一种基于历史信息的一致性哈希集群重复数据删除路由策略

邢玉轩[1]，肖侬[1]，刘芳[1]，付印金[2]，李芳[1]，巫小泉[1]

（1 国防科学技术大学计算机学院，湖南长沙 410073　　2 解放军理工大学指挥信息系统学院，江苏南京，210007）

（xinghuan1990@gmail.com）

A History-based Consistent Hashing Routing Policy for Cluster Deduplication System

Xing Yuxuan[1], Xiao Nong[1], Liu Fang[1], Fu Yinjin[2], Li Fang[1], Wu Xiaoquan[1]

(1School of Computer, National University of Defense Technology, Changsha 410073, China

2College of Command Information System, PLA University of Science and technology, Nanjing 210007, China)

Abstract: With the explosive growth of global data, single-node deduplication system can't satisfy the performance request. Then Cluster Deduplication System appears. It is a big challenge to improve data transmission efficiency, to save network bandwidth and to enhance system scalability in Cluster Deduplication System. We design a history-based consistent hashing routing policy: one hand it matches fingerprints in local index table before routing, which can reduces a large number of fingerprint message requests; on the other hand it effectively mitigates the sharp decrease of global data elimination-ratio and unbalanced data-distribution through consistent hashing routing. The experimental results on three real datasets have shown that our policy can decrease message requests by 20%~80% and keep data elimination-rate decreased less than 33% while keeping load-balancing.

Keywords: Cluster Deduplication System; routing policy; message request; load balancing

摘要：全球数据量爆炸式增长，单节点重复数据删除系统已不能满足性能需求，集群重复数据删除系统应运而生。如何提高数据传输效率、节约网络带宽和增强系统的可扩展性，成为当前面临的严峻挑战。我们提出一种基于历史数据信息的一致性哈希路由策略，通过在本地缓存热点数据块指纹，数据路由前先在本地索引，可以大大减少索引消息请求数量，并且采用一致性哈希的路由策略，有效的缓解集群系统中动态扩展存储节点导致的全局数据重删率急剧恶化与负载不均。我们在三类真实的数据集上进行试验，能减少20%~80%的指纹消息请求，动态扩展存储节点导致数据缩减率降低保持在33%以下，并且能够很好地保持系统节点间负载均衡。

关键词：重复数据删除集群；路由策略；消息请求；负载均衡

中图分类号：TP301

随着信息技术的快速发展以及互联网的普及，全球数据量呈爆炸式增长。国际数据公司（International Data Company, IDC）2012 年一项调查研究表明[1]：到 2020 年，全球数据量将达到 40ZB（1ZB=10^{12}GB）。绝大部分数据来自平板电脑、手机和嵌入式传感器等移动终端，这些数据为人类的生产和生活带来极大便利，同时也为数据存储和管理带来严峻挑战。国际数据公司另一项报告指出[2]：超过一半以上的企业一次数据丢失事故会造成至少 10 万美元的损失。因此，本地备份和远程备份被广泛用于企业的数据保护，如何高效的传输数据以及提高存储设备利用率成为亟待解决的难题。

重复数据删除技术将文件划分成小的数据块，只传输和备份唯一的数据块，可以有效节约网络带

基金支持：国家自然科学基金项目（61025009,61232003,61170288,61332003,61402518）；国家“八六三”高技术研究发展计划基金项目（2013AA013201）

宽、节省存储空间，因而被广泛应用于备份系统。它首先按照一定策略：如固定大小分块（static chunking, SC）、基于文件内容分块（content defined chunking, CDC）、全文件分块（whole file chunking, WFC）等[3]，将数据流进行分块，然后使用加密算法（SHA-1、MD5 等）计算每个数据块的指纹用来唯一标识该数据块，在备份服务器端维护一张指纹到数据块存储地址的映射表，数据块指纹被发送至服务器，服务器查询该指纹是否在映射表中存在，如果存在说明该指纹对应的数据块已经存在，不需要再次发送存储，否则，将数据块备份到存储服务器，同时更新映射表。当数据丢失后，备份终端发送文件恢复请求，存储服务器根据所需恢复的文件的指纹列表，在本地获取相应的数据块然后重组返回终端，直到全部数据恢复。但是，随着数据迅速增长，需要备份的文件数量越来越庞大，单节点的备份服务器已经不能满足性能需求，多节点在线的重复数据删除集群系统应运而生。它提供高吞吐量的同时也面临以下几个挑战：节点间通信开销越来越大；存储节点互相独立进行消重以及动态扩展存储节点使得全局重删率急剧下降。因此，设计新的系统架构和路由策略解决以上问题具有重大研究意义。

