Heat-based algorithm for identifying hot and cold data in database

Fang Peng*
Dadu River Hydropower Development Co., Ltd., Chengdu 610041, Sichuan, China

ABSTRACT

In the database system based on hybrid storage, the hot and cold data identification algorithm aims to distinguish the hot and cold data when data migration occurs and eliminate the cold data to the storage media with lower performance, and its accuracy directly affects the access efficiency of the hybrid storage system. The traditional hot and cold data identification algorithm mainly identifies hot and cold data based on single features such as access time and access frequency, which has certain limitations. For this reason, a hot and cold data identification algorithm for database based on heat is proposed, which integrates two important features of data access frequency and recent access time to quantify the heat of data, and distinguishes hot and cold data based on this heat. The experimental results show that the algorithm has good stability under different conditions and can improve the cache hit rate by up to about 8% compared with the traditional cache replacement algorithm.

Keywords: Hybrid storage, heat, hot and cold data

1. INTRODUCTION

In the era of big data, the amount of data shows explosive growth, and how to store massive data efficiently becomes a difficult problem that people need to face. Based on the consideration of access cost and performance, hybrid storage is a better solution, and most modern databases also adopt this architecture[1]. In a hybrid storage system, data is scattered and stored in storage media with different performances, such as SSD (Solid State Disk) and HDD (Hard Disk Driver). Due to the high cost of high-speed storage media, to make the system have better performance while minimizing the storage cost, it is better to store only frequently and recently accessed data, i.e., hot data, on high-speed storage media, and store cold data, which is accessed infrequently or even not accessed for a recent period, on low-speed storage media. Therefore, in hybrid storage systems, an algorithm is needed to recognize how hot or cold the data is in the high-speed storage media and migrate the cold data to the low-speed storage media at the appropriate time[2].

Hot and cold data identification can refer to the cache replacement strategy in the operating system. The classical cache replacement algorithm decides whether the data needs to be replaced out of the memory based on the access time or access frequency of the data, and its consideration of the data characteristics is often relatively single, and cannot be adapted to different data access patterns. To address this problem, and to more effectively identify the degree of hot and cold data in the database, this paper comprehensively considers the multidimensional features of the data to model the heat of the data and distinguishes between hot and cold data by the quantified heat attribute, and the data with a higher degree of heat is hot data, and vice versa is cold data.

2. RELATED WORK

The research on identifying hot and cold data originated from the cache replacement policy in the operating system, based on the different kinds of data features considered, the traditional cache replacement policy is mainly classified into two kinds: one is to differentiate hot and cold data based on the most recent access time, and the representative algorithm is LRU (Least Recently Used)[3]; the other is to differentiate between hot and cold data based on the frequency of access, a representative algorithm is LFU (Least Frequently Used)[4]. The LRU algorithm considers that the data that has just been accessed recently is still very likely to be accessed next, and it mainly considers the principle of localization of data access, in many scenarios, the LRU algorithm performs very well, but it ignores the access frequency characteristics of the data, and if a piece of data has been accessed at the earliest time and is accessed frequently at a certain period in the future, then the data should usually be considered as hot data, and it should not be considered as cold data based only on the access time should be considered as cold data[5]. The LFU algorithm considers data that has been accessed more

*12044616@ceic.com

International Conference on Optics, Electronics, and Communication Engineering (OECE 2024), edited by Yang Yue, Proc. of SPIE Vol. 13395, 133953G · © 2024 SPIE · 0277-786X · Published under a Creative Commons Attribution CC-BY 3.0 License · doi: 10.1117/12.3049264

frequently in the recent period to be hotter, which compensates for the shortcomings of the LRU algorithm, but it is also undesirable to consider only the access frequency of the data, if a data has been accessed frequently in a very early period, and has not been accessed since then, then the data should not be considered hot, and if it is still retained in cache for a long period, this will cause a "Cache Pollution[6]".

