# 服务于风电系统的改进缓存替换算法研究

鲁尔洁 陈 峦 李 坚 黄 琦 张真源 井 实 周统汉
(电子科技大学能源科学与工程学院 成都 611731)

**摘 要** 针对风电系统中缓存命中率较低等问题,在最近最少使用算法(Least Recently Used,LRU)、最不经常访问算法(Least Frequently Used,LFU)、SIZE以及Hybrid算法的基础上,提出了一种基于综合因素的替换算法FST(Frequency,Object Size,Access Time),从而解决了传统算法考虑因素单一、系统性能较低等问题。该算法结合了访问频率、对象大小、访问时间间隔及最久未访问等特性,并根据最近访问时间长短采取分段的处理方法。在风电系统的缓存服务器中,将FST算法与LRU,LFU和SIZE算法进行实验对比,实验结果显示FST算法在提高命中率、减少延迟时间方面具有更好的性能。

**关键词** 风电系统,综合因素,替换算法,命中率,延迟时间
**中图法分类号** TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.09.043

## Research on Improved Cache Replacement Algorithm Serving for Wind Power System

LU Er-jie CHEN Luan LI Jian HUANG Qi ZHANG Zhen-yuan JING Shi ZHOU Tong-han
(School of Energy Science and Engineering,University of Electronic Science and Technology of China,Chengdu 611731,China)

**Abstract** Aiming at the problems such as the low cache hit rate in wind power system,after researching on LRU (Least Recently Used),LFU (Least Frequently Used),SIZE and Hybrid,a replacement algorithm FST (Frequency,Object Size,Access Time) based on comprehensive factors was proposed to solve the problems such as single factor and low system's performance which are caused by traditional algorithms. This algorithm combines the features of access frequency,object size,access time interval and the longest time without access,and it takes the method of segmentation according to the length of the recent access time. By taking contrast test with LRU,LFU and SIZE in the wind power system cache server,FST algorithm shows better performance in improving the hit rate and reducing the delay time.

**Keywords** Wind power system,Comprehensive factors,Replacement algorithm,Hit rate,Delay time

风电是新能源发电的重要组成部分,随着风力发电机装机容量的上升,风电在电网中的比例逐渐增加,因此分析风电场数据对电网运行有极其重要的意义。一个风电场体系包含测风塔信息、风机信息以及风场运行信息,它们每天更新的数据多达数百万条,如何对风电系统的海量数据进行高效管理和存储成为了亟待解决的问题。在风电系统的数据库服务器中,如果每次都向原始数据库发送访问请求,不仅会增加用户访问延迟时间,还会加重服务器的负载,同时也会引起网络拥堵问题[1]。因此,通过缓存技术把用户访问过的数据存入计算机的本地存储空间,当数据再次被访问时,直接从缓存中获取数据,不必请求数据库服务器,从而有效提高了系统的整体性能。然而每个缓存系统的容量都是有限的,当需要缓存的数据量超过系统的内存空间时,就要适当替换缓存中的一部分数据,以保证新数据的加入[2]。

缓存替换算法是影响缓存性能的关键因素,好的替换算法能保证缓存具有较高的命中率。文献[3]介绍了几种传统的缓存替换策略及其各自的特点。文献[4]提出了评价缓存替换算法的几项重要指标。文献[5]针对Hybrid缓存算法提出了改进方法,但是其对保存价值的定义过于复杂,实现起来比较麻烦。针对于此,本文设计一种综合访问频率、对象大小、访问时间间隔及最久未访问特性的缓存替换算法FST。该算法公式简单,易于实现,并且充分利用了文档的已知信息。通过访问风电系统数据,比较缓存系统中LRU,LFU,SIZE和FST算法的性能指标,验证了FST算法在提高命中率、降低访问延迟时间方面的优势。