我们做出以下两点贡献：通过在备份终端缓存经常出现的数据块指纹，构建本地指纹索引表，大大减少与存储节点的通信数量；采用基于超块的一致性哈希路由策略，将动态扩展存储节点带来的负面影响控制在一定范围。下文第二章介绍相关背景，第三章详细介绍系统和路由策略的设计与实现，第四章做实验进行分析比较，第五章归纳总结，并说明将来要做的工作。

1. 背景

下面我们介绍集群复数据删除系统的结构原理和国内外相关研究工作。

1.1 集群重复数据删除系统

集群重复数据删除系统通常由备份终端、元数据管理服务器和存储服务器集群（包括众多存储节点）构成，其中：备份终端将数据流分块并计算数据块指纹，然后与元数据服务器和存储节点进行交互开始备份；元数据管理服务器接收备份终端发送的文件数据块指纹列表和整个文件的校验信息（用于验证文件恢复正确性）；多个存储节点组成存储服务器集群，每个存储节点独立进行数据消重备份，其维护一张数据块指纹到存放地址的映射表，用于块指纹检索和数据块的获取。按照消重的时机可以分为在线的重复数据删除集群和离线的重复数据删除集群，按照消重位置分为源端重复数据删除集群和服务器端重复数据删除集群，按照是否利用先前的存储信息来指导路由分为有状态路由重复数据删除集群和无状态路由重复数据删除集群。

1.2 相关工作

根据 Broder 最小值独立置换理论[4]，Bhagwat 等人提出 Extreme Binning 策略[5]，它选择一个文件中所有数据块的最小指纹值来代表该文件所有指纹，如果两个文件最小指纹相同则表明这两个文件相似，并使用最小的指纹进行路由，在存储节点选择相似的文件进行去重，加速指纹索引过程。但是当文件间的相似度不高时，该策略不能发挥其优势。NEC 设计的 HYDRAstor 系统[6]，选用 64KB 大小的数据块粒度，使用分布式哈希表（Distributed Hash Table, DHT）将数据块路由到不同存储节点，然后在节点内部独立进行消重。HYDRAstor 虽然能够很好地平衡数据缩减率和元数据以及通信开销，但是如此大粒度对一些应用数据集并不适用，并且当存储节点扩展或者出现故障时，会导致全局数据缩减率急剧下降。IBM 公司的 PretecTier[7]充分利用备份数据流中数据的相似性，将数据流划分为 16MB 的大粒度数据块，计算每个数据块签名，通过签名判断可能相似的数据块，然后再通过字节级的比较删除冗余。我们之前提出的$\sum -Dedupe$[8]同样基于 Broder 定理理论证明并且通过大量实验验证多个数据块组成的超块其代表性指纹（我们称之为掌纹）的相似度就是超块间众多数据块的相似度，采用超块粒度并且兼顾存储节点负载进行路由。但是，上述两种路由机制均不能很好地解决存储节点扩展带来的系统性能下降。

2. 系统设计与实现

为了减少备份终端和存储节点间数据通信量，并且降低存储节点动态扩展带来的负面影响，同时保证全局数据缩减率以及存储节点负载均衡，我们基于指纹索引历史信息提出一种新的路由策略。

2.1 基本结构和路由算法

经过大量实验我们发现：不同应用间共享的数据量很小，即不同应用类型之间的重复数据很少几乎可以忽略[9]。此项发现，也被微软之后通过统计分析分布于全球的 15 台服务器上数据证实[10]。除此之外，企业和个人多次归档的数据集，不同版本之间重复的数据块绝大部分是不变的。

