基于动态区间映射的文档聚类算法

孙永林 刘 仲
(国防科技大学计算机学院 长沙 410073)

摘 要 随着信息数字化的快速发展,新兴的归档存储成为研究热点,空间利用率和扩展性是其关键问题。利用基于内容分块存储实现重复数据删除,是提高存储空间利用率的有效途径,但由于归档数据规模巨大,在所有数据中寻找共享分块的做法十分低效。将动态区间映射思想引入信息聚类,提出了基于动态区间映射的文档聚类算法DC-DIM;利用分块和特征提取方法产生文档的分块特征集合,将分块特征集合映射在区间链上,依据文档分块特征集合的映射分布确定文档的存储容器,实现文档聚类;将内容相似度高(共享内容多)的文档聚集在一起,为分块存储和方便数据管理创造有利条件。

关键词 文档聚类,归档存储,动态区间映射,空间利用率,扩展性
中图法分类号 TP391 **文献标识码** A

Document Clustering Algorithm Based on Dynamic Interval Mapping

SUN Yong-lin LIU Zhong
(School of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract Archival storage is becoming a research hotspot with information digitization accelerating, where space utilization and scalability are very important. Using content-based chunking storage to achieve data deduplication is an effective way to improve storage space utilization, however, it is inefficiency to find shared chunks in all of the huge scale of archival data. We introduced the thought of dynamic interval mapping to information clustering, and presented the DC-DIM(Document Clustering algorithm based on Dynamic Interval Mapping). The algorithm uses chunking and feature extraction methods to generate the feature-set of document, and map it on interval links, then choose the document's storage container according to its feature-set's distribution on interval links. By this way, those documents with high similarity(shared a lot of contents) will be clustered, then, it will be very convenient to improve the space utilization and data management.

Keywords Document clustering, Archival storage, Dynamic interval mapping, Space utilization, Scalability

随着信息数字化的快速发展,出于对法规遵从[1]和信息潜在价值的考虑,迫切需要对内容固定、极少访问、规模巨大、迅猛增长的归档数据进行有效存储,因此归档存储成为研究热点[2-5],空间利用率和扩展性是亟待解决的关键问题。内容分块存储方法[6],通过共享分块单例存储实现重复数据删除,提高存储空间利用率;然而大规模分布式存储环境下,在所有存储内容中寻找共享分块的做法并不可取[7]。如果能够先将内容相似度高的文档聚集在一起,再进行分块存储,将显著提高效率并方便数据管理。Deepavali Bhagwat 等人[8],依据Border's Theory将文档分块并提取特征得到特征集合,依据特征集合的计算结果,将文档存储在 $R(>1)$ 处,这样能以很高的概率保证最相似的两个文档会在某处共同存储,达到一定的文档聚类效果,但文档副本率高和扩展性不好。本文将动态区间映射思想[9,10]引入信息聚类,提出了基于动态区间映射的文档聚类算法(DC-DIM, Document Clustering algorithm based on Dynamic Interval Mapping),通过分块和特征提取方法获得文档的分块特征集合,依据分块特征集合在区间链上的映射分布确定文档的存储服务点(根据存储系统的规模,可以大到局部存储网络也可以小到磁盘分区),实现文档聚类,为通过分块存储方法提高存储空间利用率和方便数据管理创造有利条件。通过理论分析和实验验证表明,算法具有复本率低、聚类效果稳定、检索可控、扩展灵活等优点。

1 基本定义与符号约定

为叙述方便,表1列出了文中用到的符号及其含义。

表1 符号约定

符号	含义	符号	含义
Doc	文档	TTTD	分块算法
Chunk	分块	R_F	特征提取算法
CHUNKs	分块集合	DIM	动态区间映射
ChunkID	分块特征	θ	相似度
CHUNKIDs	分块特征集合	η	特征包含率
Server	存储服务点	α	检索启动阈值
ServerID	存储服务点编号	φ	迁移率
ServerID_{max}	存储服务点最大编号	Θ	相似度阈值向量

到稿日期:2009-07-14 返修日期:2009-10-11 本文受国家自然科学基金(60503042)资助。
孙永林(1984-),男,硕士,主要研究方向为网络存储、密码学,E-mail:leomuyi@126.com;刘 仲(1971-),男,博士,副研究员,主要研究方向为网络存储、并行与分布处理。