## 1 服务于风电系统的缓存特性分析

### 1.1 缓存系统的性能指标

当用户对风电系统服务器进行访问时,如果要访问的数据对象在缓存中,则称为一次命中;否则称为一次缺失。设总的请求次数为 $m$,当请求 $i(1\leqslant i\leqslant m)$ 在缓存中时,$\delta_i=1$;反之,$\delta_i=0$。下面介绍缓存性能的几种评价指标。

到稿日期:2016-08-23 返修日期:2017-01-05 本文受国家自然科学基金(61503063,51277022),四川省科技计划项目(2016GZ0143)资助。
鲁尔洁(1992-),女,硕士生,主要研究方向为电力系统分析、运行及控制,E-mail:15608178050@163.com;陈 峦(1973-),男,硕士,副教授,硕士生导师,主要研究方向为计算智能及其应用、电力系统自动化;李 坚(1984-),男,博士,副教授,硕士生导师,主要研究方向为遥操作系统稳定性控制、电力系统广域测量与控制。

(1)请求命中率

请求命中率 HR(Hit Rate)是评价缓存性能的最主要指标，指请求命中的次数与总的请求次数之比[6]。HR 可表示为：

$$HR=\frac{\sum_{1}^{m}\delta_i}{m},1\leqslant i\leqslant m \tag{1}$$

(2)字节命中率

字节命中率 BHR(Byte Hit Rate)是指请求命中的字节数与总的字节数之比[7]。由于缓存中小文档的存在可能使得文档的请求命中率较高而字节命中率较低，从而影响缓存性能，因此将字节命中率作为评价缓存系统的评价指标。用 $S_i$ 表示第 $i$ 次请求文档的大小，则 BHR 可表示为：

$$BHR=\frac{\sum_{1}^{m}\delta_i S_i}{\sum_{1}^{m}S_i},1\leqslant i\leqslant m \tag{2}$$

(3)分组命中率

分组命中率 PHR(Packet Hit Ratio)与字节命中率的意义大致相同，只是把单位由字节数换成了分组数。根据 TCP/IP 协议，按照一次访问约等于(2＋文档大小字节数/536)进行计算[8]，令 $P_r$ 表示用户总的请求分组数，$P_h$ 表示命中分组数，则 PHR 可表示为：

$$PHR=\frac{P_h}{P_r} \tag{3}$$

(4)平均延迟时间

平均延迟时间 ALT(Average Latency Time)是指从客户端请求数据到收到数据所需的平均时间[9]。平均延迟时间越短，缓存性能越好，ALT 的计算公式表示如下：

$$ALT=\frac{\sum_{1}^{m}T_i}{m},1\leqslant i\leqslant m \tag{4}$$

其中，$T_i$ 为客户端第 $i$ 次请求文档的延迟时间。

### 1.2　影响缓存替换的因素

影响缓存替换算法的因素很多[10]，结合风电系统的数据特性，下面主要从缓存空间大小和文档对象大小两个方面进行分析研究。

(1)缓存空间大小

缓存空间大小与文档命中率成正比，缓存空间越大，文档命中率越高，当缓存空间足够大时，命中率接近 1。由于受到经济成本和设备条件的限制，风电系统的缓存服务器空间不可能无限大，因此需要设置合理的缓存大小以确保缓存有较高的命中率，从而减少用户对风电数据请求的延迟时间。

(2)文档对象大小

用户请求的文档中，其大小存在差别，但总体上符合重尾(Heavy Tailed，HT)分布特征，其关系表达式如式(5)所示：

$$P[X>x]\approx x^{-\alpha},x\to+\infty,0<\alpha<2 \tag{5}$$

其中，$\alpha$ 是重尾度索引，影响分布的重尾度。该分布说明用户请求的文档集中为较小的文档，而对较大的文档的请求并不频繁。如果缓存中优先存储小文档对象，请求命中率则会提高，但是大文档对象的缺失会导致字节命中率降低；反之，如果缓存中优先存储大文档对象，则会降低请求命中率，提高字节命中率。

