Dynamic Data Storage and Management Strategies for Distributed File System

Feng Liu[1(✉)], Di Lin[1], Yao Qin[1], Yuan Gao[2], and Jiang Cao[2]

[1] University of Electronic Science and Technology of China, Chengdu, Sichuan, China
201921090328@std.uestc.edu.cn, lindi@uestc.edu.cn
[2] Military Academy of Sciences, Beijing, China

Abstract. HDFS has a very wide range of applications in the field of big data, but HDFS was designed for a homogeneous environment at the beginning. HDFS adopts a static replica management strategy, the storage location and number of file replicas will not change after determination. This strategy will low overall system performance. In this paper, we propose optimized replica management strategy, abbreviated as ORMP to fix this problem. ORMP is based on file heat value and LSTM. File heat value is proposed to evaluate the activity of files. LSTM is used to predict the access times of files. Based on LSTM, the file heat value can be updated regularly, so we can dynamically change the storage location and number of replicas. Experiments show that ORMP is 22.08% faster in reading speed compared with the default replicas management strategy.

Keywords: HDFS · File heat value · ORMP · LSTM

1 Introduction

Hadoop Distributed File System (HDFS [1]) is a widely used distributed file storage system. HDFS has the ability to store ultra-large-scale data [2]. The use of multiple replica technology and erasure coding technology makes the system highly fault-tolerant. HDFS allows clusters to be built in a number of inexpensive general-purpose computer devices, and performs well in homogeneous computer clusters with similar node performance [3].

With the development of storage technology, solid state drive (SSD) with high read-write performance has been widely used in various applications. Compared with traditional mechanical hard disk drive (HDD), SSD has absolute performance advantages. However, the price of solid state disk is very expensive, so HDD is still the mainstream storage medium in HDFS system. SSD is mainly used to improve the performance of the system [4]. This kind of HDFS cluster using HHD, SSD and other storage media is called heterogeneous HDFS cluster [5].

HDFS adopts multi replica technology to ensure the reliability of data and fault tolerance of cluster. The replica placement, replica management strategy and replica retrieval algorithm are a series of strategies related to multi replica

X. Wang et al. (Eds.): AICON 2021, LNICST 396, pp. 579–592, 2021.
https://doi.org/10.1007/978-3-030-90196-7_50

mechanism [6], because HDFS is mainly based on homogeneous cluster environment. The default replicas management strategy is not suitable for the heterogeneous cluster environment mentioned above. The storage location of replicas will not change after the first placement. Those files placed in low-performance storage devices may be read frequently. However, HDFS is unable to automatically adjust the storage location of those file to other nodes with better performance. The storage resources in heterogeneous HDFS system are not effectively utilized.

The contribution of this paper is as follows.

- We proposed a new strategy based on LSTM to calculate HDFS file heat values.
- We proposed a heterogeneous replicas management strategy based on file heat value. The number and storage position of replicas can change with the heat value dramatically.

2 Related Works

The topic of optimization of replica strategies and algorithms in heterogeneous HDFS environment is one of the significant research directions, and many optimization schemes have been proposed in the related literature.

Literature [7] proposes a heterogeneous perception layered storage scheme based on heterogeneous HDFS cluster named hatS. hatS is a multi-tier storage system that integrates heterogeneous storage technologies into the Hadoop ecosystem. It can actively recognize various hardware devices with different storage performance in the cluster, and create diverse hierarchical structures according to the I/O performance of the storage devices. The three-layer structure mentioned in the paper includes a fast layer, an intermediate layer, as well as a slow layer.

With respect to the replica placement strategy, a hybrid awareness strategy that comprehensively considers layer awareness and network awareness. Compared with the default strategy, a layer awareness technology is proposed to improve the utilization of storage devices by reasonably using different storage devices, while retaining the network-aware technology to ensure data fault tolerance and security. With respect to the replica selection strategy, by considering both the performance of the layer where the replica is located and the distance of network topology, different weights are assigned to different replicas of the same file, and then the weighted random function is further used to access the replica, thereby mitigating the overload problem of the nodes with good performance. Results show that by directing 64% of I/O access to the SSD layer, in terms of Hadoop job execution speed, hatS is 32.6% faster than native HDFS.