归档存储研究方兴未艾，许多名词术语没有明确定义，为消除歧义和方便讨论，表2列出了本文涉及的名词术语的基本定义。下面简单说明各定义的依据或用途。

表2 基本定义

名词术语	基本定义
文档相似度	两文档分块特征集交集的模与并集的模之比
副本率	平均每个文档在存储系统中存储的次数
空间利用率	存储数据量与存储空间开销之比
特征包含率	区间链包含特征数与特征集合元素总数之比
检索启动阈值	控制检索范围的阈值，若某区间链的特征包含率超过此值，将检索其对应存储服务点
相似度阈值向量	划分检索结果的一组相似度数值
扩展迁移率	系统扩展迁移数据量与存储数据总量之比

文档相似度。进行文档聚类，首先要定义文档的相似度。对于归档存储，文档聚类的主要目的是聚集内容相似的文档，以便发现共享内容，提高空间利用率；文档内容的相似程度取决于共享内容的多少，进而取决于共享分块的多少，即：文档的相似程度与共享分块数正相关。经分块和特征提取得到文档分块特征集合，两文档分块特征集合的交集元素数同并集元素数之比即可反映两文档的相似程度，因此将文档相似度定义为分块特征集合交集的模与并集的模之比，即：

$$\theta(Doc_1,Doc_2)=\frac{|CHUNKIDs(Doc_1)\cap CHUNKIDs(Doc_2)|}{|CHUNKIDs(Doc_1)\cup CHUNKIDs(Doc_2)|}$$

副本率。对于存储系统，理想情况是每个文档只存储一次，但诸如安全性、完整性、负载等约束，文档在系统中往往不止存储一次。定义副本率为系统中平均每个文档存储的次数。在满足系统应用需求的条件下，副本率越低越好。

空间利用率。存储系统首要目标就是存储空间的充分利用，副本率在一定程度上反映了系统的空间效率，但不够直接和充分，而存储数据量与存储空间开销之比恰能直接和充分反映系统的空间效率，因此将其作为空间利用率的定义。

特征包含率。指区间链对文档特征集合的覆盖率，由于相似文档的分块特征集合在区间链上的映射分布具有相似性，特征包含率体现了文档在其所对应存储服务点存储的概率，可以作为聚类和检索的判定参数。

检索启动阈值。用户的检索行为受结果获取时间、完整性和准确度等因素影响。通过设定检索启动阈值来控制检索范围，进而控制检索结果获取时间、完整性和准确度等参数。

相似度阈值向量。为了方便分析检索结果随检索启动阈值变化，设置一组依相似度划分检索结果的数值。

扩展迁移率。迁移最优性[9]是衡量系统扩展性的重要指标，因此定义扩展迁移率：系统扩展迁移数据量与存储数据总量之比，来衡量系统扩展的灵活性。

2 基于动态区间映射的文档聚类算法

2.1 算法思想

DC-DIM算法的基本思想是：通过基于内容的分块和特征提取方法生成文档的分块特征集合；利用伪随机数产生器[11]生成均匀分布于区间[0,1)上的伪随机数集合，建立分块特征与伪随机数集合的映射；依据容量、性能等因素对系统存储服务点赋权，依据系统存储服务点的赋权划分[0,1)区间，建立子区间与存储服务点的映射，并将映射到同一存储服务点的子区间链成区间链，得到区间链与存储服务点的一一对应关系；依据文档的分块特征集合在各区间链上的映射分布，选取特征包含率最高的存储服务点存储该文档。将文档聚类问题转化为文档特征集的区间分布问题，特征数越多，聚类的效果越好，有效避免了高维聚类的“维数祸根”[12]问题。

由于相似度高的两个文档的共享分块具有很高的比重，分块特征集合交集将主导两文档的区间映射分布，经过两次映射，相似度高的文档会以很高的概率被同一存储服务点接收，达到了文档聚类的目的，同时副本率接近1，只有在具有最高特征包含率的区间链不唯一时，文档才会在多处存储，而这种情况发生的概率是很小的。

在检索时，同样依据检索样例内容计算分块特征集，只是在区间映射时引入检索启动阈值 α，只有特征包含率高于检索启动阈值时，区间链所对应的存储服务点才会被访问，这样就可以根据系统负载或检索者的需求来调整检索启动阈值，在适当的范围内检索，获取满意的结果。