Given the limitations of LRU and LFU algorithms, many scholars at home and abroad have proposed many improvement algorithms for these two algorithms. In 1993, O'neil[7] and others proposed the LRU-K algorithm, which extends the criterion for judging the hot data from being recently accessed once in the LRU to being recently accessed K times; In 2002, Song et al.[8] proposed the LIRS algorithm, which uses two parameters, Recency (Recently Accessed Time) and Inter-Reference Recency (Access Interval between the last two accesses), to measure the data hot and cold, and in 2003, Megiddo et al.[9] proposed the ARC algorithm, which uses two LRU chained lists to separately store data that has been accessed only once and more than once recently. These improved hot and cold data identification methods based on cache replacement strategies have good performance and identification accuracy, but they cannot fully reflect the degree of hot and cold data in the database and are more dependent on specific data structures.

In addition to the hot and cold data identification method based on the cache replacement strategy, there are many domestic and foreign studies to determine hot and cold data from other perspectives. 2013, Levandoski et al.[10] analyzed database logs through an exponential smoothing model to predict the likelihood of data access in the future, and the identification accuracy is higher than that of the current better hot and cold data identification method based on cache replacement, but the method has complex implementation logic and high storage and computational overhead; In 2019, Xie[11] proposed a temperature model based on Newton's Law of Cooling and quantified the degree of heat and cold of the data by the model, and the experiments showed that the method can effectively improve the recognition accuracy of hot and cold data, but the model complexity and computational overhead are still large.

To overcome the limitations of the existing database hot and cold data determination methods and to improve the storage efficiency and performance of the database, this paper proposes the heat-based Hot and Cold Data Identification algorithm HHDI (Heat-based Hot Data Identification) for databases. The algorithm is based on the Hacker News ranking algorithm, which quantifies the heat of the data in terms of two dimensions: access frequency and recent access time, and determines the hotness or coldness of the data based on the heat, and eliminates some of the identified cold data to the cold database used for storing cold data when the data volume of the hot database used for storing the hot data reaches the peak value or a set storage threshold. The experimental results show that the heat-based database hot and cold data recognition algorithm proposed in this paper can effectively improve the hit rate in the cache, and the recognition accuracy of hot and cold data is high, which has a certain application value.

3. HEAT-BASED HOT AND COLD DATA IDENTIFICATION ALGORITHM HHDI

By analyzing the existing hot and cold data identification algorithms, it is not difficult to find that the access frequency and access time of data are the two main features considered by the majority of algorithms, on the one hand, the data with high access frequency is likely to become hot data, on the other hand, the recently accessed data is also likely to become hot data, but if you only consider any one of the two then it will produce a lot of defects if you analyze the two in combination. If the two are analyzed together, the real situation of hot and cold data can be reflected more comprehensively. Based on the above conjecture, this paper proposes a formula for calculating the heat of data by drawing on the Hacker News ranking algorithm[12], and further categorizes the data into cold and hot data based on the heat, and finally verifies the reasonableness of the conjecture through experiments.

3.1 Algorithm design

Hacker News is a news site that focuses on programming and technology-related issues, where users can submit links to Internet content and other users can vote and comment on those links. Unlike other news sites, on Hacker News users can't vote against a story, but only for it or not at all. Based on the number of votes, the site ranks all the news and provides users with a list of top stories, but the news with the highest number of votes doesn't necessarily sit at the top of the list, as the ranking also takes into account the time factor and the most recently published news is more likely to rank higher than news that has been published for a long time.

Hacker News' ranking algorithm:

$$score = \frac{(p-1)}{(T+2)^G} \tag{1}$$

where p stands for the number of votes, p-1 is to filter out the votes given by the author of the published news to himself, T stands for the time interval between the news release and the present, T+2 is to avoid the divisor being too small, G stands for the gravity factor, the size of its value determines how fast the news rankings will fall over time, G is usually taken as 1.2, 1.5, 1.8.