Literature [8] presents a hybrid design (Triple-H) that can minimize the I/O bottlenecks in HDFS and ensure efficient utilization of the heterogeneous storage devices (e.g. RAM, SSD, and HDD) available on HPC clusters. This paper also propose an effective data placement policies to speed up Triple-H. Our design integrated with parallel file system (e.g. Lustre) can lead to significant storage space savings and guarantee fault-tolerance. Performance evaluations show that

Triple-H can improve the write and read throughputs of HDFS by up to 7× and 2×, respectively. The execution times of data generation benchmarks are reduced by up to 3×. Triple-H also improves the execution time of the Sort benchmark by up to 40%. Over default HDFS and 54% over Lustre. The alignment phase of the Cloudburst application is accelerated by 19%. Triple-H also benefits the performance of SequenceCount and Grep in PUMA over both default HDFS and Lustre.

Literature [9] optimizes the data placement strategy on heterogeneous HDFS nodes by considering the disk space difference of each node. Literature [10] proposes a new replica placement strategy, which considers the disk utilization and the real-time status of each node when selecting the node to place the replica, so as to balance the workload of the node.

Literature [11] proposes a variable number of replica placement strategy, which highlights node performance, data access frequency and storage capacity. Literature [12] proposes a load balancing algorithm to balance the read and write load of heterogeneous nodes. Literature [13] proposes a serpentine data placement mechanism, which divides different nodes into multiple virtual storage layers, and uses the hot and cold proportional replication strategy to determine the replication factor of each file block. Literature [14] summarizes the relationships between files by analyzing file upload requests, and then allocates the high correlated files to the initial position of nodes, thereby improving the efficiency of data access.

3 Proposed Method

In this section, we propose the concept of file heat value. File heat value is composed of two parts: subjective heat value and objective heat value. The subjective heat value can be calculated by the initial storage policy specified by user. The Objective heat value is based on the LSTM. And then we use the file heat value to optimize the replica management strategy of HDFS.

3.1 File Heat Value

File heat value is used to evaluate the activity of files. The more times a file is read, the higher the value is. By considering the user's expectation of the files' popularity and the real-time access characteristics, a hybrid calculation method based on subjective and objective heat is proposed to calculate the real-time popularity of files.

Subjective Temperature, abbreviated as St: indicates the initial heat value of a file, $St \in [0, 100]$. The subjective heat value depends on the user's estimation on file for a period of time after the file is created, this value will not change over time. $St(f)$ represents the subjective heat value of file f.

The calculation formula of $Temperature_0(f)$ is described as follows:

$$St(f) = \frac{max[Heat(policy_0)] + min[Heat(policy_0)]}{2} \tag{1}$$

$policy_0$ is the initial storage policy of file specified by user. So St is actually determined by user when uploading the file. The relationship between file heat value and storage policy is shown in Table 1.

Table 1. Relationship between file heat value and storage policy

Strategy name	Storage strategy	Explanation	Heat range
Hottest	RAM_DISK:1 SSD:3	One replica puted in RAM_DISK, other three puted in SSD	[80.100]
Hot	SSD:3	All three replicas puted in SSD	[60.80)
Warm	RAM_DISK:1 SSD:2	One replica puted in RAM_DISK, other two puted in SSD	[40.60)
Cold	DISK:3	All three replicas puted in DISK	[20.40)
Coldest	DISK:1	ALL one replica puted in DISK	[0.20)

Objective Temperature, abbreviated as Ot: indicates the predicted heat of a file obtained by prediction algorithm in a period of time, $Ot \in [0, 100]$. In order to calculate the objective popularity, we use LSTM to predict the number of times a file is read. $O_t(f, T_c)$ indicates the objective heat value of the file in the time period T_c.