动态区间映射具有均衡性，当数据规模很大时，文档的聚类分布也具有均衡性，不会出现“热点”和“冷点”现象；动态区间映射具有扩展性和迁移最优性，可以灵活地扩展存储服务点，以适应归档数据迅猛增长的存储需求。

2.2 算法描述

2.2.1 存储过程

如图1所示：输入Doc，算法第一步：用TTTD算法[6]对Doc分块生成CHUNKs；第二步：用Rabin's fingerprint算法[13]计算CHUNKs中分块Chunk的特征ChunkID生成分块特征集CHUNKIDs；第三步：对CHUNKIDs中每个ChunkID进行DIM映射，统计映射分布；第四步：选取特征包含率最多的区间链对应ServerID组成目标存储服务点标识集ServerIDtarget；第五步：发送Doc至相应存储服务点。

算法1 存储过程

```
Input: Doc
Step1:CHUNKs=TTTD(Doc);
Step2:CHUNKIDs=∅;
      for (every Chunk in CHUNKs)
      {     ChunkID=R_fingerprint(Chunk);
            add ChunkID to CHUNKIDs;
      }
Step3:for (ervry ServerID)
            flag [ServerID] = 0;
      for ( every ChunkID in CHUNKIDs)
      {     ServerID=DIMapping(ChunkID);
            flag[ServerID]++;
      }
Step4:ServerIDtarget=
      {     ServerID| ∀ ServerIDx,1≤ServerIDx≤ServerIDmax,
            flag[ServerID]≥flag[ServerIDx]
      };
Step5:for (every ServerID ServerIDtarget)
            Send Doc to Server[ServerID];
```

图1 存储过程

2.2.2 检索过程

如图2所示：输入检索内容DocSearch，算法前三步与存储过程相同；第四步：计算特征包含率 η[ServerID]，若 η[ServerID]达到了检索精确度 α，则将ServerID添加进检索存储服务点标识集ServerID_SEARCH；第四步：检索ServerID_

SEARCH 所标识的存储服务点。

算法 2 检索过程

```
Input:DocSearch
Step1:CHUNKs=TTTD(DocSearch);
Step2:CHUNKIDs=∅;
      for (every Chunk in CHUNKs)
      {     ChunkID=R_fingerprint(Chunk);
            add ChunkID to CHUNKIDs;
      }
Step3:for(ervry ServerID)
            flag [ServerID]=0;
      for (every ChunkID in CHUNKIDs)
      {     ServerID=DIMapping (ChunkID);
            flag[ServerID]++;
      }
Step4:for(ervry ServerID )
            η[ ServerID ] =flag[ServerID]/|CHUNKIDs|;
ServerID_ SEARCH={ServerID|η [ServerID]≥α } ;
Step5:for (every ServerID in ServerID_SEARCH)
            Search similar documents in Server [ServerID];
```

图 2 检索过程

2.2.3 系统扩展过程

如图 3 所示：输入扩展节点数 AddServerNum，第一步：更新区间映射表示并初始化 ServerID 为 1；第二步：对 Server [ServerID]中存储的每个文档 Doc，在新的区间映射表示下，计算目标存储服务点标号集 ServerIDtarget，并依据集合 {ServerID} 与 ServerIDtarget 的包含关系更新迁移文档集 NeedMoveDocs 和删除文档集 NeedRemoveDocs；第三步：迁移 NeedMoveDocs 中的文档并删除 NeedRemoveDocs 中的文档；第四步：将 ServerID 加 1，判断是否所有旧存储服务点都已处理，是则结束，否则转第三步。

算法 3 系统扩展过程

```
Input: AddServerNum
Step1:updateIntervalMapping (AddServerNum);
      ServerID = 1;
Step2:for (every Doc stored in Server [ServerID])
      {     compute ServerIDtarget for Doc;
            if (ServerIDtarget={ServerID})
                  continue;
            else
            {     add Doc to NeedMoveDocs;
                  if ( ServerID∉ ServerIDtarget )
                        add Doc to NeedRemoveDocs;
            }
      }
Step3:for (every Doc in NeedMoveDocs)
            move (Doc);
      for (every Doc in NeedRemoveDocs)
            remove (Doc);
Step4:ServerID++;
      if (ServerID>OldServerNum)
          exit();
      else goto Step2;
```

