一种高效的闪存存储系统偏好访问模式识别技术研究

雷 娟[1] 朱 珠[1] 符云清[2] 石 亮[2]
(国网重庆市电力公司电力科学研究院 重庆 401123)[1] (重庆大学计算机学院 重庆 400044)[2]

摘 要 闪存已经是目前使用最为广泛的存储设备。众所周知,闪存对访问模式具有极大的敏感性,比如随机与顺序访问模式、冷热访问模式以及写聚集和分段顺序写模式等。此外,闪存设备的很多部件也对这些访问模式具有较大的敏感性。因此,闪存偏好访问模式的识别技术对闪存存储系统的性能和设计具有重要意义。首先给出闪存存储的偏好模式定义,然后提出了闪存存储系统偏好模式的识别技术。实验表明,所提出的偏好模式识别技术具有很高的准确性。

关键词 闪存存储系统,访问模式,识别技术
中图法分类号 TP333 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.2.018

Research of Efficient Recognition Technique for Preferred Access Pattern of Flash Storage System

LEI Juan[1] ZHU Zhu[1] FU Yun-qing[2] SHI Liang[2]
(Electric Power Research Institute of Chongqing Electric Power Company, National Power Grid, Chongqing 401123, China)[1]
(College of Computer Science, Chongqing University, Chongqing 400044, China)[2]

Abstract Flash memory is the most widely used storage device. Flash memory, as is well-known, has great sensitivity to its access pattern, such as random and sequential access patterns, hot and cold access mode, focused writing and partitioned sequential writing, etc. In addition, many parts of flash device also have higher sensitivity to access patterns. Therefore, the recognition technology of preferred access pattern has great influence on the performance and design of flash storage system. The definition of preferred access pattern of flash storage system was first introduced in this paper, and the re-cognition technology of preferred pattern was followed in detail. The experimental results show that the proposed recognition technology has very high accuracy.

Keywords Flash storage system, Access pattern, Recognition technique

1 引言

闪存存储是目前使用最为广泛的二级存储设备,其优越性源自于闪存具有较好的随机访问性能、防震能力以及轻便尺寸小等特征。与常规的存储设备(如磁盘)一样,闪存也存在工作负载访问模式的敏感性。文献[2,4]指出了闪存的访问模式对性能的影响非常大。不同的访问模式有着截然不同的性能特征,如读操作性能相比于写性能要好很多,顺序写操作较随机写操作也要好很多。在闪存存储设备的设计中,各种存储部件(如闪存转换层[10,14,16,17]、缓存管理[23]、均衡磨损算法[6]以及垃圾回收机制[23]等)的设计与访存模式也密切相关,因而在设计过程中亦需充分考虑访问模式,如闪存转换层一般要考虑如何缓解随机写操作导致的写性能下降,均衡磨损算法和垃圾回收算法则要考虑数据冷热等。本文提出了高效识别闪存系统的偏好访问模式算法,为闪存存储系统的性能改善提供基础和依据。

不同于磁盘的模式敏感特征,闪存的敏感模式则是另一组访问模式,如读的访问模式在闪存上差异很小,而写模式则有较大差异。闪存对访问模式具有极大的依赖性。通过闪存偏好模式的识别能对闪存性能特别是系统设计产生重要作用。我们将闪存存储系统中具有较好性能的访问模式称为闪存偏好模式,当前极少有研究考虑闪存偏好模式的识别。

本文将提出一系列闪存偏好模式识别技术。不同于以往技术,所提技术将从真实的系统负载中识别闪存偏好模式。具体来说,我们提出了一个基于地址域的模式识别方法和一个局部性程度的模式识别方法用于识别偏好模式。通过模拟实现和分析,表明我们所提出的识别模式具有较好精确性和高效性。

