第 34 卷 第 6 期 计 算 机 工 程 2008 年 3 月
Vol.34 No.6 Computer Engineering March 2008
·软件技术与数据库· 文章编号：1000—3428(2008)06—0063—02 文献标识码：A 中图分类号：TP301

基于内存池的空间数据调度算法

郭丙轩，张京莉，张志超
(武汉大学测绘遥感信息工程国家重点实验室，武汉 430079)

摘 要：计算机处理海量空间数据时，内外存之间数据的频繁交互导致内存占用高、处理效率低。使用内存池方式调度空间数据可以提高计算机效率。在多种特定地图使用模式下，不同的内存池页面置换算法能有效降低操作过程中的内外存交互，提高空间数据调度效率。实验表明，该算法为内存容量有限的嵌入式设备上的 GIS 提出了高效处理空间数据的方案。

关键词：内存池；空间数据；页面置换

Algorithm of Spatial Data Scheduling Based on Memory Pool

GUO Bing-xuan, ZHANG Jing-li, ZHANG Zhi-chao
(State Key Laboratory for Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079)

【Abstract】 Due to the characteristics of geographical information data, when massive spatial data processed with computer, frequent data interaction between external memory and internal memory will lead to higher occupancy, and lower efficiency. But memory pool can be used to dispatch spatial data and enhance efficiency. Based on a variety of specific map using models, adoption of the technology utilizing different memory pools page replacement algorithm can effectively reducee the interaction between external memory and internal memory. Experimental results show the proposed solution is ideal for GIS system in the embedded equipment with limited memory capacity.

【Key words】 memory pool; spatial data; page displacement

操作系统存储管理的主要目的是提高内存利用率，方便用户使用内存。在多道程序环境中，多个作业须共享内存资源，能否合理有效地利用内存对计算机整体性能影响很大。目前操作系统大多采用页式或段页式内存管理方法[1]，但这种方法并不适用于调度 GIS 中具有尺度维的空间数据。由于空间数据具有数据量大、类型复杂等特点，因此在浏览、导航、编辑时，需要频繁申请释放内存空间，容易产生页面“抖动”等问题[2]而降低 GIS 的效率。解决抖动问题最根本的方法是控制多道程序的数量，使每个用户作业都有足够的内存空间。但如果作业的个数太少，就会影响处理器的利用率[3]。为了提高 GIS 调度空间数据的效率，本文提出基于内存池的空间数据调度算法。

1 内存池原理及页面置换算法

1.1 BerkeleyDB 内存池

BerkeleyDB 内存池是嵌入式数据库 BerkeleyDB 的一个通用内存共享缓冲区，它允许同时访问数据库的多个进程或进程的多个线程共享一个高速缓存，负责将修改后的页写回文件并为新调入的页分配内存空间。BerkeleyDB 内存池采用平均划分页面大小的管理机制，在启动内存池前可以自行设定整个内存池的大小和页面大小，内存池在运行过程中始终保证有适量空闲页面。它适用于灵活的、基于页面的、带缓冲的文件共享访问，如图 1 所示。

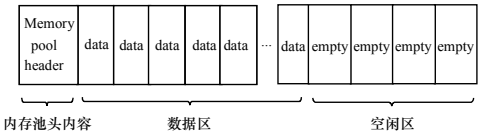

图 1 BerkeleyDB 内存池结构

BerkeleyDB 内存池采用 LRU 页面置换算法，本文在此基础上增加 FIFO 和 LFU 页面置换算法，分析 GIS 的不同功能在空间数据调度上分别使用这些置换算法产生的效率差异。

1.2 内存池页面置换

页式管理按一定的粒度划分内存，如何更新这种内存页面是内存与磁盘交互时的关键。作为操作系统内存管理的延伸，内存池采用页面管理，须使用合适的页面置换算法。经典的操作系统页面置换算法有：