基于以上事实，我们在备份终端开辟一块空间

保存备份文档共有的数据块指纹，在进行路由之前首先在本地进行指纹索引，如果存在则不需要再发送到存储节点进行查询。具体实现如下：在备份终端内存中开辟一块空间保存数据块的指纹列表，我

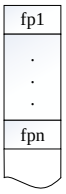

图 1　本地索引表

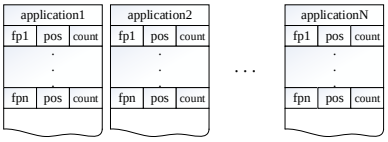

图 2　存储节点索引表

们使用 SHA-1 算法生成指纹，每个指纹占用 20byte（160bit），保存 1000000 条需要 19MB 的内存，其开销可以接受，本地索引表（如图 1）每项只包括数据块的指纹值。存储节点为每类应用分别建立一张索引表（如图 2），表中每项包括数据块指纹值、存放地址和该数据块出现次数，不同应用在各自的索引表中进行匹配。存储服务器新来一个指纹在索引表中匹配，如果没有，说明之前没有存储该数据块，将指纹加入索引表继续处理下一个指纹；如果命中，判断出现次数（count）是否大于我们设定的阈值（N），大于则继续判断本地索引表项是否还有空间，若有将其加入本地索引表，若没有按照最近不使用替换策略（LRU）删除固定数量指纹，然后将该指纹添加进去，如果出现次数（count）没有超过阈值（N），则处理下一条指纹（流程如图 3）。通过本地保存经常出现的指纹，在指纹路由前先在本地索引表中进行匹配，如果指纹存在则说明存储节点已经保存该数据块，不需再次发送查询，这样可以大大减少备份终端与存储服务器集群之间的消息数量。

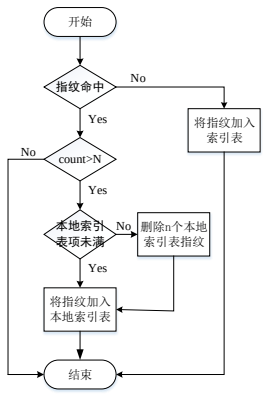

图 3　本地索引表生成

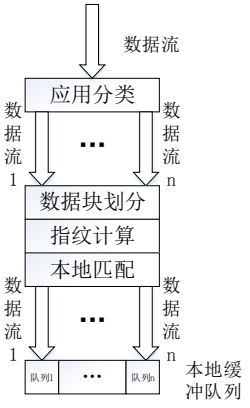

图 4　备份终端数据处理

备份终端首先将数据流按照应用类型进行分类，然后选择合适的分块策略进行数据块划分并计算指纹，然后在本地索引表中匹配，如果命中则该数据块已经在存储节点存在，若没有命中，则将指纹加入每个存储节点缓冲队列，达到一定数量一起发送至存储节点进行索引，其流程如图 4 所示。

我们采用一致性哈希的思想来对数据块进行路由，首先利用哈希算法将数据块指纹映射到 0~MAX（由我们设定）范围内的一个值，然后对 8 取模，然后将该指纹路由至对应的存储节点（如图 5（a））。

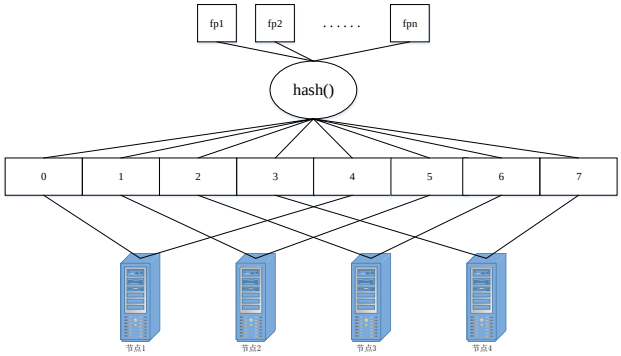

图 5（a）　路由机制

当集群动态扩展节点时（假设有 N 个节点扩展到 2N 个），我们将 8 变为 24，保证能够落在原节点上的 1/3 指纹仍然落在原节点，2/3 指纹落在新增的节点上（如图 5（b））。这样可以有效避免全局数据缩减率急剧下降。

