IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 21, 2019, accepted April 19, 2019, date of publication April 30, 2019, date of current version May 15, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2913961

LR-LRU: A PACS-Oriented Intelligent Cache Replacement Policy

YINYIN WANG[1], YUWANG YANG[1], CHEN HAN[1], LEI YE[1], YAQI KE[1], AND QINGGUANG WANG[2]

[1]School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China
[2]Yancheng 1st People's Hospital, Yancheng 224005, China

Corresponding author: Yuwang Yang (yuwangyang@njust.edu.cn)

ABSTRACT An intelligent cache replacement policy suitable for picture archiving and communication systems (PACS) was proposed in this work. By combining the logistic regression (LR) algorithm with the classic least recently used (LRU) cache replacement policy, we have created a new intelligent cache replacement policy called LR-LRU. The LR-LRU policy is unlike conventional cache replacement policies, which are solely dependent on the intrinsic properties of the cached items. Our PACS-oriented LR-LRU algorithm identifies the variables that affect file access probabilities by mining medical data. The LR algorithm is then used to model the future access probabilities of the cached items, thus improving cache performance. In addition, $\ell 1$-regularization was used to reduce the absolute values of the variables' coefficients. This screens some variables that have little influence on the model by causing their coefficients to approach zero, which achieves the effect of screening the variables. Finally, a simulation experiment was performed using the trace-driven simulation method. It was shown that the $\ell 1$-regularized LR model is superior to the LR and $\ell 2$-regularized LR models. The LR-LRU cache algorithm significantly improves PACS cache performance when compared to conventional cache replacement policies, such as LRU, LFU, SIZE, GDF, and GDSF.

INDEX TERMS PACS, cache replacement policy, logistic regression, hybrid storage, LRU.

I. INTRODUCTION

The picture archiving and communication system (PACS) is a computer system that was specifically designed to process, store, and transmit medical images. In a PACS, the image data acquired by medical imaging techniques, such as computed tomography (CT), computed radiography (CR), digital subtraction angiography (DSA), magnetic resonance imaging (MRI), digital gastrointestinal imaging, ultrasound, and endoscopy, are digitalized by a secure network and then transmitted to a server for classification and storage. The data being handled by PACSs is growing rapidly in volume. For example, the PACS data volumes of the Yancheng 1st People's Hospital are growing by over 20% per annum. This incredible rate of growth poses a massive challenge for data storage systems. Although data storage systems are becoming faster, larger, and cheaper, neither hard disk drive (HDD) nor solid-state drive (SSD) storage systems are able to satisfy the requirements of PACS data storage, due to the inherent limitations of SSD and HDD technologies. To solve this problem, HDD and SSD storage systems have been combined to form hybrid storage systems, which fully leverage the advantages of both technologies [1]–[3].

The associate editor coordinating the review of this manuscript and approving it for publication was Jun Wu.

A hybrid data storage system combines storage media with contrasting features, so that each data request will be handled by the optimal storage media for the given data access characteristics and system load, thus enhancing overall system performance. In most cases, SSDs are used as cache memory for HDDs. This configuration exploits the speed advantage of SSDs while retaining the storage capacity advantage of HDDs, thus creating a fast and high-capacity storage space. However, the storage space of SSD caches is relatively limited due to the high cost of SSDs; a cache replacement policy is therefore necessary to manage hybrid storage systems [1]–[7].

Conventional cache replacement policies tend to perform poorly in PACSs. That is because these policies only focus on one specific factor (e.g., filesize, last access time, access frequency, and so on). References [1]–[5] proposed the use of machine learning techniques to improve conventional cache replacement policies. In most of these studies, a machine learning algorithm (e.g., artificial neural networks (ANNs), support vector machines (SVMs), and the C4.5 decision tree algorithm) was combined with a conventional cache replacement policy to predict the future access probabilities of the cached items, thus forming an intelligent cache replacement policy that improves cache performance. These policies generally consider the intrinsic factors of the cached items, but overlook the characteristics of real-world entities. Consequently, the performance of intelligent cache replacement policies in PACSs still leaves something to be desired.

The treatment of a patient tends to generate large amounts of medical data, which are stored in various medical databases, e.g., health information system (HIS), radiology information system (RIS), and electronic health record (EMR) databases. A variety of variables can be obtained by mining these data. Some interesting insights were obtained by analyzing these variables:

- If the imaging process has yet to be completed, the corresponding image data will be accessed repeatedly.
- The image data of inpatients is much more likely to be accessed than that of outpatients.
- Images that led to positive diagnoses are much more likely to be accessed than images that led to negative diagnoses.
- Doctors vary significantly from one to another in their probability of viewing medical images.
- The image data of surgery patients and critically-ill patients have a much higher probability of being accessed than that of ordinary patients.

In our method, medical data is mined to identify the patients who correspond to the PACS's cached items. A machine learning algorithm is then used to construct a predictive model that is based on the variability of the treatment process. This model is then used to predict the future access probability of each cached item, thus improving cache performance.

The remainder of this paper is organized as follows: hybrid storage architectures, cache replacement policies, and the $\ell 1$-regularized logistic regression (LR) algorithm are described in Section II. Section III describes the framework and algorithm of the LR-LRU policy. Section IV describes the procedures for variable extraction, data acquisition, and data preprocessing. Section V shows how the $\ell 1$-regularized LR algorithm is used to construct a predictive model. Section VI evaluates the performance of classifier models and the LR-LRU policy. Section VII provides the conclusion of this paper and suggests future directions for research.

II. MATERIALS AND METHODS

A. ARCHITECTURE OF HYBRID STORAGE SYSTEMS

PACSs often use hybrid storage systems that combine SSDs and HDDs [6]–[8]. Hybrid storage systems are generally configured in one of two basic architectures. In the first

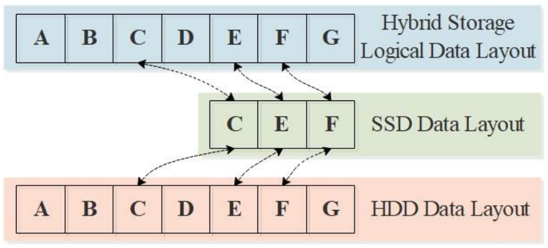


FIGURE 1. Hybrid storage architecture where the SSD is used as cache memory for the HDD.

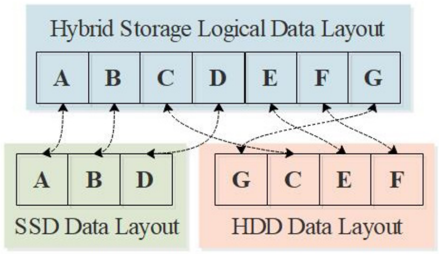


FIGURE 2. Tiered SSD-HDD architecture.

architecture, the SSD is used as cache memory for the HDD, and the logical addresses of the hybrid storage system correspond one-to-one to physical addresses in the HDD. The SSD only caches "hot" data, and the total capacity of the HDD is the total capacity of the hybrid storage system. A schematic of this architecture is shown in Figure. 1. When an access request occurs, the system will first search the SSD. If a "cache hit" occurs, the file will be read from the SSD. Otherwise, the data will be transferred from the HDD to the SSD before being read. If the SSD is already full, some of the cached data will be replaced according to the cache replacement policy. This architecture is suitable for file storage systems. Since a PACS mainly stores image files, this architecture is very suitable for PACSs.