(1)最佳置换算法(OPTIMAL)，淘汰以后不再被使用的或是在最长时间内不会再被访问的页面，是理想的置换算法。

(2)先进先出置换算法(FIFO)，淘汰最先进入内存的页面，即淘汰在内存中驻留时间最长的页面。

(3)最近最久未使用置换算法(LRU)，赋予每个页面一个访问字段，记录该页面自上次被访问以来经历的时间 T，当须淘汰一个页面时，淘汰现有页面中 T 值最大的页面，即淘汰最近一段时间内使用最少的页面。

(4)最近使用最多置换算法(MRU)，赋予每个页面一个访问字段，记录该页面自上次被访问以来经历的时间 T，当须淘汰一个页面时，淘汰现有页面中 T 值最小的页面，即淘汰最近一段时间内使用最多的页面。

(5)使用频率最低置换算法(LFU)，淘汰内存中访问次数最少的页面。

上述的传统页面置换算法都是根据过去的情况预测未来

作者简介：郭丙轩(1970－)，男，副研究员、博士，主研方向：3S 集成应用，图形图像处理，模式识别；张京莉、张志超，硕士研究生
收稿日期：2007-03-25 **E-mail：**zhichao_z@163.com

的趋势，没有考虑不同的应用程序在运行时有不同的页面访问模式。对不同的页面访问模式应选用不同的置换算法，否则可能会增加系统的缺页中断率，降低系统性能。

2 BerkeleyDB 内存池空间数据调度算法

自动探测不同应用程序的页面访问模式，针对不同的模式使用不同的操作系统内存管理页面置换算法[4]，此方法虽然可以提高应用程序的效率，但因为涉及操作系统核心问题，所以难以实现。本文在内存池上根据不同的页面访问模式使用不同的页面置换算法，易于实现。由于 mapinfo 地图的.map 文件由固定个数的、大小为 512 B 的块组成，因此本文将内存池的页面大小设置为 512 B，以方便调度该数据。通过大量实验，得出进程访问内存池页面的一些模式，如下：

(1)顺序型页面访问模式，每个页面按先后顺序被依次访问，被访问过的页面不能再次被访问。

(2)循环型页面访问模式，每个页面间隔一定时间被重新访问一次。

(3)时间聚类型页面访问模式，最近被访问的页面将很快再次被访问。

(4)概率型页面访问模式，每个页面有各自固定的被访问概率。

确定页面访问模式后，就可以选用相应的页面置换算法。对顺序型和循环型页面访问模式一般采用 FIFO 页面置换算法；对时间聚类型页面访问模式一般采用 LRU 页面置换算法；对概率型页面访问模式一般采用 LFU 页面置换算法。

实验表明在不同模式下选用有针对性的内存池置换算法可以有效减少磁盘 I/O 次数，提高应用程序的运行效率。

3 BerkeleyDB 内存池空间数据调度算法实验结果

采用 BerkeleyDB 内存池，对 mapinfo 的 tab 格式文件进行调度，模拟地图导航、地图编辑和地图浏览，使用 rational quantify 分别测算直接操作文件调度时间和内存池调度时间。调度时间指调度空间数据的函数执行时间，该时间可以有效表达算法调度空间数据的效率，针对不同的地图使用模式采用不同的置换算法，并进行了实验分析。

实验环境：2.40 GHz Pentium(R) 4 CPU，512 MB 内存。

实验条件与结果如下：

(1)地图导航：选取武汉市一级道路层 mapinfo 地图，路线为沿武汉市汉宜公路指定点到关山路指定点，实验结果如表 1 所示。

表 1 地图导航时间

置换算法	采用内存池及不同页面置换算法花费时间/ms	文件大小/KB	内存池大小/KB	直接读写文件花费时间/ms
FIFO	121 872.77	3 542	256	271 243.85
LRU	193 623.54	3 542	256	271 243.85
LFU	165 842.36	3 542	256	271 243.85

