文章编号:1000-582X(2008)06-0658-05

基于 3-gram 模型和数据挖掘技术的元数据预取

李学明,唐相桢
(重庆大学 计算机学院,重庆 400030)

摘 要:在大规模的文件存储系统中,针对大多数算法的设计没有考虑到元数据访问的特征与元数据本身较小的特点,提出了一种利用存储系统中的元数据操作日志文件,运用 3 元(3-gram)预测模型和数据挖掘的方法对用户未来可能要操作的元数据进行组预取。实验证明,对于从日志文件中提取出的文件元数据访问序列,新预取模式的缓存(Cache)命中率与基于权重图的预取算法(NEXUS)相比平均提高了 3.9%,与最近最少使用算法(least recently used,LRU)比较平均提高了 16%。

关键词:元数据;*n*-gram 模型;数据挖掘;组预取
中图分类号:TP311 文献标志码:A

Metadata prefetching based on 3-gram model and datamining technology

LI Xue-ming, *TANG Xiang-zhen*
(College of Computer Science, Chongqing University, Chongqing 400030, P. R. China)

Abstract: An efficient and accurate metadata-oriented prefetching scheme is critical for achieving the best metadata service performance in large distributed storage systems. Most previously developed algorithms, however, do not consider the characteristics of metadata, such as small size. In light of this issue, we presented a 3-gram based model and datamining technique to use the traces of metadata accessing to prefetch groups of users' future metadata operations. By using large trace-driven simulations in which our new prefetching scheme was adopted, it is shown that the hit rate for metadata access extracted from the traces can be increased by up to 3.9% and 16 % compared with NEXUS and least recently used(LRU), respectively.

Key words: metadata; *n*-gram model; data mining; prefetch groups

在传统的文件存储系统中,文件系统元数据与文件数据是由相同的设备统一管理和存储的[1]。由于有些文件操作无需涉及它所包含的数据,而只与文件属性或者目录数据等元数据信息相关,所以在现代存储系统中为了减轻 I/O 瓶颈问题所带来的压力,把数据存储在通过网络能直接访问的设备上,元数据由一个或多个专门的元数据服务器来进行管理,不失为一种有效的方法[2-3]。文件系统元数据与文件数据的分离管理和存储,可以更好地利用传输网络和相应存储设备的特性,根据需要对 2 种类型的服务器进行优化,从根本上减轻服务器的负载,从而提高用户文件操作的性能。

针对这种新的存储体系结构,以前的研究主要是采用廉价磁盘冗余阵列(RAID)条带[4]、缓存[5]、调度管理[6]、网络[7]等技术来优化文件数据管理的有效性与可扩展性,然而却忽略了文件元数据管理

收稿日期:2008-02-16
基金项目:重庆市自然科学基金资助项目(CSTC2005BB2190);重庆市自然科学基金资助项目(CSTC2007BB2178)
作者简介:李学明(1967-),男,重庆大学副教授,主要从事数据挖掘、神经网络、网络与网格计算研究,(Tel)13983137687;(E-mail)lixueming@cqu.edu.cn。

的可扩展性。事实上，在大规模的文件系统中针对元数据的 I/O 操作超过了半数。特别是在数据量达到 2 的50 次方字节（Petabyte）级、元数据量达到 2 的 40 次方字节（Terabyte）级时，元数据访问性能的提高对于 I/O 的吞吐量和可扩展性有非常重要的作用。

采用缓存和预取机制是提高 I/O 操作性能的 2 种有效途径，因而在文件数据的管理中经常同时采用这两种方法。可是将它们应用到元数据预取时却面临 2 个方面的问题：一是预取操作是在分布式环境下进行的；由于在大多数的文件系统中存在元数据负载歪斜的问题，故在大规模存储系统中元数据管理系统性能不会随 I/O 负载的减轻而成比例提高，元数据服务的负载可能分散在一组元数据服务器中。二是现存的缓存和预取算法[8-10]可能并不适合应用于元数据预取。因为这些算法的设计没有考虑到元数据访问的特征与元数据本身较小的特点，往往只保守地预取一个数据。由于元数据具有大小较小且数量多的特点，在预取时即使预取了错误的元数据，其所付的代价要比文件数据预取出错小很多，故贪心的成组预取很适合应用于文件元数据管理。为了使元数据服务性能达到最优，设计一种充分体现元数据特点的缓存和预取的新算法势在必行。