## 2　缓存替换算法的改进研究

当缓存空间容量达到最大值时，需要按照一定算法规则替换出一部分缓存对象，以便有足够的空间存储新对象，这种算法即缓存替换算法。如果被替换的对象是访问量较高的数据，那么会降低缓存命中率，因此，需要设计一种合理的替换算法来保证缓存具有较高的命中率。

### 2.1　现有缓存替换算法

现有的缓存替换算法都是基于文档的访问特性而设计的，主要有以下 4 种。

(1)基于访问时间间隔的替换算法

该类替换算法的典型代表是 LRU 算法，其基本思想是：将缓存中所有文档按照距离最近被访问的时间间隔大小排序，当缓存空间不够时，替换时间间隔最大的文档，一般用栈来实现该算法[11]。由于该算法没有考虑文档大小的因素，可能将小文档对象替换出去，导致请求命中率降低。

(2)基于访问次数的替换算法

LFU 是这类算法中最典型的代表，其基本思想是替换出缓存中访问次数最少的文档[12]。该算法实现简单，即为每个对象设置一个计数器，文档被访问一次则计数器加 1。但是对于短时间内访问量较大的文档，如果没有失效机制，则会造成“缓存污染”，即过时文档永远保存在缓冲器中。另外，该算法没有考虑文档大小和访问获取延迟。

(3)基于文档大小特性的替换算法

在该类替换算法中，典型代表是 William 等人设计的 SIZE 算法，其基本思想是替换出缓存中最大的文档以便容纳更多小文档对象。该算法提高了请求命中率，但减少了大文档数量，造成了字节命中率的降低，且大文档再次被载入时网络开销会很大。

(4)基于保存价值的替换算法

该类算法的思想是：比较文档的保存价值，将价值较低、对命中率贡献最小的对象替换出去[13]。其典型代表是 Hybrid 算法，文档 $d$ 的保存价值 $f_d$ 的定义如式(6)所示：

$$f_d=\frac{(C_s+\frac{C_1}{b\omega_s})}{S_d}*(C_d)^{C_2} \tag{6}$$

其中，$C_s$ 代表连接 Web 服务器的时间，$b\omega_s$ 代表 Web 服务器到代理服务器的带宽，$S_d$ 表示文档 $d$ 的大小，$C_d$ 表示文档进入缓存后被访问的次数($C_1$，$C_2$ 为常数)。该算法考虑了 Web 服务器的延迟和吞吐量等多种访问属性，但实现起来比较复杂，同时没有考虑文档的最近访问时间。

### 2.2　一种基于综合因素的缓存替换算法

已有研究[14-15]表明，综合多种因素的缓存替换算法在性能上优于单一因素的算法。结合 1.2 节所述的影响因素，本文提出了一种基于综合因素的替换算法，其设计思路如下。

(1)风电场访问频率

在风电缓存系统中，用户对文档对象的访问不服从均匀分布，据已有研究[16]显示，80％的用户倾向于访问频率占前 20％的文档，即“二八原则”。该规律服从 zipf 分布，用公式表示为：

$$F_r=\frac{C}{r^{\alpha}} \tag{7}$$

其中，$F_r$ 为某个对象出现的频率，$r$ 为该对象的访问次数排名，$C$ 和 $\alpha$ 为常数（$\alpha$ 接近 1）。因此，缓存对象的存在价值与访问频率成正比。访问频率越高的对象被替换出去的概率越小。

