A Multi-Level Cache Management Approach of Scientific and Technical Information Data for Electric Power

Yan Ma¹,*, Lida Zou²,*, Dali Qi¹ and Yufeng Chen¹

¹State Grid Shandong Electric Power Research Institute, Jinan 250002, China.
²Shandong University of Finance and Economics, Jinan 250014, China.

*Corresponding author e-mail: zoulida@163.com, yanpony@126.com

Abstract. Scientific and technical information database for electric power becomes indispensable with the development of big data and think tank in electric industry. Querying information database is time-consuming and cache management is an efficient solution. In the paper we focus on a multi-level cache management in order to improve search speed of sci-tech information. The proposed computation method of cached data value considers visiting volume, data size and user experience, which is accurate for choosing key words and caching device. Our caching management framework consists of four modules, and the used machine learning method could effectively predict the number of visits next period for key words. The experiments demonstrate that our proposed MLCM approach performs better in prediction accuracy rate, query time and user experience than traditional statistic, LRU, LFU algorithms.

1. Introduction

Information work is the guarantee of rapid development for science and technology. Information data of electric power industry is an active force in the development of national economy [1]. It helps to make strategic decisions, and will play a more and more important role in power grid development.

Scientific and technical information database is an important source of obtaining sci-tech information for scientific workers. When users access information database, they hope to find the searched data as soon as possible. However, during retrieving information, the results are returned in the form of items, documents, pictures, audios, videos, etc. Their volumes are huge, especially for batch search. This causes a long query time and bad user experience.

It could effectively accelerate data access speed to put some frequently accessed data into cache [2-4]. Cache management is complicated, which is reflected in two aspects. 1) Cache media are diversified. At present memory, network cache, solid state disk (SSD), disk and so on are the common cache media. Each cache medium has different access speed. 2) Cache space is limited. Each cache medium provides limited storage space due to cost. Therefore, a multi-level cache management approach is necessary in order to improve user query and access efficiency.

Before designing cache management method, it should be known in advance that which data are frequently accessed in future. The most valuable data then could be put into the limited cache and provide the most efficient cache usage. Information database are usually searched by users in key words, such as topic, author, journal. The method of accurately predicting access frequency of key word in a future period is indispensable.
2. Cache Management Framework

To solve the above problem, we propose a cache management framework of scientific and technical information database, which use multi-level cache management to improve searching speed of electric power information. Its flowchart is shown in Fig. 1. It first gets periodic events and information data. After getting the input data, it uses machine learning method to predict the number of visits in the future periods. According to the number of visits, data size and user experience function, it estimates the value of cached data needed by each key word. Last the data are put into each level of caches through analyzing value.

In order to modularize system structure, we divide our framework into four modules, as shown in Fig. 2. They are periodic event module, data acquisition module, visits prediction module and multi-level cache placement module. Periodic event module manages the period of cache framework. Data acquisition module gathers data of sci-tech information for electric power. Visits prediction module predicts the hot key words searched by users and their visiting volumes next period. Multi-level cache placement manages the stored data by multi-level caches.

3. Cache Management Approach

In the section we introduce our proposed Multi-Level Cache Management method (MLCM) in details. The concrete steps of each module will be given.

3.1. Periodic Events Module

Periodic events module initiates an event of cache management every once in a while. It sends instructions to data acquisition module periodically.

3.2. Data Acquisition Module

Data acquisition module gets the real number of visits for each key word recently, and the weight of each key word on recent scientific literatures. It sends these data to visits prediction module in order to predict the number of visits next period. The concrete steps are as follows.

1) First it obtains scientific literatures from sci-tech information database for electric power in the recent time. The length of recent time is defined as \( n \) times of a prediction period. If the time span of a period is one week and \( n = 10 \), it obtains the literatures in recent 10 weeks.

2) Assume the literature set from a literature database is denoted as \( T_i \), \( i \in I \), where \( I \) is the identity set. For example, scientific literatures from journal library is labeled as \( T_i \), scientific literatures from conference library is set as \( T_i \), scientific literatures from dissertation library is represented as \( T_i \), etc.

