used to control the influence of the mean value M on the heartbeat interval to avoid excessive changes in the heartbeat interval and increase the burden on the system. $round\{\}$ is a function for rounding, which can be used to simplify system RPC configuration.

3.2 Adaptive Heartbeat Interval Setting

The mathematical model dynamically adjusts the heartbeat sending interval of each DataNode by sensing the stored data. Assume that the length of the heartbeat information sent between nodes is the same without considering other influencing factors in the interior HDFS network, in jth cycle, the system network load Q_{ij} generated by all the nodes is:

$$Q_{ij} = \sum_{i=1}^{N} \frac{T}{t_{ij}} \cdot L = \sum_{i=1}^{N} \frac{T}{round\{t' \cdot \frac{\omega \cdot M}{D_{ij}}\}} \cdot L \ (i = 1,2,\cdots,N; j = 1,2,\cdots,k) \tag{3}$$

where T is the interval time of one running cycle, and $\sum_{i=1}^{N} \frac{T}{t_{ij}}$ is the total number of heartbeat packets sent by the DataNodes in the system.

For distributed systems, the fault tolerance of the system is also crucial. The time interval between two continuous heartbeats determines the system response time for node fault. If the NameNode receives no heartbeats from one DataNode within a heartbeat interval, the DataNode is considered to be in an abnormal state. Normally, the abnormal response time of a distributed system equals the heartbeat interval. Assuming that the number of data blocks in the node i is large, it has a high probability that it will affect the completion of the task in the node when an abnormal state occurs. Conversely, if the number of data blocks is small, the task affects probability when an abnormal state occurs will also decrease. For this reason, in jth cycle, the total abnormal response time of the system is:

$$T_{ART} = \sum_{i=1}^{N} \eta_{ij} \cdot t_{ij} = \sum_{i=1}^{N} \frac{D_{ij}}{\sum_{i=1}^{N} D_{ij}} \cdot round\left\{t' \cdot \frac{\omega \cdot M}{D_{ij}}\right\} \ (i = 1,2,\cdots,N; j = 1,2,\cdots,k) \tag{4}$$

In order to obtain better overall system performance, the network load Q of the system and the abnormal response time T_{ART} are needed to be optimized. To sum up, we define the system optimization function with the adaptive heartbeat interval as:

$$W = \frac{Q}{Q'} + \frac{T_{ART}}{T'_{ART}} = \frac{\sum_{i=1}^{N} \frac{T}{t_{ij}} L}{\sum_{i=1}^{N} \frac{T}{t'} L} + \frac{\sum_{i=1}^{N} \frac{D_{ij}}{\sum_{i=1}^{N} D_{ij}} \cdot t_{ij}}{\sum_{i=1}^{N} \frac{D_{ij}}{\sum_{i=1}^{N} D_{ij}} \cdot t'} \tag{5}$$

In above mathematical model, Q' and T'_{ART} respectively refer to the network load and total system abnormal response time in a typical HDFS system using the default fixed heartbeat interval t' in each running cycle. Our goal of this article is to get a smaller W. When $\frac{Q}{Q'}$ and $\frac{T_{ART}}{T'_{ART}}$ are both less than 1, it is considered that our proposed method outperform the compared system method with the default settings.

Table 2 The definition of relevant symbols.

Symbol	Description
N	The number of nodes in the system
D_{ij}	The number of data blocks stored by node i in jth cycle
M	The expected mean number of data blocks stored by all nodes in the cluster derived from the historical records
ω	Heartbeat factor
$round\{\}$	Rounding function
t_{ij}	The adaptive heartbeat interval time
Q_{ij}	The network load of node i in jth cycle