In the second architecture, the SSD is used as an extension of the HDD and addressed in unison with the HDD. The total capacity of the hybrid storage system is therefore the sum of SSD and HDD capacities [6]–[8]. This hybrid storage architecture is illustrated in Figure. 2. This architecture is best suited for database storage systems. A page is either placed on the SSD or HDD, depending on the workload; the page may also be migrated between the SSD and HDD according to its migration cost. Read-intensive pages are placed on the SSD, while write-intensive and frequently updated pages are placed on the HDD, thus making full use of the contrasting strengths of SSDs and HDDs.

B. CACHE REPLACEMENT POLICIES

1) CONVENTIONAL CACHE REPLACEMENT POLICIES

This section provides a review of conventional cache replacement policies, which may generally be divided into five categories.

TABLE 1. The conventional replacement policies.

Policy	Brief description	Advantages	Disadvantages
RAND	A random number generator is used to determine the replaced object	The algorithm is the simplest and easy to implement	•As no factors are considered, the performance is unstable •Low hit ratio
LRU	The least recently used objects are removed first	It is easy to implement and the hit rate is low	•Only time factors were considered and other factors were ignored •Cache contamination
LFU	The least frequently used objects are removed first	It can avoid cache contamination	•Only the frequency factors are considered, and other factors are ignored •Hard to implement
SIZE	Big objects are removed first.	It is easy to implement, keep small objects first and has a high cache hit ratio	•Stores small Web objects first even if these objects are never accessed again. •Low byte hit ratio
GDSF	The factors of frequency and size are integrated, and the aging factor is introduced as well as factors of time	Overcomes the weakness of the SIZE policy by removing objects which are no longer requested by users	•The calculation cost is high and the parameter adjustment is complex •Low byte hit ratio

- Random algorithm (RAND)-based policies: These policies use a software or hardware-based random number generator to determine the files that will be replaced in the cache. This is by far the simplest type of cache replacement policy, and therefore the easiest to implement. However, RAND policies do not account for any of the factors that affect hit ratio and thus have low hit ratios. This is why they are rarely used in practice.
- Least recently used (LRU) algorithm-based policies [9]: In these policies, cached items that have not been accessed for the longest time are targeted for replacement. LRU policies are relatively simple to implement and have low time complexities. However, they are susceptible to cache pollution, since factors like filesize and access frequency are overlooked in these policies.
- Least frequency used (LFU) algorithm-based policies [10]: In these policies, the items that are accessed least frequently are selected for replacement, which prevents the occurrence of cache pollution. Clearly, this is a very reasonable policy. However, the implementation of this policy is extremely difficult; a counter must be set up for each cached item, and a fixed clock must be selected to periodically evaluate all counters. Furthermore, this policy only considers the access frequency of the cached items and ignores all other factors.
- Filesize-based (SIZE) policies [11]: In filesize-based SIZE policies, the largest items in the cache are replaced when space is needed to store new items. These policies are simple and easy to implement. However, the cache could become contaminated by small items, as it is very difficult to replace small items even if they are never accessed again. SIZE policies are suitable as web cache replacement strategies, as they have high hit ratios but low byte hit ratios.
- Function-based policies, such as greedy dual size (GDS) and greedy dual-size frequency (GDSF)-based cache replacement [12]: With these policies, cache performance can be optimized by selecting the appropriate weighting parameters. These policies are thus applicable to a variety of usage scenarios, as they can account for multiple influencing factors. However, the selection of appropriate weighting parameters is extremely challenging and new problems may arise from the computation of function values.

Table 1 summarizes the strengths and weaknesses of the aforementioned cache replacement policies. It may be observed that these policies only consider a single factor or use mathematical methods to predict access probabilities. As the storage environment changes very rapidly and is constantly being updated, these policies are not generally very efficient. Due to the need to improve cache performance, a multitude of intelligent cache replacement policies have been proposed in the literature.

2) INTELLIGENT CACHE REPLACEMENT POLICIES

Numerous intelligent cache replacement policies have been proposed in recent years. The strategies proposed in References [1]–[5] can generally be divided into two categories. In the first category, intelligent algorithms are used independently as cache replacement policies; in the second category, intelligent algorithms are combined with conventional cache replacement policies. Both of these approaches rely on the prediction of future access probabilities to enhance cache performance.

Reference [1] proposed a Markov chain-based predictive model called the predictive greedy dual-size frequency artificial intelligence (PGDSF-AI) policy. Firstly, a Markov chain model is constructed according to user usage habits. This is then used to predict user requests. When the cache runs out of space, the cached items that are not in the set of predicted items and have the lowest key values will be replaced. Reference [2], semantic analysis concepts like semantic segments, probe queries, and remainder queries were used to construct a segment access-aware semantic cache, which predicts the items that will be accessed by users in the future to improve cache hit ratio. In the method proposed by Reference [3], web log mining is performed to construct an algorithm that predicts the future access frequency of each page. The frequency predicting model was then incorporated into the GDSF caching policy to improve the GDSF policy's performance. Reference [4], multiple logistic regression (MLR) was used to categorize cache items into multiple classes. The MLR classifier was then combined with the LRU algorithm to form a new cache replacement policy called LRU-M, where the cache is divided into multiple class-based queues that are managed by the LRU algorithm. Reference [5], three intelligent cache replacement policies were developed by combining machine learning with conventional cache replacement policies; each of these policies was designed to handle a different application.

C. THE $\ell 1$-REGULARIZED LOGISTIC REGRESSION ALGORITHM

LR is a generalized linear algorithm [13] that is commonly used in machine learning for data mining, webpage classification, and economic forecasting. LR excels in describing the relationship between dependent and independent variables. The probability of an event occurring can be predicted simply by fitting data to an LR function, which is very useful for applications that rely on probabilistic decision-making. The logistic probability function of the LR algorithm is:

$$y = \phi(x) = \frac{1}{1+e^{-x}} \tag{1}$$

The dependent variable is $y \in \{0, 1\}$, and the eigenvector, which consists of n independent variables, that determines the predicted result is $x = (x_1, x_2, x_3, \cdots, x_n)$. If the regression coefficient is expressed as a vector $\beta = (\beta_1, \beta_2, \beta_3, \cdots, \beta_n)$, the a posteriori probability estimation is:

$$p\left(y=1\,|x\right) = \phi\left(\beta^T x\right) = \frac{1}{1+e^{-\beta^T x}} \tag{2}$$