Hacker News ranking algorithm on the one hand through the user vote to the news "heating", on the other hand, through the index attenuation to the news has been published "cooling", fully considering the user access and release time on the news The impact of user access and release time on news heat, and these two factors are very similar to the access frequency and access time in the hot and cold data identification strategy, so drawing on the algorithm, the Formula (1) deformation, you can get the following formula:

$$heat = \frac{F}{(T+1)^a} \tag{2}$$

where, F denotes the number of times the data has been accessed; T indicates the time interval between the last time the data was accessed and the present time; α is time decay factor, indicating the speed of data heat cooling over time, the value is generally in the range of 1 to 2, the larger the value, the faster the data heat cooling over time.

By using equation (2), the heat of the data in the database can be quantified, and the quantified heat attribute is used as a criterion to measure the hot and cold data, and the data with higher heat is regarded as hot data, while the data with lower heat is regarded as cold data. When the amount of data in the hot database reaches the set storage threshold, the part of cold data with lower heat is migrated to the cold database.

The basic principle of heat-based database hot and cold data recognition algorithm is shown in Figure 1, which is mainly divided into the following aspects in the algorithm design:

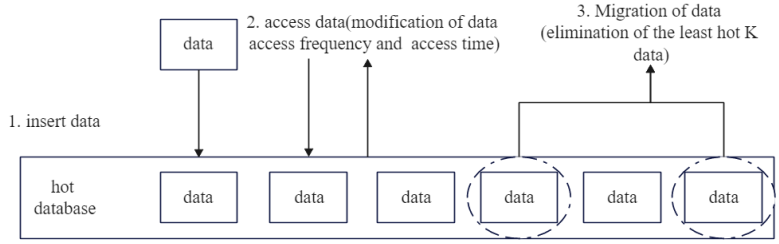

Figure 1. Principle of the HHDI algorithm.

(1) Storage structure: in the hot database, not only saves the data information, but also stores the number of times the data has been accessed and the timestamp of the most recent access, so you can consider using a map-type data structure to store, each data storage structure {data:(frequency, timestamp)}, where data can be a specific block of data information, but also for the index information.

(2) Data insertion: When data is inserted into the hot database for the first time, the number of times the data has been accessed is recorded as 1, and the most recent access time is the insertion time.

(3) Data access: when accessing the data in the hot database, two situations may occur: one is that the data is not in the hot database, at this time it is necessary to read the data from the cold database and insert it into the hot database; the other is that there is the data in the hot database, then access it directly and add 1 to the number of times that the data is accessed, and at the same time change the most recent access time to the time of the current access.

(4) Data Migration: When the amount of data stored in the hot database reaches the storage threshold of the hot database, it is necessary to carry out the data migration operation, at this time, the heat calculation formula is utilized to calculate the heat of all the data in the hot database, and the part of the cold data with lower heat is migrated to the cold database.

3.2 Algorithm Implementation

The flowchart of the heat-based algorithm for recognizing hot and cold data in the database is shown in Figure 2.

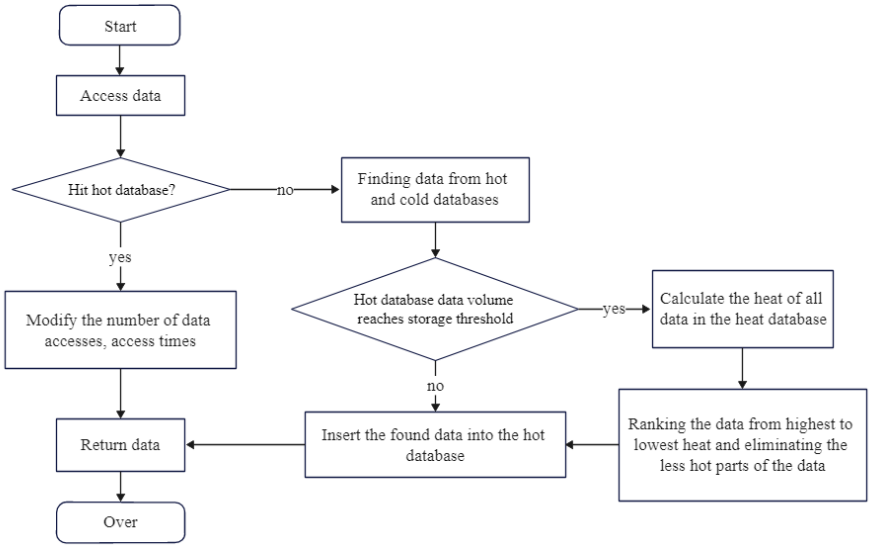

Figure 2. Flowchart of HHDI algorithm.