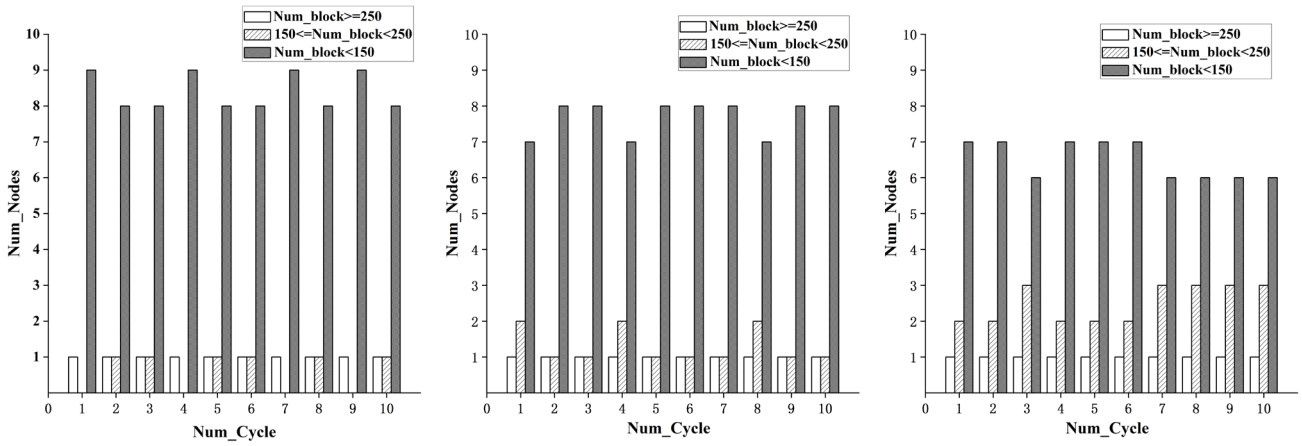
Q'	The default network load
T_{ART}	The abnormal response time
T'_{ART}	The default abnormal response time
W	System evaluation function

4. EXPERIMENTAL ANALYSIS

In this section, we verify the remote procedure call optimization method in big data systems based on data awareness through extensive experiments. The proposed method can adjust the heartbeat transmission interval between NameNode and DataNodes in a HDFS system in our designed experiments, and it can reduce the network throughput generated by the system information communication and the node abnormal response time.

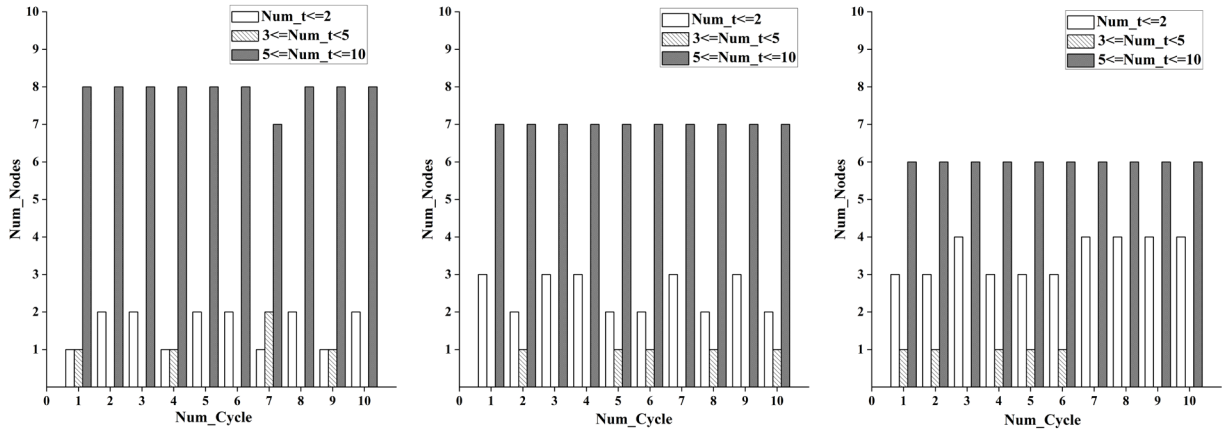
To validate the node effect on the heartbeat sending interval, we first conducted experiments with 10 experimental sets. Each experimental set records the D_{ij} value of every node in one running cycle. We set up 10 running cycles, each cycle T has 30 minutes. In each running cycle, we upload 1000 data blocks the HDFS cluster, and each data block size is 128M. These 1000 data blocks are randomly distributed and stored in 10 DataNodes in the cluster. In order to study the possibility of different data allocation, we simulate the cluster with different allocation conditions in 3 cases. For Case 1, 1000 data blocks are mainly distributed in 10%-20% nodes in the cluster. For Case 2 and Case 3, the distribution probabilities are 20%-30% and 30%-40% respectively. The data block distributions in our experiments are shown in Figure 4.