The maximum likelihood method may then be used to estimate the value of β. Given that the training dataset is $D_m = \{(x_i, y_i)\}_{i=1}^m$, the maximum log-likelihood of the LR model is: [14]:

$$\arg\max_\beta\{\ln p\left(\beta\,|D_m\right)\} = \arg\max_\beta\left(\sum_{i=1}^m \ln y_i\phi(\beta^T x_i)+(1-y_i)\right.$$
$$\left.\times\left(1-\phi(\beta^T x_i)\right)\text{-lnp}\left(\beta\right)\right) \tag{3}$$

We created a penalty function based on the concepts of the LASSO penalty function [15], that is, by adding an $\ell 1$-norm function to the model's coefficients to obtain a more refined LR model, the $\ell 1$-regularized LR model:

$$\arg\max_\beta\{\ln p\left(\beta\,|D_m\right)\} = \arg\max_\beta\left(\sum_{i=1}^m \ln y_i\phi(\beta^T x_i)+(1-y_i)\right.$$
$$\left.\times\left(1-\phi(\beta^T x_i)\right) - \lambda\,\|\beta\|_1\right) \tag{4}$$

Highly-efficient algorithms have been proposed by Shevade and Keerthi [16], Zhang [17], Kin *et al.* [18] to solve the $\ell 1$-regularized LR model. Here, we used the "Shooting" algorithm proposed by Balakrishnan and Madigan to solve the $\ell 1$-regularized LR model [14].

Firstly, the LR log-likelihood is:

$$\sum_{i=1}^m \ln\left(p\left(y_i\,|\beta\right)\right) = \sum_{i=1}^m \ln\left(y_i\phi(\beta^T x_i)\right)$$
$$+\,(1-y_i)\left(1-\phi(\beta^T x_i)\right) \tag{5}$$

When yi = 1 or 0, the log-likelihood functions are ln $\phi(\beta$Txi) and ln $\phi(1-\beta$Txi), respectively, as shown below:

$$\sum_{i=1}^m \ln\left(p\left(y_i\,|\beta\right)\right) = \begin{cases} \sum_{i=1}^m \ln\phi(\beta^T x_i), & y_i = 1 \\ \sum_{i=1}^m \ln\left(1-\phi(\beta^T x_i)\right), & y_i = 0 \end{cases} \tag{6}$$

In both of these cases, the log-likelihood function can be approximated by a Taylor expansion around $\beta_{i-1}^T x_i$, thus converting the original optimization problem into an approximate optimization problem.

$$\arg\max_\beta\left\{\ln p\left(\beta\,|D_m\right)\right\} \approx \arg\max_\beta\left(\beta^T\Psi_m\beta+\beta^T\theta_m-\lambda\,\|\beta\|_1\right) \tag{7}$$

where $\Psi_m = \sum_{i=1}^m x_i(x_i)^T$ and $\theta_m = \sum_{i=1}^m x_i.\Omega$ is defined as $\Omega = 2\Psi\beta+\theta$, while ai and bi are obtained by the Taylor expansion of the log-likelihood function around $\beta_{i-1}^T x_i$. The procedure of the "Shooting" algorithm is thus:

- Select an initial value for iteration, β_0.
- Iterate the algorithm; on the m-th step, perform the following operation for i = 1 to p:

$$\beta_i = \begin{cases} 0, & if\ \Omega_i \le \lambda \\ \dfrac{\lambda-\Omega_i}{2\Psi_{ii}} & if\ \Omega_i \succ \lambda \\ \dfrac{-\lambda-\Omega_i}{2\Psi_{ii}} & if\ \Omega_i \prec -\lambda \end{cases} \tag{8}$$

  After the p-th calculation, use the newly-estimated value to replace the result of the previous step.
- Repeat Step 2 until β_m converges.

Compared to the LR model and $\ell 2$-regularized LR model, the $\ell 1$-regularized LR model is very effective in sparsifying variables when solving for model coefficients, since the $\ell 1$-regularization reduces the absolute values of the coefficients. This also causes the coefficients of variables that have a minimal impact on the dependent variable of the model to approach 0, thus screening these independent variables.

III. THE PACS-ORIENTED LR-LRU INTELLIGENT CACHE REPLACEMENT POLICY

A. THE LR-LRU INTELLIGENT CACHE REPLACEMENT POLICY

LRU is a conventional cache replacement policy. However, the LRU policy often results in cache pollution, which means that unwanted items are often left in the cache for long periods of time. In the LRU policy, new objects are inserted onto the top of the cache stack. Even if this object is not accessed in the future, it will take a long time before it reaches the bottom of the stack and is removed from the cache.

Algorithm 1 LR-LRU

```
Input: each file X requested by user
1: If X is in SSD
2:     Cache hit occurs   return X //file X read completion
3:   Else
4:     nFetch X from HDD to SDD
5:   End
6: While not enough space in SDD for X
7:     Evict Y such that Y is at the bottom of the
       cache stack
8:     Insert X at the bottom of the cache stack
9:     Cache hit occurs   return X //file X read completion
10: End
11:     Use LR-LRU algorithm to update probability
        $P_X$ of X
12:     Predict whether X is cold data or hot data
13: If X is cold data
14:     Move X to the middle of the cache stack
15:   Else
16:     Move X to the top of the cache stack
17:   End
Output: X
```

To mitigate the cache pollution problems of the LRU policy, we have combined the LR algorithm with LRU to form the LR-LRU intelligent cache replacement policy. The workflow of the proposed LR-LRU policy is shown below. When a user requests a file, X, the LR algorithm predicts the future access probability of this file, P. If P is greater than the threshold, ξ, X is adjudged to be "hot" data and stored at the top of the cache stack. Otherwise, X is adjudged to be "cold" data and stored in the middle of the cache stack, so as to accelerate its eviction from the stack. The pseudocode of the LR-LRU policy is shown below:

B. ARCHITECTURE OF THE LR-LRU INTELLIGENT CACHE REPLACEMENT POLICY

The architecture of the proposed LR-LRU intelligent cache replacement policy is illustrated in Fig. 3. This architecture consists of three functional components: the extract-transform-load (ETL) component, offline component, and online component.

The ETL component is responsible for extracting distributed and differentiated medical data (e.g., HIS, RIS, and EMR data) and depositing it in a temporary intermediary layer, where the data is cleaned, converted, and collected before being uploaded to the target data center. The uploaded database is the basis of all subsequent data analysis and data mining operations [18], [19].

The offline component does not directly handle user requests, as its role is to train the LR-LRU policy when the server is idle. The updated LR-LRU policy is then used by the cache manager in the online component.

The online component is responsible for managing the cache manager. When a user inputs an access request, the variable of the cached item is first obtained from the target data center. The variable is then forwarded to the cache manager, where the LR-LRU policy is used to perform cache management.

IV. VARIABLE SELECTION AND DATA PREPROCESSING

In this work, a trace-driven simulation was used to experimentally examine the LR-LRU policy. The experimental data was derived from the PACS of Yancheng 1st People's Hospital, and the data was collected between 7th May 2018 and 27th May 2018 (a total of 21 days). Firstly, a trace recorder was constructed to analyze the PACS logfiles in real time. In the log, each record represents a query, which may be treated as a transaction; a transaction may query one or many cached items. For the purposes of this experiment, each line of the log was interpreted as a transaction file.