After the LSTM model is trained, it can predict the read number of file in the next time interval through the access of file in the first n time intervals, as shown in the formula:

$$\hat{c}(f, T_c) = LSTM\left\{c(f, T_{c-n}), c(f, T_{c-n+1}), \cdots, c(f, T_{c-2}), c(f, T_{c-1}))\right\} \tag{2}$$

$\hat{c}(f, T_c)$ represents the predicted access times obtained by LSTM. $c(f, T_i)$ represents the access times of files in the period of T_i. $LSTM$ represents the prediction process using LSTM model. We can calculate the access prediction values of all files in the T_C time period, and then calculate the average of the access prediction values in the T_C time period system. The formula is described as follows:

$$\overline{\hat{C}(f^*, Tc)} = \frac{\sum_n^{i=0} \hat{c}(f, T_c)}{n} \tag{3}$$

n in this formula is the number of files in system. Now we can calculate the Objective heat value of each file: Ot. Ot reflects the access level of files in the whole system, the formula is described as follows:

$$Ot(f, T_c) = \frac{\hat{c}(f, T_c)}{\hat{C}(f^*, Tc)} * 50 \tag{4}$$

50 is the middle of the file heat value. The final formula to calculate real-time heat of file is described as:

$$Temperature(f, T_c) = \lambda * St(f) + (1 - \lambda) * Ot(f, T_c) \tag{5}$$

λ is a attenuation factor, which indicates the degree of attenuation of the influence of subjective heat on real-time heat. It will gradually decrease from 1 to 0 over time. This means subjective heat has a certain timeliness.

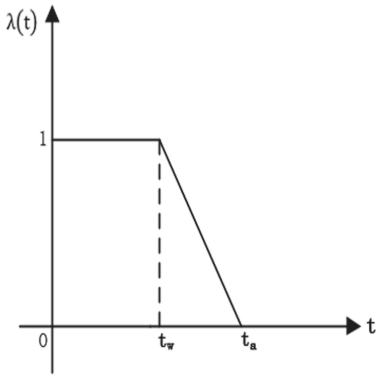

Fig. 1. The decrease process of λ

The heat values of files are re-calculated at every fixed time interval. This time interval is called as the update cycle of temperature, abbreviated as Tc_{temp}. If the update cycle is too long, the heat value cannot be updated in time, resulting in the resources in the system cannot be fully utilized; if the update cycle is too short, the update process will costs lots of system resources. We set Tc_{temp} to 8.

In order to provide sufficient data support for the subsequent optimization work, the new storage strategy and the duration of the storage strategy will be saved in the metadata of the file. New fields are shown in Table 2.

Table 2. New field added in the metadata of HDFS file

Field name	Description
StoragePolicy	The storage policy after updated
Duration	The duration of current storage policy, the unit of this field is Tc, when policy is updated, this field is setted to zero
UpdateTime	The update time of policy

3.2 Optimization of Replica Placement and Management Strategy

In this section, we will use the file heat value to optimize the replica placement and management strategy of HDFS. In our Strategy, When users need to store files to HDFS, they must specify an initial storage policy ($policy_0$), this parameter decides the number of replica and the position of replica shown in

Table 1. Then we use $policy_0$ to calculate the initial heat value of files as shown in Formula 1.

After files are stored in HDFS. The heat value of files will constantly changes with time. So we use LSTM to predict their future heat value. And we will use the predicted heat value to change the position and replica number of files. This strategy is called Optimized replica management policy, abbreviated as ORMP.

4 Experiment and Analysis

This chapter will design experiments based on cloudsim [15] simulation tool to verify the actual optimization effect of ORMP strategy proposed in Sect. 3.