图 3 系统扩展过程

2.3 算法分析

2.3.1 聚类效果分析

通过修改(增加或删除)文档小部分内容，分析聚类结果。

前提约定：

①|CHUNKs(Doc)|=|CHUNKIDs(Doc)|=N；

②$\eta=\{\eta_1,\cdots,\eta_{first},\cdots,\eta_{second},\cdots,\eta_{ServerIDmax}\}$：$\eta_{first}$ 和 η_{second} 为最高和次高特征包含率；

③$|temp|<|Chunk_{min}|$：temp 为修改内容，$Chunk_{min}$ 为最小分块；

④Add(Doc,temp)：添加 temp 进 Doc；Sub(Doc,temp)：删除 Doc 中的 temp；

⑤$InLink_x$：η_x 所对应的区间链；

⑥num[x]：$InLink_x$ 包含文档分块特征的个数；

⑦ChunkID(Chunk)：Chunk 的特征值。

动作：Add(Doc,temp)

由于 temp 的长度不足以单独产生一个分块，增加动作后有两种情况。

(1)情况一：temp 恰好插入两分块间，$Chunk_{follow}$ 为紧随其后的分块。

在这种情况下有两种可能结果：1) temp 同 $Chunk_{follow}$ 合并产生新的分块 $Chunk_{new}$；2) $Chunk_{follow}$ 分裂，前半部分同 temp 合并产生新的分块 $Chunk_{new1}$，后半部分产生新的分块 $Chunk_{new2}$。对于 1)，仅当 num[first]≤num[second]+2，且 $ChunkID(Chunk_{follow})\in InLink_{first}$ 与 $ChunkID(Chunk_{new})\in InLink_{second}$ 至少有一个满足，才有可能使得 $\eta_{first}\leqslant\eta_{second}$。对于 2)，仅当 num[first]≤num[second]+3，且以下 3 个条件 $ChunkID(Chunk_{follow})\in InLink_{first}$，$ChunkID(Chunk_{new1})\in InLink_{second}$ 以及 $ChunkID(Chunk_{new2})\in InLink_{second}$ 至少满足一个时，才有可能使得 $\eta_{first}\leqslant\eta_{second}$。除此之外，$\eta_{first}>\eta_{second}$。

(2)情况二：temp 插入某一分块 $Chunk_{old}$ 内部。

在这种情况下也有两种可能结果：1) temp 同 $Chunk_{old}$ 合并产生新的分块 $Chunk_{new}$；2) $Chunk_{old}$ 同 temp 先合并，再分裂产生新的分块 $Chunk_{new1}$ 和 $Chunk_{new2}$。无论 1)或 2)都与情况一的结论相同。

由于分块的大小被设定在一个范围内，temp+$Chunk_{old}$(或 $Chunk_{follow}$)的长度一般情况下小于分块上限；即使超出上限，temp+$Chunk_{old}$(或 $Chunk_{follow}$)被硬截断，并且截剩部分小于分块下限，则截剩部分再与后面分块合并的长度几乎不可能再次超过分块上限，于是分裂终止，分裂链式反应的情况不会发生。

经以上分析，Add(Doc,temp)后，仅当 η_{first} 与 η_{second} 相差无几，且新增减的分块的特征值的分布恰好在 $InLink_{first}$ 与 $InLink_{second}$ 间此消彼长时，才会有 $\eta_{first}\leqslant\eta_{second}$ 的变化。据此得出结论：文档增加一小部分内容对文档区间映射的结果产生的影响非常小。

对于 Sub(Doc,temp)，经分析有类似的结论：文档减少一小部分内容对文档区间映射的结果产生的影响非常小。

此外，即使对文档的修改致使出现 $\eta_{first}\leqslant\eta_{second}$ 的变化，在文档检索的过程中，η_{first} 和 η_{second} 所对应的存储服务点仍然在最先考虑的检索范围内，因此对检索结果几乎没有影响。

综合以上分析，得到结论 1。

结论 1 文档的微小改动几乎不影响聚类结果。

随着改动的增多，新文档分块特征集合的区间映射分布与旧文档的分块特征集合的区间映射分布的差异逐渐显著，同时新旧文档的相似度也随之降低，这也反映了依据分块特征集合的区间映射分布进行文档聚类的连续稳定性。