A. INITIAL SCREENING OF VARIABLES

The selection of appropriate variables is crucial for the construction of a mature and efficient predictive model, and it is also an important challenge in this work. Although a very large number of patient treatment-related variables can be obtained using the ETL component, only some of these variables are suitable for model building. The conditions for the selection of variables in this work are as follows:

- The variable is related to cached file requests
- The variable can be quantified or coded
- The variable is uniquely defined at each point in time
- The variable can be obtained via simple computations or queries

A total of 124 suitable variables were obtained after the initial screening process.

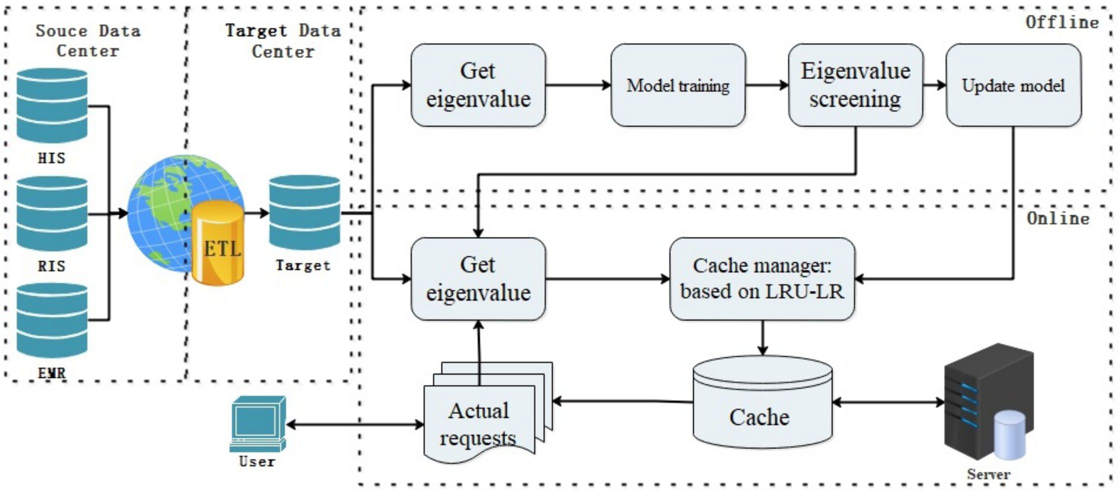


FIGURE 3. Architecture of the intelligent LR-LRU cache replacement policy.

B. DATA PREPROCESSING

The transaction files collected by the ETL must be preprocessed before they can be used in the simulation experiment. The data preprocessing procedure deletes irrelevant or invalid requests, adds relevant variables, and labels the attributes of the cached files. This procedure is as follows:

- Filtering: Filter irrelevant or invalid tasks like un-cached requests, unsuccessful tasks, and tasks with incomplete records.
- Interpretation: Interpret the meaning of each field in the transaction files, convert all files to a unified format, and delete unnecessary fields.
- Completion: Fill in the variable information of the transaction files' requests through the ETL component.
- Labeling: If a cached item is accessed within 24 hours of its previous query, it is labeled as hot data. Otherwise, the cached item is labeled as cold data.
- Finalization: The 21-day transaction file dataset obtained through the abovementioned data preprocessing procedure was divided into two parts. The first 14 days of the dataset were used to train the model, and the last 7 days of the dataset were used in the simulation experiment.

The preprocessed transaction file dataset includes the cached items' records, variables, and labels.

V. CONSTRUCTION OF THE ℓ1-LR MODEL

After the transaction file dataset has been prepared, the LR algorithm may then be trained using the training dataset to construct a classifier for the cached items. The testing dataset is then used to test and optimize the model. LASSO and the LR library were used to construct the LR-LRU policy in Matlab [20].

A. MODEL TRAINING

The first 14 days of the preprocessed data were selected to form a dataset, D by stratified sampling method. 10-fold cross validation was used to divide D into 10 mutually exclusive subsets of equal size, i.e., $D = D_1 \cup D_2 \cup \cdots \cup D_{10}, D_i \cap D_j = \emptyset (i \neq j)$. The data distribution of each D_i subset was made to be as consistent as possible; each subset was obtained from D via time-stratified sampling. The union of 9 different subsets was used as the training set, while the last set was used as the testing set. In this way, we obtained 10 training sets and 10 testing sets, which allowed the model to be trained and tested 10 times. The returned results correspond to the averaged results of 10 tests. 10-fold cross validation is illustrated in Fig. 4.

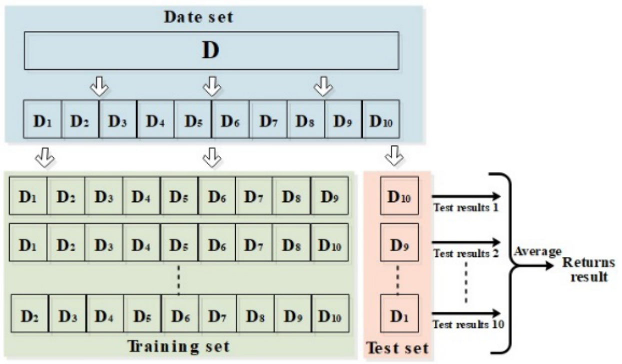

Date set
D
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10
D1 D2 D3 D4 D5 D6 D7 D8 D9
D10
D1 D2 D3 D4 D5 D6 D7 D8 D10
D9
D2 D3 D4 D5 D6 D7 D8 D9 D10
D1
Training set
Test set
Test results 1
Test results 2
Test results 10
Average
Returns result

FIGURE 4. 10-fold cross validation.

B. CONSTRUCTION OF THE ℓ1-LR MODEL AND VARIABLE SELECTION

Firstly, suitable values were selected for the 124 previously-selected variable through Wald test-based backward selection. The access probability (dependent variable) of each file was defined as $y \in \{0, 1\}$, while the eigenvector, which consists of n variables, that influences the result of the prediction was defined as $x = (x_1, x_2, x_3, \cdots, x_n)$. The regression coefficient was expressed as a vector, $\beta = (\beta_1, \beta_2, \beta_3, \cdots, \beta_n)$. Hence, the i-th variable and regression coefficients are x_i and β_i, respectively.

TABLE 2. The selected variables and their definitions.