In this algorithm, the search, modification, and deletion operations in the database are regarded as access operations to the data, and when the data access operation occurs, the first is to find the hot database, if found then we directly return data information, at the same time to the number of times the data access plus 1 and the access time to modify the access time to the current system time. If we don't find it in the hot database, we need to go to the cold database to find it, get the data, return the data information, and insert the data into the hot database. When inserting data into the hot database, we need to first determine whether the amount of data in the current hot database reaches the storage threshold, if it does not, the data will be directly inserted into the hot database and the number of accesses to this data will be recorded as 1, and the access time will be set to the current insertion time; On the contrary, it is necessary to calculate the heat of all the data in the hot database at this time, and eliminate the data outside the set heat threshold to the cold database, where the heat threshold refers to the proportion of hot data, i.e., the proportion of the part of the data with a higher heat that needs to be retained after all the data are sorted according to the heat of the data from the highest to the lowest.

In addition, in this algorithm, new data is saved in the hot database by default, in order not to affect the insertion of new data when data migration occurs, the hot database storage threshold is set to 80%, i.e., when the amount of data in the hot database reaches 80% of the total capacity of the hot database, the data migration operation needs to be performed, in practice, this value can also be realized as dynamically adjusted according to the workload[13], but this issue is not the focus of discussion in this paper.

The pseudo-code description of the hot and cold data recognition algorithm based on heat is as follows:

Algorithm 1 HHDI Algorithm

Input: access data(data), storage threshold(s), heat threshold(h), time decay factor(α)

Output: access data information

1. if data in the hot database do
2. //Get the old frequency of data and the current time
3. old_frequency←data.frequency
4. access_time=System. time
5. //update the frequency and the access time
6. map[data]←(old_frequency+1, access_time)
7. end if

8. else

9. //Get the data value from the cold database and the current time

10. data←cold_data

11. access_time = System. time

12. if the size of the hot database=s do

13. //Get the current time

14. current_time←System. time

15. for data in a hot database do

16. //compute the heat of each data in a hot database

17. heat←data. frequency/Math.pow(current_time-access_time+1, α)

18. MinHeap.put(data, heat)

19. end for

20. cold_data←Select(MinHeap, h)

21. move the cold data from the hot database to the cold database

22. //put the data into a hot database

23. map[data]←(1, access_time)

24. end else

25. return the value of the data

3.3 Algorithm overhead

The overheads of the HHDI algorithm in space and time are as follows:

(1) Space overhead: The HHDI algorithm in storing each piece of data in addition to storing the original data information, also needs to additionally store the number of times the data is accessed and the most recent access time, in the implementation of the two long integer variables can be used to store the information, for the 100,000 pieces of data, the space overhead is probably about 1.5 M.

(2) Time overhead: The time overhead of the HHDI algorithm is mainly because when data migration occurs, it is necessary to traverse the data in the entire hot database and calculate the hotness, and then it is also necessary to sort the hotness of the data so that the part of the data with the lowest hotness will be eliminated to the cold database. Assuming that the hot database can hold 100 pieces of data, the access hit rate is 50%, and only one piece of data with the lowest hotness is eliminated each time it is eliminated, then 10,000 accesses will occur 10,000∗(1-50%)-100=4,900 migration operations and the lowest hotness data needs to be found each time it is migrated, which has a large computational overhead, so in order to reduce the algorithm's overhead each time a data migration should be batch eliminated as much as possible to reduce the number of migration operations occurring.