(2)访问对象大小

由前文的分析可知，用户倾向于访问大文档对象，即缓存对象的存在价值与其大小成反比，数据量越大的对象被替换出去的概率越大[17]。其关系表示为：

$$K_1=\frac{S}{F} \tag{8}$$

其中，$S$ 为缓存对象的大小，$F$ 为缓存对象总的访问次数，当替换发生时，移除 $K_1$ 值最大的对象。

(3)访问时间间隔

访问时间间隔是指某个文档连续两次被访问的平均时间差。访问时间间隔的大小对文档的下一次访问有一定预测作用[18]，因此在替换算法中，也应该考虑访问时间间隔的影响，其计算公式为：

$$K_2=\frac{T_L-T_B}{F} \tag{9}$$

其中，$T_L$ 为最近一次访问时间，$T_B$ 为文档建立时间，当缓存数据需要发生替换时，移除 $K_2$ 值最大的对象。

(4)最近访问时间

访问时间间隔短的缓存对象可能会成为某一段时期的“热点”对象，但是随着时间的推移，它的流行度逐渐降低，其 $K_2$ 值却一直不变。如 2014 年 4 月 10 日－20 日的测风塔数据信息是本月下旬访问量最高的对象，但到了 2016 年 4 月就几乎没有访问价值了。这样导致一些已经过时的缓存对象永久保存在缓存中，占用内容空间，降低请求命中率和字节命中率。因此，最近访问时间是影响缓存性能的一个重要因素。它服从时间局部性原理，指在 $t$ 时刻被访问的文档在 $t+\Delta t$ 时刻再次被访问的概率与 $\Delta t$ 的大小成反比。用 $K_3$ 表示最近一次的访问时间间隔，其计算公式如下：

$$K_3=T_R-T_L \tag{10}$$

其中，$T_R$ 为系统当前时间，$K_3$ 越小说明该缓存对象被再次访问的概率越大。

基于此，本文设计了一种综合访问频率、对象大小、访问时间间隔及最久未访问特性的缓存替换算法 FST。为降低算法对文档大小的依赖性，将式(9)中的 $S$ 改为 $\log(S)$。改进后的算法公式为：

$$K=\frac{\log(S)}{F}*K_2*f(K_3) \tag{11}$$

其中，$f(K_3)$ 表示评价值 $K_3$ 的函数，$f(K_3)$ 是 $K_3$ 的增函数。根据时间局部性可知，$\Delta t$ 越小，文档再次被访问的可能性越大。现在根据最近一次的访问时间间隔按照 1h 和 24h 将 $K_3$ 分为 3 部分，因此分 3 种情况考虑函数 $f(K_3)$：$0<K_3\leqslant 3600$，$3600<K_3\leqslant 86400$，$K_3>86400$（$K_3$ 的单位是秒）。即改进的方式是将 $f(K_3)$ 设置为分段函数：

$$f(K_3)=\begin{cases} f_1(K_3), & 0<K_3\leqslant 3600 \\ f_2(K_3), & 3600<K_3\leqslant 86400 \\ f_3(K_3), & K_3>86400 \end{cases} \tag{12}$$

其中，$f_1(K_3)$，$f_2(K_3)$，$f_3(K_3)$ 分别为 $K_3$ 的增函数。对于式(12)中的 $f_1(K_3)$，$f_2(K_3)$，$f_3(K_3)$，需要确定它们与 $K_3$ 的关系。如果其值太小，则 $K_3$ 的变化对函数的影响不大；如果其值太大，则会掩盖前几项因素的影响。

当 $0<K_3\leqslant 3600$ 时，该部分数据在近期被访问的概率较大，此改进算法把该数据当作在近期内还会被访问的对象，设置其造成的影响值为 0.5，即 $f_1(K_3)=0.5$。$3600<K_3\leqslant 86400$ 代表最后一次访问时间是 24h 之内，$K_3>86400$ 代表最后一次访问时间是 24h 之前。为减小 $K_3$ 在这两个区间的取值造成的差值对缓存系统性能的影响，对其取对数，令 $f_2(K_3)=\log(K_3)$，$f_3(K_3)=2\log(K_3)$，则式(11)变为：

$$K=\frac{\log(S)}{F}*\frac{T_L-T_B}{F}*\begin{cases} 0.5, & 0<K_3\leqslant 3600 \\ \log(K_3), & 3600<K_3\leqslant 86400 \\ 2\log(K_3), & K_3>86400 \end{cases} \tag{13}$$

