倒排文件索引缓存机制的优化①

杨晓波

(浙江财经学院 信息分院，杭州 310018)

摘 要：为了有效提高搜索引擎检索服务系统的整体性能，提出了一种基于倒排文件索引的缓存机制优化方法。具体研究过程是：首先分析倒排文件缓存的体系结构和数据加载，接着讨论负载数据对倒排文件缓存和缓存替换算法的影响，最后通过设计仿真实验研究倒排文件的缓存优化。研究结果表明，采用倒排文件索引的缓存机制优化方法可以明显减少磁盘系统 I/O 访问次数，提高磁盘系统带宽的利用率。

关键词：倒排索引；缓存优化；替换算法；负载特性

Optimization of Inverted File Index Caching Mechanism

YANG Xiao-Bo

(Dept. of Information, Zhejiang University of Finance & Economics, Hangzhou 310018, China)

Abstract: In order to improve the whole performance of search service system effectively, it proposes a method of caching mechanism optimization based on inverted file index in this paper. The specific studying process is as follows. Firstly, the system structure and data loading of inverted file caching are analyzed, and then discuss the impact of loading data to inverted file caching and cache replacement algorithm; finally, the cache optimization of inverted file is studied through designing simulation experiment. The result shows that the method of caching mechanism optimization based on inverted file index can reduce disk system I/O access times significantly, and also improve the bandwidth utilization of disk system.

Key words: inverted index; cache optimization; replacement algorithm; loading character

搜索引擎每天要处理海量的 Web 数据，满足百万次以上的用户查询请求，同时其索引数据规模和用户数量还在不断增长，这样一个庞大的信息系统需要很强的系统性能和可扩展性。缓存技术是提高系统性能和可扩展性的一种重要手段，在计算机各个应用领域都有广泛的应用。如何有效的在搜索引擎检索服务系统中使用缓存技术被广泛关注。

缓存技术的有效性是建立在被缓存对象访问序列存在的局部性特征基础之上。搜索引擎检索系统中通常被研究的可缓存对象分为三种：即查询结果，布尔操作的中间结果，以及倒排文件。Xie[1]和 Wang[2]等详细分析了搜索引擎用户查询日志，发现用户查询具有很强的局部性，并提出了缓存搜索引擎查询结果的可行性；Markatos[3]和 Saraiva[4]等又进一步研究了缓存替换算法、缓存大小等因素对系统性能的影响；Chidlovskii[5]等提出语义缓存，把布尔查询的中间结果作为缓存对象，并利用查询结果间的语义关系加速后续查询的执行，该方法可以充分利用不同查询之间的相关性提高缓存命中率，缺点是限制在布尔查询上，可能影响结果相关性排序。关于倒排文件的缓存，用户查询经过查询器执行，转换为对倒排文件数据的访问序列，这些数据也可以作为缓存对象。Jonsson[6]等研究了 IR 背景下用户交互式查询的倒排文件缓存与查询执行结合的方法；Saraiva[4]等研究了一个实际搜索引擎（TodoBR）中的倒排文件缓存对系统效率的影响。

① 收稿时间:2011-08-21;收到修改稿时间:2011-09-22

本文采用天网搜索引擎的实际运行数据，重点研究倒排文件索引的缓存优化设计，根据倒排文件访问数据特点，分析缓存性能评估指标的选取，以及数据组织对缓存效率及系统性能的影响等。

1 倒排文件缓存

1.1 体系结构

天网搜索引擎检索系统采用分布式体系结构，按文档划分的方式组织数据到多个索引服务节点，它们独立且并行处理用户查询，将各自检索结果提交给查询服务器汇总并返回给用户。该系统的缓存结构如图 1 所示：

