DOI：10.7652/xjtuxb201310009

重复数据删除中的无向图遍历分组预测方法

王龙翔，张兴军，朱国峰，朱跃光，董小社
（西安交通大学电子与信息工程学院，710049，西安）

摘要：针对重复数据删除系统中存储容量受内存限制难以进行扩展的问题，提出了一种基于无向图遍历的重复数据删除分组预测方法。该方法将索引表保存在磁盘中，并在内存中维护索引表缓存，以此提高系统最大可支持的存储容量。对于索引表缓存命中率低、系统性能差的问题，采用了图遍历分组方法予以解决，根据数据块访问序列特征信息建立无向图并进行分析，基于分析结果对索引项进行分组，并以组进行缓存替换，从而提高缓存命中率和系统性能。实验结果表明，基于缓存预取原理和无向图遍历分组，在将缓存设置为索引表大小的 10％时，重复数据删除存储系统最大存储容量比原有方法提高了 7.5 倍，缓存命中率由不进行索引项分组时的 47％提高到 87.6％。

关键词：重复数据删除；分组预测；大规模存储系统
中图分类号：TP333 **文献标志码**：A **文章编号**：0253-987X(2013)10-0051-06

A Grouping Prediction Method Based on Undirected Graph Traversal in De-Duplication System

WANG Longxiang, ZHANG Xingjun, ZHU Guofeng, ZHU Yueguang, DONG Xiaoshe
(School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract: An index table grouping prediction method is proposed based on undirected graph traversal to solve the problem that the data storage capacity of the de-duplication system is limited by the memory and is difficult to expand into large-scale. The method saves the index in a disk and maintains the cache of index in memory to expand the maximum storage capacity of system. The hit rate of the grouping prediction and the system performance are improved by grouping index entries based on undirected graph traversal. The method sets up and analyzes the undirected graph based on the features of data chunk sequences, and the groups generated by analyzing the graph is used in cache replacement. Experimental results show that since the proposed method bases on the cache prefetching and the hash table grouping, the index table cache hit rate of the method increases from 47% to 87.6%, and the maximum storage capacity of IDSMS system is 7.5 times higher than that of the existing method for the cache consuming only 10% of the index table size.

Keywords: data de-duplication; grouping predict; large-scale storage system

随着数据规模的急速增长，重复数据删除技术已经得到了越来越广泛的应用，根据 IDC 的报告，80％的公司正在计划采用重复数据删除技术[1]。国内的研究者也分别从不同角度展开了研究：顾瑜等

收稿日期：2012-12-20。 作者简介：王龙翔(1988—)，男，博士生；张兴军(通信作者)，男，副教授，博士生导师。 基金项目：国家科技支撑计划资助项目(2011BAH04B03)；国家自然科学基金资助项目(61173039)；国家高技术研究发展计划资助项目(2011AA01A204)。
网络出版时间：2013-07-25 网络出版地址：http://www.cnki.net/kcms/detail/61.1069.T.20130725.1545.002.html

提出了一种重复数据删除中的可靠性保证方法[2]；付印金等描述了如何将重复数据删除应用在虚拟桌面存储技术中[3]。

数据规模的增长要求重复数据删除系统能够提供更大的存储容量，由于重复数据删除本身的机制原因，其所能支持的存储容量有限。解决该问题的关键是如何处理索引表的增长。该问题主要是由于重复数据删除系统必须维护索引表用来保存数据块元信息，而在备份、归档等应用场景下，需要存储的数据规模十分庞大，索引表的容量远远超过了系统可用内存容量。如果想扩大系统可支持的最大存储容量，可以考虑将索引表保存在磁盘中，然而，由于索引项需要频繁读写，而磁盘访问速度比内存要慢几个数量级，这种方法会导致读写性能极其低。一种解决此问题的方法是在内存中建立缓存机制来加速索引表读写，然而传统的缓存机制在重复数据删除中命中率十分低，因此系统提高的性能十分有限，即使采用 8 块硬盘对索引表进行并行读写，吞吐率也只有 6.5 MB/s[4]。本文将讨论如何建立一种提高缓存命中率的方法。