当发生替换时，移除缓存中 $K$ 值最大的对象。如图 1 所示，该算法的处理过程如下：

(1)当客户端有数据请求时，首先查看缓存中是否保存了该对象，如果保存有，则更新对象的访问时间并将访问次数加 1，返回对象结束，否则执行步骤(2)；

(2)向数据库服务器请求该数据对象，如果缓存空间未满，将数据加入缓存中，将对象的文档建立时间设置为当前时间，访问次数初始化为 1，返回对象结束，否则执行步骤(3)；

(3)采用 FST 算法机制处理缓存中的对象，将所有缓存对象按照 $K$ 值大小排序；

(4)移除排序后位于首位的对象，把新对象加入缓存，将对象的文档建立时间设置为当前时间，访问次数初始化为 1，返回对象结束。

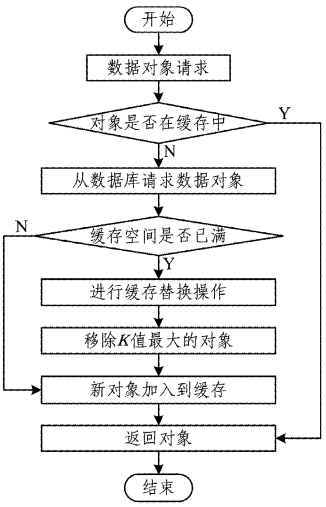


图 1 FST 处理机制

## 3 缓存替换算法的性能测试

以风电数据为测试对象，设定缓存容量分别为 5MB，10MB，15MB，20MB，25MB，30MB，每次输入 2000 个查询请求，分别调用 LRU，LFU，SIZE 和 FST 算法，得到它们的请求命中率、字节命中率、分组命中率及访问延迟时间曲线图，如图 2－图 5 所示。

由图 2 可知，当缓存容量较小时，SIZE 的请求命中率最高，这是因为以文档大小作为替换依据可以腾出较大的空间给更多小文档。当缓存容量增大时，SIZE 的优势逐渐减弱，这是因为载入大文档对象会导致很大的网络开销。LFU 和 LRU 则平稳增长，但其请求命中率低于 FST 和 SIZE。

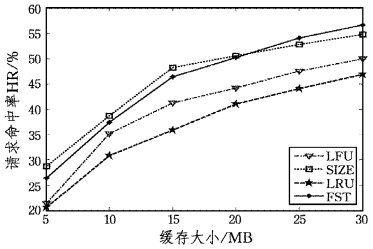


图 2　请求命中率变化图

由图 3 可知，4 种替换算法的字节命中率相差不大，FST 的字节命中率最高。大文档对象的缺失导致 SIZE 算法的字节命中率降低，LFU 的字节命中率与 SIZE 相近，LRU 的字节命中率最低。

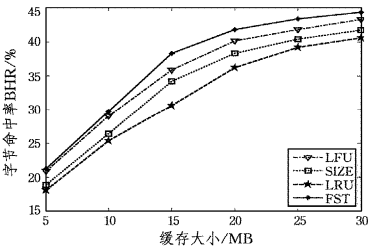


图 3　字节命中率变化图

由图 4 可知，分组命中率和字节命中率的变化情况相似，与图 3 类似，4 种算法的差别不大，改进的新算法 FST 的分组命中率最高。

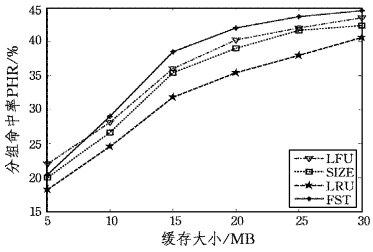


图 4　分组命中率变化图

图 5 中 LFU 和 LRU 的延迟时间较长，SIZE 和 FST 的时间相差不大。当缓存容量较小时，由于 SIZE 的请求命中率最高，因此其延迟时间最短；随着缓存容量的增大，FST 的优势逐渐明显。