2 背景及相关工作

2.1 访问模式识别

工作负载访问模式已经被研究了很多年[8,20]。访问模式由于对性能的影响非常大,因此对系统部件的设计具有重要意义。目前在闪存的缓存管理中,访问模式识别技术被广泛

到稿日期:2014-03-09 返修日期:2014-06-23 本文受重庆市自然基金项目(CSTC,2010BB2248),中央高校基金项目(106112014,CDJZR185502),国网重庆电力科学研究院基于网络大容量存储在线监测项目资助。

雷 娟(1982—),女,硕士生,主要研究方向为计算机体系结构、网络与信息安全等;朱 珠(1976—),男,主要研究方向为信息安全、企业信息管理等;符云清(1969—),男,博士,教授,主要研究方向为计算机体系结构、计算机网络与通信、现代远程教育、软件工程等,E-mail:yqfu@cqu.edu.cn;石 亮(1987—),男,博士,讲师,主要研究方向为嵌入式与实时系统、计算机系统结构与分布式计算、新型非易失性存储器管理与优化等。

使用[7,9,15,19,25]。此外，访问模式识别技术对系统性能的预测也具有重要意义。

通常来说，访问模式可分为顺序访问模式和随机访问模式。因为不同的访问模式会导致不同的系统性能，因此目前有许多研究致力于识别不同的访问模式。DEAR[7]是第一个关于识别访问模式的工作，并应用于高速缓存。DEAR 考虑了顺序、循环、时间等可能的应用访问模式，通过对复用距离、频率和前进距离等的衡量来识别并预测访问模式；UBM[15]利用在文件层访问的数据预测顺序或循环访问模式；PCC[9]和 AMP[19]则利用计数器来控制评测频度；RACE[25]是最新的研究工作，是 UBM 和 PCC 方法的结合，可识别高速缓存的访问模式。所有这些识别访问模式的工作都应用在管理高速缓存空间上，而最少最近使用(LRU)的替换策略在管理顺序和循环访问模式并不高效。本文的工作不同于现有的这些工作，主要是识别闪存偏好访问模式。对于高速缓存，主要通过分析访问频率和局部性来识别偏好访问模式；而对于闪存设备，则通过地址空间的挖掘来识别闪存偏好访问模式。

2.2 闪存设备的模式

众所周知，闪存设备对负载访问模式非常敏感。研究表明，从低端到高端闪存设备，采用不同访问模式其性能各不相同。文献[12]分析了闪存顺序和随机访问模式的性能，并提出了衡量系统性能的一些高效标准；文献[4]对不同访问模式的性能开展了详细研究，尤其针对顺序写、局部写和分段顺序写等写操作。同时，还提出了称为 uFLIP 的系列标准，用以评估从低端到高端各种类型的闪存设备在不同访问模式下的闪存性能。

当前，大部分现有的模式识别策略[11,22]都着重于识别闪存的热数据。本文受到 uFLIP 研究的启发，提出了一系列访问模式识别策略，用来高效识别闪存偏好访问模式，包括顺序写、聚集写和分段顺序写。

3 闪存的高效模式识别

3.1 模式定义

不同的访问模式会对闪存造成不同的性能影响，表 1 给出了从高端到低端闪存设备在不同访问模式下的性能。从表中可见，在所有类型写模式中，顺序写模式拥有最好的性能，而另外两种闪存偏好写模式——分段写和聚集写具有和顺序写相似的性能。Nath 等[21]发现一种半随机写也有与顺序写相似的性能。事实上，在 uFLIP 中，半随机写就是分段顺序写。相比顺序写，随机写的性能差了 10 到 89.6 倍。

表 1 从高端到低端闪存设备在不同写模式下的响应时间
(单位：ms，页大小：32kB；“—”表示没有数据)

规格型号	顺序写模式	分段顺序写模式	随机写模式	聚集写模式
Memoright/MR25.2-032S	0.3	0.3	5	0.3
Mtron/SATA7035-016	0.4	0.6	9	0.8
Samsung/MCBQE32GMPP	0.6	1.2	18	0.9
Transcend Module/TS32GSSD25S-M	1.7	3.4	18	3.4
Transcend MLC/TS4GDOM40V-S	2.6	5.2	233	2.6
Kingston DTI/DTI 4GB	2.9	14.5	256	—