目前，研究者已经针对重复数据删除的存储容量扩展问题进行了大量研究，解决该问题的方法主要有三类。

第一类方法主要是通过某种途径来创造索引项局部性以提高缓存命中率。具有代表性的为 Zhu 等提出的解决方法，按照备份中数据块的访问顺序对数据以及索引项进行分组，当组中的某一数据块被访问时，将该块所属组的全部索引项读入内存中，同时利用 bloom filter 来加快速度[5]，然而这种方法只适用于备份的应用场景。

第二类方法主要通过相似性来减少索引表占用空间。Bhagwat 等提出将所有的数据块按照其所在文件之间的相似性进行分组，将所有相似的文件切分后形成的数据块及其索引项分为一组，然后只在相似的分组之间进行重复数据删除[6]。与上述方法类似，Lillibridge 等提出了利用稀疏索引来解决该问题，该方法主要通过抽样哈希值并以这些哈希值来进行相似检测[7]。

第三类方法通过改进查找数据结构来加快索引表的查询速度，以此提高系统性能。Min 等提出了基于 LRU 的索引表切分方法，通过把哈希值存放到一系列小索引表中来创造空间局部性[8]。Bobbarjung 等在数字查找树基础上进行改进，提出了一种新的查找数据结构，通过该数据结构能够减少查找索引项所需时间[9]。DBLK 是用于主存储的重复数据删除系统，其中使用了 MBF 技术来加快索引表查询[10]。Tomazic 等提出了一种新的基于 hash 的文件检测方法(FHFEC)来判断某一个给定的指纹值是否存在于系统中[11]。

上述方法存在的问题是局限于特定的应用场景。Wildani 等提出了基于邻居划分的分组预测法来解决该问题[12-13]。分组预测法对于不同的应用场景具有很好的适应性，其主要步骤为：首先记录每一次数据访问的特征信息，然后每隔一段时间通过对已经收集到的数据信息进行计算得到距离矩阵，最后按照距离将其对应的索引项划分为一系列的组，分组之后，当某一索引项被访问时，将该索引项所属组的其他索引项读入内存中。该方法可以同时满足主存储系统与二级存储系统的要求，但是，邻居划分法的分组预测率有待提高。

本文在 Wildani 等所提方法[12-13]的基础上，提出了一种预测率更高的分组方法，该方法的时间复杂度为 $O(N+E)$。实验结果表明，与邻居划分法相比，该方法具有更好的预测率与性能。

1 分组预测法框架

分组预测法框架如图 1 所示，该方法主要用到三种数据结构：指纹索引缓存 i_c，指纹索引表 i_f，组信息索引表 i_g，三者的关系为 $i_c \subseteq i_f \approx i_g$。其中，指纹索引缓存 i_c 采用 LRU 算法进行组的替换。指纹索引表 i_f 中保存了所有已经写入磁盘的数据块所对应的元数据信息，一条索引项记录包括指纹值、数据块在磁盘中的地址以及所属组信息。组信息索引表 i_g 保存了所有的分组信息，根据组 id，就可以得到该组中的所有索引项。指纹索引缓存 i_c 是索引表 i_f 的一个子集，指纹索引缓存 i_c 保存在内存中，而索引表 i_f 以及索引表 i_g 保存在磁盘中。为了提升系统性能，必须尽可能少地访问磁盘。当有 I/O 请求时，经过分块以及指纹计算得到指纹序列，在系统中指纹访问所经过的路径分为 3 种情况：①情况 A 为指纹在缓存中命中；②情况 B 为指纹在缓存中没有命中，但通过查找指纹索引表找到了该指纹并且该指纹属于某个组，则把该指纹所在组中的全部索引项读入缓存中；③情况 C 为指纹在缓存中没有命中，同时在索引表中也没有查找到，或者虽然在索引表中查找到但是不属于某个组，则把该指纹对应的特征信息发送给分组模块进行分组。