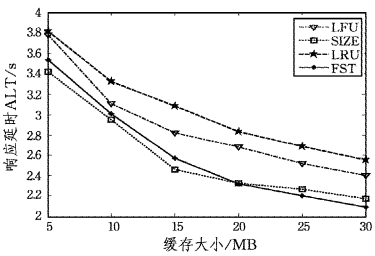


图 5　平均延迟时间变化图

各种指标的测试结果如表 1 所列。

表 1　缓存替换算法指标测试结果

| 缓存大小/MB | | 5 | 10 | 15 | 20 | 25 | 30 |
|---|---|---|---|---|---|---|---|
| 请求命中率/% | LFU | 21.3 | 35.2 | 41.2 | 44.2 | 47.6 | 50.0 |
| | SIZE | 28.7 | 38.6 | 48.2 | 50.6 | 52.8 | 54.8 |
| | LRU | 20.5 | 30.8 | 35.8 | 41.0 | 44.0 | 46.8 |
| | FST | 26.4 | 37.4 | 46.4 | 50.2 | 54.1 | 56.7 |
| 字节命中率/% | LFU | 20.8 | 29.0 | 35.8 | 40.1 | 41.8 | 43.3 |
| | SIZE | 18.8 | 26.4 | 34.2 | 38.3 | 40.4 | 41.7 |
| | LRU | 18.0 | 25.3 | 30.6 | 36.2 | 39.2 | 40.6 |
| | FST | 21.2 | 29.7 | 38.3 | 41.8 | 43.4 | 44.3 |
| 分组命中率/% | LFU | 22.0 | 28.1 | 36.0 | 40.2 | 42.0 | 43.5 |
| | SIZE | 20.0 | 26.6 | 35.4 | 39.0 | 41.6 | 42.3 |
| | LRU | 18.2 | 24.5 | 31.8 | 35.4 | 37.9 | 40.6 |
| | FST | 20.4 | 29.0 | 38.5 | 42.0 | 43.6 | 44.5 |
| 平均延迟时间/s | LFU | 3.77 | 3.11 | 2.82 | 2.67 | 2.51 | 2.40 |
| | SIZE | 3.42 | 2.94 | 2.45 | 2.32 | 2.26 | 2.17 |
| | LRU | 3.81 | 3.32 | 3.08 | 2.83 | 2.68 | 2.55 |
| | FST | 3.53 | 3.00 | 2.57 | 2.32 | 2.20 | 2.09 |

对一种缓存替换算法的评价需要考虑多方面的指标，好的替换算法不仅应该具有较高的请求命中率，而且应该有较高的字节命中率、分组命中率以及较短的延迟时间。由以上图表显示的结果可以看出，新算法的整体性能优于其他几种算法。

分析算法的设计原理可知，LFU，LRU 和 SIZE 算法只考虑了单一的影响因素，没有充分利用文档的已知信息，导致其缓存性能不能达到较好的效果。FST 是一种基于多种因素的替换算法，由于考虑到文档大小的因素，其请求命中率得到提高。通过引入最近访问时间这一因素，避免了过时“热点”文档占用缓存空间所造成的资源浪费现象。此外，访问时间间隔及访问次数可以保证缓存有较高的字节命中率和分组命中率，从而缩短了访问延迟时间。

**结束语**　为加快用户对风电系统的访问速度，本文提出了缓存技术的概念及重要性，并针对风电系统分析了缓存对象的访问特性以及影响缓存替换的因素。在现有的替换算法的基础上，设计了一种综合访问频率、对象大小、访问时间间隔及最久未访问特性的缓存替换算法 FST。将该算法应用于风电系统的缓存服务器中，以风电系统数据为测试对象，并通过与 LRU，LFU 和 SIZE 算法进行实验对比，验证了 FST 算法在提高命中率和减少延迟时间方面具有更好的性能。该算法省略了网络开销和预取机制的分析，这些方面将是今后重点研究的方向。

## 参 考 文 献

[1] HAN X C，TIAN Y G. Web cache replacement algorithm based on prediction[J]. Computer Engineering and Design，2010，31(1)：110-113. (in Chinese)
韩向春，田玉根. 基于预测的 Web 缓存替换算法[J]. 计算机工程与设计，2010，31(1)：110-113.