由于读模式相比写模式对闪存性能影响很小，本文重点研究闪存的 3 种写模式：顺序写、分段顺序写和聚集写。假设一段连续块的地址为：$\{d_1, d_2, d_3, \cdots, d_i, \cdots, d_n\}$，其中 d_i 代表第 i 个访问地址。这 3 种类型的写模式定义如下：

1)顺序写模式

其特点是所有的块只能被一个接一个地访问，并且不能被重复访问。在这个模式中，两个连续写的地址差值为 1。因此，如果两个连续写地址满足 $d_{i+1} - d_i = 1$，则该写模式为顺序写。例如，写序列{1,2,3,4,5,6,7,8,…}为顺序写。

2)分段顺序写模式

分段顺序写的特性为，所有块将被一个接一个地访问，并且同一地址域内的地址不能被重复访问。地址域为一系列连续的地址。在这个模式中，同一地址域的两个连续写的地址满足 $d_{i+1} - d_i = 1$。因此，若每段区域内都是顺序写，那么这个写模式即为分段顺序写。例如，写序列为{1,1001,2,1002,3,1003,4,5,1004,6,7,1005,8,1006,…}就是两个分段顺序写，分别为{1,2,3,4,5,6,7,8,…}和{1001,1002,1003,1004,1005,1006,…}。

3)聚集写模式

聚集写模式的特点为，所有的块都只能在固定的地址空间中被访问，比如 4M、8M 或 16M 的聚集大小。对于这个模式，任意两个写的地址满足 $\forall i, \forall j \leqslant n, |d_j - d_i| \leqslant$聚集大小。例如，{1,2,3,2,3,1,2,3,1,2,3,1,…}就是一个分段大小为 3 的聚集写。

3.2 闪存偏好访问模式的识别

基于上面对 3 种类型闪存偏好写模式的定义，我们在本节提出模式识别规则。这里定义的顺序模式不同于文献[7]，文献[7]根据频率定义顺序模式，每个块只能被访问一次。而在这里，我们为顺序模式诠释了一个新的定义。

1)模式识别规则和分析

顺序写模式规则：块访问序列满足 $d_{i+1} - d_i = 1$。这里会涉及到阈值 T_s。如果有 T_s 个地址满足这个公式，则这个访问序列就是顺序写模式。为了识别顺序写，识别策略只需要始终维护两个连续的地址$\langle d_i, d_{i+1}\rangle$和阈值 T_s。

分段顺序写模式规则：可访问块地址被划分成几个地址域。每个区域内的访问地址都满足顺序写模式识别规则。顺序写模式可看作一种特殊的分段顺序写模式，其地址域数目为 1。在这种情况下，识别策略需要维持一系列的连续地址$\langle d_i, d_{i+1}\rangle$。为了确保这些区域不会相互覆盖，区域的开始地址也应该被记录下来。

聚集写模式规则：访问块满足 $\forall i, \forall j \leqslant n, |d_j - d_i| \leqslant$聚集大小。这个识别策略需要维持最大的地址 d_{max}和最小的地址 d_{min}。但对于实际的工作负载，块地址始终远大于聚集大小。因为聚集写是在一段时间内发生在固定地址空间的写操作，故应提出一种策略来量化访问序列的局部性。若在一段时间内的局部性程度很高，那么这段时间内这个序列为聚集写模式。

模式检验规则可以用来检测这 3 种闪存偏好访问模式。然而，这在实际的工作负载中是一个挑战。大多数情况下，实际工作负载有很多复杂的模式。图 1 展示了 Financial 负载和 PC 负载这两个实际工作负载的情况，其中 x 轴表示时间，y 轴表示逻辑块号。从图中可看出 Financial 负载的访问在固定大小的区间内频繁波动，且这个地址空间很小。因应用仅仅访问了分段地址域中的有限空间，这个空间很小或序列是顺序访问，所以这种工作负载的写模式为分段顺序写和聚集写。而在 PC 负载中，块访问在一段很长的时间内一直处于一个固定的区域，并且地址空间很大，所以在其工作负载中，