NO.	Variable	Definition	Assignment
1	Reporting state	Report completed or not	0: not completed, 1: completed
2	Patient status	Patient discharged or not	0: discharged, 1: not discharged
3	Report results	Result of examination report	0: positive, 1: negative
4	Critical Patient	Critical patient or not	0: No, 1: Yes
5	Surgical patient	Surgical patients or not	0: No, 1: Yes
6	Attendings	Doctor 1	0: No, 1: Yes
		Doctor 2	
		…………	
		Doctor *n*	
7	Check time	<24 h	
		24-48 h	0: No；1: Yes
		>48 h	
8	Inspection project	CT	
		DR	0: No；1: Yes
		MIR	
		B-ultrasound	
		Other	
9	Classification of disease	I	
		II	0: No；1: Yes
		III	
		IV	

Significance tests were performed on all x_i variables and y dependent variables; the variables that failed to meet significance requirements were removed one by one. The null hypothesis is denoted as H_0: $\beta_i = 0$, which indicates that the independent variable x_i did not alter the occurrence probability of the event. If the null hypothesis is disproved, x_i then has an effect on the occurrence probability of the event. The Wald test (Eq. (9)) is often used to test the significance of regression coefficients:

$$Wald = \left[\frac{\beta_i}{SE\left(\beta_i\right)}\right]^2 \tag{9}$$

where $SE\left(\beta_i\right)$ is the standard deviation of β_i. However, the standard deviation of a regression coefficient will be embellished if its absolute value is very large; this causes the value of the Wald statistic to shrink, which increases the probability of erroneous results. In other words, the Wald test may fail to disprove the null hypothesis even if the regression coefficient is indeed significant. $\ell 1$-regularization was incorporated to solve this problem. During the calculation of the model's coefficients, $\ell 1$ regularization shrinks the absolute values of the coefficients and screens insignificant variables (with respect to the model's dependent variables) by having their coefficients approach 0 in value. Nine variables were selected by training the $\ell 1$-LR model, as shown in Table 2. These variables were used as the variables of the final model.

Variables 1-5 are binary variables, which are either 0 or 1 depending on whether they are true or false, whereas variables 6-8 are multiclass variables. Multiple classifiers are needed to handle multiclass problems. Given a dataset $D \in \mathbb{R}^{m\times n}$, whose labels are $Y \in \mathbb{R}^k$, these samples will have k different classes. The samples that were labeled as $c\,(c \leq k)$ are selected and marked as '1', whereas the samples that were not labeled as c are marked as '0'. These data are then used to train a classifier. An LR classifier function for c-labeled samples, $h_c\left(x\right)$, was thus obtained. The procedure above can be used to obtain k different β values.

SPSS 21.0 was then used to perform statistical tests on the nine variables and β_0 constant that were selected, which are shown in Table 3, for the final model. Here, '*Sig*.' represents the P value of the variable in the model; the variables' differences are of statistical significance if $P \leq 0.010$ [21]. '$Exp(\beta)$' is the odds ratio (OR) of the variable, i.e., the ratio between high and low variable values in terms of the number of event occurrences. Hence, $OR = 1$ or OR values close to 1 indicate that the variable does not affect event occurrence. In LR, a multiclass variable is always treated as a single entity, that is, as long as one of the classes of the multiclass variable is statistically significant in terms of P and OR, the multiclass variable will be included in the model.

When using Wald test-based backward selection, the correlation between the variables must be taken into account,

TABLE 3. Coefficients and Wald tests for L1-LR.

Variable	Classification	β[1]	S.E[2]	Wald[3]	Df[4]	Sig.[5]	Exp(β)[6]
Reporting state	Bipartition	0.435	0.111	15.358	1	0.002	2.569
Patient status	Bipartition	0.291	0.059	24.327	1	0.001	3.256
Report results	Bipartition	1.488	0.365	16.620	1	0.000	5.231
Critical Patient	Bipartition	2.378	0.416	32.677	1	0.000	1.654
Surgical patient	Bipartition	2.111	0.281	56.437	1	0.000	1.701
Attendings	Doctor 1	-0.097	0.119	0.664	879	0.383	1.001
	Doctor 2	0.394	0.132	8.909	879	0.010	1.488
	…………						
	Doctor *n*	0.879	0.076	133.767	879	0.001	2.365
Check time	<24 h	0.459	0.105	19.109	3	0.000	1.529
	24-48 h	0.031	0.180	0.030	3	0.000	1.029
	>48 h	0.602	0.123	23.954	3	0.000	0.550
Inspection project	CT	0.917	0.121	57.434	5	0.000	2.371
	DR	0.301	0.104	8.377	5	0.001	1.171
	MIR	0.523	0.123	18.080	5	0.000	1.345
	B-ultrasound	0.084	0.109	0.594	5	0.000	1.005
	Others	0.371	0.115	10.408	5	0.001	1.219
Classification of disease	I	0.934	0.123	57.661	4	0.001	2.339
	II	0.584	0.122	22.914	4	0.000	1.488
	III	0.402	0.116	12.010	4	0.000	0.550
	IV	0.822	0.131	39.373	4	0.002	0.397
Constant		4.186	0.211	393.580	1	0.003	0.474

1 β = logistic coefficient

2 S.E. = standard error of estimate

3 Wald = Wald Chi-square values

4 Df = degrees of freedom

5 Sig. = Significance

6 Exp(β) = exponentiated coefficient

since this method should not be used if the variables are strongly correlated [21]. SPSS 21.0 was used to analyze correlations between the selected variables. A correlation matrix was computed, which shows how the variables are correlated to each other. The correlation coefficient ranges within $[-1,1]$; a pair of variables are strongly correlated if the absolute correlation approaches 1, and weakly correlated if the absolute correlation approaches 0. The correlation matrix is shown in Fig. 5.

Fig. 5 indicates that only 4 sets of variables have absolute correlations greater than 0.6. The other 32 sets have absolute correlations less than 0.5. Hence, the selected variables are only weakly correlated to each other and the model has a high level of sparsity.

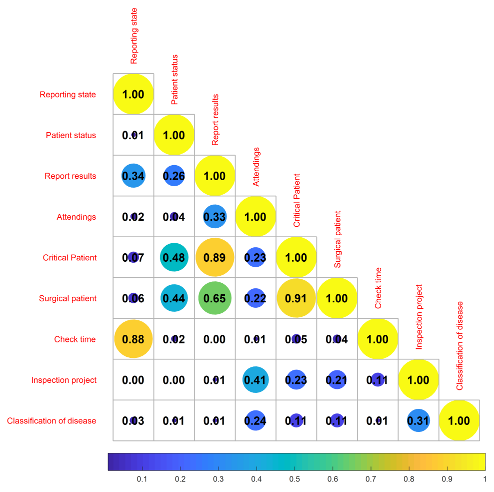


FIGURE 5. Correlation matrix of the variables.

VI. RESULT ASSESSMENT AND DISCUSSION

A. VALIDATION OF THE ℓ1-LR MODEL

The results of binary classification problems can be categorized as true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) depending on how the predicted result compares to reality. The confusion matrix of the classification results is shown in Table 4.

TABLE 4. Confusion matrix.