当前最新的 NEXUS 算法[11]对元数据管理进行过相应的研究，但由于它在预取时组大小只取了 2 个元数据，没能很好地利用元数据可以成组预取的特性，浪费了缓存（Cache）空间资源。故做此研究的目的是充分利用 Cache 空间，提高 Cache 命中率，从而实现元数据组预取 。

笔者利用 n-gram[12-13]中的 3-gram 预测模型和数据挖掘相结合的方法找出最合理的预取规则，实现文件元数据组预取。实验结果表明：该算法与 NEXUS 算法相比，基于单个用户的 Cache 命中率平均提高了 3.9%，与 LRU 算法比较提高了 16%。

1　NEXUS 算法介绍

NEXUS 算法是一种基于加权图的预取算法，通过调节预测窗口的大小，它不但可以找出 2 个相邻元数据之间的关系，还可以找出预测窗口影响范围内的 2 个元数据间的关系。在构造加权图时，采用了线性递减的分配权值的方法，2 个元数据离得越近，关系越强，权值也越大。新的元数据请求到来时，如果加权图中已经存在该元数据节点，则修改与其相应的权重，否则把该节点加入加权图中并生成相应的权重。测试中，如果预取的组大小为 n，那么当出现新的元数据请求而未命中 Cache 时，则在与该元数据节点相邻的节点中选取权重较大的 n 个元数据进行预取。在预测窗口大小为 5、预取的组大小为 2 时，NEXUS 算法的 Cache 命中率达到最高。事实上，因为元数据很小，仅为文件数据的 6.25%，NEXUS 算法预取的组大小仅为 2，明显没有很好地利用 Cache 空间资源。

2　新的文件元数据预取模式

新的文件元数据预取模式算法的基本思想是产生所有长度为 3 的元文件组（3-gram），同时利用数据挖掘技术产生长度大于 3 的闭集频繁访问组，从中选出频度最大的若干组构成规则，各规则合并产生最后要进行预取的组。

2.1　闭集频繁访问组的关联规则挖掘

定义 1　在某个时间段内被频繁访问的文件元数据组 $\mathrm{Fre}(G)\geqslant \mathrm{MinFre}$，其中 $\mathrm{Fre}(G)$ 是 G 在文件元数据访问序列中出现的频度，通常 MinFre 是一个预先给定的阈值，G 被称为频繁访问组。

定义 2　若 G 是一个频繁访问组，当不存在频繁访问组 G_S，其中 G_S 包含 G，且 $\mathrm{Fre}(G_S)=\mathrm{Fre}(G)$，则 G 又被称为闭集频繁访问组。

把由闭集频繁访问组产生的关联规则作为对文件元数据进行预取的前提条件。

2.2　*n*-gram 预测模型中参数 *n* 的选择

受自然语言中 n-gram 语言模型的启发，利用 n-gram的概率预测模型对元数据访问请求进行预测。假定文件元数据请求访问序列 $S=M_1M_2M_3M_4\cdots M_k\cdots$，由于链式规则，$P(S)=p(M_1)\cdot p(M_2|M_1)p(M_3|M_1M_2)\cdots p(M_k|M_1M_2\cdots M_{k-1})\cdots$，对 $p(M_k|M_1M_2\cdots M_{k-1})$ 而言，$M_1M_2\cdots M_{k-1}$ 为历史；为了便于计算，历史不能太长，一般只考虑 n-1 个元数据构成的历史，即 $p(M_k|M_{k-n+1}\cdots M_{k-1})$，考虑前面 n-1 个元数据构成历史的模型即为 n-gram 模型。模型的元数越高，预测的准确度就越高，但计算的代价也越大。笔者用 2-gram、3-gram、4-gram 模型分别对 HP 文件系统日志文件[14]中的元数据进行预取，图 1给出了 3 个模型在 Cache 大小不同的情况下的 Cache 命中率的变化情况，可以看出 3-gram、4-gram 模型的 Cache 命中率基本相同，但明显高于 2-gram 模型。