写模式很可能为顺序写和聚集写。

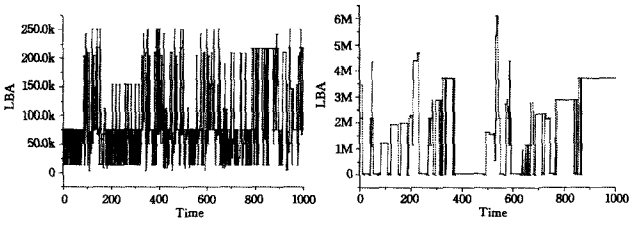

图 1 Financial 负载和 PC 负载的逻辑块地址和访问时间的对应关系

2)高效的闪存偏好访问模式识别

本节将使用上述 3 种规则来识别闪存偏好访问模式。首先,使用基于地址域方案来识别顺序和分段顺序写模式。然后根据局部性程度方案来识别聚集写模式。

图 2 展示了基于地址域模型的组织结构,所有块访问地址被划分成多个区域。每个区域中,访问地址被有序计算,并且每个区域属于不同的地址空间。

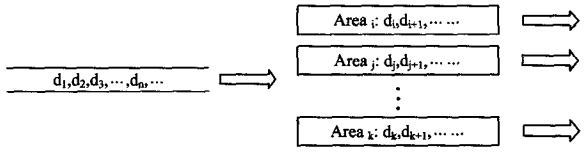

图 2 地址域方案的组织结构

基于地址域的处理流程详见算法 1。

Algorithm 1: Process of the address area-based approach

```
Input:
  d_r- The address of the current block access
  Area_r- The area of the current block access
1  Index d_r in the area list;
2  if d_r belongs to Area_r then
3    if (d_r-Area_r.d_last) !=1 then
4      Area_r.count is cleared;
5      return random ;
6    else if (Area_r.count++) >=T_f Then
7      return sequence ;
8      Area_r.d_last= d_r;
9  else
10     Create a new Area_r;
11     Area_r.d_start= Area_r.d_last=d_r;
12     Area_r.count =1;
13   return random ;
14   if Area_r.d_last>Area_r+1.d_start Then
15     Area.d_start =Area_r.d_start ;
16     Area.d_count =Area_r+1.d_count ;
17     Area.d_last =Area_r+1.d_last;
18     Delete Area_r and Area_r+1;
19   Insert Area_r to area list in ascending address order ;
```

算法中的 Area 结构包括 3 个参数:d_{start},d_{last} 和 $count$。d_{start} 是区域的开始地址。d_{last} 用来维持相应区域最后一个访问地址(第 8 行)。$count$ 用于记录地址的数目,满足公式 $d_{i+1}-d_i=1$(第 6 行);如果不满足这个公式,那么 $count$ 将被清除并且处理过程将返回一个随机模式(第 4 行)。在块访问期间,我们会记录区域中每个块的访问情况,并根据结构判断是否是顺序写模式(第 2—7 行)。特殊地,如果当前的访问地址不属于任何一个区域,我们将建立一个新的区域,而这个新区域的写模式为随机模式(第 10—13 行)。

这个方法中的所有区域都是根据 d_{last} 有序排列的,而且为了有效检索都被组织成了 B+树的结构。随着区域大小的增加,前面一个区域的 d_{last} 可能会与下一个区域重叠。在这种情况下,将启动区域合并方法。区域合并方法将区域合并成一个区域并且重新构造 Area 结构(第 14—18 行)。