According to the node storage situation of data blocks in the cluster in each running cycle, the proposed method dynamically adjusts heartbeat interval time of different nodes. In the experiments, the average number M of data blocks in the cluster is 100, and the value of heartbeat factor is 1.2. In order to avoid excessively long heartbeat intervals between nodes when using the proposed adaptive heartbeat method, in our design the maximum heartbeat interval is 10s.



(a) 10%-20% (b) 20%-30% (c) 30%-40%

Figure 4 Data block distribution in different cycles.

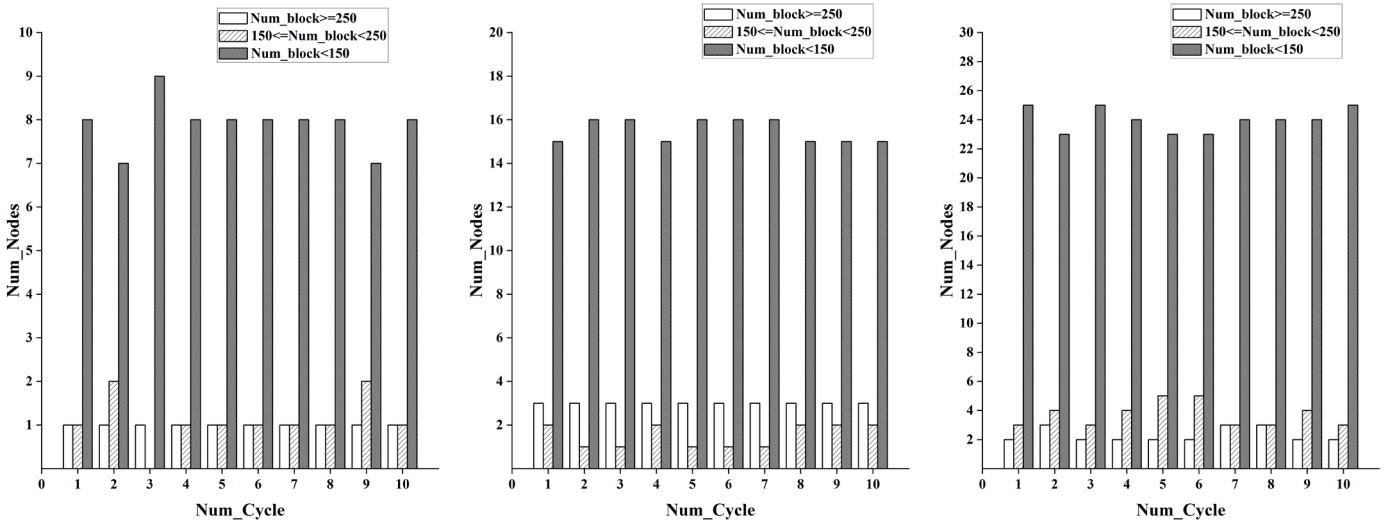


(a) 10%-20% (b) 20%-30% (c) 30%-40%

Figure 5 Distribution of heartbeat sending interval in different cycles.