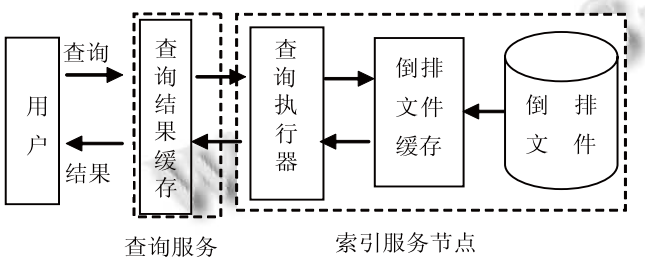

图 1 搜索引擎检索系统缓存结构图

倒排文件缓存位于索引服务节点上，对查询执行器在执行用户查询过程中访问的倒排文件数据进行缓存。大量统计研究表明查询中所查询词的序列具有良好的局部性，可以预期查询执行器读取的这些查询词，倒排数据序列也具有同样的性质，这是研究倒排文件缓存的基本出发点。与 Saraiva[4]等提出的过滤空间向量模型查询处理技术不同，天网搜索引擎使用带位置数据的全文倒排索引，对多个词的用户查询计算邻近权值。查询执行器访问倒排文件的数据分为两类：文档数据和位置数据。Navarro[7]等通过随机访问的倒排索引组织技术，可以减少数据读取量。

天网查询执行器读取倒排文件的数据序列包括文档数据序列和位置数据序列。本文提出一种新的缓存替换算法，以查询词位置邻近关系为基础，采用带位置数据的倒排索引，并使用索引压缩和块随机访问技术提高性能[7]，解决了因倒排文件访问数据不固定导致索引性能下降的问题。

1.2 数据加载

下面采用踪迹驱动（trace driven）的方法来研究倒排文件缓存的性质[8]。本文采用了天网搜索引擎的用户查询日志，查询日志中记录了用户查询和该查询是否被天网的查询结果缓存命中。将被结果缓存命中的查询剔除，就得到实际到达索引服务节点的查询序列。

同时，还需要形成一个该查询序列所针对的文档集合。为此，可从天网搜集的网页集合中随机抽取一批网页，建立索引，修改查询程序，把访问倒排文件的每次操作记录到日志文件，提取经过滤的查询序列中连续的 10 万个查询，送入查询程序执行。

最后，为了更有效的进行数据处理，把两个序列中的对象标识（I/O 序列中是索引词编号与访问类型，PAGE 序列是页面编号）转换为从 0 开始的连续整数。数据集的统计信息如表 1 所示：

表 1 数据集基本统计信息

名称	数值
用户总查询数	7,341,383
结果缓存未命中个数	3,522,968
文档总数	2,603,035
文档数据原始大小	30.18GB
倒排文件大小	5.77GB
I/O 序列长度	1,887,198
I/O序列唯一对象数	112,145
PAGE序列长度	20,808,025
PAGE序列唯一对象数	965,929

2 负载特性

通过分析负载数据的性质，讨论它们对倒排文件缓存和缓存替换算法的影响。

2.1 序列对象的频度和时间间隔分布

I/O 序列中的对象大小不同，其中由位置数据访问产生的部分是固定长度(32KB)，而对文档数据访问产生的对象大小分布很不均匀，以 4KB 为单位对其分布统计，统计结果表明：当值为 7.59KB，79%的请求对象长度在 64KB 以下，同时也有少数较大的数据访问。在考虑缓存替换策略时，偏向小对象的方法预计可以获得更好的性能。

对象被访问的频率是缓存设计的一个重要因素。如果序列中对象访问频率分布非常不均匀，则需要考虑两个问题，一是缓存少数高频对象可以提高性能，另一个是不区分出大量低频对象将降低性能。

I/O 序列和 PAGE 序列的序号与访问频度的分布

情况是：随着序列序号的增加，访问频度呈下降趋势。序列的时间局部性可以从序列中对同一个对象的两次连续访问时间间隔分布来考察。Jin[9]等认为使用访问在序列中的位置间隔，而不使用绝对时间，可以屏蔽用户查询密度在各个时间段内的周期性对分析结果的影响。I/O 序列和 PAGE 序列的时间间隔分布情况是：将距离数据按 2000 为单位分组，并计算各组的频度，可以得出在对数坐标下，序列的时间间隔分布接近直线，说明具有良好的时间局部性。

2.2 序列的重复模式

序列的空间局部性是指序列中固定模式的重复，Almeida[10]等认为它可以通过原始序列和随机排列处理后的序列中唯一定长串的个数来说明。空间局部性也是缓存设计需要考虑的一个因素。