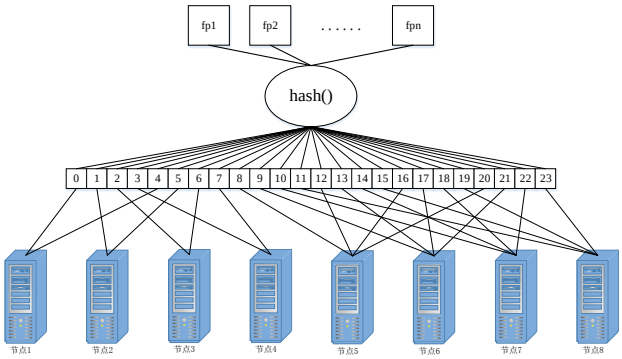

图 5（b）　路由机制

详细路由算法如下所示：

基于历史信息的一致性哈希路由算法

1. 先在本地索引表中匹配该数据块指纹，如果存在，继续处理下一个数据块指纹，如果不存在，转至步骤 2；
2. 利用特定哈希算法（BKDRHash）将数据块指纹进行处理产生 0~MAX 之间的一个值，找到该值对应的存储节点 id；
3. 按照上述策略处理数据块指纹，直到到达同一存储节点的指纹数量达到固定数量，将所有指纹发送至该节点进行索引；
4. 如果在存储节点索引表中命中，说明该数据块存在，将没有命中的指纹和需要保存的指纹反馈给备份终端；

5. 备份终端将数据块和指纹发送到对应存储节点进行存储，同时更新本地索引表。

2.2 系统结构

我们设计的在线重复数据删除集群系统结构如图 6 所示，它包括备份终端、元数据管理服务器以及重复数据删除服务器集群（由多个存储节点组成），它们之间通过以太网连接。备份终端对数据流按应用进行分类，然后根据应用类型选择数据分块策略划分数据块并计算指纹。元数据管理服务器处理备份终端发送的文件备份和恢复请求，保存备份文件的相关信息：如文件校验值和数据块指纹列表等。重复数据删除服务器集群是整个系统的核心，它检测备份终端发送过来的数据块指纹判断该数据块是否已经存在，如果不存在则通知客户端将该数据块发送至存储节点存储，更新索引表；在恢复过程，接收备份终端发送的文件指纹列表，根据指纹获取数据块并返回备份终端。具体实现流程如下：

备份过程

1. 备份终端向元数据服务器发送一个备份请求，包括文件的 id（基于整个文件内容的抗冲突哈希值，用来检验文件恢复正确性）、文件包含所有数据块的指纹列表及指纹对应的存储节点，如果元数据接受该请求，则返回一个开始备份标志；
2. 备份终端按照基于历史信息的一致性路由策略将打包的指纹发送至存储节点；
3. 存储节点通过索引，将所需发送的数据块和满足条件的指纹反馈给备份终端；
4. 备份终端更新本地索引表，同时将数据块和指纹打包发送给相应存储节点；
5. 存储节点保存数据块，更新索引表；
6. 重复 2~5 过程，直到所有文件备份结束。

恢复过程

1. 备份终端发送一个文件恢复请求给元数据服务器，元数据服务如果接受请求则返回一个开始恢复标志；
2. 备份终端将要恢复的所有文件 id 发送至元数据服务器，元数据服务器查找对应的文件信息返回给备份终端；
3. 备份终端将文件的数据块指纹列表发送给对应的存储节点将数据块取回本地进行文件重组，每恢复一个文件，计算恢复文件的抗冲突哈希值，检验文件恢复正确性；
4. 重复上述过程，直到所有文件恢复完毕。