4. DESIGN AND ANALYSIS OF EXPERIMENTS

In this section, the hit rate and performance of the HHDI algorithm proposed in this paper will be examined through experiments. Considering that the data accesses in practical scenarios are often characterized by "locality", the data in the experiments are 10,000 pieces of data generated by using the Zipf distribution of s=1, and the number of accesses is 100,000 times. To verify the effectiveness of the HHDI algorithm proposed in this paper, LRU, LFU, and LRU-2 algorithms are selected for experimental comparison, all the algorithms are implemented in Java, and the value of the HHDI algorithm is fixed to 1.2, and the experiments are run on Windows 10, with a processor of Intel i5-7200U at a frequency of 2.50 GHZ, and a memory of 16 GB.

4.1 Cache hit rate

The cache hit rate represents the ratio of cache hits to the total number of accesses, which is the most important index to measure the effectiveness of the cache replacement algorithm, the higher the cache hit rate, the more accurate the algorithm's replacement strategy is, and also indicates that the algorithm's accuracy in recognizing hot and cold data is higher. In this section, we compare the hit rate of LRU, LFU, LRU-2, and the HHDI algorithm proposed in this paper according to different hot database sizes and different heat thresholds, and the experimental results are shown in Figures 3 and 4.

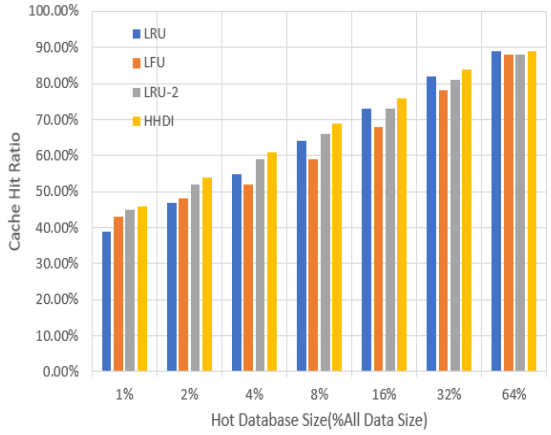

Figure 3. Comparison of hit rates for different hot database sizes.

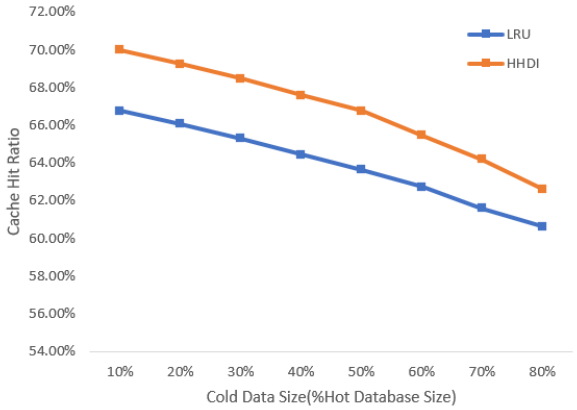

Figure 4. Comparison of hit rates for different heat thresholds.

Figure 3 reflects the change in hit rate of various algorithms in different hot database capacity by eliminating only one piece of data at a time, from which it can be seen that the hit rate of various algorithms is improving with the increasing hot database capacity, among which the hit rate of HHDI algorithm has been maintaining the highest status, slightly higher than that of LRU-2 algorithm, and about 5% higher than that of LRU and LFU algorithms on average, and the advantage of HHDI algorithm is more obvious in the case of a hot database with a smaller capacity. When the capacity is small, the advantage of the HHDI algorithm is more obvious. From the experimental results, it can be obtained that the performance of HHDI and LRU-2 algorithms based on the consideration of multidimensional features of the data has been better than that of LRU and LFU algorithms based on a single feature, which also verifies the correctness of the conjecture given in this paper when proposing the HHDI algorithm.