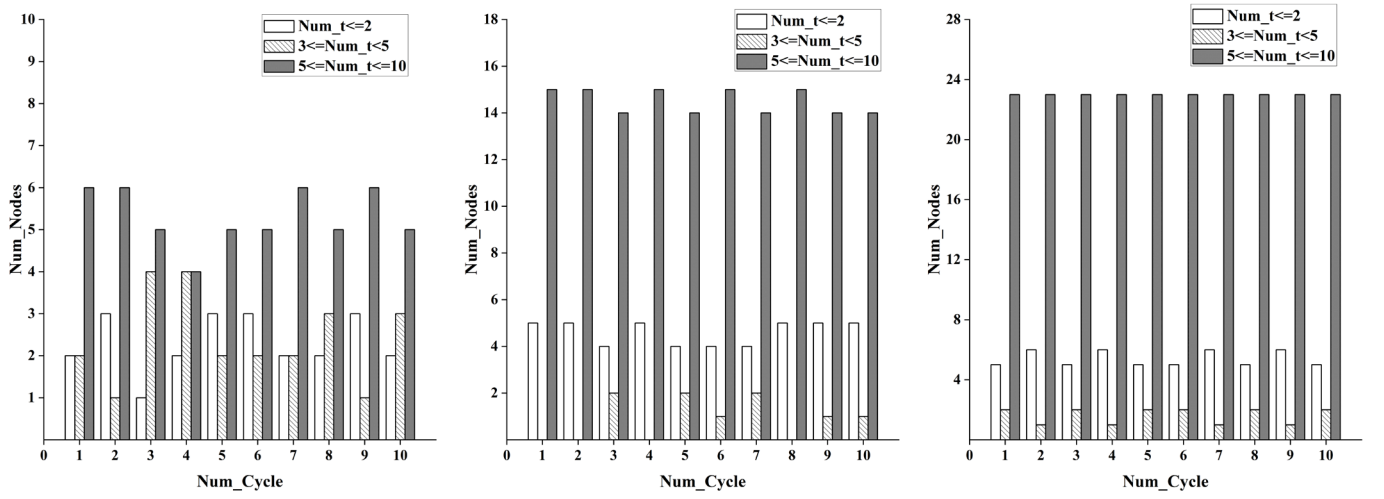
It can be seen from Figure 5 that for Case 1 in 10 running cycles, the node number with heartbeat interval from 5s to 10s ranges from 7 to 8. The node number with heartbeat interval from 3s to 5s ranges from 0 to 1, and the node number with heartbeat interval less than 3s ranges from 1 to 2. The uploaded data blocks in Case 1 are mainly distributed in 10%-20% nodes. It can be seen from Figure 5, the node number with reduced heartbeat interval after the adjustment of Case 1 is also within 20%. Similarly, the node number with reduced heartbeat interval in Case 2 is within 30%, and the node number with reduced heartbeat interval in Case 3 is also within 40%.

In addition, in order to study the influence of block number stored by nodes in clusters with different heartbeat sending intervals. We also simulated three different clusters in three cases. For Case 4, 10 nodes are distributed in the cluster. For Case 5 and Case 6, the node numbers are 20 and 30 respectively. The experimental configuration is the same as Case 2, which has ten 30-minute running cycles. In each running cycle, 2, 000 blocks are uploaded to the different clusters in the three cases. 2, 000 data blocks are mainly distributed in 20%-30% nodes in the cluster. The mean M of each cluster is calculated by the historical number records, and the heart factor was 1.2. The designed experiments are used to observe the effect of node storage in different clusters on the heartbeat interval transmission. The distribution of data blocks in the cluster is shown in Figure 6, and the distribution of heartbeat transmission interval is shown in Figure 7.



(a) 10 nodes (b) 20 nodes (c) 30 nodes

Figure 6 Data block distribution in different cycles.



(a) 10 nodes (b) 20 nodes (c) 30 nodes

Figure 7 Distribution of heartbeat sending interval in different cycles.

As can be seen from Figure 6 and Figure 7, although the cluster scale is different in the three cases, the data block distributions have a relatively consistent effect on the heartbeat sending interval. For nodes with more data blocks, the heartbeat sending interval time decreases. On the contrary, nodes with less data block distribution increase the time between two RPC heartbeats.