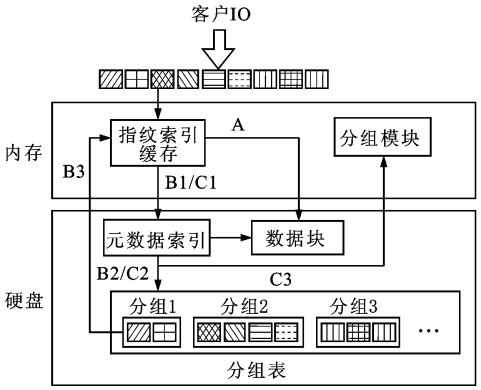

图 1　系统框架

2　基于无向图的分组预测法

分组处理流程如图 2 所示。下面对分组过程进行具体描述。

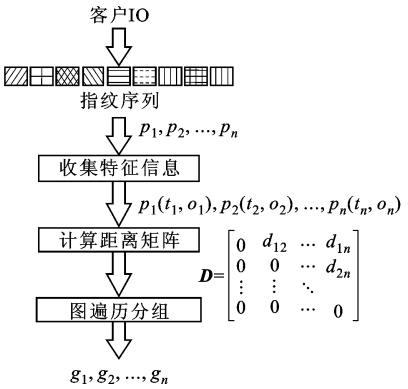

图 2　分组处理流程

2.1　记录数据访问信息

数据访问的特征信息是对索引项进行分组的基础，特征信息由访问时间 t、hash 值对应数据块的 LBA(Logical Block Addressing)地址 o 组成。为了进行分组，对访问序列 $p_1, p_2, \cdots, p_n$，记录下其对应的特征信息 $(t_1, o_1), (t_2, o_2), \cdots, (t_n, o_n)$，当访问次数达到阈值时，对访问序列 $p_1, p_2, \cdots, p_n$ 计算距离矩阵。

2.2　距离矩阵计算

为了进行分组，需要计算所有访问点之间的距离，通过欧氏距离来定义访问点 p_i 与访问点 p_j 之间的距离，如式(1)所示

$$d(p_i, p_j) = (t_s(t_i - t_j)^2 + o_s(o_i - o_j)^2)^{1/2} \quad (1)$$

式中：t 代表访问时间；o 代表 LBA 地址；t_s 和 o_s 分别代表权重因子。通过调整权重因子，可以使某一个特征量对距离的影响变大。距离 d 用于表示无向图中两个点之间边的权值，而无向图具有性质 $d_{i,j}=d_{j,i}$，因此，为了得到 $p_i, p_2, \cdots, p_n$ 之间的关系，只需计算矩阵

$$\boldsymbol{D} = \begin{bmatrix} 0 & d_{12} & \cdots & d_{1n} \\ 0 & 0 & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \quad (2)$$

式中：$\boldsymbol{D}$ 为严格上三角矩阵并且为稀疏矩阵。

2.3　图遍历法

由距离矩阵 $\boldsymbol{D}$ 可以生成无向图 G，由于访问点之间具有传递性，即如果 A 与 B 相连，B 与 C 相连，则 A 与 C 相连，因此分组的思想是尽量将一个点附近的其他点分为一组。具体步骤为，首先使用广度优先遍历法从任一节点开始遍历图 G，把所有遍历过的点 $v_1, v_2, \cdots, v_i$ 所对应的距离 $d_1, d_2, \cdots, d_{i-1}$ 进行累加得到和 s，当 s 大于某一阈值 t 时，已遍历过的点构成组 g_1，之后从图 G 中删除 $v_1, v_2, \cdots, v_i$，一直重复该步骤直到遍历结束，最后得到分组 $g_1, g_2, \cdots, g_n$。算法描述如下。