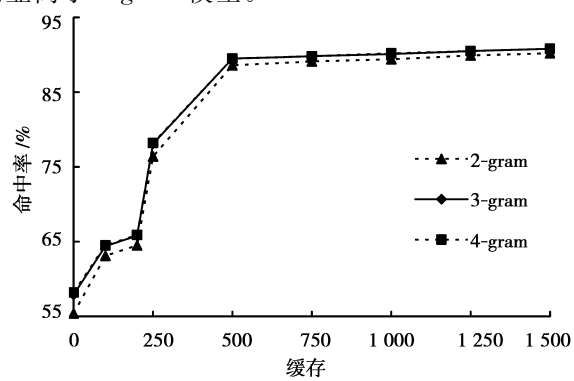


图 1　不同缓存下命中率的变化情况

以模型的计算代价和Cache命中率2个方面为评价标准，这种新的文件元数据预取模式最适合采用3-gram模型。

2.3 模型的构造

在进行数据挖掘之前，首先把文件元数据请求构成的序列中的每个不同的元数据请求生成对应一棵树，然后对文件元数据请求构成的序列进行分组。设预测窗口大小为 w，当有新的元数据请求到来时，用新来的元数据请求代替在预测窗口中最旧的元数据请求，从而形成新的组 G。由于采用了3-gram模型，故对于每个组 G，固定前面2个文件元数据请求，且作为有序的序列，后面的 $w-2$ 个作为无序序列。分组完成后开始对生成的组进行冗余处理，去掉无序序列中相同的元数据请求以及与有序序列相同的元数据请求。把经过处理的组逐个添加到相应的树上构成访问树(其中节点存有支持度记数)。从每棵树中找出频度最大的若干个闭集频繁访问组构成规则，各规则合并生成最后要进行预取的组。下面给出预测模型构造的算法描述。

算法 PredictionModelConstruction ()

// 设元数据请求序列 $M=m_1m_2m_3\cdots m_i\cdots m_n$，$n$ 为序列中元数据请求的总数。

1 L 为空链表；
2 for M 中的每个元数据请求，do
3 if(m_i不在链表 L 中)
4 将 m_i 加入表 L 中；
5 end for
6 for $i\leftarrow 1$ to $n-1$
7 $G_i=m_im_{(i+1)}\cdots m_{(i+w-1)}$；// 预测窗口的大小为 w；
8 $G_i\leftarrow$filter(G_i)；// filter 函数的功能：组 G_i 中前2个项固定，去除 G_i 中相同的项，处理后的 G_i 可以表示为 $m_im_{(i+1)}\cdots m_k$，$i+1\leqslant k\leqslant i+w-1$；
9 组 G_i" $\leftarrow m_im_{(i+1)}+$subset($m_{(i+2)}\cdots m_k$)；// 组 G_i 前2个项固定不变，加上组 G_i 后面剩余的项的子集合，分别保存到 G_i"；
10 for 每个组 G_i"，do// 构造访问树；
11 找出链表 L 中的节点 m_i；
12 if(节点 m_i 的子节点中不含节点 $m_im_{(i+1)}$)
13 添加节点 $m_im_{(i+1)}$ 到节点 m_i 下，形成它的另一个孩子节点；
14 else；
15 节点 $m_im_{(i+1)}$ 的支持度增加1；
16 while $j\leqslant$length(G_i") // j 初始值为3；
17 找出节点 $m_im_{i+1}\cdots m_{j-1}$；
18 if(节点 $m_im_{i+1}\cdots m_{j-1}$ 的子节点不含节点 $m_im_{i+1}\cdots m_j$)
19 添加节点 $m_im_{i+1}\cdots m_j$ 到节点 $m_im_{i+1}\cdots m_{j-1}$ 下，形成它的另一孩子节点；
20 else
21 节点 $m_im_{i+1}\cdots m_j$ 的支持度增加1；
22 $j++$；
23 end while
24 end for
25 end for
26 rule(Group1, Group2, …) $\leftarrow$ MaxGroups (SubTree T)；// 针对每棵子树，比较第二层节点以下的每个节点，比较它们的支持度，选出几个支持度最高的闭集频繁访问组；
27 将规则放入子树的第二层节点里；// 算法结束。