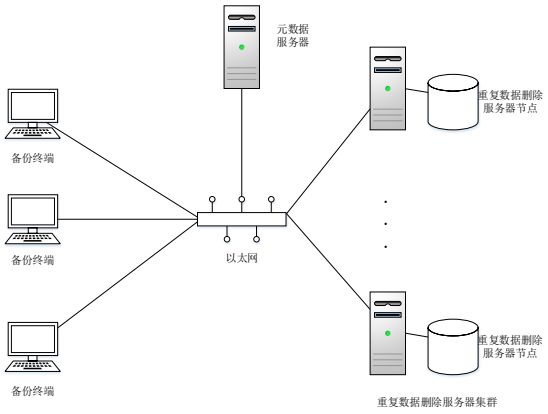

图 6　系统结构图

3. 性能评价

为了验证上文路由算法的性能，我们在 Linux 平台使用 C++完成系统原型，并采用三类真实的数据集进行测试。

3.1 实验平台和工作负载

我们使用 14 台电脑进行实验验证，其中一台作为备份终端，一台作为元数据管理服务器，其它作为存储集群节点。它们都运行 CentOS6.4 系统，主频 3.3Hz 2 核 4 线程，内存 8G。它们通过一台锐捷千兆以太网交换机相连，使用基于 TCP 协议之上的 RPC 进行通信。我们收集三类真实的数据集对系统进行测试（如表 1）。

表 1　测试数据集

数据集	大小（GB）	所有数据块个数	唯一数据块个数	重删率
Home	750.96	99609670	7148870	13.9336
Mail	525.51	137759031	12440532	11.0734
VM	42.64	11177702	5885997	1.899

其中 Home 来自网站[11]，由加州大学伯克利分校发布的对其文件服务器内容采用基于内容分块的多次备份，我们使用其中 13 次的备份进行测试；Mail 和 VM 来自网站[12]，它们均采用 4K 固定大小分块策略，其中 Mail 是邮件服务器内容的多次备份，VM 是虚拟机镜像的多次备份。重删率为所有数据块个数与唯一数据块个数的比值。

3.2 消息数量

当存储节点数据块指纹个数超过一定值后，我们就将该指纹保存在备份终端索引表，图 7、8、9 分别为三类数据集不同阈值时备份终端索引表的命中率和替换率（命中率和替换率均为命中指纹数量和发生替换指纹数量与所有数据块指纹数量的比值）。

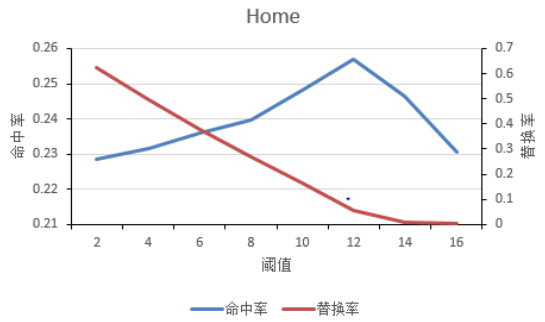

图 7　Home 随阈值变化趋势

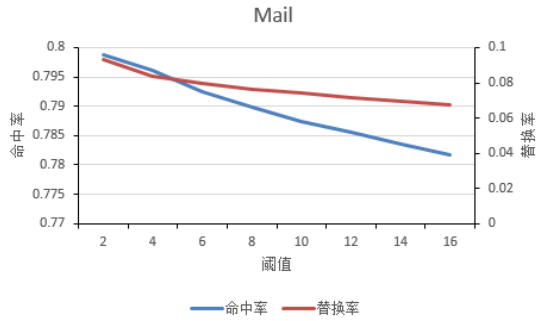

图 8　Mail 随阈值变化趋势

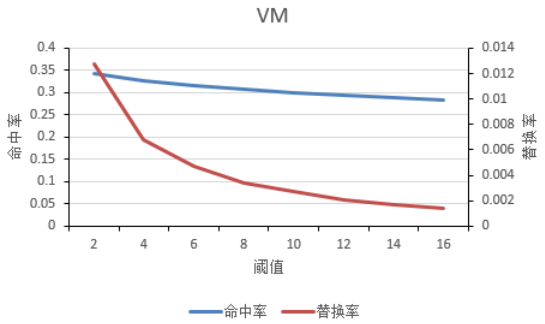

图 9　VM 随阈值变化趋势

从上图中可以看到，替换率均是随着阈值的升高而下降，因为阈值变大满足要求进入备份终端索引表的指纹变少，所以指纹替换次数也相应减少。Home 数据集命中率最高点出现在阈值为 12 附近，可能是由于该数据集中热点指纹出现频次为 12，如果阈值太小满足条件的指纹数量增多会造成热点指纹来回替换，如果阈值过高一些热点指纹不能进入索引表，均导致命中率下将。对于 Mail 和 VM 数据集阈值越大进入本地索引表指纹越少，并且热点指纹出现频次本来也不高，所以命中率随着阈值的增大而降低。