提取 I/O 序列和 PAGE 序列前 10 万个数据，处理其中长度从 1 到 9 的连续串，统计唯一的串的个数，再将序列进行随机重排并统计，得到序列中指定长度的唯一串的个数，计算结果表明：随着串长度的增加，唯一模式串的个数也随之增加。其中，随机排列的序列增加速度最快，其空间局部性最差；I/O 序列增加得最为平缓，其空间局部性较强，PAGE 序列次之。

综上分析，在不同条件的缓存实验中，可以采用系统的实际性能参数为指标。定义系统实际执行的 I/O 次数比率 Rt,以及系统实际读取的数据量比率 Rd,如式(1)所示：

$$Rt=\frac{T}{T_{base}}=\frac{\mathrm{Re}questNum*(1-HitRatio)}{\mathrm{Re}questNum_{base}*(1-HitRatio_{base})}$$
$$Rd=\frac{D}{D_{base}}=\frac{\mathrm{Re}questBytes*(1-ByteHitRatio)}{\mathrm{Re}questBytes_{base}*(1-ByteHitRatio_{base})} \quad (1)$$

其中 T 表示系统实际执行的 I/O 次数，D 表示实际读取的数据量，HitRatio 是缓存命中率，ByteHitRatio 是缓存字节命中率，RequestBytes 和 RequestNum 分别表示需求字节和需求数量，当 Rt 和 Rd 小于 1 时说明效率比基准更高。

3 仿真实验与结果分析

3.1 仿真实验设计

在分析了负载数据序列的性质后，通过仿真实验来进一步研究倒排文件缓存优化问题，下面详细介绍仿真实验的设计过程。

在倒排文件数据的访问过程中，文件系统的作用十分重要。天网运行在 Linux 操作系统上，缓存在 page cache 中[11]，Linux 系统使用一种综合 LRU 与 LFU 的缓存替换策略，并且可以使用全部空闲的物理内存作为缓存空间。

首先研究 I/O 序列的缓存设计，以缓存命中率为性能指标，这和文件系统对定长页面缓存的设计不同。I/O 序列中的对象是变长数据，需要在缓存替换算法中考虑。这种变长数据缓存问题在 Web 代理缓存领域有较多研究，其中 GD-SIZE 系列算法性能较好[12]，通过与基本的 SIZE 算法与基本的 LRU、LFU 算法比较，选取出 I/O 序列下缓存的最佳替换算法。

接着研究 PAGE 序列的缓存设计，以缓存字节命中率为性能指标，对页面大小为 4KB 的情况，倒排文件缓存与文件系统缓存的差异在于替换算法的选择。O'Neil[13]等提出的 LRU-K 算法被广泛研究，很多应用中具有比 LRU/LFU 更好的性能，本文选取它作为一种替换算法进行比较。

最后研究 PAGE 序列的缓存设计中，缓存效率和页面大小的关系。倒排文件缓存可以选择最优的页面大小，从而可以得到比文件系统缓存在固定 4KB 页面下更好的性能。这时使用 Rd 作为性能指标，基准为 4KB 页面。因为无法确切实现文件系统的缓存替换算法，实验统一使用 LRU。

3.2 仿真实验结果

通过仿真实验，首先分析 I/O 序列缓存替换算法的性能。实验结果表明：以命中率为指标，GD-SIZE1 性能最优，其次依次为 GD-SIZE(T)、LRU-2、LRU、LFU 和 SIZE。根据前述对负载特性的分析，对于倒排文件索引，LRU 算法比考虑访问频率的 LFU 性能好，命中率平均高 7%。性能最好的 GD-SIZE1 比 LRU 平均高出 5%。如果从减少搜索引擎检索的磁盘 I/O 次数，缓解磁盘系统 IOPS 参数瓶颈的角度设计倒排文件缓存，可以使用 GD-SIZE1 算法，对 I/O 序列进行缓存，获得最好的性能。

以字节命中率为指标，分析结果是：LRU-2 性能最优，其次依次为 LRU、GD-SIZE(T)、GD-SIZE1、LFU 和 SIZE。LRU-2 在缓存容量较小，且在 128MB-512MB 时，性能比 LRU 高 4%左右，随着缓存容量增大，两者性能逐渐趋于一致；GD-SIZE(T)比 GD-SIZE1 性能稍好，如果考虑磁盘 I/O 次数和磁盘