以上为基于地址域方案的基本实现。此外,仅仅有限个区域将维持在程序中。区域如果在很长一段时间内未被访问,将被删除。LRU 列表用来维护这些区域。值得注意的是,基于地址域方案的存储开销非常小。对于一个最多有 1024 个区域的设计,只需花费 10 比特来记录 $count$,32 比特记录 d_{start},32 比特记录 d_{last}。总共存储的开销为 9.25kB。

聚集写模式的识别在文献[4]中作为反映写序列的局部性之用,这里反过来以写序列的局部性作为衡量聚集程度的标准。对于具有高度局部性的序列,写比较聚集。我们定义局部性程度(DOL)作为衡量在一段时间内写序列的聚集度。高度局部性意味着在这段时间内只有少数块被访问。

定义 1 写序列的 DOL 定义如下:

$$DOL=\frac{N}{time}$$

其中,N 表示从开始到现在访问过的不同块的数目,$time$ 表示到现在写序列处理的时间。

因为闪存的写操作不能就地更新,我们不需要为 DOL 计算地址空间。这也是我们仅需要考虑这段时间内不同地址数目计算写模式局部性的原因。我们发现小的 DOL 表示更多的聚集写,这里 N 的值要小于聚集大小。基于对写序列局部性程度的简单定义,我们可以用来识别闪存的聚集写模式。如果 DOL 小于阈值 T_f,那么这个写序列在这段时间内为聚集写模式;否则,将识别为基本的随机模式。

为了实现对聚集写模式的识别,需要维持两个参数:N 和 $time$。我们用一个哈希表来记录在这段时间内访问过的块,一个哈希结点记录一个块,那么这个哈希表中的结点数就为 N。对于 $time$,我们使用一个计数器来记录访问哈希表的次数。在识别处理过程中,计算 DOL 并与 T_f 进行比较。对于 T_f,假设在一段时间 T 内以 F 个扇区作为聚集大小,那么 $T_f \leqslant F/T$,这里 $T \geqslant F$。实现 DOL 的存储开销就是维持哈希表。哈希表最多有 $time$ 个结点,而每个结点仅需要记录块号。

4 实验和分析

4.1 实验配置

本节主要评估了本方案在识别 3 种闪存偏好访问模式的效果。基于请求大小的方案使用 16kB 作为随机和顺序模式的阈值[23]。本方案中只需维持 1024 个地址域。除此之外,我们还考虑了地址域个数不作限制的方案。对于聚集写模式,采用文献[4]中发现的闪存偏好模式,使用 DOL 来评估负载的局部性程度,以识别是否为聚集写模式。

本方案中使用的工作负载包括两个 TPC-C 基准测试程序[24]和两个 PC 负载[12],PC 负载收集的是一台笔记本在两个不同时间运行不同应用产生的记录。TPC-C 基准测试程序已经被广泛使用,用于模拟企业级磁盘中应用程序的行为,比如数据库程序,其中请求主要包括请求较小数据的随机写请求。PC 负载记录包括收发邮件、上网、编辑文档、播放音频等。

4.2 实验结果

首先将顺序和分段顺序模式识别方案与基于请求大小的方案进行对比,然后评估聚集写模式识别方案的效果,最后对参数的选择进行分析。

1)顺序和分段顺序模式:对于基于请求大小方案(Size)、有限个区域数目方案(Area)以及地址域不作限制方案(Un-

limited) 3 种方案，图 3 展示了每种方案下写访问被识别为顺序和分段顺序写模式的数目。所有数据均以地址域不作限制方案为标准进行规范化。图 3 中，T_s 设为和基于请求大小方案一样大的 16kB。如果基于请求大小方案识别请求为顺序模式，那么本方案也识别为顺序模式。从图可看出，Area 方案比 Size 方案准确度更高。此外，还可看到 Financial 负载相比 PC 负载有更多请求被识别为顺序模式。究其原因，是 Financial 负载中的平均请求大小远小于 16kB，而 PC 负载中的请求大小通常都大于 16kB。