[2] SHEU J P，CHUO Y C. Wildcard rules caching and cache replacement algorithms in software-defined networking[J]. IEEE Transactions on Network and Service Management，2016，13(1)：19-29.

[3] KHARBUTLI M，SHEIKH R. LACS：A locality-aware cost-sensitive cache replacement algorithm[J]. IEEE Transactions on Computers，2014，63(8)：1975-1987.

(下转第 238 页)

using rough set theory[J]. International Journal of Approximate Reasoning,2011,52(6):881-893.

[4] MARTINEZ I G,PEREZ R E B. Making decision in case-based systems using probabilities and rough sets[J]. Knowledge-Based Systems,2003,16(4):205-213.

[5] YANG X B,YAN X,XU S P,et al. New heuristic attribute reduction algorithm based on sample selection[J]. Computer Science,2016,43(1):40-43. (in Chinese)
杨习贝,颜旭,徐苏平,等. 基于样本选择的启发式属性约简方法研究[J]. 计算机科学,2016,43(1):40-43.

[6] YANG X B,QI Y S,SONG X N,et al. Test cost sensitive multigranulation rough set:Model and minimal cost selection [J]. Information Sciences,2013,250(11):184-199.

[7] HU Q H,PEDRYCZ W,YU D R,et al. Selecting discrete and continuous features based on neighborhood decision error minimization [J]. IEEE Transactions on Systems Man & Cybernetics-Part B,2010,40(1):137-150.

[8] HU Q H,YU D R,XIE Z X. Neighborhood classifiers [J]. Expert Systems with Applications,2008,34(2):866-876.

[9] HU Q H,YU D R,XIE Z X. Numerical attribute reduction based on neighborhood granulation and rough approximation [J]. Journal of Software,2008,19(3):640-649. (in Chinese)
胡清华,于达仁,谢宗霞. 基于邻域粒化和粗糙逼近的数值属性约简[J]. 软件学报,2008,19(3):640-649.

[10] WRIGHT J,YANGA Y,GANESH A,et al. Robust face recognition via adaptive sparse representation [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence,2009,31(2):210-227.

[11] ZHANG L,YANG M,FENG X C. Sparse representation or collaborative representation:Which helps face recognition? [C]// Proceedings of the 2011 International Conference on Computer Vision. IEEE Computer Society,2011:471-478.

[12] WILSON D R,MARTINEZ T R. Improved heterogeneous distance functions [J]. Journal of Artificial Intelligence Research,1997,6(5):1-34.

[13] XU S P,YANG X B,SONG X N,et al. Prediction of protein structural classes by decreasing nearest neighbor error rate [C]// Proceedings of the 2015 International Conference on Machine Learning and Cybernetics. IEEE Computer Society,2015:7-13.

[14] XU S P,YANG X B,YU H L,et al. Multi-label learning with label-specific feature reduction[J]. Knowledge-Based Systems,2016,104:52-61.

(上接第 233 页)

[4] LIU L,XIONG X P. Least cache value replacement algorithm [J]. Journal of Computer Applications,2013,33(4):1018-1022. (in Chinese)
刘磊,熊小鹏. 最小驻留价值缓存替换算法[J]. 计算机应用,2013,33(4):1018-1022.