算法 1　无向图广度优先遍历法

输入　图 $G=(V,E)$ 距离矩阵 $\boldsymbol{d}[u,v]$

输出　分组序列 $g_1, g_2, \cdots, g_n$

```
1:  for each vertex u∈V[G]−s  do
2:    color[u] ←WHITE
3:    π[u] ←NIL
4:  end for
5:  color[s] ←GRAY
6:  π[s] ←NIL
7:  i ← 0;
8:  Q ← NIL
9:  ENQUEUE(Q,s)
10:  while Q≠∅ do
11:    u ← DEQUEUE(Q)
12:    for each v∈Adj[u] do
13:      if color[v]=WHITE then
14:        color[v] ← GRAY
15:        π[v] ← u
16:        ENQUEUE(Q,v)
17:      end if
18:    end for
19:    color[u] ← BLACK
20:    sum ← sum+d[u, π[u]]
21:    if sum<threshold then
22:      g[i++] ← u;
23:    else
24:      write g to disk and empty g;
```

```
25:        sum ← 0;
26:      end if
27:   end while
```

该算法在无向图用邻接矩阵存储时的时间复杂度为 $O(N+E)$。

3 实验分析

为了验证分组方法的有效性，本文在课题组所开发的智能重复数据删除存储系统(Intelligent Deduplication Storage Management System，IDSMS)[14]中加入分组模块来进行性能评测，实验代码用 java 实现，实验环境为 ubuntu 12.04 操作系统，其中 CPU 为 Pentium® Dual-Core CPU E5500 @ 2.80 GHz，内存大小为 2 GB，IDSMS 系统可用内存为 1.5 GB，硬盘为希捷 ST3500418AS (500 GB/7 200 r/min)。

3.1 数据集

本文选择了两组数据集来进行实验，数据集 1 由多个版本的 Linux 内核源码文件组成，数据集 2 由 PDF 文件组成，这两组数据集都是以文件为单位进行读写，文件之间的到达顺序是随机的，主要信息如表 1 所示。

3.2 结果分析

本文主要从缓存大小与系统最大存储容量大小的关系和图遍历法、邻居划分法[12-13]以及不分组法对命中率和性能的影响结果进行分析。

3.2.1 *系统存储容量分析* IDSMS 系统在之前的实现中采用将索引表完全保存在内存中的方法，其最大存储容量 c 为

$$c = nk \tag{3}$$

式中：n 为系统最大可存储的数据块数量；k 为数据块大小。由于数据块的指纹值与实际存储的数据块是一一对应的，因此系统最大可存储的数据块数量 n 与系统最大可存储的指纹值数量 m 具有以下关系

$$n = m \tag{4}$$

而 m 可由以下公式得到

$$m = te^{-1} \tag{5}$$

式中：t 为系统可用内存；e 为索引表中的一条索引项大小

$$e = \sum_{i=1}^{a} s_i \tag{6}$$

其中 s_i 代表索引项中的一个索引属性，例如指纹值或者数据块在磁盘中的 LBA 地址，a 为索引项属性的个数。

结合式(3)～式(6)，可得系统最大存储容量为

$$c = tk\left(\sum_{i=1}^{a} s_i\right)^{-1} \tag{7}$$

如果将索引表保存在磁盘中，同时使用缓存来加速系统读写性能，缓存大小 o 与索引表大小 p 有以下关系

$$o = lp \tag{8}$$

式中：l 为缓存容量系数，表示缓存与索引表的容量关系。由式(7)、式(8)可得到系统最大存储容量为

$$c = tk\left(l\sum_{i=1}^{a} s_i\right)^{-1} \tag{9}$$

在本文所测试的环境中，存储系统可用内存为 1.5 GB，如果不使用分组方法，则索引项大小为 24 B，其中指纹采用 tiger128 函数计算，大小为 16 B，数据块所在磁盘地址为 8 B。如果使用分组方法，则索引项大小为 32 B，除了指纹值与数据块所在磁盘地址之外，还要保存所在组编号(8 B)。由式(7)、式(9)可分别计算出不使用缓存与使用不同大小的缓存后系统所能支持的最大存储容量，如图 3 所示。