Actual situation	Predicted results	
	Predicted positive	Predicted negative
Positive	True positive (TP)	False negative (FN)
Negative	False positive (FP)	True negative (TN)

The predicted value is compared to a threshold value, ξ; a prediction is positive if the predicted value is greater than ξ and negative otherwise. Hence, reducing the value of ξ will produce a larger number of positive results and thus increase the percentage of actual positives among the positive predictions, i.e., the true positive rate (TPR). However, this also increases the number of negative examples among the positive predictions, i.e., the false positive rate (FPR). The receiver operating characteristic (ROC) curve was plotted to visualize these changes. TPR and FPR are the vertical and horizontal axes of the ROC curve, respectively. Based on Table 4, the definitions of TPR and FPR are:

$$TPR = \frac{TP}{TP + FN} \tag{10}$$

$$FPR = \frac{FP}{TN + FP} \tag{11}$$

LR and ℓ2-LR models were constructed using the same methods, and then compared to the ℓ1-LR model. Fig. 6 illustrates the ROC curves of these models.

The ROC curves allow the strengths and weaknesses of the three models to be directly evaluated. If the ROC curve of a model is completely covered by the ROC curve of another model, the latter model is superior to the former. In other words, the area under the ROC curve (AUC) can be used to determine the performance of a classifier, as the performance of a classifier is proportional to its AUC. Figure. 6 shows that the ROC curve of the ℓ1-LR model completely covers those of the LR and ℓ2-LR models. The ℓ1-LR model also has the largest AUC. Hence, the performance of the ℓ1-LR model is superior to that of the ℓ2-LR and LR models.

B. EVALUATION OF THE LR-LRU INTELLIGENT CACHE REPLACEMENT POLICY

1) EVALUATING METRICS FOR CACHE REPLACEMENT POLICIES

The two most important metrics for evaluating cache performance are hit ratio (HR) and byte hit ratio (BHR). HR is the number of accesses obtained from the cache as a percentage of all access requests, while BHR is the percentage of bytes obtained from the cache over the total number of requested bytes.

The equations for calculating HR and BHR are shown in Eqs. 12 and 13, respectively. N is the number of access requests, and each request $i(1 \leq \mathrm{i} \leq \mathrm{N})$ corresponds to a cached item. m_i is the filesize of the i-th cached item. qi indicates whether the i-th item is a hit ($q_i = 1$ for hits and $q_i = 0$ for misses).

$$HR = \frac{\sum_{i=1}^{N} q_i}{N} \times 100\% \tag{12}$$

$$BHR = \frac{\sum_{i=1}^{N} q_i m_i}{\sum_{i=1}^{N} m_i} \times 100\% \tag{13}$$

HR and BHR have different purposes, as HR mainly focuses on reducing user response times and improving the user experience, whereas BHR is about reducing bandwidth consumption. It is extremely difficult for a cache replacement policy to simultaneously optimize HR and BHR [22], [23]. To increase HR, the cache has to store as many small cached items as possible and maximize the density of cached items. However, this will reduce BHR. Conversely, given a cache of finite size, maximizing the storage of large files in the cache will increase BHR but reduce the density of the cached items, thus reducing HR. Therefore, the aim of this work is to create a cache replacement policy that is able to achieve excellent results in one of these metrics without significantly degrading the other metric.

2) COMPARISON BETWEEN THE LR-LRU POLICY AND CONVENTIONAL CACHE REPLACEMENT POLICIES

The trace-driven simulation method was used to perform a simulation experiment [24]. The data used in this experiment is the transaction file dataset that was prepared in Section IV.B. Prior to the experiment, it is necessary to determine the termination point of the experiment, that is, the size of the "infinite" cache. An infinite cache is a cache that is large enough to store all cached items without needing to replace any items. Hence, the infinite cache size is the sum of the file sizes of all cached items. An infinite cache will also allow HR and BHR to reach their maximum values. However, it is very difficult to realize an infinite cache due to cost factors. It was experimentally determined that the infinite cache size was 161341.79 GB, and the corresponding maximum HR and BHR values were $HR_{max} = 42.36\%$ and B$HR_{max} = 45.18\%$. 10 cache sizes were used in the simulation experiment, and the cache sizes ranged from 25 GB to 214 GB.

The experiment was performed using the LR-LRU policy and the conventional GDSF, GDS, LRU, LFU, and SIZE policies. Figures. 7 and 8 show the HR and BHR corresponding to each policy for each cache size. An increase in cache size always increases HR and BHR, for any given policy. When the size of the cache approaches the infinite cache size, HR and BHR values will plateau and approach their maximum values.

In terms of HR (Figure. 7), it is clear that the LR-LRU policy has better levels of performance than the conventional LRU policy, thus proving the effectiveness of combining the LR algorithm with the LRU policy. The access frequencies of the PACS storage system's image cache do not vary significantly, as there are very few excessively "hot" or

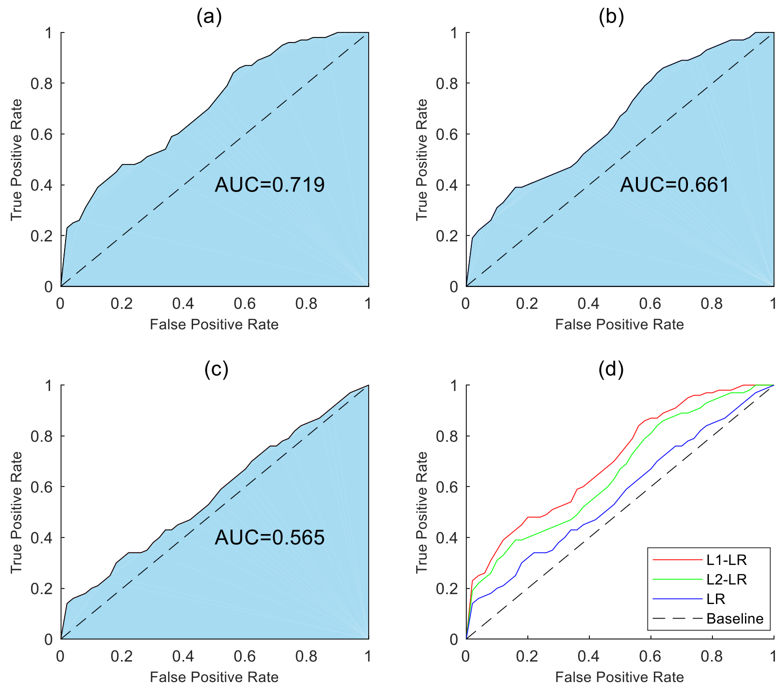


FIGURE 6. Comparison between the ROC curves of the LR, ℓ2-LR, and ℓ1-LR models.

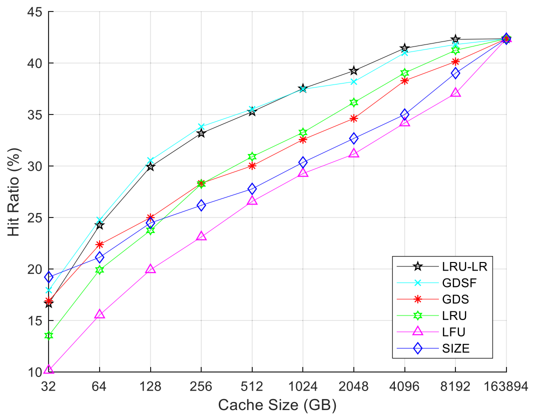


FIGURE 7. Comparison between the HR of several algorithms.