The default heartbeat interval of HDFS distributed systems is 3s. If the DataNode sends a heartbeat packet to the NameNode at 3s interval, the network throughput generated by a node in a running cycle is $\frac{T}{t} \cdot L = 600$ heartbeat packets. The system overall network throughput generated in each running cycle is 600*10 = 6000 heartbeat packets, and in 10 running cycles 600*10*10 = 60000 heartbeat packets will be generated in the system.

As shown in Figure 6, by dynamically adjusting the heartbeat sending interval of different nodes, the network throughput generated by Cases 1, 2, 3 in each cycle is less than 6000. Due to the different distribution of data blocks in Cases 1, 2, 3, the node numbers with reduced heartbeat interval are also different. According to the experiments, the network throughputs generated by the three cases in each running cycle are different. The total network throughputs of Cases 1, 2, 3 in 10 cycles are 41598, 48613, 54864 respectively. From the above results, the node number with reduced heartbeat sending interval in each running cycle follows the order: Case 1 < Case 2 < Case 3. Also, the generated network loads follows the following order: Case 1 < Case 2 < Case 3.

Using the adaptive adjusting of the heartbeat sending interval, the abnormal response time of the system also changes. In our experiments, the default heartbeat interval to detect the abnormal behaviors is 3s. As shown in Figure 6, the average abnormal response times of Cases 1, 2, 3 are all less than the default abnormal response time. From our designed experiments, the node number with reduced heartbeat sending interval in each cycle follows the order: Case 1 < Case 2 < Case 3, and the average abnormal response time order is: Case 1 > Case 2 > Case 3. Via our comprehensive experiments, the proposed method improves both the network throughput and abnormal response time of different testing cases in different running cycles, and experimental performance comparisons are shown in Figure 8.

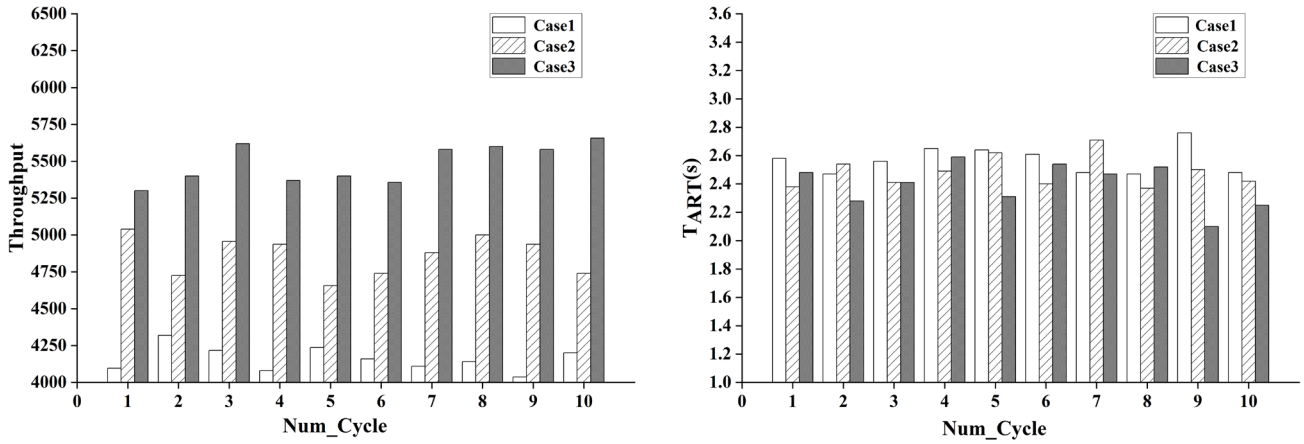


Figure 8 Network throughput and Abnormal response time.