4.1 Simulation Environment Configuration

This experiment simulation uses a tool of CloudSim, which is an open source cloud computing framework developed by CLOUDS Lab at University of Melbourne. CloudSim has built-in rich modeling and simulation functions, covering almost all cloud computing scenarios. It supports modeling and simulation of large cloud data centers, federal cloud modeling and simulation, as well as virtual server modeling. In this paper, we run the relevant simulation program on a single physical host, and the detailed information of software and hardware configuration is shown in Table 3.

Table 3. Hardware and software configuration of experimental environment

Hardware&Software	Configuration
CPU	Intel(R)Core(TM)i7-8700k@3.70 GHz
Memory	LPX DDR4 16G
Hard disk	Intel 660P SSD 215G
Operating system	Windows 10
Compiling environment	JDK 1, 8.0_172
Development tool	Intellig IDEA 2019.1.1

In this experiment, a host node is configured as a NameNode in CloudSim, and 24 host nodes are used as DataNodes, from DN1 to DN18 respectively. They are scattered on three racks (Rack1, Rack2, Rack3). Specifically, the DataNodes of DN_1–DN_6 are located on Rack1, the DataNodes of DN_7–DN_{12} are located on Rack2, and the DataNodes of DN_{13}–DN_{18} are located on Rack3. The detailed information on the configuration of each host node is shown in Table 4.

This experiment need to simulate the reading and writing of data from various storage media, so the respective I/O speeds are set according to the performance of various storage media in the real environment as follows. The reading

Table 4. Detailed configuration parameters of each node

Node number	CPU	Memory (GB)	Hard disk (GB)
1, 2, 7, 8, 13, 14	3200/16	8192	SSD1024
3, 4, 9, 10, 15, 16	2400/12	4096	SSD512
5, 6, 11, 12, 17, 18	2000/8	4096	HDD1024

and writing speed of the solid state hard disk are set to 400 MB/s, the reading speed and writing speed of the mechanical hard disk are set to 150 MB/s, as well as the reading speed and writing speed of the memory are set to 8 GB/s. In addition, in reference with the parameters in the actual cluster, the network card bandwidth of each host is set to 1000 MB/s, the maximum transmission bandwidth in the same rack is set to 1200 MB/s, and the maximum transmission bandwidth between racks is set to 600 MB/s.

4.2 Experiment on Replica Management Strategy

The purpose of this experiment is to verify the effect of optimized replica management policy (ORMP) on dynamically adjusting the location and number of replicas in heterogeneous HDFS cluster environment. This experiment can be divided into two parts in structure: the first part verifies the feasibility and actual effect of LSTM network predicting the access times of files in HDFS. The second part verifies the optimization effect of ORMP strategy based on LSTM prediction results compared with DRMP strategy.

Prediction of File Heat Value Based on LSTM. We take 180 day file access records in heterogeneous HDFS cluster environment as data set. The original data includes 4269 file access records.

Through data cleaning, 50 files are selected, which have been in HDFS for 180 days. Through data preprocessing, the access records of each file are divided into 8-h intervals to get the total access times of each interval. Finally, we get 27000 pieces of data from 50 files in 180 days. The structure of LSTM cell is shown in Fig. 2.

The structure of the neural network consists of an input layer, a hidden layer and an output layer. The number of neurons in the hidden layer is 200. After the input value enters the input layer, it is connected to a fully connected layer through LSTM. In the fully connected layer, an output value is obtained through linear operation, which is the predicted value. In order to enhance the nonlinearity of the neural network model, the activation function is relu and the loss function is MSE. In order to solve the problem of large swing amplitude and accelerate the convergence speed of the function, *RMSprop* [16] algorithm is selected as the optimization method of the model.