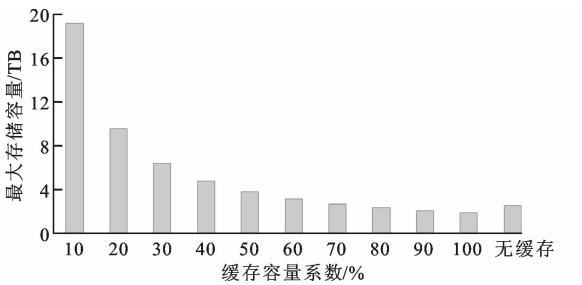

图 3 缓存容量系数与系统最大存储容量的关系

表 1 数据集信息

数据集名称	数据集类型	文件总数(去重后)	数据集大小/MB(去重前)	数据集大小/MB(去重后)	文件访问次数	索引项访问次数	指纹数量
数据集 1	Linux 内核源码文件	17 023	443	151	50 000	782 925	48 192
数据集 2	PDF 文件	380	10 842	412	10 000	2 868 124	87 459

考虑到缓存系数设置太小会降低系统性能，设置太大会减少系统最大可支持的存储容量，为了达到两者的平衡，实际中系数取 10% 比较合适，此时系统最大存储容量从完全使用内存索引表时的 256 GB 增加到了 1 920 GB，增加了 7.5 倍。

3.2.2　缓存命中率分析　缓存命中率决定了系统的性能，图 4、图 5 分别是对数据集 1 与数据集 2 在不同缓存系数下的命中率实验结果。数据集 1 和数据集 2 的缓存理想命中率分别是 93.8%、97.0%，缓存理想命中率为

$$R = (t - f)t^{-1} \tag{10}$$

式中：t 代表指纹访问次数；f 代表指纹数量。在缓存系数为 10% 时，数据集 1 测试结果表明不分组法、邻居划分法、图遍历法的缓存命中率分别为 47%、86.2%、87.6%。数据集 2 测试结果表明不分组法、邻居划分法、图遍历法的缓存命中率分别为 13.2%、96.4%、96.6%。