带宽利用的均衡，GD-SIZE(T)是一个较好的选择；同样 LFU 和 SIZE 性能不佳。

接着分析 PAGE 序列（大小为 4KB）缓存替换算法的性能。采用替换算法 LRU、LFU 和 LRU-2，仿真结果表明： LRU-2 性能最好，在缓存容量中等的情况下，最多比第二位的 LRU 高 5%，在大缓存容量下，两者性能差异不大。同 I/O 序列类似，LRU 比 LFU 的性能好。

另外，为了了解 PAGE 序列的缓存性能与页面大小的关系，还需分析 PAGESIZE 对 PAGE 序列缓存性能的影响。仿真结果表明：以 LRU 算法的命中率为指标，随着 PAGESIZE 增加，PAGE 序列的缓存命中率也随之增加。考虑到不同 PAGESIZE 下以 4KB 为基准的 Rd 随缓存大小的变化。以 Rd 为性能指标，4KB 页面为基准，可以得出：文件系统 4KB 页面的缓存在不同的应用和配置环境下不是最优，倒排文件缓存可以通过使用较大的页面，获得更好的系统性能。

最后分析在页面对齐倒排文件组织方式下的缓存性能。仿真实验结果表明： Rd 和 Rt 的性能提升主要来自按页面对齐减少数据请求总数，随着缓存容量变化，性能可以提高 5%-10%左右，对系统性能提升较为明显。

通过四组缓存仿真实验，验证了倒排文件缓存经过优化设计后，可以比文件系统缓存性能更好。以缓存命中率为指标，采用 GD-SIZE1 算法，对 I/O 序列进行缓存，可获得最好的性能；以字节命中率为指标，LRU-2 算法性能最优；对 4KB PAGE 序列的缓存，不同替换算法的性能差异不大；另外，随着 PAGESIZE 增大，PAGE 序列的缓存命中率也随之增大，倒排文件缓存可以通过使用较大的页面，获得更好的系统性能；缓存命中率通常不会随着页面的增加而增加，只有在新的组织方式下，大页面的优势才能显现出来。

4 结语

本文研究了搜索引擎中倒排文件缓存技术。通过分析数据访问序列的局部特性，采用基于真实数据的缓存仿真实验，探讨了倒排文件缓存优化设计中的性能指标选择问题、替换算法、页面大小和倒排文件组织方式等对缓存性能的影响，并得到如下结论。

1）通过缓存变长的 I/O 序列对象，采用 GD-SIZE1 替换算法，可以明显减少磁盘系统 I/O 访问的次数；

2）通过按页面对齐方式组织倒排文件，选取大的页面作为访问倒排文件的单位，可以使磁盘系统带宽利用率得到优化。

这些结论对提高搜索引擎检索系统的效率具有重要的意义。进一步的工作可以研究查询器执行算法对倒排文件缓存系统的影响。如何结合语义缓存的思想，把查询器处理的中间结果缓存与倒排文件缓存结合起来，也是值得进一步探讨的工作。

参考文献

1 Xie Y, O'Hallaron D. Locality in search engine queries and its implications for caching. Proc. of Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. New York: Infocom, 2002: 1238−1247.

2 Wang J, Shan S, Lei M, Xie Z, Li X. Web search engine: characteristics of user behaviors and their implication. Science in China, 2001,44(5):351−365.

3 Markatos EP. On caching search engine query results. Computer Communications, 2001,24(2):137−143.

4 Saraiva PC, De Moura ES, Ziviani N, Meira W, Fonseca R, Neto BR. Rank-preserving two-level caching for scalable search engines. Proc. of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM, 2001: 51−58.

5 Chidlovskii B, Roncancio C, Schneider ML. Semantic cache mechanism for heterogeneous Web querying. Computer Networks, 1999,31(11):1347−1360.

6 Jonsson BT, Franklin MJ, Srivastava D. Interaction of query evaluation and buffer management for information retrieval. Proc. of the ACM SIGMOD International Conference on Management of Data. New York: ACM, 1998: 118−129.