In this experiment, the access data of each file is divided according to the ratio of 7:3, with the first 70% of the data (from May 1, 2019 to September 4,

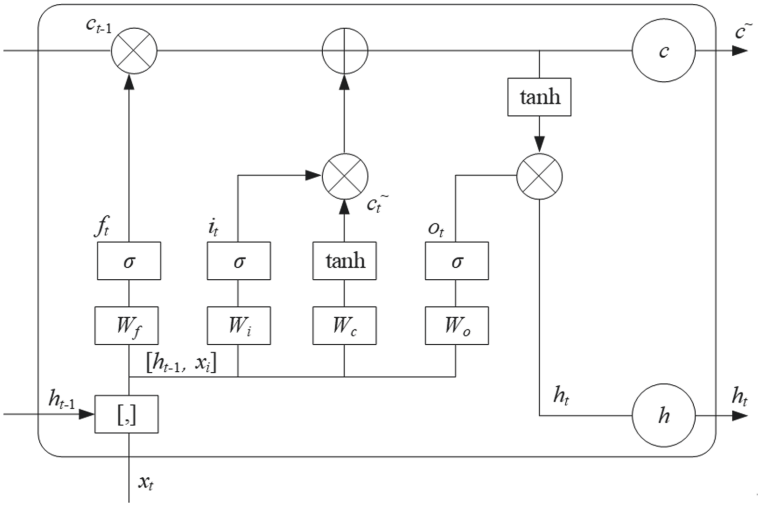

Fig. 2. The structure of LSTM cell

2019) as the training set, and the last 30% of the data (from September 5, 2019 to October 28, 2019) as the test set. The step size of training is 8, that is, the real value of file access times in the past 8 time periods is used to predict the file access times in the next time period in the future. After 100 epoch iterations, the loss function converges and a LSTM neural network model is obtained.

After this model is trained, we randomly select the last 30% data of the file to test the network model. The test results are shown in Fig. 3. The MSE of the prediction results is 73.014.

As shown in the figure, the overall trend of the data can be obtained through the prediction. The short-term prediction effect in most areas is better, but in some areas with abrupt changes in quantity, the prediction deviation is larger, especially the fit of the valley bottom is not ideal.

The root cause of inaccurate prediction in some regions is that the amount of data in the training set is still too small. In general, LSTM can be used to predict the file access, which can basically meet the needs of calculating the real-time heat value in the optimized replica management strategy.

Optimization Effect of ORMP. The ORMP policy dynamically manages the replica based on the prediction of file access. We will analyze the impact of replica management strategy on the read performance by reading files from HDFS. Before the experiment, 200 files are written to HDFS. Each file is 1 GB and divided into 16 file blocks. Since it is impossible to simulate users' subjective judgment on files, the initial heterogeneous storage policy of files is set to warm, one copy stored in SSD, the other two copies stored in HDD.

Select files A, B and C with different activity from the cluster as the research object. We will track six heat value update cycles Tc_{temp}, which are recorded