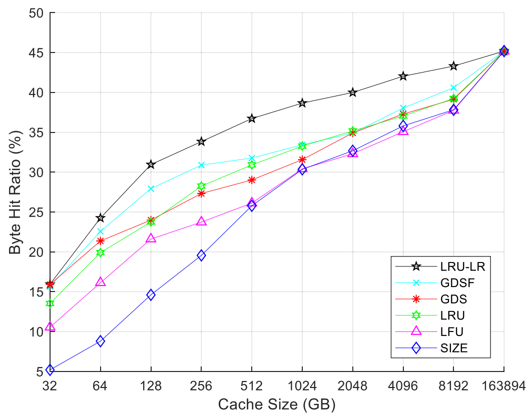


FIGURE 8. Comparison between the BHR of several algorithms.

"cold" cached files. Consequently, the LFU policy (which depends on a frequency factor) exhibits the worst performance between all caching policies. However, the filesizes in the PACS storage system's image cache do vary quite significantly. For example, B-scan ultrasonography and digital radiography (DR) files only range between 1 MB to 10 MB in size while CT and MRI files can be several hundreds of MBs in size. Therefore, filesize-dependent policies like SIZE, GDS, and GDSF hold very large advantages over other caching policies for small cache sizes. However, HR grows more slowly with increasing cache size for these policies than other policies, as they tend to discard large cached files with excessive frequency.

In terms of BHR (Figure. 8), the LR-LRU, LRU, and LFU policies display similar BHR and HR trends with respect to cache size since these policies are not filesize-dependent. The SIZE, GDS, and GDSF policies tend to favor small cached items; hence, these policies sacrifice BHR to obtain high HR values. At small cache sizes, these policies perform better than the LR-LRU policy in terms of HR, but significantly poorer in terms of BHR.

To provide a more comprehensive assessment of these policies, the improvement ratio (IR) was used to evaluate these policies [5]. The equation for calculating IR is shown in Eq. 14. PM is the "proposed method," while CM is the "comparative model." The IR values of the LR-LRU policy (PM)

TABLE 5. IR of the LR-LRU policy with respect to conventional cache replacement policies.

Cache Size (GB)	LR-LRU Over GDSF (%)		LR-LRU Over GDS (%)		LR-LRU Over LRU (%)		LR-LRU Over LFU (%)		LR-LRU Over SIZE (%)	
	HR	BHR	HR	BHR	HR	BHR	HR	BHR	HR	BHR
32	-3.25	1.88	-1.60	0.19	22.99	15.12	63.94	33.81	-13.38	67.38
64	-2.10	6.93	8.40	11.88	21.80	17.90	56.15	33.48	14.77	63.71
128	-2.03	9.76	19.77	22.49	26.02	23.26	50.25	30.18	22.31	52.79
256	-1.92	8.72	17.16	19.24	17.45	16.49	43.57	29.86	26.74	42.18
512	-0.70	13.48	17.49	20.97	14.11	15.82	32.79	28.87	26.96	29.79
1024	4.78	13.53	20.51	18.34	18.01	13.97	34.10	21.34	29.32	21.50
2048	1.20	12.75	11.64	12.68	6.86	12.05	24.04	19.23	18.27	18.28
4096	1.10	9.49	8.23	11.28	6.15	11.87	21.24	16.56	18.40	14.82
8192	1.20	6.26	5.38	9.61	2.57	9.38	14.17	12.82	8.41	12.66
163894	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

with respect to other cache replacement policies (CMs) are summarized in Table 5.

$$IR = \frac{(\text{PM-CM})}{CM} \times 100\% \tag{14}$$

As a whole, the LR-LRU policy is superior to all other policies in terms of BHR, at all cache sizes. In terms of HR, the LR-LRU policy is slightly inferior to the SIZE, GDS, and GDSF policies (which use size factors) at small cache sizes. This is because these policies sacrifice BHR to maximize HR. It is thus shown that the LR-LRU policy has successfully obtained a large advantage in BHR with a relatively small cost in HR.

The LR-LRU policy has significantly better HR and BHR values than the LRU policy, and the maximum IRs were obtained when the cache size was 128 GB ($IR_{HR} = 26.03\%$ and $IR_{BHR} = 23.26\%$). By analyzing the PACS logs, it was found that the LR-LRU policy is faster than the LRU policy at predicting and replacing cold data in the cache, especially at lower cache sizes. Hence, the combination of the LR algorithm and LRU policy has been proven to be highly effective. The HR and BHR values of the LR-LRU policy are also excellent compared to the LFU policy. Compared to GDS and SIZE, the LR-LRU policy only has slightly lower HR values when the cache size was 32 GB; under all other circumstances, the HR and BHR of the LR-LRU policy are superior to these policies. The GDSF policy, which combines frequency and size factors and also utilizes aging and time factors, has a better HR but a poorer BHR than the LR-LRU policy at small cache sizes. At small cache sizes, the LR-LRU policy achieves a large advantage in BHR for a small cost in HR. LR-LRU gradually begins to outperform GDSF in both HR and BHR as cache size increases.

In summary, the LR-LRU policy outperforms all conventional cache replacement policies in terms of BHR, while retaining an acceptable level of performance in HR, especially at larger cache sizes. As a whole, the LR-LRU policy is superior to conventional cache replacement policies. This is in-line with the aims stated in Section V.B.1 as we have created a cache replacement policy that performs excellently in one metric, while achieving adequate levels of performance in the other metric.

VII. CONCLUSION AND OUTLOOK

In this work, we have proposed an intelligent cache replacement policy that is suitable for PACS storage systems. This policy combines the LR algorithm with the LRU policy to form the LR-LRU intelligent cache replacement policy. Unlike currently-existing policies that solely rely on the cached items' intrinsic properties (e.g., filesize, last access time, and access frequency), our PACS-oriented LR-LRU policy identifies the variables that affect file access probabilities by mining medical data and models the future access probabilities of cached items using the LR algorithm, thus improving the performance of the caching policy. $\ell 1$-regularization was also used to shrink the absolute values of the variables' coefficients, thus screening less important variables by causing their coefficients to approach zero. It was experimentally demonstrated that the LR-LFU policy significantly improves cache performance compared to the LRU, LFU, SIZE, GDS, and GDSF policies.

Although the LR-LRU policy proposed in this work is oriented towards hospital PACSs, the core concepts of this policy can be extended to other fields, such as electric utilities, banking, aerospace, and telecommunications. These industrial systems are similar to PACSs as they handle large amounts of data, much of which contains a large number of dimensions and are strongly correlated. Data mining can be used to identify the most relevant variables for cached items; machine learning algorithms may then be used to construct predictive models that predict the future access probabilities of the cached items, thus improving cache performance.