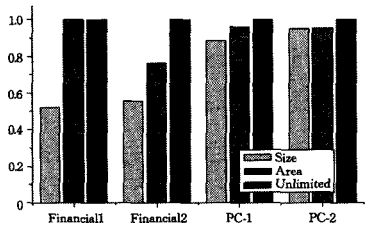

图 3　被识别为顺序和分段顺序模式的写请求数目

2)聚集写模式：图 4 示出不同聚集大小下被识别为聚集写模式的时间段的个数，这里选了 4M、8M 和 16M 3 个聚集大小，其中 T_f 设为 0.5。如图 4 所示，Financial 负载比 PC 负载有更多的情况被判定为聚集写模式，这是因为与 PC 负载相比，Financial 负载有更高程度的局部性。此外，可以看到对于不同的聚集大小，聚集写区域的数目随着应用不同而不同。对于 Financial1 来说，随着聚集大小的增大，聚集区域的个数也在增加，Financial2 却与之相反。出现这种现象的原因与两者的区域局部性有关，Financial1 的区域局部性比较小，并且每个区域的局部性各不相同，而 Financial2 与其不同。

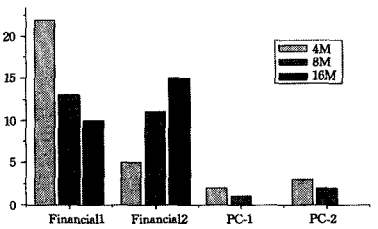

图 4　识别为聚集写模式的时间段个数

3)参数敏感性研究：在这部分，我们主要分析基于地址域方案的区域个数和 DOL 方案中的 T_f 参数。

图 5 示出基于地址域方案中不同区域数目情况下，识别为顺序和分段顺序模式的个数。如图 5 所示，区域数目越多，识别为顺序和分段顺序模式的数目就越多。对于 Financial 负载，随着区域数目的增加，识别准确度也相应增加。而对于 PC 负载，区域数目即使增加很多，对识别准确度的影响也很小，区域数目从 128 到 1024 的识别准确度几乎不变。

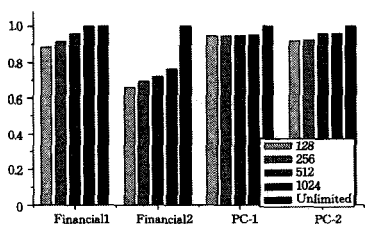

图 5　识别为顺序和分段顺序模式的写访问的个数

图 6 示出识别为聚集写的时间段的个数。为了分析 T_f 的影响，聚集大小固定为 4M，我们从 0.1 到 0.9 改变 T_f 的值。随着 T_f 的增大，更多的时间段被识别为聚集写模式。此外，还可以看到 Financial1 的局部性程度比 Financial2 要好，所以 Financial1 有更多时间段为聚集写模式。

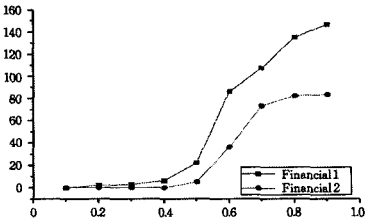

图 6　识别为聚集写模式的时间段个数

结束语　本文提出了一系列高效识别闪存偏好访问模式的方案。这些方案可识别的模式包括顺序、分段顺序以及聚集写模式。不同于现有识别基于高速缓存顺序模式的工作，我们提出了一个基于地址域的方案，用来识别闪存的顺序模式。此外，DOL 方案用来识别另一种闪存中的偏好模式——聚集写模式。在实际工作负载上的模拟分析表明，其有很高的识别准确度。

参 考 文 献

[1] Agrawal N, Prabhakaran V, Wobber T, et al. Design tradeoffs for ssd performance[C]// USENIX Annual Technical Conference on Annual Technical Conference. 2008

[2] Birrell A, Isard M, Thacker C, et al. A design for highperformance flash disks[J]. SIGOPS Oper. Syst. Rev., 2007, 41(2): 88-93