3.3 负载分布

我们统计随着节点数量变化各节点负载变化情况，从图 10、11、12 中可以看出，对于三类数据集，不论存储集群节点为何值时，系统均能很好保持负载均衡。

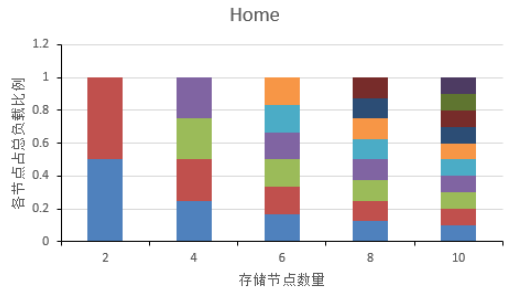

图 10　Home 节点负载

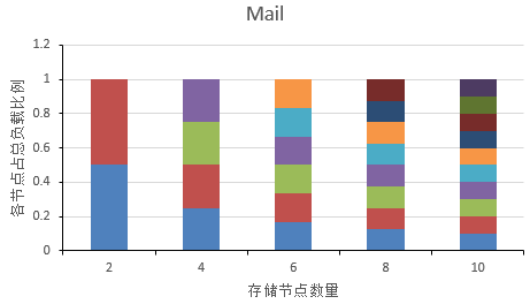

图 11　Mail 节点负载

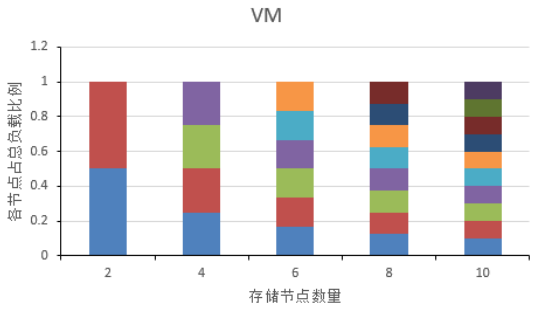

图 12　VM 节点负载

3.4 动态扩展存储节点

为了测试当系统节点动态扩展系统性能如何变化，我们设计当系统备份次数分别为 0、3、5、7、9 时，节点数量由 4 变为 8，使用 Home 数据集测试统计数据重删率和负载两项指标。

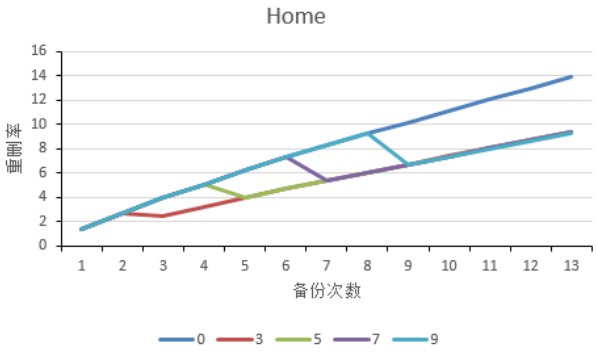

图 13　重删率变化曲线

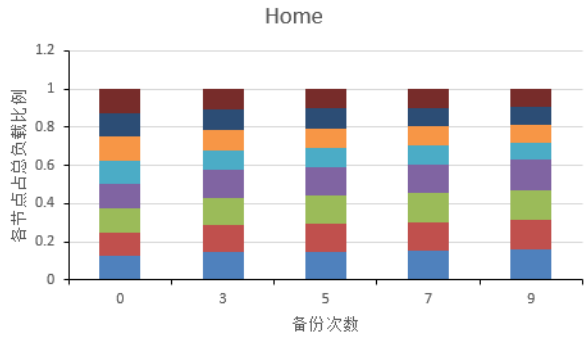

图 14　负载分布

从图 13 中可以看出不论是从哪次备份开始动态扩展系统节点都会导致数据重删率下降，之后随着备份次数增加慢慢升高，不同时机开始备份的重删率曲线几乎重合，到第 13 次备份结束时，数据缩减率下降 32%~33%。从图 14 中看出系统各节点负载分布比较均衡。