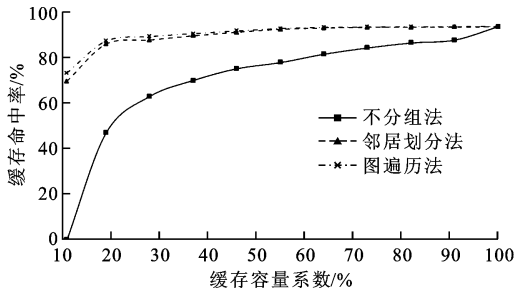

图 4　数据集 1 的缓存命中率结果

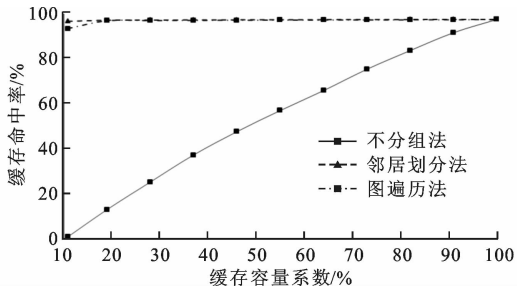

图 5　数据集 2 的缓存命中率结果

实验结果表明，图遍历法比邻居划分法具有更高的命中率，这是由于图遍历法能更加准确地反映索引项之间的关系，此方法生成的分组能够预测到更多将来会被命中的索引项。对于访问序列 p_1，p_2，…，p_3，邻居划分法只计算 p_i 与 p_{i+1} 之间的关系，并以此作为划分分组的依据。图遍历法则会计算 p_i 与 p_1，p_2，…，p_{i-1}，p_{i+1}，p_{i+2}，…，p_n 之间每一对访问点之间的关系，因此能够划分出更加精确的分组。在极端情况下，如果一个访问序列，对于任意的 i，p_i 与 p_{i+1} 都无访问关系，则使用邻居划分法不会对此序列形成分组，从而退化成不使用分组方法的情况。这种情况比较容易出现在文件平均大小较小时，此时图遍历法相对邻居划分法能够得到更多的性能提升。在文件平均大小较大时，由于有较多的数据块属于同一个文件，邻居划分法已经形成了较为精确的分组，图遍历法相比邻居划分法能够提升的缓存命中率有限。

3.2.3　系统性能分析　本文通过比较不同方法进行数据读写所消耗的时间来进行性能评测，实验结果如图 6、图 7 所示，在缓存系数为 10% 时，数据集 1 中不分组法、邻居划分法、图遍历法所需的运行时间分别为 3 935 s、1 030 s、964 s。数据集 2 中不分组法、邻居划分法、图遍历法所需的运行时间分别为 23 654 s、990 s、996 s。由 3.2.2 小节可知，数据集 2 中相比邻居划分法，图遍历法提升的缓存命中率较小，但需要更多的计算时间，因此处理数据访问所需的总时间较多，性能比邻居划分法略差。该实验结果表明，图遍历法适合于平均文件大小较小的情况。

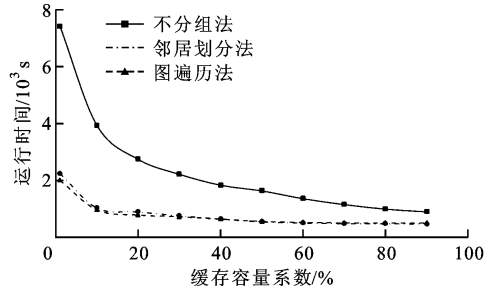

图 6　数据集 1 的运行时间

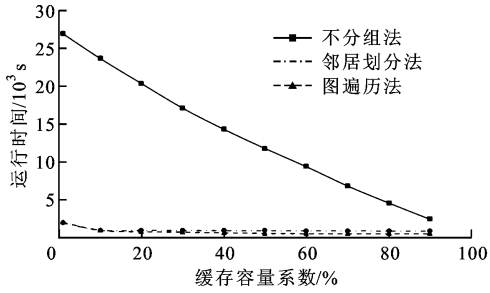

图 7　数据集 2 的运行时间

4　结　论

本文提出了一种大规模数据环境下对重复数据删除系统存储容量进行扩展的方法，该方法的主要原理是将索引表保存在磁盘中，并在内存中维护缓存。针对缓存命中率低的问题，本文通过分析数据块之间的关系，以此为基础将数据块所对应的索引

项进行分组。数据块之间的关系主要通过数据块访问时间与数据块在磁盘中的地址来进行计算,分组方法采用无向图遍历法,进行分组之后,缓存的替换以组为单位。实验结果表明,图遍历法能够显著提高缓存命中率,采用此方法之后在缓存系数设置为10%时,IDSMS系统比原有的方法存储容量扩展了7.5倍。

参考文献:

[1] DUBOIS L, AMALDAS M, SHEPPARD E. Key considerations as deduplication evolves into primary storage, White Paper 223310 [R]. Framingham, MA, USA: IDC, 2011.

[2] 顾瑜, 刘川意, 孙林春, 等. 带重复数据删除的大规模存储系统可靠性保证 [J]. 清华大学学报:自然科学版, 2010, 50(5): 739-744.
GU Yu, LIU Chuanyi, SUN Linchun, et al. Reliability provision mechanism for large-scale de-duplication storage systems [J]. Journal of Tsinghua University: Science and Technology, 2010, 50(5): 739-744.