[3] Boboila S, Desnoyers P. Write endurance in flash drives: measurements and analysis[C]// Proceedings of the 8th USENIX Conference on File and Storage Technologies(FAST'10). 2010

[4] Bouganim L, Jnsson B T, Bonnet P. uflip: Understanding flash io patterns[C]//CIDR'09: Fourth Biennial Conference on Innovative Data Systems Research. 2009

[5] Chang L-P. A hybrid approach to nand-flash-based solid-state disks[J]. IEEE Transactions on Computers, 2010, 59(10): 1337-1349

[6] Chang L-P, Du C-D. Design and implementation of an efficient wear-leveling algorithm for solid-state-disk microcontrollers[J]. ACM Trans. Des. Autom. Electron. Syst., 2009, 15(1)

[7] Choi J, Noh S H, Min S L, et al. An implementation study of a detection-based adaptive block replacement scheme[C]// Proceedings of the Annual Conference on USENIX Annual Technical Conference. 1999

[8] Dan A, Yu P S, et al. Characterization of database access pattern for analytic prediction of buffer hit probability[J]. VLDB Journal, 1995, 4(1): 127-154

[9] Gniady C, Butt A R, Hu Y C. Program-counter-based pattern classification in buffer caching[C]// Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation. 2004

[10] Gupta A, Kim Y, Urgaonkar B. Dftl: a flash translation layer employing demand-based selective caching of page-level address mappings[C]// Proceeding of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems. 2009

(下转第 122 页)

图 3 的横轴记录爬取时间，纵轴记录爬取主题相关网页的数量。由实验数据可知，在相同时间内，SVM-topical 算法抓取的主题相关目标网页数最高，其次为 HITS 算法，而 Breadth-First 算法最低。在时间较短时，3 种爬取策略爬取的主题相关目标网页数量相差不大。由于 SVM-topical 算法有利于主题相关网页的挖掘，因此，随着所爬取时间的增加，本文的 SVM-topical 算法爬取的网页数量逐渐高于其他两种算法。

结束语 本文提出了一种基于 SVM 分类算法的主题爬虫技术，通过两次爬取获取与主题相关的网页。其中，在第一次爬取时，采用种子选取的方式，将与主题相关的关键词加入到爬虫中，之后在获取的页面列表中提取与主题相关的一类集合作为种子。第二次爬取时，采用 HITS 算法对已爬取的网页集合进行计算处理获取种子。通过这种基于预测的主题相关性算法，在考虑网页文本的同时，也借助网页之间的链接关系，选取权威度或中心度高的页面作为种子。其中，权威度高的网页与主题相关性高，而中心度高的网页有利于挖掘出主题相关网页。经实验表明，基于 SVM 分类算法的主题爬虫技术可提高主题相关网页收获率和召回率，进而提高搜索引擎的检索效率。

参 考 文 献

[1] Boanjak M, Oliveira E, et al. TwitterEcho: a distributed focused crawler to support open research with twitter data[C]// WWW'12 Companion Proceedings of the 21st International Conference Companion on World Wide Web. 2012

[2] Kazai G. In Search of Quality in Crowdsourcing for Search Engine Evaluation[J]. Advances in information retrieval, Lecture Notes in Computer Science, 2011, 66(11): 165-176

[3] 许笑，张伟哲，张宏莉，等. 广域网分布式 Web 爬虫[J]. 软件学报，2010，21(5)：1067-1082

[4] 张宪超，徐雯，高亮，等. 一种结合文本和链接分析的局部 Web 社区识别技术[J]. 计算机研究与发展，2012，49(11)：2352-2358

[5] de Groc C. Babouk: Focused Web Crawling for Corpus Compilation and Automatic Terminology Extraction[J]. Web Intelligence and Intelligent Agent Technology (WI-IAT), IEEE/WIC/ACM International Conference, 2011, 3(1), 497-498