5. CONCLUSION

With the rapid increase of the information data in big data systems, the traditional RPC-based periodic information interaction method between master and slave nodes can no longer meet the requirements in terms of network throughput and abnormal response. This paper presents a data-aware RPC optimization method to improve HDFS performance based on data sensing. Due to the different upload locations of users, the number of data blocks stored in DataNodes may be unevenly distributed over time in the HDFS system. The proposed method is aware of perceives the data blocks stored in each DataNode, and it can dynamically adjust the heartbeat sending time interval of different DataNodes in different running cycles. The adaptive RPC heartbeat method can effectively reduce the processing pressure of the NameNode. Through out designed experiments, the proposed method can reduce the network throughput generated by the information transmission between NameNode and DataNodes, and it also improves the overall abnormal response time. Experimental results show that this method can effectively optimize the performance for distributed systems.

ACKNOWLEDGEMENT

We thank Researchers Supporting Project (No. RSP-2020/102) King Saud University, Riyadh, Saudi Arabia, for funding this research. We would also thank the support from the National Natural Science Foundation of China (Nos. 61772454, 61811530332, 61811540410, 61802031), the Natural Science Foundation of Hunan Province, China (No. 2019JGYB177), the Research Foundation of Education Bureau of Hunan Province, China (No. 18C0216), the "Practical Innovation and Entrepreneurial Ability Improvement Plan" for Professional Degree Graduate students of Changsha University of Science and Technology (No. SJCX201971) and Hunan Graduate Scientific Research Innovation Project, China (No. CX2019694). We appreciate the help of the Programs of Transformation and

Upgrading of Industries and Information Technologies of Jiangsu Province (No. JITC-1900AX2038/01).

FUNDING STATEMENT

This work is supported by Researchers Supporting Project (No. RSP-2020/102) King Saud University, Riyadh, Saudi Arabia, the National Natural Science Foundation of China (No. 61802031, 61772454, 61811530332, 61811540410), the Natural Science Foundation of Hunan Province, China (No. 2019JGYB177), the Research Foundation of Education Bureau of Hunan Province, China (No. 18C0216), the "Practical Innovation and Entrepreneurial Ability Improvement Plan"" for Professional Degree Graduate students of Changsha University of Science and Technology (No. SJCX201971) and Hunan Graduate Scientific Research Innovation Project, China (No. CX2019694). This work is also supported by the Programs of Transformation and Upgrading of Industries and Information Technologies of Jiangsu Province (No. JITC-1900AX2038/01).

CONFLICTS OF INTEREST

The authors declare no conflicts of interest to report regarding the present study.