下面举例说明模型构造过程，设有文件元数据请求序列 ABCADEFABEF，预测窗口大小为8，产生的组如下：

组1 AB(CADEFA) $\xrightarrow{\text{经过冗余处理后}}$ AB(CDEF)；
组2 BC(ADEFAB) $\xrightarrow{\text{经过冗余处理后}}$ BC(ADEF)；
组3 CA(DEFABE) $\xrightarrow{\text{经过冗余处理后}}$ CA(DEFB)；
组4 AD(EFABEF) $\xrightarrow{\text{经过冗余处理后}}$ AD(EFB)；
组5 DE(FABEF) $\xrightarrow{\text{经过冗余处理后}}$ DE(FAB)；
组6 EF(ABEF) $\xrightarrow{\text{经过冗余处理后}}$ EF(AB)；
组7 FA(BEF) $\xrightarrow{\text{经过冗余处理后}}$ FA(BE)；
组8 AB(EF) $\xrightarrow{\text{经过冗余处理后}}$ AB(EF)；
组9 BE(F) $\xrightarrow{\text{经过冗余处理后}}$ BE(F)；
组10 EF $\xrightarrow{\text{经过冗余处理后}}$ EF。

由于篇幅关系，不尽详述，以文件元数据A为例，其所生成的访问树如图2所示。

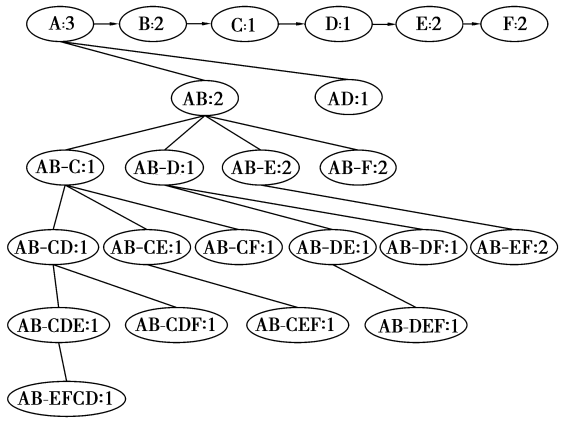


图2 文件元数据A所生成的子树

对应的闭集频繁访问组为：AB－EF：2，AB－EFCD：1。

从已经找到的闭集频繁访问组中选出频度最大

的若干组构成规则，各规则合并生成最后要预取的组，并将它放到第二层相应的节点中，以便测试时快速进行预取。对于上述给出的例子，当选取规则的数量为 1 时，选取的组为 AB-EF，最后的预取组为 AB－EF，将 EF 放到第二层 AB 节点中；当选取规则的数量为 2 时，选取的组为 AB－EF、AB－EFCD，最后的预取组为 AB－EFCD，同样把 EFCD 放入第二层 AB 节点中。