2.3.2 检索性能分析

通过调整检索启动阈值，观察检索结果，分析检索启动阈值在检索中的作用，以及设定检索启动阈值的参考因子。

前提约定：

①N：系统当前存储服务点总数；

②系统中已经存储了足够多的文档，即：检索结果数足够反映检索启动阈值对检索结果分布的控制作用；

③η_i(CHUNKIDs(DocSearch))：区间链 $InLink_i$ 对检索内容 DocSearch 的分块特征集的特征包含率；

④η_{max}，η_{min}最大和最小特征包含率；

⑤检索结果集 Results={Doc|θ(Doc,DocSearch)$\geqslant\Theta_{min}$，Doc$\in$Server[i]，η_i(CHUNKIDs(DocSearch))$\geqslant\alpha$}，其中 Θ_{min} 为 Θ 中最小的相似度阈值；

⑥Θ(Results)={R_1，…，R_i，…，R_{top}}：经 Θ 划分的结果集；

⑦Search(DocSearch，α，Θ)：在检索启动阈值 α 的控制下检索与 DocSearch 相似的文档并依据 Θ 划分结果集。

动作：Search(DocSearch，α，Θ)

先分析两种极端情况，若 $1\geqslant\alpha>\eta_{max}$，则 Results=$\emptyset$；若 $0\leqslant\alpha\leqslant\eta$ min，则 Results={Doc|θ(Doc，DocSearch)$\geqslant\Theta_{min}$}。可以看出，如果 α 设置得太高，将不会检索任何存储服务点，得不到任何检索结果；如果 α 设置得太低将会检索所有存储服务点，并返回所有满足相似度最低要求的结果。理想情况是检索尽可能少的存储服务点，返回尽可能多的与 DocSearch 相似度高的文档。相似度高的文档意味着少量修改即可从一个文档得到另一个，依据结论 1，有结论 2。

结论 2 对 CHUNKIDs(DocSearch) 包含率最高的少数几个区间链对应的存储服务点，存储了绝大多数与 DocSearch 相似度高的文档。

2.3.3 扩展性分析

通过扩展系统规模，分析系统扩展的难易程度，以及系统扩展对其它性能的影响。

前提约定：

①旧区间链 $InLink_{old}$={$Inlink_1$，…，$Inlink_i$，…，$Inlink_N$}；

②假定各存储服务点的权值都为 1；

③Extent(N，M)：N 个存储服务点扩展 M 个。

动作：Extent(N，M)

对于每个 $Inlink_i$($1\leqslant i\leqslant N$)，截取长度为 $1/(M+N)$生成新的 $Inlink_i$'，剩余部分归并成长度为 $M/(M+N)$ 的区间链 $Inlink_{temp}$，再将其 M 等分，依次赋给 $Inlink_j$($N+1\leqslant j\leqslant N+M$)，得到新的区间链 InLink'={$Inlink_1$'，…，$Inlink_i$'，…，$Inlink_N$'，$InLink_{N+1}$，…，$InLink_j$，…，$InLink_{N+M}$}。对 Server[$i$]中的每个文档，重新依据 $Inlin_k$'进行区间映射确定是否迁移以及迁移的目标。一般情况 $M<N$，对于当前分析的 Server[i]中的文档 Doc，在 $Inlink_{old}$ 中 $\eta_i=\eta_{max}$，扩展只分割了 $Inlink_i$ 的小部分并分散到若干 $Inlink_j$'中，在 InLink'中 $\eta_i'=\eta_{max}'$的概率仍很高，Doc 很可能仍留在原存储服务点。对于 $M>N$，系统更像是重组而非扩展，迁移量已不再重要，但分析结果表明，扩展后 Doc 留在原处的概率仍很高。经以上分析得到结论 3。

结论 3 系统扩展灵活，迁移率低。

在 Deepavali Bhagwat 等人的方法中，文档存储在 R(>1)处，即副本率至少为 2；而且系统的存储服务点总数 M 固定，如果将 M 初始值设置较大，则存在明显的资源浪费，虽然各存储服务点的存储能力可以扩展，但总会受到限制。