Figure 4 reflects the change in the hit rate of the LRU algorithm and HHDI algorithm at different heat thresholds with fixed hot database capacity (10% of total data volume), from which it can be seen that the hit rate of both algorithms decreases as the heat threshold decreases, and the reason for analyzing this may lie in the fact that, as the heat threshold decreases, the proportion of cold data eliminated each time a data migration occurs is increasing, and These eliminated data may contain some of the hotter data, and that part of the data will be revisited soon, which ultimately leads to a decrease in the hit rate of the algorithms. According to the results of this experiment, the recognition accuracy of the hot and cold data algorithm will be improved with the increase of the heat threshold, however, in practical applications, the threshold should not be raised blindly, too high a threshold means that the proportion of data eliminated each time will be very small, which will lead to the hot database has been in the state of close to saturation, and the storage space cannot be utilized more effectively. In order to balance the hit rate of the algorithm and the utilization of storage space, the hotness threshold should preferably be dynamically adjusted according to the changes in the current workload, which is also one of the future research directions of this paper.

From the results of cache hit experiments, it can be seen that the HHDI algorithm proposed in this paper has a higher cache hit rate than the traditional cache replacement algorithm and has good stability under different conditions, and the feasibility of the algorithm has been tested.

4.2 Algorithm performance

To evaluate the performance of the algorithms, this experiment records the running time of LRU, LFU, LRU-2, and HHDI algorithms with different hot database sizes and different heat thresholds, respectively, and the experimental results are shown in Figures 5 and 6.

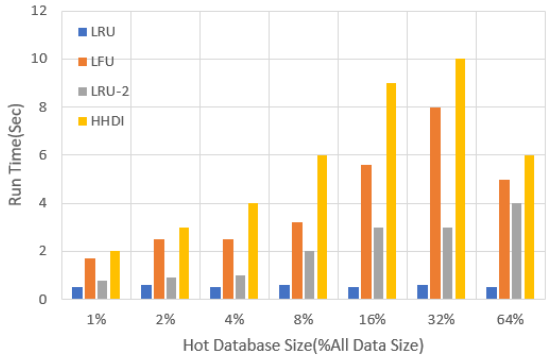

Figure 5. Comparison of runtime for different thermal database sizes.

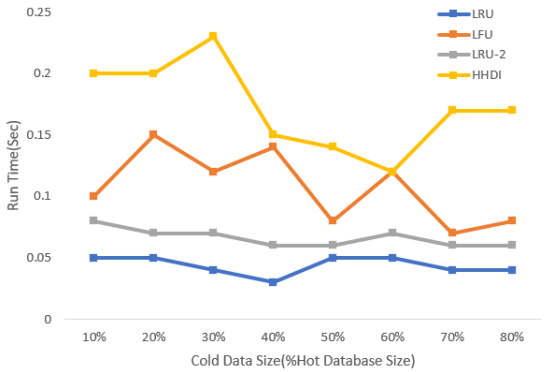

Figure 6. Comparison of runtime for different heat thresholds.

Figure 5 reflects the change of the running time of each algorithm under different hot database sizes when only one data is eliminated at a time, and it can be seen from the figure that the running time of the LRU algorithm, which is the simplest to implement, basically remains unchanged and is stable within 1 s, while other algorithms increase with the increase of the hot database capacity, but the running time increases instead, the reason for this is, as analyzed in Section 2.3, when only one data is eliminated at a time, The migration operations triggered by the algorithm will become more, and the HHDI algorithm and LFU and LRU-2 algorithms proposed in this paper will incur computational overheads when migration operations occur, and the larger the database capacity is, the larger this computational overhead will be, and thus the algorithms' performances will be continuously reduced.

Figure 6 reflects the running time changes of each algorithm under different heat thresholds at a fixed hot database capacity (10% of the total data volume), from which it can be seen that as the proportion of cold data eliminated each time continues to increase (and the heat threshold continues to decrease), the running time of the LFU and HHDI algorithms roughly shows a decreasing trend, and the algorithms' performances continue to improve, whereas the running time of the LRU and LRU-2 algorithms remains unchanged, and the running time of the LFU and HHDI algorithms is getting closer and closer to that of the LRU and LRU-2 algorithms, and the running time of all four algorithms stays within 1 s.