3) The weight of word frequency for \( T_i \) is obtained by TF-IDF algorithm. Let \( A_i \) denote the weight set of word frequency for \( T_i \), i.e., \( A_i = \{ h_{i,j} \mid i \in I, j \in Q \} \), where \( h_{i,j} \) is the frequency weight of \( j \)-th key word in \( T_i \) and \( Q \) is identity set of key word in sci-tech lexicon.

4) It next gets the number of visits for each key word. Let \( S \) denote the visits set of key words in
recent $n$ periods, i.e., $S = \{s_{k,j} | k = 1, 2, \ldots, n, j \in Q\}$, where $s_{k,j}$ denotes the number of visits for $j$-th key words in $Q$ in the previous $k$-th period, and $n$ is the sequential number of the earliest period since $S$ begins to record data.

5) It last sends the data in $A_i | i \in I$ and $S$ to visits prediction module and stores them into prediction database to provide training data.

3.3. Visits Prediction Module
Visits prediction module gathers training data, trains time series models and predict the number of visits for next period. It then sends the hot key words and their visits data to multi-level cache placement module. The concrete steps are as follows.

1) Data preparation. First the key words with a small access amount are eliminated. Then the trained data are gathered from prediction database. Let $X$ and $Y$ respectively denote the input data and real visiting volumes, i.e., $X = \{x_i | i < n\}$, $Y = \{y_i | i < n\}$, where $i$ is integer and $n$ is the number of trained data records. The input data is denoted as $\{a_j | 1 \leq j \leq k \}$, where $j$ is the identity of $Q$ and $k$ is integer. The number of visits is represented as $0, y_{j} s_{k,j}$, where $s_{0,j}$ is the number of visits for $j$-th key word next period.

2) Model training. In the paper we choose LSTM algorithm [5] as our training model. It uses $X$ and $Y$ to do training.

3) Visits prediction. It adopts the trained model to predict the number of visits next period. Let $\hat{S} = \{\hat{s}_j | j \in Q'\}$ denote the visits set, where $\hat{s}_j$ is the visits of $j$-th key words and $Q'$ is the set of key words after eliminating ones with a small access amount from $Q$.

4) Send $\hat{S}$ to multi-level cache placement module.

3.4. Multi-level Cache Placement Module
Multi-level caches include memory, network cache, solid state disk (SSD) and disk media. In general the faster the read-write speed is, the smaller the cache capacity is. Multi-level cache placement module manages different levels of cached data, which helps improve total access efficiency and user experience. Its concrete steps are as follows.

1) It first gets the size of need cached data for each key word, i.e., the size of returned data when a user queries a key word. If these data are put into cache in advance, the access efficiency is improved. We set $d_j$ as the data size of the $j$-th key word, and $j \in Q'$ stands.

2) User experience function is defined as $f(t)$, where $t$ is return time of a query. The function shows the change of user experience with $t$. The higher the index is, the worse user experience is. We can define $f(t) = t$, as shown Fig.3 (a). It can also be set as discontinuous piecewise function, as shown in Fig. 3(b) (c).

![Figure 3. Several cases of user experience function](image)

3) Next the cached data for each cache are generated according to placement strategy. It first estimates the value of cached data according to user experience function, data size and access frequency.
Then the data with high value are placed into the cache with fast read-write speed. The concrete algorithm is as follows.

a) Different types of cache media are sorted in non-ascending order of read-write speed, and denoted as \( C = \{c_1, c_2, \ldots, c_n\} \), where \( c_i \) represents a caching device, \( n \) is the number of caching devices.

b) Let \( e_i \) denote the cache capacity of \( c_i \).

c) A caching device is taken from \( C \) in turn, and denoted as \( c_i \).

d) Assume that the value of \( \hat{j}_s \) is \( x_j \), and computed as

\[
x_j = \frac{f(t_{j,s}) - f(t_{j,\infty})}{d_j}
\]

Where \( t_{j,s} \) means its query time if just putting the needed data by \( \hat{j}_s \) into \( c_i \).

e) The values \( x_j \), \( j \in Q' \) are sorted, and the needed data by \( \hat{j}_s \) are in turn put into \( c_i \), until \( e_i \) is used up. Then update \( Q' \), i.e., delete the already cached key words in \( Q' \).

f) Repeat step c), until all the caching devices are used.

4. Experiments and Analysis

In the paper we propose our MLCM approach. Next we validate its performance through extensive experiments. The comparison indicators are accuracy rate, query time and user experience score. We choose three traditional algorithms: statistic method, LRU and LFU [6].