7 Navarro G, De Moura ES, Neubert M, Ziviani N, Yates RB. Adding compression to block addressing inverted indexes. Information Retrieval, 2000,3(1):49−77.

8 Anthony T, Hector GM. Caching and database scaling in distributed shared-nothing information retrieval systems. Proc. of the 1993 ACM SIGMOD International Conference on Management of Data. New York: ACM,1993: 129−138.

9 Jin S, Bestavros A. Sources and characteristics of Web temporal locality. Proc. of the 8th Interantional Symposium on Modeling, Analysis and Simulation of Computer and

(下转第 117 页)

3 结语

本容错系统采用的仍然是传统的双机容错原理，但是很好的结合了实际使用情况，而且能很有效的保证考试的正常运行。因为网络考试一般应用于学校的机房中，因此找两台甚至多台拥有完全一样的硬件和软件系统的计算机比较简单，这样不仅能实现设备的冗余，而且能实现操作系统、应用软件、数据等的同步和冗余，最关键的是系统相同可以在发生故障时实现无缝切换。经过多次对考试系统可靠性和稳定性，特别是容错模块的反应速度的测试，模拟现实中所发生的各种错误操作及紧急情况，观察得出考试系统在出现很多常见错误时仍能正常运行，基本可以达到 2 秒级的实时效果，真正达到无断点运行。

参考文献

1 栾好利.基于局域网的计算机考试系统研究与实现.东北大学学报, 2006,13(10):33−35.

2 梅晓勇,颜君彪,侯识忠.网络环境下的考试系统应用设计与实现.计算机工程与应用, 2003,8(26):43−46.

3 全渝娟,范荣强.基于 Web 的远距离考试系统.计算机应用与软件, 2003,15(7):92.

4 陈胜功.容错计算机技术及应用研究.北京:航空工业出版社, 2000:57−59.

5 王珍熙.可靠性、冗余及容错技术.北京:航空工业出版社, 1991:243−247.

6 郝莹.网上考试系统中的安全机制.微机发展, 2007,26(3):88−92.

7 淡战平,候义斌.面向应用级的纯软件双机热备份机制设计与实现.计算机工程与应用, 2000,56(6):104−105,140.

8 南英,陈士格,戴冠中.容错控制进展.北京:航空工业出版社, 1993:62-67.

9 Flavin Cristian. Understanding fault-tolerant distributed systems. Comm. of ACM, 2001,34(2):57-58.

10 Nam HC, Kim J, Hong SJ, Lee SG. Probabilistic Check pointing IEICE Transactions on Information and Systems, 6(17), June 2002:1093−1104.

11 姚耀文,刘希挥,王作新.双计算机容错系统的故障诊断模型研究.华南理工大学学报(自然科学版)，1999,78(7):39−44.

12 朱幸辉.大规模事务处理监测系统的研究与实现.长沙:国防科技大学, 2005.

13 简新红.Corba component model 系统监控与管理的设计与实现.长沙:国防科技大学, 2005.

14 Michael N.Nelson, Brent B. Welch, et al. Caching in the Sprite Network File System. ACM Transactions on Computer Systems, 1988,6(1):134−154.

(上接第 99 页)

Telecomunication System. Washington, DC: IEEE Computer Society, 2000: 28.

10 Almeida V, Bestavros A, Crovella M, De Oliveira A. Characterizing reference locality in the WWW. Proc. of the 1996 4th International Conference on Parallel and Distributed Information Systems. Washington, DC: IEEE Computer Society, 1996:92−103.

11 Otoo E, Olken F, Shoshani A. Disk Cache Replacement Algorithm for Storage Resource Managers in Data Grids. Proc. of the 2002 ACM/IEEE Conference on Supercomputing. Los Alamitos: IEEE Computer Society, 2002. 1−15.

12 Cao P, Irani S. Cost-Aware WWW Proxy Caching Algorithms. Proc. of the 1997 Usenix Symposium on Internet Technologies and Systems (USITS-97). Berkeley: USENIX Association, 1997:193−206.

13 O'Neil EJ, O'Neil PE, Weikum G. The LRU-K page replacement algorithm for database disk buffering. Proc. of the 1993 ACM SIGMOD International Conference on Management of Data. New York: ACM, 1993: 297−306.