[6] 张伟哲，张宏莉，许笑，等. 分布式搜索引擎系统效能建模与评价[J]. 软件学报，2012，23(2)：253-265

[7] 王上，于海，王钲旋. Deep Web 垂直搜索引擎设计与实现[J]. 计算机研究与发展，2009，46：359-365

[8] 蒋华荣，郁雪. 应用遗传算法优化子空间的 SVM 分类算法[J]. 计算机科学，2013，40(11)：255-260，275

[9] 黄仁，王良伟. 基于主题相关概念和网页分块的主题爬虫研究[J]. 计算机应用研究，2013，30(8)：2377-2380

[10] 李晓明，闫鸿飞，王继民. 搜索引擎：原理技术与系统[M]. 北京：科学技术出版社，2004：29-33

[11] Chang Chih-chung, Lin Chih-jen. LIBSVM: A library for support vector machines[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2011, 2(3): 280-292

[12] 李稚楹，杨武，谢治军. PageRank 算法研究综述[J]. 计算机科学，2011，38(Z10)：185-188

(上接第 89 页)

[11] Hsieh J-W, Kuo T-W, Chang L-P. Efficient identification of hot data for flash memory storage systems[J]. ACM Transactions on Storage, 2006, 2(1): 22-40

[12] Huang P-C, Chang Y-H, Kuo T-W, et al. The behavior analysis of flash-memory storage systems[C]// Proceedings of the 2008 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing. 2008: 529-534

[13] Jiang S, Zhang L, et al. S-ftl: An efficient address translation for flash memory by exploiting spatial locality[C]// IEEE 27th Symposium on Mass Storage Systems and Technologies. 2011

[14] Kim J, Kim J M, Noh S, et al. A space-efficient flash translation layer for compactflash systems[J]. IEEE Transactions on Consumer Electronics, 2002, 48(2): 366-375

[15] Kim J M, Min S L, Choi J, et al. A low-overhead highperformance unified buffer management scheme that exploits sequential and looping references[C]// Proceedings of the 4th Symposium on Operating Systems Design and Implementation. 2000: 119-134

[16] Lee S, Shin D, Kim Y-J, et al. Last: locality-aware sector translation for nand flash memory-based storage systems[J]. SIGOPS Oper. Syst. Rev., 2008, 42: 36-42

[17] Lee S-W, Park D-J, Chung T-S, et al. A log buffer - based flash translation layer using fully-associative sector translation[J]. ACM Transactions on Embedded Computing Systems, 2007, 6(3): 1-27

[18] Li Y, Xu J, Choi B, et al. Stablebuffer: optimizing write performance for dbms applications on flash devices[C]// Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM'10). 2010

[19] Lin L, Li X, Jiang H, et al. Amp: An affinity-based metadata prefetching scheme in large-scale distributed storage systems[C]// Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid. 2008

[20] Madhyastha T M, Reed D A. Input/output access pattern classification using hidden markov models[C]// Proceedings of the Fifth Workshop on I/O in Parallel and Distributed Systems (IOPADS'97). 1997

[21] Nath S, Gibbons P B. Online maintenance of very large random samples on flash storage[J]. The VLDB Journal, 2010, 1(1): 970-983

[22] Park D, Du D. Hot data identification for flash-based storage systems using multiple bloom filters[C]// 2011 IEEE 27th Symposium on Mass Storage Systems and Technologies. 2011

[23] Park S, Park J, Kim S, et al. A pattern adaptive nand flash memory storage structure[J]. IEEE Transactions on Computers, 2010, 61(1): 134-138

[24] Repository U T. Oltp application i/o [OL]. http://traces.cs.umass.edu

[25] Zhu Y, Jiang H. Race: A robust adaptive caching strategy for buffer cache[J]. IEEE Transactions on Computers, 2008, 57(1): 25-40

[26] 唐世庆，李云龙，田凤明，等. 基于 Hadoop 的云计算与存储平台研究与实现[J]. 四川兵工学报，2014，35(8)：97-100