4. 结束语

我们设计实现一种基于历史信息的一致性哈希路由算法，通过本地缓存热点指纹减少 20%~80%的消息请求数量，并且采用一致性哈希的分发策略有效缓解系统动态扩展节点时数据重删率的急剧下降，同时保持节点间负载均衡。

下一步利用新型存储介质（如 SSD）作为备份终端本地 cache，在文件恢复过程中，缓存热点指纹和对应的数据块，以此减少文件恢复过程中数据传输量，加快文件恢复速度。

参　考　文　献

[1] John Gantz, David Reinsel. The Digital Universe Decade – Are You Ready? [R] IDC White Paper, 2012: 1~16.

[2] C.J. Kolodg. Effective Data Leak Prevention Programs: Start by Protecting Data at the Source-Your Databases [R]. White Paper, IDC, Aug. 2011.

[3] 付印金，肖侬，刘芳。重复数据删除关键技术研究进展，计算机研究与发展，49（1）：12-20，2012.

[4] A. Z. Broder. On the resemblance and containment of documents [C]. In SEQUENCES'97: Proceedings of the Compression and Complexity of Sequences, 1997: 21~29.

[5] Benjamin Zhu, Kai Li, Hugo Patterson. Avoiding the Disk Bottleneck in the Data Domain Deduplication File System [C]. In Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST'08), Berkeley, CA, USA: USENIX, 2008: 269~282.

[6] Cezary Dubnicki, Leszek Gryz, Lukasz Heldt, Michal Kaczmarczyk, Wojciech Kilian, Przemyslaw Strzelczak, Jerzy Szczepkowski, Cristian Ungureanu, Michal Welnicki. HYDRAstor: a Scalable Secondary Storage [C]. In Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST'09), Berkeley, CA, USA: USENIX, 2009: 197~210.

[7] IBM ProtecTIER Deduplication Gateway [EB/OL]. http://www-03.ibm.com/systems/storage/tape/ts7650g/index.html, 2011.

[8] Yinjin Fu, Hong Jiang, Nong Xiao. A Scalable Inline Cluster Deduplication Framework for Big Data Protection. The ACM/IFIP/USENIX 13th International Conference on Middleware (Middleware'12), Montreal, Quebec, Canada, December 3-7, 2002: 354~373.

[9] Yinjin Fu, Hong Jiang, Nong Xiao, Lei Tian, Fang Liu. AA-Dedupe: An Application-Aware Source Deduplication Approach for Cloud Backup Services in the Personal Computing Environment [C]. Proceedings of the 13th IEEE International Conference on Cluster Computing (Cluster'11), 2011: 112~120.

[10] A. El-Shimi, R.Kalach, A. Kumar, J. Li, A. Oltean, and Sudipta Sengupta. Primary Data Deduplication – Large Scale Study and System Design [C]. Proceedings of the 12th USENIX Annual Technical Conference (ATC'12), 2012: 285~296.

[11] http://tracer.filesystems.org/.

[12] SNIA. IOTTA repository (January 2009), http://iotta.snia.org/.

邢玉轩，男，1990 年生，硕士研究生，主要研究方向为网络存储。

肖侬　男，1969 年生，博士，教授，主要研究方向为网格计算、存储以及计算机体系结构。

刘芳　女，1976 年生，博士，副教授，主要研究方向为网络存储和信息安全。

付印金　男，1984 年生，博士研究生，主要研究方向为网络存储。

李芳　女，1989 年生，硕士研究生，主要研究方向为网络存储。

巫小泉　男，1989 年生，硕士研究生，主要研究方向闪存文件系统。

校对负责人：邢玉轩
联系电话：13574165446
E-mail：xinghuan1990@gmail.co