REFERENCES

1. I. M. El-Hasnony, S. I. Barakat, M. Elhoseny and R. R. Mostafa, "Improved feature selection model for big data analytics," *IEEE Access*, vol. 8, pp. 66989-67004, 2020.
2. Y. Chen, M. Zhou and Z. Zheng, "Time-aware smart object recommendation in social Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 2014-2027, 2019.
3. Y. Chen, M. Zhou and Z. Zheng, "Learning sequence-based fingerprint for magnetic indoor positioning system," *IEEE Access*, vol. 7, pp. 163231-163244, 2019.
4. A. Shukla and Y. Simmhan, "Model-driven scheduling for distributed stream processing systems," *Journal of Parallel and Distributed Computing*, vol. 117, pp. 98-114, 2018.
5. Y. Chen, M. Zhou and Z. Zheng, "Toward practical crowdsourcing-based road anomaly detection with scale-invariant feature," *IEEE Access*, vol. 7, pp. 67666-67678, 2019.
6. J. Wang, Y. Yang, T. Wang, R. S. Sherratt and J. Zhang, "Big data service architecture: a survey," *Journal of Internet Technology*, vol. 21, pp. 393-405, 2020.
7. M. N. Oo, S. Parvin and T. Thein, "Forensic investigation through data remnants on hadoop big data storage system," *Computer Systems Science and Engineering*, vol. 33, no. 3, pp. 203-217, 2018.
8. L. F. Yuan, E. L. Yao and G. M. Tan, "Automated and precise event detection method for big data in biomedical imaging with support vector machine," *Computer Systems Science and Engineering*, vol. 33, no. 2, pp. 105-114, 2018.
9. H. Sim, A. Khan, S. S. Vazhkudai, S. Lim and A. R. Butt, "An integrated indexing and search service for distributed file systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 10, pp. 2375-2391, 2020.
10. S. Ghemawat, H. Gobioff and S. Leung, "The google file system," in *Proc.SOSP*, Bolton Landing, NY, USA, pp. 29-43, 2003.
11. M. H. Hajeer and D. Dasgupta, "Handling big data using a data-aware HDFS and evolutionary clustering technique," *IEEE Transactions Big Data*, vol. 5, no. 2, pp. 134-147, 2019.
12. K. Shvachko, H. Kuang, S. Radia and R. Chansler, "The Hadoop distributed file system," in *Proc. MSST*, Lake Tahoe, Nevada, USA, pp. 1-10, 2010.
13. T. Ma, F. Tian and B. Dong, "Ordinal optimization-based performance model estimation method for HDFS," *IEEE Access*, vol. 8, pp. 889-899, 2020.
14. D. Han and B. Nam, "Improving access to HDFS using NVMeoF," in *Proc. CLUSTER*, Albuquerque, USA, pp.1-2, 2019.
15. A. U. Khan and S. Bagchi, "Software architecture and algorithm for reliable RPC for geo-distributed mobile computing systems," *Future Generation Computer Systems*, vol. 86, pp. 185-198, 2018.
16. Y. Wu, T. Ma, M. Su, M. Zhang and K. Chen, "RF-RPC: remote fetching RPC paradigm for RDMA-Enabled network," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 7, pp. 1657-1671, 2019.
17. H. H. Le, S. Hikida and H. Yokota, "Namenode and datanode coupling for a power-proportional Hadoop distributed file system," in *Proc. DASFAA*, Wuhan, China, pp. 99-107, 2013.
18. A. Ganesan, R. Alagappan, A. C. Arpaci-Dusseau and R. H. Arpaci-Dusseau, "Redundancy does not imply fault tolerance: analysis of distributed storage reactions to file-system faults," *ACM Transactions Storage*, vol. 13, no. 3, pp. 20:1-20:33, 2017.
19. B. Memishi, S. María, P. Hernández and G. Antoniu, "Enhanced failure detection mechanism in MapReduce," in *Pro. HPCS*, Madrid, Spain, pp. 690-692, 2012.
20. Q. Zhang, S. Yang and M. Liu, "A new crossover mechanism for genetic algorithms for steiner tree optimization," *IEEE Transactions on Cybernetics*, vol. 99, pp. 1-12, 2020.
21. Q. Zhang, L. Ding and Z. Liao, "A novel genetic algorithm for stable multicast routing in mobile ad hoc networks," *China Communications*, vol. 16, no. 8, pp. 24-37, 2019.
22. Y. Chen, J. Tao, Q. Zhang, K. Yang, X. Chen *et al.*, "Saliency detection via improved hierarchical principle component analysis method," *Wireless Communications and Mobile Computing*, 2020.