The above experimental results show that there is some gap between this algorithm and the traditional cache replacement algorithm in terms of running time, but this gap is shrinking as the proportion of cold data eliminated each time continues to increase. Considering that the algorithm proposed in this paper is mainly used in database systems based on hybrid storage, the requirements on time are not as strict as in the cache replacement scenario in the operating system, the performance is acceptable in general, but there is still a lot of room for optimization.

5. CONCLUSIONS

Aiming at the limitations of existing hot and cold data recognition algorithms, this paper proposes a hot and cold data recognition algorithm for databases based on heat, which quantifies the heat of the data from the two dimensions of access frequency and recent access time, and then categorizes the data into hot and cold data according to the heat. Experimental results show that the hot and cold data recognition method proposed in this paper is better than the traditional cache replacement algorithm in terms of hit rate, and the accuracy of hot and cold data recognition is higher, but there is a slight gap between the performance and the implementation of the simpler cache replacement algorithm, which can be followed by the combination of the algorithm proposed in this paper and the cache replacement algorithm, to form a multilevel recognition queue, to improve the overall performance of the algorithm.

REFERENCES

[1] Liu, X. and Salem, K., "Hybrid storage management for database systems," Proceedings of the VLDB Endowment 6(8), 541-552 (2013).
[2] Niu, J., Xu, J. and Xie, L., "Hybrid storage systems: A survey of architectures and algorithms," IEEE Access 6, 13385-13406 (2018).

[3] Dan, A. and Towsley, D., “An approximate analysis of the LRU and FIFO buffer replacement schemes,” Proceedings of the 1990 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems 143-152 (1990).
[4] Lee, D., Choi, J., Kim, J. H., et al., “On the existence of a spectrum of policies that subsumes the least recently used (LRU) and least frequently used (LFU) policies,” Proceedings of the 1999 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, 134-143 (1999).
[5] Liang, Y., Chen, Y., et al., “Database cold and hot data identifying algorithm for transaction scenery adaption,” Modern Electronics Technique 7, 107-111 (2022). (in Chinese)
[6] Seshadri, V., Mutlu, O., Kozuch, M. A., et al., “The evicted-address filter: A unified mechanism to address both cache pollution and thrashing,” 2012 21st International Conference on Parallel Architectures and Compilation Techniques (PACT), 355-366 (2012).
[7] O’neil, E. J., O’neil, P. E. and Weikum, G., “The LRU-K page replacement algorithm for database disk buffering,” ACM Sigmod Record 22(2), 297-306 (1993).
[8] Jiang, S. and Zhang, X., “LIRS: An efficient low inter-reference recency set replacement policy to improve buffer cache performance,” ACM SIGMETRICS Performance Evaluation Review 30(1), 31-42 (2002).
[9] Megiddo, N. and Modha, D. S., “ARC: a self-tuning, low overhead replacement cache,” Proceedings of the 2nd USENIX Conference on File and Storage Technologies, 9 (2003).
[10] Levandoski, J. J., Larson, P. Å. and Stoica, R., “Identifying hot and cold data in main-memory databases,” 2013 IEEE 29th International Conference on Data Engineering (ICDE), 26-37 (2013).
[11] Yulin, X., [Research on Recognition Mechanism of Hot and Cold Data Based on Data Tenwerature], Zhejiang: Zhejiang University, Master’s Thesis, (2019). (in Chinese)
[12] Stoddard, G., “Popularity dynamics and intrinsic quality in reddit and hacker news,” Proceedings of the International AAAI Conference on Web and Social Media 9(1), 416-425 (2015).
[13] Xu, B. and Tao, L., “Design of data migration time in tiered storage,” Computer Engineering and Design 34, 725-729 (2013). (in Chinese)