2.4　预测算法

基于 2.3 中构造的预测模型，预测算法描述如下：

算法 Prediction（S：用户的元数据访问请求序列）// S= $s_1 s_2 s_3 \cdots s_i \cdots$；

1　if（s_i不在 Cache 里面）

2　将 s_i加入 Cache 中；

3　找出链表 L 中的 s_{i-1}节点及 s_{i-1}节点的孩子节点 s_{i-1} s_i；

4　取出节点 s_{i-1} s_i 中的规则，将规则放入组 Prediction；

5　将组 Prediction 加入 Cache 中，剔除掉 Cache 中与 Prediction 重复的项；

6　return（“prediction”）；

7　else

8　命中率 Hitcache 加 1；

9　end if

10　return（“no prediction”）

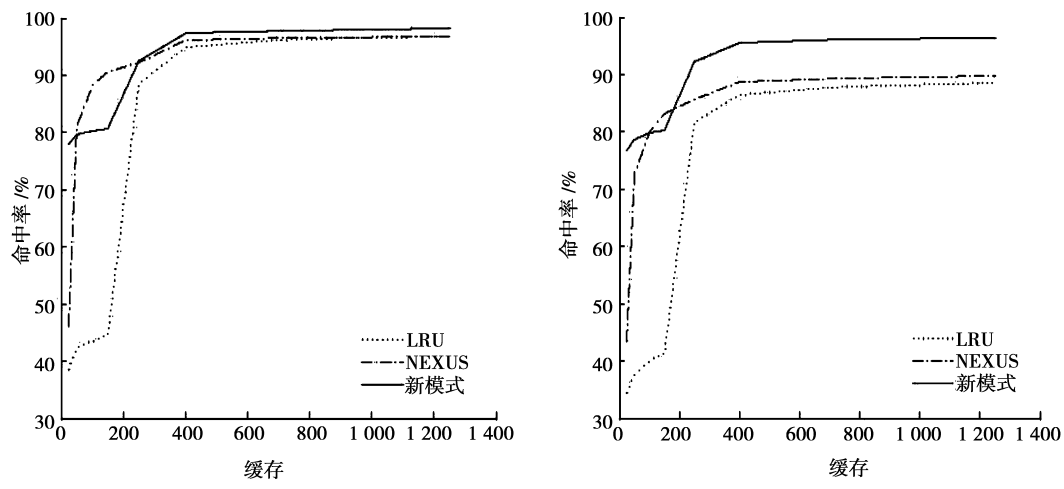
对于 2.3 中的实例，当测试序列为 ABE 时，当 A 在 Cache 中，而 B 不在时，首先搜索访问树，产生 AB 的预取组为 EF 或 EFCD，则对于 B 而言，是必须的系统访问，对 EF 或 EFCD 而言，相当于进行预取。

3　实验结果

笔者使用的数据文件来自于 2001 年 11 月惠普（HP）实验室收集的 HP 文件系统日志[14]，这些日志文件从总大小为 500 GB 的一个分时服务器中收集了 10 d 的文件数据和文件元数据的 I/O 操作事件。在仿真实验中，过滤掉文件数据的 I/O 操作事件，保留了文件元数据的部分。

从用户的角度来看，总的 Cache 命中率包括 3 个部分：本地命中（客户端 Cache 命中）；远程命中（元数据服务器命中）；协作缓存命中。而本地命中最能反映预取算法效率的高低[15]。由于预取的准确度不能直接反映系统的性能，所以比较“Cache 命中率”这个更能直接反映系统性能的指标。

从经过处理后的日志文件中分离出不同用户的文件元数据操作日志数据，从而使用 LRU、NEXUS 和这种新的文件元数据预取算法来对该数据分别测试，由于篇幅有限，选取其中的元数据操作最频繁的 2 个用户，在预测窗口大小为 7 时，比较情况如图 3 所示。



(a)用户一　　(b)用户二

图 3　两个用户的缓存命中率比较

不同算法模型下单个用户的平均 Cache 命中率比较结果如图 4 所示。

从图 4 中可以得出，虽然在 Cache 小于 200 时，这种新算法的 Cache 命中率有一小段略低于 NEXUS 算法，但综合比较新的文件元数据预取算法的平均 Cache 命中率以 3.9％优势明显高过 NEXUS 算法，且与 LRU 算法相比，高出了 16％。

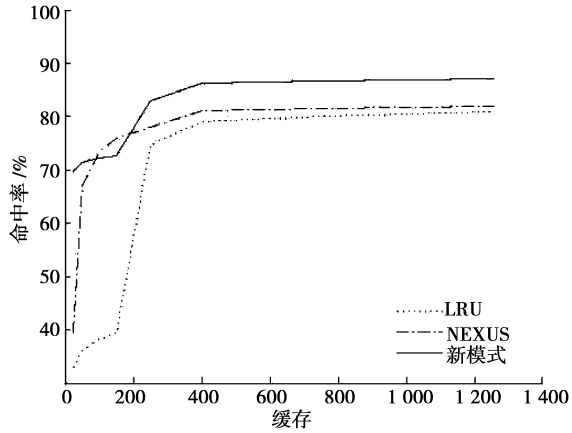


图4 单个用户的平均缓存命中率比较

4 结 语

笔者提出了一种利用存储系统中的元数据操作日志文件,运用3-gram预测模型和数据挖掘相结合的方法对用户未来可能要操作的元数据请求进行成组预取,其Cache命中率与NEXUS算法相比平均高出3.9%,与LRU算法相比高出16%。实验结果表明了模型的有效性。

参考文献:

[1] 王召福,章文嵩,刘仲. LCFS中元数据服务器的可靠性分析模型[J]. 计算机工程与科学, 2005,27(5):54-58.
WANG ZHAO-FU, ZHANG WEN-SONG, LIU ZHONG. A reliability anaysis model of the meta-data server in LCFS[J]. Computer Engineering & Science, 2005,27(5):54-58.