(2)地图编辑：选取武汉市水系面色图层 maoinfo 地图，依次增加 20 个地物点，10 条线，5 个面，实验结果见表 2。

表 2 地图编辑时间

置换算法	采用内存池及不同页面置换算法花费时间/ms	文件大小/KB	内存池大小/KB	直接读写文件花费时间/ms
FIFO	985 643.78	7 659	256	1 574 683.62
LRU	863 751.33	7 659	256	1 574 683.62
LFU	768 531.64	7 659	256	1 574 683.62

(3)地图浏览：选取武汉市汉阳小城区图层 mapinfo 地图，按照地图幅面从上至下，从左至右浏览整幅地图，实验结果见表 3。

表 3 地图浏览时间

置换算法	采用内存池及不同页面置换算法花费时间/ms	文件大小/KB	内存池大小/KB	直接读写文件花费时间/ms
FIFO	205 365.23	4 983	256	365 678.36
LRU	161 234.10	4 983	256	365 678.36
LFU	193 264.28	4 983	256	365 678.36

从实验结果中可以看出：地图导航模式的时序关联性很强，应采用时间关联的 FIFO 算法；地图编辑模式与空间对象的使用频度有关，应采用 LFU 算法；地图浏览模式的时间联度差和频度关联度类似，应采用 LRU 算法。

参考文献

[1] 任满杰, 刘树刚, 李军红, 等. 操作系统原理实用教程[M]. 北京: 电子工业出版社, 2006.

[2] Mosberger D, Eranian S. Linux 内核设计与实现[M]. 陈莉君, 译. 北京: 机械工业出版社, 2004.

[3] 陆松年, 薛　质, 潘　理, 等. 操作系统教程[M]. 北京: 电子工业出版社, 2006.

[4] 殷联甫, 汪承焱. 基于探测的自适应页面置换算法研究[J]. 计算机应用与软件, 2005, 22(6): 142-144.

(上接第 50 页)

被激活。由于 du_1 只由一个委托步 ds_{JD} 组成，因此 ds_{JD} 被激活，寿命开始倒计时。然后 $DA_{JD}=\{(e, JD, JD\_s, 7, ds_{JD})\}$ 中的 e 调用 JD_s 为 JD 服务，但是否成功还要看其委托凭证 dc_{JD} 是否通过虚拟组织和服务资源所在自治域的双重验证。若 du_1 成功且与 du_2 的分权依赖得到满足，则 du_2 被激活。du_2 由 2 个委托步 ds_{CK_1} 和 ds_{CK_2} 组成，且满足分权依赖的要求和 du_1, du_2 的分权依赖要求(委托凭证 dc_{CK_1} 和 dc_{CK_2} 表明 du_1 和 du_2 的服务资源来自不同的自治域，由于篇幅限制，未在表 3 中示出)，因此，它们的执行结果共同决定了 du_3 的激活状态。其他情况类似。

由上述分析可见，整个流程委托授权满足职责分离和最小特权原则，因此，只要每个任务是可验证和可信的，那么系统的完整性就能得到保证。

3 结束语

本文提出的 GWACM 充分结合了角色委托技术与基于任务的访问控制技术的优点，为网格工作流授权管理提供了细粒度的支持。

参考文献

[1] Ferraiolo D, Sandhu R. Proposed NIST Standard for Role-based Access Control[J]. ACM Transactions on Information and System Security, 2001, 4(3): 224 -274.

[2] 孙为群, 单保华, 张　程, 等. 一种基于角色代理的服务网格虚拟组织访问控制模型[J]. 计算机学报, 2006, 29(7): 1199-1208.

[3] Thomas R K, Sandhu R S. Task-based Authorization Controls: A Family of Models for Active and Enterprise-oriented Authorization Management[C]//Proceedings of the IFIP WG11.3 Workshop on Database Security. Lake Tahoe, California: [s. n.], 1997.

[4] 邓集波, 洪　帆. 基于任务的访问控制模型[J]. 软件学报, 2003, 14(1): 76-82.