[5] GAO M,WANG N H,LI D,et al. Data pre-fetching and caching algorithm based on templates[J]. Application Research of Computers,2014,31(11):3240-3242,3246. (in Chinese)
高萌,王霓虹,李丹,等. 一种基于模板的数据预取和缓存算法[J]. 计算机应用研究,2014,31(11):3240-3242,3246.

[6] HEFEEDA M,NOORIZADEH B. On the benefits of cooperative proxy caching for peer-to-peer traffic[J]. IEEE Transactions on Parallel and Distributed Systems,2010,21(7):998-1010.

[7] LIANG W,BAYHAN S,KANGASHARJU J. Effects of cooperation policy and network topology on performance of in-network caching[J]. IEEE Communications Letters,2014,18(4):680-683.

[8] WU J L,YANG Q. A Web cache replacement algorithm based on collaborative filtering[J]. Computer Engineering & Science,2015,37(11):2128-2133. (in Chinese)
吴俊龙,杨清. 基于协同过滤的 Web 缓存替换算法研究[J]. 计算机工程与科学,2015,37(11):2128-2133.

[9] HU Y Q,LI X N. A new proxy cache replacement mechanism of multimedia streams based on recommendation [J]. Journal of Yanshan University,2015,39(2):139-144,151. (in Chinese)
胡玉琦,李晓娜. 一种新的基于推荐的流媒体代理缓存替换机制[J]. 燕山大学学报,2015,39(2):139-144,151.

[10] CAO M,LIU W Z. Research on cache replacement model based on multi-request mode under Hybrid architecture model [J]. Computer Science,2015,42(6):175-180. (in Chinese)
曹旻,刘文中. 混合架构下多请求模式的缓存替换算法模型研究[J]. 计算机科学,2015,42(6):175-180.

[11] LIN M W,YAO Z Q,XIONG J B. History-aware page replacement algorithm for NAND flash-based consumer electronics[J]. IEEE Transactions on Consumer Electronics,2016,62(1):23-29.

[12] WAN S G,HE X B,HUANG J Z,et al . An efficient penalty-aware cache to improve the performance of parity-based disk arrays under faulty conditions[J]. IEEE Transactions on Parallel and Distributed Systems,2013,24(8):1500-1513.

[13] HSIEH J W,KUAN Y H. DCCS:Double circular caching scheme for DRAM/PRAM Hybrid cache[J]. IEEE Transactions on Computers,2015,64(11):3115-3127.

[14] XIE R L. Research and implementation of the cache system In distributed search engine [D]. Xi'an: Northwest University,2009. (in Chinese)
谢瑞莲. 分布式搜索引擎中缓存系统的研究与实现[D]. 西安:西北大学,2009.

[15] WANG W J. Research on web cache and prefetching model based on access path mining[D]. Chengdu: Southwest Jiaotong University,2011. (in Chinese)
王文建. 基于访问路径挖掘的 Web 缓存与预取模型研究[D]. 成都:西南交通大学,2011.

[16] YANG L,DONG H Q,LIU G L. Current progress of caching techniques in storage [J]. Microcomputer Applications,2015,4(5):1-9. (in Chinese)
杨琳,董欢庆,刘国良. 存储领域缓存技术的现状[J]. 网络新媒体技术,2015,4(5):1-9.

[17] CHEN N J,LIN P. An user access feature driven semantic cache replacement policy in middleware[J]. Journal of Guangxi University(Nat Sci Ed),2010,35(5):787-792. (in Chinese)
陈宁江,林盘. 用户访问特征驱动的中间件语义缓存替换策略[J]. 广西大学学报(自然科学版),2010,35(5):787-792.

[18] MA H Y,WANG B. Query results caching and prefetching in web search engines based on user characteristics[J]. Journal of Chinese Information Processing,2012,26(6):19-26. (in Chinese)
马宏远,王斌. 基于用户特性的搜索引擎查询结果缓存与预取[J]. 中文信息学报,2012,26(6):19-26.