We first evaluate the accuracy rate of visits prediction for next period. Historical statistic method is used as a benchmark. It assumes that the visiting volume of a key word in a future period is similar to historical visits. Visiting volumes of multiple historical periods are counted and their mean value is the visiting volume in the future period. Fig. 4 shows the comparison results, where horizontal coordinate represents the number of periods for input data. It is shown that our MLCM approach performs better in whole than statistic method. When the number of periods is larger than 11, the accuracy rate of MLCM no longer increases obviously.

![Figure 4. Accuracy rate with the number of periods](image)

![Figure 5. Accuracy rate with key words](image)

Next accuracy rate with different key words is verified, as shown in Fig. 5. We divide key words into five types according to their access frequency. First all the key words are sorted in non-ascending order of visiting volume. B0-20 denotes the key words ranking the top 20%, B20-40 means the key words ranking from 20% to 40%, and so on. From the figure we see that the key words with proper visits have a high accuracy rate, while the accuracy rate for hot or unpopular key words descends.

After validating accuracy rate, we see query performance of MLCM. We use query time as our index. The classical algorithm LRU and LFU are compared. In order to adapt to multi-level cache, the two algorithms are improved as follows. 1) First put data with high priority into advanced cache. 2) The replaced data from advanced cache are placed into secondary cache. 3) The replaced data from secondary cache are placed into tertiary cache, and so on. We use three kinds of caching devices:
memory, SSD and disk. The ratio of capacity for three-level caches is set to 1:10:100. Fig.6 gives the comparison on query time, where horizontal coordinate represents memory capacity. The capacities of the other caches are computed. Our MLCM performs 68% and 61.3% better than LRU and LFU in query time.

![Query time with memory capacity](image1)

**Figure 6.** Query time with memory capacity

![Query time and user experience score](image2)

**Figure 7.** Query time and user experience score

We last see the effects of different caching devices on user experience score and query time. Three combinations are set: just memory, memory+SSD, memory+SSD+disk. Fig.7 shows the experimental results, where histograms denote query time and area graphs denote user experience score. We use linear function as user experience model. Our MLCM is better than LRU and LFU for both query time and user experience score. MLCM performs best in user experience, because the computation method of data value considers thoroughly and has a precise equation.

5. Conclusion
In the paper we propose a multi-level cache management approach for power information database. It counts and analyses which data would be frequently accessed, and uses LSTM model to predict visiting volume of key words in the future period. The prediction is accurate and best for key words with proper visits. Through multi-level cache management, query efficiency is improved and user experience becomes better. The experiments shows that the proposed MLCM has 22.3% better accuracy rate than statistic method, and has over 60% better query time than LRU and LFU. It has the best user experience as well, no matter for which combination of caching devices.

**Acknowledgments**
This work was supported by Science and Technology Plan for Colleges and Universities in Shandong Province (KJ2018BAN046).
References

[1] Ma Yan, Qi Dali, Yu Naihai, Zou Lida. A Prediction and Push Method of Scientific Researching Issue Based on RNN [C], IOP Conf. Series Materials Science and Engineering, 452 (2019) 042079.

[2] X.L. Qin, W.B. Zhang, J. Wei, W. Wang, H. Zhong and T. Huang, Progress and Challenges of Distributed Caching Techniques in Cloud Computing, Journal of Software, 24,1, pp. 50-66 (2013).

[3] Storm A J, Garcia-Arellano C, Lightstone S S, Diao Y, surendra M, Adaptive self-tuning memory in DB2, Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB), Seoul, Korea, pp. 108-1092 (2006).

[4] Malik T, Burns R, Chaudhary A, Bypass caching: making scientific databases good network citizens, Proceedings of the 35th International Conference on Data Engineering (ICDE), Tokyo, Japan, pp. 94-105 (2005).

[5] Hochreiter s, Schmidhuber J. Long Short-Term Memory Neural Computation [J], 1997, 9 (8): 1935-1970.

[6] O’Neil Elizabeth J, O’Neil Patrick E, Weikum Gerhard, “The LRU-K page replacement algorithm for database disk buffering”, Proceedings of the 19th International Conference on Management of Data (SIGMOD). Washington, DC, USA, pp. 297-306 (1993).