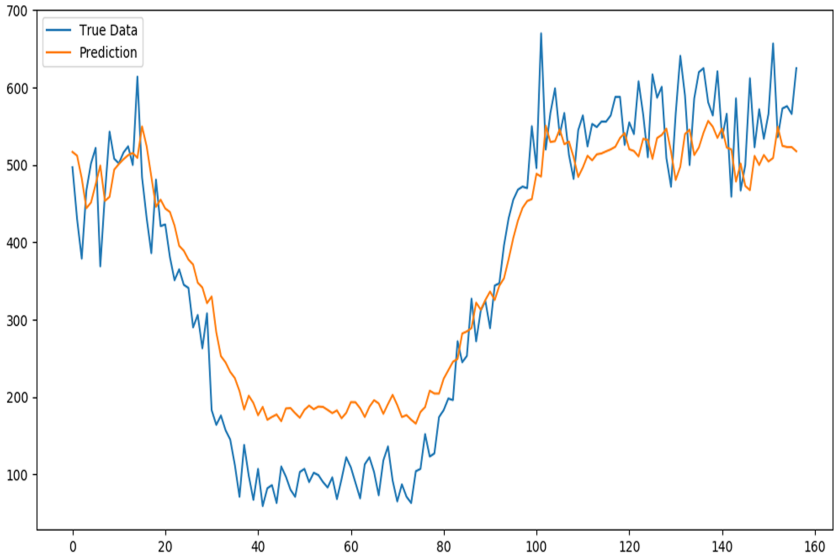

Fig. 3. Comparison between predicted value and real value

as $T_1 - T_6$. Because it is difficult to simulate the user's subjective judgment of file access characteristics, in order to eliminate the influence of subjective heat on the experimental results, the six Tc_{temp} selected in this experiment are later than the subjective heat active period τ_a of each file.

Table 5, 6, 7 and 8 shows the predicted and real file access values of files A, B and B in T_1–T_6 cycle, as well as the average of the predicted file access values in the whole system.

Table 5. Predicted and real file access values of files

Time period	Predicted num			Real num			Error			Average predicted num
	A	B	C	A	B	C	A	B	C	
T1	730	202	22	758	210	15	28	8	−7	170.62
T2	759	199	11	761	181	9	2	−18	−2	154
T3	786	157	5	799	352	4	13	195	−1	202.86
T4	840	310	8	857	574	0	17	264	−8	157.4
T5	897	869	3	872	947	5	−25	78	2	199.06
T6	909	968	7	928	941	0	19	−27	−7	272.76

During T_1–T_6, the real value of access times of file A fluctuates slightly, but it has always been at a high level; the real value of access times of file B fluctuates greatly in the observation period, the overall trend is gradually increasing; the real value of access times of file C has always remained at a very low level. At

the same time, it can be found that the error between the predicted value and the real value of files A and C is very small, which means the predicted value can reflect the real change of file activity; while the predicted value and the real value of file B are different in some time periods, but it can still reflect the change trend to a certain extent.

Figure 4, 5 and 6 show the adjustment of the location and number of replicas of files A, B and C after $ORMP$ strategy optimization during T_1–T_6, which are marked on the horizontal axis; the three pictures also show the location and number of replicas of files under $ORMP$ strategy, which are marked with $Default$ on the horizontal axis.

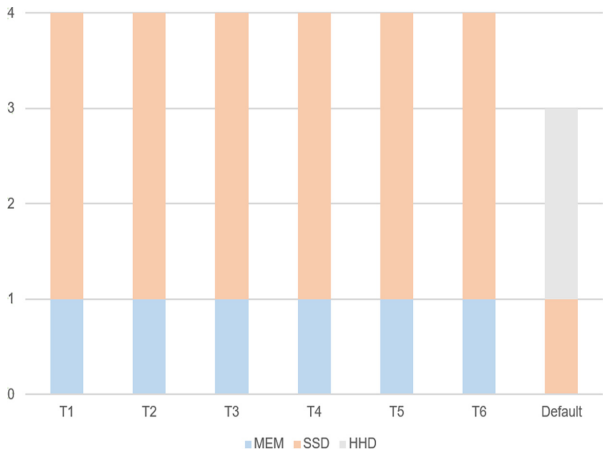

Fig. 4. Replica change trends of file A

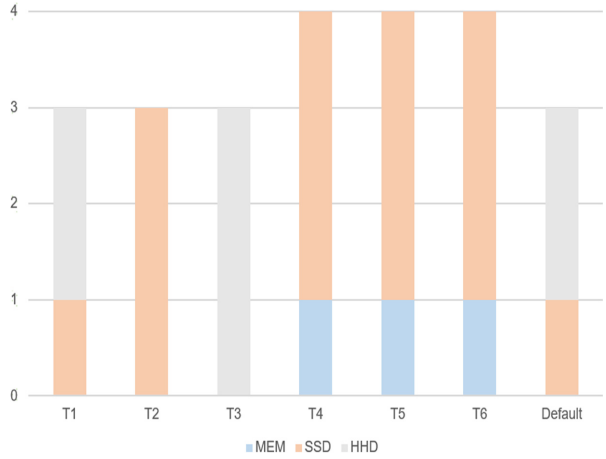

Fig. 5. Replica change trends of file B

Under the ORMP strategy, the access times of file A are always very high, and all of them are always in the hottest policy, that is, three copies are placed in SSD and one copy is placed in memory.