REFERENCES

[1] H.-Y. Huang and Z. Q. Zhong, "Web cache replacement algorithm based on multi-Markov chains prediction model," *Microelectron. Comput.*, vol. 31, no. 5, pp. 123–125, 2014.

[2] K. Ma, B. Yang, Z. Yang, and Z. Yu, "Segment access-aware dynamic semantic cache in cloud computing environment," *J. Parallel Distrib. Comput.*, vol. 110, pp. 42–51, Dec. 2017.

[3] Q. Yang, H. H. Zhang, and T. Li, "Mining web logs for prediction models in WWW caching and prefetching," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2001, pp. 473–478.

[4] J. H. Chang and W. S. Lee, "*estWin*: Adaptively monitoring the recent change of frequent itemsets over online data streams," in *Proc. 12th Int. Conf. Inf. Knowl. Manage.*, Nov. 2003, pp. 536–539.

[5] W. Ali, S. M. Shamsuddin, and A. S. Ismail, "Intelligent Web proxy caching approaches based on machine learning techniques," *Decis. Support Syst.*, vol. 53, no. 3, pp. 565–579, Jun. 2012.

[6] R. Micheloni, L. Crippa, and M. Picca, *Hybrid Storage*, Springer, 2013, pp. 37–61.

[7] R. Hemmati, M. Shafie-Khah, and J. P. S. Catalão, "Three-Level Hybrid Energy Storage Planning under Uncertainty," *IEEE Trans. Ind. Electron.*, vol. 66, no. 3, pp. 2174–2184, Mar. 2019.

[8] C. Zhen, L. Wenjie, Z. Xiao, and B. Hailong, "Review on HDD-SSD hybrid storage," *J. Comput. Appl.*, vol. 37, no. 05, pp. 1217–1222, 2017.

[9] D. D. Sleator and R. E. Tarjan, "Amortized efficiency of list update and paging rules," *Commun. ACM*, vol. 28, pp. 202–208, Feb. 1985.

[10] D. Lee *et al.*, "On the existence of a spectrum of policies that subsumes the least recently used (LRU) and least frequently used (LFU) policie," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Modeling Comput. Syst.*, May 1999, pp. 134–143.

[11] G. P. Sajeev, "A novel content classification scheme for web caches," *Evolving Syst.*, vol. 2, no. 2, pp. 101–118, Jun. 2011.

[12] J. B. Patil and B. V. Pawar, "GDSF#, A better caching algorithm that optimizes both hit rate and byte hit rate in web proxy servers," *Int. J. Comput. Sci. Appl.*, vol. 5, no. 4, pp. 1–10, Jan. 2008.

[13] V. Bewick, L. Cheek, and J. Ball, "Statistics review 14: Logistic regression," *Crit. Care*, vol. 9, no. 1, pp. 112–118, Jan. 2005.

[14] S. Balakrishnan and D. Madigan, "Algorithms for sparse linear classifiers in the massive data setting," *J. Mach. Learn. Res.*, vol. 9, pp. 313–337, Feb. 2008.

[15] R. Tibshirani, "Regression shrinkage and selection via the lasso: A retrospective," *J. Roy. Statist. Soc. B (Statist. Methodol.)*, vol. 73, no. 3, pp. 273–282, 2011.

[16] S. K. Shevade and S. S. Keerthi, "A simple and efficient algorithm for gene selection using sparse logistic regression," *Bioinformatics*, vol. 19, no. 17, pp. 2246–2253, 2003.

[17] T. Zhang, "On the dual formulation of regularized linear systems with convex risks," *Mach. Learn.*, vol. 46, no. 1, pp. 91–129, Jan. 2002.

[18] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale $l1$-regularized least squares," *IEEE J. Sel. Topics Signal Process.*, vol. 1, no. 4, pp. 606–617, Dec. 2007.

[19] Q. Yao, Y. Tian, P. F. Li, L. L. Tian, Y. M. Qian, and J. S. Li, "Design and development of a medical big data processing system based on Hadoop," *J. Med. Syst.*, vol. 39, no. 3, p. 23, Mar. 2015.

[20] J. Yu, "Tool condition prognostics using logistic regression with penalization and manifold regularization," *Appl. Soft Comput.*, vol. 64, pp. 454–467, Mar. 2018.

[21] S. Sakai, K. Kobayashi, S. Toyabe, N. Mandai, T. Kanda, and K. Akazawa, "Comparison of the levels of accuracy of an artificial neural network model and a logistic regression model for the diagnosis of acute appendicitis," *J. Med. Syst.*, vol. 31, no. 5, pp. 357–364, Oct. 2007.

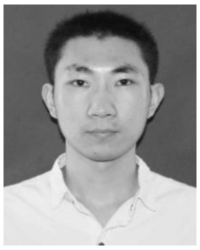
[22] G. Hughes, "Tjur's R2 for logistic regression models is the same as Youden's index for a 2×2 diagnostic table," *Ann. Epidemiology*, vol. 27, no. 12, pp. 801–802, Dec. 2017.

[23] S. Romano and H. Elaarag, "A neural network proxy cache replacement strategy and its implementation in the squid proxy server," *Neural Comput. Appl.*, vol. 20, no. 1, pp. 59–78, Feb. 2011.

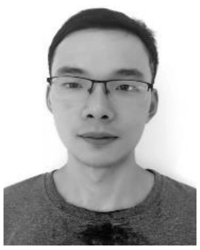
[24] R. A. Uhlig and T. N. Mudge, "Trace-driven memory simulation: A survey," *ACM Comput. Surv.*, vol. 29, no. 2, pp. 128–170, Jun. 1997.

YINYIN WANG was born in Yancheng, Jiangsu, China. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Nanjing University of Science and Technology, China. His research interests include machine learning, high performance computing, and medical big data.

YUWANG YANG received the B.S. degree from Northwestern Polytechnical University, in 1988, the M.S. degree from the University of Science and Technology of China, in 1991, and the Ph.D. degree from the Nanjing University of Science and Technology (NUST), in 1996. He is currently a Professor with the School of Computer Science and Engineering, NUST. His research interests include high performance computing, machine learning, and intelligent systems.



CHEN HAN was born in Nanjing, Jiangsu, China. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Nanjing University of Science and Technology, China. His research interests include network coding, high performance computing, and big data analysis.



LEI YE was born in Taizhou, Jiangsu, China. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Nanjing University of Science and Technology, China. His research interests include network coding, big data analysis, and machine learning.

YAQI KE received the master's degree in the discipline of phonology from Nanjing Agriculture University, China, in 2018. She is currently a Research Assistant with the Nanjing University of Science and Technology, Nanjing, China. Her research interests include the areas of agricultural model, agricultural informatics, and machine learning.

QINGGUANG WANG received the B.S. degree from Peking University. He is currently the Chief Pharmacist of Yancheng 1stPeople's Hospital and an Associate Professor with the Medical School, Nantong University.