[3] 付印金, 肖侬, 刘芳, 等. 基于重复数据删除的虚拟桌面存储优化技术 [J]. 计算机研究与发展, 2012, 49(S1): 125-130.
FU Yinjin, XIAO Nong, LIU Fang, et al. Deduplication based storage optimization technique for virtual desktop [J]. Journal of Computer Research and Development, 2012, 49(Supp 1): 125-130.

[4] QUINLAN S, DORWARD S. Venti: a new approach to archival storage [C] // Proceedings of the FAST 2002 Conference on File and Storage Technologies. Berkeley, CA, USA: USENIX, 2002: 89-101.

[5] ZHU B, LI Kai, PATTERSON H. Avoiding the disk bottleneck in the data domain deduplication file system [C] // Proceedings of the 6th USENIX Conference on File and Storage Technologies. Berkeley, CA, USA: USENIX, 2008: 269-282.

[6] BHAGWAT D, ESHGHI K, LONG D D E, et al. Extreme binning: scalable, parallel deduplication for chunk-based file backup [C] // Proceedings of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. Piscataway, USA: IEEE, 2009: 1-9.

[7] LILLIBRIDGE M, ESHGHI K, BHAGWAT D, et al. Sparse indexing: large scale, inline deduplication using sampling and locality [C] // Proceedings of the 7th Conference on File and Storage Technologies. Berkeley, CA, USA: USENIX, 2009: 111-123.

[8] MIN J, YOON D, WON Y. Efficient deduplication techniques for modern backup operation [J]. IEEE Transactions on Computers, 2011, 60(6): 824-840.

[9] BOBBARJUNG D R, JAGANNATHAN S, DUBNICKI C. Improving duplicate elimination in storage systems [J]. ACM Transactions on Storage, 2006, 2(4): 424-448.

[10] TSUCHIYA Y, WATANABE T. DBLK: deduplication for primary block storage [C] // Proceedings of the 27th IEEE Symposium on Mass Storage Systems and Technologies. Piscataway, USA: IEEE, 2011: 1-5.

[11] TOMAZIC S, PAVLOVIC V, MILOVANOVIC J, et al. Fast file existence checking in archiving systems [J]. ACM Transactions on Storage, 2011, 7(1): 24-44.

[12] WILDANI A, MILLER E L, RODEH O. Hands: a heuristically arranged non-backup in-line deduplication system, UCSC-SSRC-12-03 [R]. Piscataway, USA: IEEE, 2012.

[13] WILDANI A, MILLER E, WARD L. Efficiently identifying working sets in block I/O streams [C] // Proceedings of the 4th Annual International Conference on Systems and Storage. New York, USA: ACM, 2011: 46-57.

[14] ZHU Guofeng, ZHANG Xingjun, WANG Longxiang, et al. An intelligent data de-duplication based backup system [C] // Proceedings of the 15th International Conference on Network-Based Information Systems. Piscataway, USA: IEEE, 2012: 771-776.

[本刊相关文献链接]

吕正,陈昊,陈峰,等.一种ARM存储模型的快速检测方法.2013,47(6):68-72.[doi:10.7652/xjtuxb201306012]

边根庆,高松,邵必林,等.面向分散式存储的云存储安全架构.2011,45(4):41-45.[doi:10.7652/xjtuxb201104008]

余思,桂小林,黄汝维,等.一种提高云存储中小文件存储效率的方案.2011,45(6):177-181.[doi:10.7652/xjtuxb 201106011]

黄汝维,桂小林,余思,等.支持隐私保护的云存储框架设计.2011,45(10):1-6.[doi:10.7652/xjtuxb201110001]

颜秉珩,钱德沛.一种支持负载均衡的存储调度算法.2009,43(10):61-65.[doi:10.7652/xjtuxb200910013]

颜秉珩,钱德沛.面向存储资源管理的多协议存储系统.2009,43(6):10-14.[doi:10.7652/xjtuxb200906003]

(编辑 武红江)