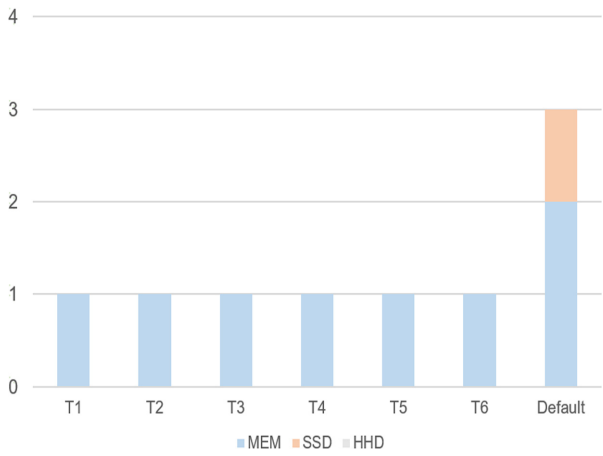

Fig. 6. Replica change trends of file C

The access times of file B are unstable, and the storage policies are warm, hot, cold and hottest in turn, which are consistent with the trend of the real access times of file B.

The access times of file C are close to zero 0, so file C is the coldest storage strategy, only one copy placed in the HDD. Table show the experiment results of three files under two different replica management strategies from T_1 to T_6.

Table 6 shows the experimental results of continuously reading file A based on DRMP strategy and ORMP strategy in T_1–T_6. It can be found that the number of times to read the replica in SSD using ORMP is significantly higher than that using DRMP policy. We can read replicas in memory when using ORMP strategy. Therefore, the cumulative read time of ORMP strategy is much lower than that of DRMP strategy. The average single read time of ORMP is 27.78% less than that of DRMP strategy.

Table 6. Experiment results of file A

	DRMP	ORMP	Optimization degree
Total read num	4975	4975	
Read SSD	26556	70354	
Read memory	0	9246	
Read time (s)	57692.738	41665.563	
Average read time (s)	11.597	8.375	29.78%

Table 7 shows the experimental results of file B. The experimental results of file B are similar to file A. The ORMP strategy performs better in hit SSD and hit memory. The average single read time when using the ORMP strategy is 14.11% less than that of DRMP strategy.

Table 7. Experiment results of file B

	DRMP	ORMP	Optimization degree
Total read num	3205	3205	
Read SSD	17084	39971	
Read memory	0	3697	
Read time (s)	37184.847	31937.356	
Average read time (s)	11.602	9.9065	14.11%

Table 8 shows the experimental results of file C. The reading times of file C is much lower than file A and file B. We can see that the efficiency of reading file is higher than that of ORMP when using DRMP strategy. This is due to the fact that the real-time heat value of file C is low and will not be accessed frequently according to the ORMP policy, so the replicas of file C is placed in the HDD with relatively poor reading performance. So the reading time using the DRMP policy is much lower than that of ORMP policy.

Table 8. Experiment results of file C

	DRMP	ORMP	Optimization degree
Total read num	33	33	
Read SSD	177	0	
Read memory	0	0	
Read time (s)	379.328	617.520	
Average read time (s)	11.495	18.713	−62.07%

In order to analyze the impact of ORMP strategy on the whole file system, we comprehensively analyze the experimental results of files A, B and C, as shown in Table 9.