综上所述，结论 1 表明 DC-DIM 算法具有显著和稳定的聚类效果，副本率接近 1；结论 2 表明系统的检索具有很好的可操控性；结论 3 表明系统扩展灵活，迁移率低。

3 实验与性能分析

本文的研究背景是归档数据存储管理，实验数据选取具有归档数据典型特征的 Linux 内核源码。设计了测试副本率和空间利用率变化的聚类效果实验、测试检索结果分布变化的检索性能实验、测试系统扩展迁移率变化的系统扩展实验。

3.1 聚类效果实验

在存储数据量分别达到 10，20，30，40，50，60，70，80，100，120，140，160，180，200(M)时，统计存储文档数、文档副本数、空间开销，并计算副本率和空间利用率。

如图 4 所示，副本率和空间利用率随存储数据量的变化，副本率始终稳定在 1.75，较 Deepavali Bhagwat 等人的方法(>2)有显著降低；空间利用率随着存储数据量的增加稳定增长并逐渐呈现线性增长趋势，说明 DC-DIM 算法结合分块存储能够显著稳定地提高存储空间利用率，聚类效果好。

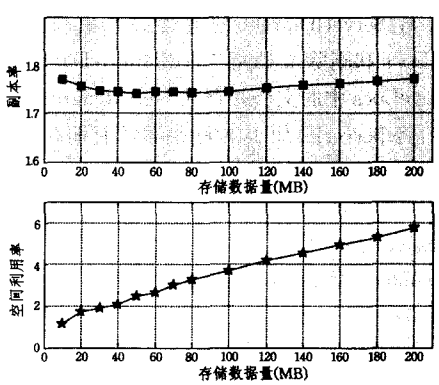

图 4 副本率和空间利用率变化图

3.2 检索性能实验

选取检索文档 DocSearch，设定相似度阈值向量 Θ=(0.2，0.3，0.4，0.5，0.6，0.7，0.8)，在存储数据量达到 200(M)时，设定不同的检索启动阈值 α={0，0.04，0.06，0.08，0.1，0.12，0.14，0.16}。

图 5 是检索结果分布变化图，检索结果随待启动阈值的提高从最低的相似度区间开始丢弃并逐步向高相似度区间蔓延，最高相似度区间的结果数恒定并最终突然变为 0，这表明文档聚类效果的显著性和稳定性并验证了结论 2，通过调整检索启动阈值，在尽可能小的范围内检索到尽可能多的相似度高的结果，提供了灵活的检索策略。在 Deepavali Bhagwat 等人的方法中，需要根据检索样例文档的计算结果检索 R(≥2)个存储服务点才能检索到最相似的文档；而 DC-DIM 方法只需检索相似度最高的存储服务点(通常只有 1 个)，并且还可以通过调整检索启动阈值获取整体相似度逐步降低的相似

文档。需要说明的是，检索启动阈值的调整范围受存储规模和存储对象大小的影响，如本实验中 Linuxl. 1 内核源文件小，存储规模也小，致使阈值调整范围比较小，在应用中应当注意。

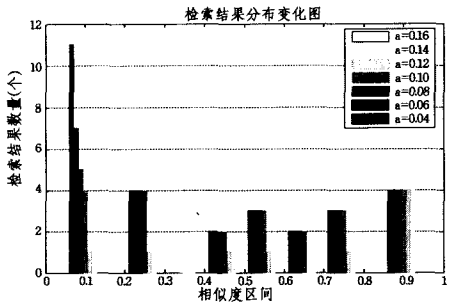

图 5　检索结果分布变化图

3.3　系统扩展实验

系统初始存储服务点 10 个，选取 500 个文档存入系统，连续扩展 3 次，每次扩展 5 个节点，统计迁移副本数和数据量，计算文档和数据迁移率，实验结果如表 3 所列。

表 3　系统扩展实验数据

系统扩展	系统副本数	迁移副本数	空间开销	迁移数据量
10	839	—	5.871M	—
10—>15	898	105	6.138M	1.113M
15—>20	1035	307	6.527M	1.539M
20—>25	1174	878	6.898M	3.969M