[2] ZHU Y,JIANG H,WANG J. Hierarchical bloom filter arrays(hba): a novel, scalable metadata management system for large cluster-based storage[C]// 2004 IEEE International Conf. California:IEEE Computer Society, 2004:165-174.

[3] 庞丽萍,何飞跃,岳建辉,等. 并行文件系统集中式元数据管理高可用系统设计[J]. 计算机工程与科学,2004,26(11):87-88.
PANG LI-PING, HE FEI-YUE, YUE JIAN-HUI, et al. High availability system design of centralized metadata management for parallel file systems[J]. Computer Engineering & Science,2004,26(11):87-88.

[4] 谢长生,刘艳,李怀阳,等. 基于分条单元热度的RAID数据分布优化[J]. 计算机科学, 2006,33(3):275-278.
XIE CHANG-SHENG, LIU YAN, LI HUAI-YANG, et al. RAID data distribution optimization based on stripe unit heat[J]. Computer Science, 2006,33(3):275-278.

[5] HU Y, YAN Q. Rapid-cache: a reliable and inexpensive write cache for high performance storage systems[J]. IEEE Transactions on Parallel and Distributed Systems,2002,13(3):290-307.

[6] THYAGARAJ T, SIVARAMA D. An efficient adaptive scheduling scheme for distributed memory multicomputers[J]. IEEE Transactions on Parallel and Distributed Systems,2001,12(7):758-768.

[7] ATHICHA M,CHEN B,DAVID M. A low-bandwidth network file system[J]. ACM SIGOPS Operating Systems Review, 2001,35(5):174-187.

[8] ALEXANDROS N,DIMITRIOS K,YANNIS M A. Data mining algorithm for generalized web prefetching[J]. IEEE Transactions on Knowledge and Data Engineering, 2003,15(5):1155-1169.

[9] 苏中,马少平,杨强,等. 基于Web-Log Mining的N元预测模型[J]. 软件学报,2002,13(1):136-141.
SU ZHONG,MA SHAO-PING,YANG QIANG, et al. An *n*-gram prediction model based on web-log mining[J]. Journal of Software,2002,13(1):136-141.

[10] 刘爱贵,陈刚. 一种基于用户的LNS文件预测模型[J]. 计算机工程与应用,2007,43(29):14-16.
LIU AI-GUI,CHEN GANG. User-based last n successors file prediction model[J]. Computer Engineering and Applications, 2007,43(29):14-16.

[11] GU P, ZHU Y, JIANG H, et al. NEXUS: a novel weighted-graph-based prefetching algorithm for metadata servers in petabyte-scale storage systems[C]// Sixth IEEE International Symposium on CCGrid2006 Conf. Singapore: IEEE Computer Society, 2006: 409-416.

[12] OLSEN J, ORIA D. Profile based compression of *n*-gram language model[C]// 2006 IEEE International Conference on Acoustics Speech and Signal Processing. [S. l.]: IEEE Signal Processing Society, 2006: 1041-1044.

[13] CHIEN J T. Association pattern language modeling[J]. IEEE Transactions on Audio Speech and Language Processing,2006,14(5):1719-1728.

[14] RIEDEL E,KALLAHALLA M,SWAMINATHAN R. A framework for evaluating storage system security[C]// In FAST. Monterey: USENIX Press, 2002:15-30.

[15] SIKALINDA P G, KRITZINGER P S, WALTERS L O. Analyzing storage system workloads[C/OL]. [2005-01-01]. http://pubs.cs.uct.ac.za/archive/00000218/, Jan. 01 2005.

(编辑 李胜春)