23. T. Zhang, X. Zhang, F. Liu, H. Leng and Q. Yu, "ETrain: making wasted energy useful by utilizing heartbeats for mobile data transmissions," in *Pro. ICDCS*, Columbus, OH, USA, pp. 113-122, 2015.
24. H. Zhu and H. Chen, "Adaptive failure detection via heartbeat under Hadoop," in *Pro. APSCC*, Jeju, Korea (South), pp. 231-238, 2011.
25. D. Bui, S. Hussain, E. Huh and S. Lee, "Adaptive replication management in HDFS based on supervised learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 6, pp. 1369-1382, 2016.
26. J. Wang, W. Wu, Z. Liao, R. S. Sherratt and A. Tolba, "A probability preferred priori offloading mechanism in mobile edge computing," *IEEE Access*, vol. 8, no. 1, pp. 39758-39767, 2020.
27. J. Wang, Y. Tang, S. He, C. Zhao and A. Tolba, "Logevent2vec: Logevent-to-vector based anomaly detection for large-scale logs in Internet of Things," *Sensors*, vol. 20, no. 9, pp. 2451, 2020.
28. J. Zhang, S. Zhong, T. Wang, H. Chao and J. Wang, "Blockchain-based systems and applications: a survey," *Journal of Internet Technology*, vol. 21, no. 1, pp. 1-14, 2020.
29. W. Li, H. Xu, H. Li, Y. Yang and J. Wang, "Complexity and algorithms for superposed data uploading problem in networks with smart devices," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5882-5891, 2020.
30. J. Wang, Y. Zou, L. Peng, L. Wang, O. Alfarraj et al., "Research on crack opening prediction of concrete dam based on recurrent neural network," *Journal of Internet Technology*, vol. 21, no. 4, pp. 1161–1169, 2020.
31. J. Zhang, C. Wu, D. Yang, Y. Chen and X. Meng, "HSCS: a hybrid shared cache scheduling scheme for multi-programmed workloads," *Frontiers of Computer Science*, vol. 12, no. 6, pp. 1090-1140, 2018.
32. D. Tao, B. Wang, Z. Lin and T. Y. Wu, "Resource scheduling and data locality for virtualized Hadoop on IaaS cloud platform," in *Pro. BigCom*, Shenyang, China, pp. 332-341, 2016.
33. Q. Zhang and W. Shi, "Firework: data processing and sharing for hybrid cloud-edge analytics," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 9, pp. 2004-2017, 2018.
34. B. Shu, H. Chen and M. Sun, "Dynamic load balancing and channel strategy for apache flume collecting real-time data stream," in *Pro. ISPA*, Guangzhou, China, pp. 542-549, 2017.
35. W. N. Chan and T. Thein, "Sentiment analysis system in big data environment," *Computer Systems Science and Engineering*, vol. 33, no. 3, pp. 187-202, 2018.
36. E. Qi and M. Deng, "R&D investment enhance the financial performance of company driven by big data computing and analysis," *Computer Systems Science and Engineering*, vol. 34, no. 4, pp. 237-248, 2019.
37. D. Chen, Y. Chen, B. N. Brownlow, P. P. Kanjamala and C. A. G. Arredondo, "Real-time or near real-time persisting daily healthcare data into HDFS and elasticsearch index inside a big data platform," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 595-606, 2017.
38. R. Gu, X. Yang, J. Yan, Y. Sun and B. Wang, "Shadoop: improving MapReduce performance by optimizing job execution mechanism in Hadoop clusters," *Journal of Parallel and Distributed Computing*, vol. 74, no. 3, pp. 2166-2179, 2014.
39. Y. Cao and H. Wang, "Communication optimisation for intermediate data of MapReduce computing model," *International Journal of Computational Science and Engineering*, vol. 21, no. 2, pp. 226-233, 2020.
40. J. Lee, J. Choi, H. Roh and J. S. Shin, "An empirical performance analysis on Hadoop via optimizing the network heartbeat period," *KSII Transactions on Internet and Information Systems*, vol. 12, no. 11, pp. 5252-5268, 2018.
41. Y. Jin, F. Liu, X. Yi and M. Chen, "Reducing cellular signaling traffic for heartbeat messages via energy-efficient D2D forwarding," in *Pro. ICDCS*, Atlanta, GA, USA, pp. 1301-1311, 2017.
42. Z. Hu, Z. Lu, X. Wen and W. Jing, "Qoe-based pseudo heartbeat message compression mechanism for future wireless network," *International Journal of Communication Systems*, vol. 29, no. 9, pp. 1513-1528, 2016.
43. T. Wu and W. Wang, "An improvement mechanism for heartbeat-driven MAC synchronization in wireless body sensor network," *International Journal of Communication Systems*, vol. 33, no. 4, 2020.
44. W. Jing, D. Tong, G. Chen, C. Zhao and L. Zhu, "An optimized method of HDFS for massive small files storage," *Computer Science and Information Systems*, vol. 15, no. 3, pp. 533-548, 2018.