图 6 是文档和数据迁移率变化图，前两次扩展，文档和数据迁移率都很低，验证了结论 3；但第三次扩展，迁移率出现了跃升，其原因在于 Linuxl. 1 内核源码存在大量几 kB 小文档，每个小文档的分块总数一般不足 20，其分块特征集合的映射分布很不稳定，当第三次扩展时，这种不稳定性凸显出来，在各存储服务点间出现频繁的小文档迁移，引起迁移率跃升。而空间开销的增值(6.898M—6.527M) 与前两次扩展相比没有显著差异，说明频繁的小文档迁移不会明显影响系统的空间利用率。在实际应用中，可以针对数据源特征，采取措施降低迁移率；或置之不理(既然不会明显影响空间利用率，况且系统的扩展也不经常进行)。

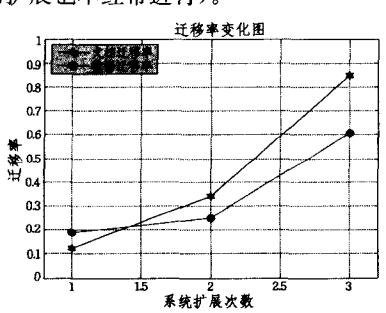

图 6　迁移率变化图

3.4　结果分析

聚类效果实验和检索性能实验表明：DC-DIM 算法具有显著和稳定的文档聚类效果，副本率低(1.75 左右)，结合分块存储能够稳定地逐步提高存储空间利用率；检索性能实验结果表明：通过调整检索启动阈值，可以在尽可能小的范围内检索到绝大多数想要的结果，检索灵活可控；系统扩展实验表明：系统扩展灵活，迁移率低，即使扩展过程中因数据源文档大小特征，出现频繁的小文档迁移而引起迁移率跃升，也可不予理睬，实际上这并不会给空间利用率带来影响。

结束语　本文将动态区间映射思想引入文档聚类，提出了基于动态区间映射的文档聚类算法 DC-DIM，通过分块和特征提取方法生成文档分块特征集合，依据分块特征集合在区间链上的映射分布确定文档的存储位置，实现文档聚类，为分块存储和方便数据管理创造了有利条件，理论分析和实验验证表明算法复本率低，聚类效果稳定，检索可控，扩展灵活，较 Deepavali Bhagwat 等人的方法，在副本率和检索性能方面都有明显优势，并克服了其扩展性的缺陷。

参 考 文 献

[1] Sarbanes-Oxley. http://www.sarbanes-oxley.com/index.php

[2] Storer M W. Secure, Energy-Efficient, Evolvable, Long-Term Archival Storage[R]. UCSC-SSRC. 2009

[3] Bradshaw P L, Brannon K W, et al. Archive storage system design for long-term storage of massive amounts of data[J]. IBM J. Res. & Dev., 2008, 52(4/5)

[4] You L L, Pollack K T, Long D D E. Deep Store: An Archival Storage System Architecture[C]// Proceedings of the 21st International Conference on Data Engineering(ICDE '05). 2005

[5] You L L, Karamanolis C. Evaluation of efficient archival storage techniques[C]// Proceedings of the 21st IEEE/12th NASA Goddard Conference on Mass Storage Systems and Technologies. 2004

[6] Eshghi K, Tang H K. A Framework for Analyzing and Improving Content-Based Chunking Algorithms[R]. Hewlett Packard Labs Technical Report TR. 2005

[7] Forman G, Eshghi K, Chiocchetti S. Finding similar files in large document repositories[C]// Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining. 2005: 394-400

[8] Bhagwat D, Eshghi K, Mehra P. Content-based Document Routing and Index Partitioning for Scalable Similarity-based Searches in a Large Corpus[C]// Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD '07). 2007: 105-112

[9] 刘仲，周兴铭. 基于动态区间映射的数据对象布局算法[J]. 软件学报，2005，16(11)：1886-1893

[10] 刘仲，周兴铭. 可伸缩分布式动态区间映射算法[J]. 计算机学报，2006，10：1757-1763

[11] Matsumoto M, Nishimura T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator[J]. ACM Transactions on Modeling and Computer Simulation(TOMACS), 1998, 8(1): 3-30

[12] 朱玉全，杨鹤标，孙蕾. 数据挖掘技术[M]. 南京：东南大学出版社，2006：130-168

[13] Chan C, Lu H. Fingerprinting using Polynomial(Rabin's method)[Z]. CMPUT690 Term Project, 2001