Table 9. Comprehensive experiment results of all three files

	DRMP	ORMP	Optimization degree
Total read num	8213	8213	
Read SSD	43827	110925	
Read memory	0	12943	
Read time (s)	95256.913	74220.439	
Average read time (s)	11.598	9.037	22.08%

Compared with DRMP, ORMP is 22.08% faster in reading speed. The strategy proposed in this paper obtains the real-time heat value of the file by predicting the number of file accesses, and then dynamically adjusts the location and number of replicas, finally reducing the unit reading time of the data. The strategy proposed in this paper can reduce the number of copies of inactive data and reduce the storage cost of the system.

5 Conclusion

In this paper, we proposed the concept of file heat value and its calculation method. Then the LSTM algorithm is used to predict the access number files in the future. Based on these predicted visits, the storage strategy of the file is updated, and the storage location and the number of copies of the file are dynamically adjusted. Experimental results show that compared with the default storage strategy of HDFS, the proposed strategy can improve the reading speed of the system.

Acknowledgments. Partially Funded by Science and Technology Program of Sichuan Province (2021YFG0330), partially funded by Grant SCITLAB-0001 of Intelligent Terminal Key Laboratory of SiChuan Province, and partially Funded by Fundamental Research Funds for the Central Universities (ZYGX2019J076).

References

1. White, T.: Hadoop: The Definitive Guide (2009)
2. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), pp. 1–10 (2010)
3. Shafer, J., Rixner, S., Cox, A.L.: The hadoop distributed filesystem: balancing portability and performance. In: 2010 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 122–133 (2010)
4. Park, N., Lee, B., Kim, K.T., Youn, H.Y.: Cold data eviction using node congestion probability for HDFS based on hybrid SSD. In: 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp. 1–6 (2015)

5. Yingxun, F., Shilin, W., Li, M.: A data distribution method under a heterogeneous HDFS cluster (2018)
6. Bui, D.M., Hussain, S., Huh, E.N., Lee, S.: Adaptive replication management in HDFS based on supervised learning. IEEE Trans. Knowl. Data Eng. **28**(6), 1369–1382 (2016)
7. Krish, K.R., Anwar, A., Butt, A.R.: hats: a heterogeneity-aware tiered storage for hadoop. In: 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 502–511 (2014)
8. Islam, N.S., Lu, X., Wasi-ur-Rahman, M., Shankar, D., Panda, D.K.: Triple-h: a hybrid approach to accelerate HDFS on HPC clusters with heterogeneous storage architecture. In: 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 101–110 (2015)
9. Zhao, W., Meng, L., Sun, J., Ding, Y., Zhao, H., Wang, L.: An improved data placement strategy in a heterogeneous hadoop cluster. Open Cybern. Systemics J. **9**(1), 8 (2015)
10. Xu, X., Yang, C., Shao, J.: Data replica placement mechanism for open heterogeneous storage systems. Procedia Comput. Sci. **109**, 18–25 (2017)
11. Ye, X., Huang, M., Zhu, D., Xu, P.: A novel blocks placement strategy for hadoop. In: 2012 IEEE/ACIS 11th International Conference on Computer and Information Science, pp. 3–7 (2012)
12. Liu, Y., Li, M., Alham, N.K., Hammoud, S., Ponraj, M.: Load balancing in mapreduce environments for data intensive applications. In: 2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), vol. 4, pp. 2675–2678 (2011)
13. Xiong, R., Luo, J., Dong, F.: SLDP: a novel data placement strategy for large-scale heterogeneous hadoop cluster. In: 2014 Second International Conference on Advanced Cloud and Big Data, pp. 9–17 (2014)
14. Shaochun, W., Xiang, S., Liang, C., Ling, Y., Bowen, Y.: A replica pre-placement strategy based on correlation analysis in cloud environment. In: 1st International Workshop on Cloud Computing and Information Security, pp. 541–544 (2013)
15. Calheiros, R.N., Ranjan, R.B.A.: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms (2011)
16. Dauphin, Y.N., de Vries, H., Chung, J., Bengio, Y.: Rmsprop and equilibrated adaptive learning rates for non-convex optimization. arXiv: Learning (2015)