事务存储系统中 PGHB 冲突检测算法改进

窦 强,王 勇
(国防科技大学计算机学院,湖南长沙 410073)

摘 要: 事务存储系统是一种全新的多核体系结构,为并行编程提供了一个简洁高效的编程环境.基于 Signature 的冲突检测算法是事务存储系统中很有前景的一种冲突检测方法,其误判率直接影响系统性能.PGHB 算法是一种优秀的冲突检测算法,具有较低的误判率,和较低的硬件实现开销.本文对 PGHB 冲突检测算法进行进一步改进,提出一种新的算法,使用单端口的 SRAM 来实现 PGHB 算法,并使用 HP 的 CACTI4.2 对硅片占用面积进行评估.结果显示,改进的 PGHB 算法与原算法相比,硅片占用面积节约了 71%,使得该算法在硬件开销和误判率之间取得了更好的折衷.

关键词: 事务存储; Signature; 冲突检测; 误判率
中图分类号: TP303 **文献标识码:** A **文章编号:** 0372-2112 (2010) 01-0195-05

The Improvement of PGHB Conflict Detection Algorithm in Transactional Memory Systems

DOU Qiang, WANG Yong
(*School of Computer Science, National University of Defense Technology, Changsha, Hunan* 410073, *China*)

Abstract: Transactional Memory is a new multiprocessor architecture intended to make parallel programming easy and efficient. Signature-based conflict detection is a promising approach in Transactional Memory systems and its rate of false positive has much influence on its performance. The PGHB algorithm is an excellent algorithm with lower false positive rate and low hardware cost. In this paper, we propose an improved algorithm based on PGHB, implemented with single-port SRAM. We use HP's CACTI4.2 to evaluate the silicon area. And experimental results are presented which show that our improvement can save 71% of the silicon area cost and gain a compromise between hardware cost and false positive rate.

Key words: transactional memory; signature; conflict detection; rate of false positive

1 引言

随着计算机技术的不断发展,多核处理器系统已经逐渐普及到桌面级应用.为了能够充分发挥多核系统的计算能力,必须采用合适的并行编程模型.然而传统的基于锁机制的并行编程模型需要考虑的问题复杂,使得只有少数的专业并行编程人员才能熟练应用;同时锁机制又存在着诸多问题[1](如:死锁、优先级反转、锁护送效应等等),这大大限制了并行程序的开发.于是人们提出了一种新的并行编程模型——事务存储系统[1](Transactional Memory,简称为 TM)来取代传统的并行编程模式.

事务的概念最早出现在数据库领域,通过事务所具有的基本特性来保证对数据库并发访问的正确性.事务存储模型发展了数据库中事务的概念,并将其引入并行计算领域,文献[2]对现有的 TM 发展状况做了全面的总结.

TM 主要包括三大基本功能:冲突检测、数据版本管理和冲突解决.冲突检测主要用于检测并发事务对同一个项(字、块、对象等等)的访问冲突,其效率直接影响系统性能.Wisconsin 大学于 2007 年在 LogTM-SE[3] 系统中提出了一种基于 Signature[3] 的冲突检测机制.它在性能和存储开销之间采取了较好的折衷,是一种非常有前景的冲突检测方法.

GHB 算法[4]就是一种优秀的基于 Signature 的冲突检测算法,该算法的动态自适应性使其在地址量较多和较少的时候都能达到较低的误判率.本文在对现有的基于 Signature 的冲突检测算法进行深入分析的基础上,对 GHB 冲突检测算法进行进一步改进,提出了一种新的算法.与 GHB 算法相比,该算法在硬件开销和误判率两者之间取得了更好的折衷.

收稿日期:2008-09-03;修回日期:2008-11-11

2 基于 Signature 的冲突检测算法

为了能准确地进行冲突检测,必须记录事务的完整读写地址集合.然而,一个事务在执行过程中,可能会对大量的地址进行读写操作,要完全的记录这些地址信息不仅将占用大量的存储空间,而且查找也将变得非常低效.同时,在进行线程切换的时候必须保存未提交的事务的读写地址集合,保证该线程在重新被调度的时候能够继续正确的被执行.如果信息过于庞大,不仅大量占用 CPU 宝贵的堆栈资源,而且将耗费大量 CPU 节拍来处理保存过程,这严重影响了系统的性能.

为解决上述问题,基于 Signature 的冲突检测算法采用了 Bloom Filter[5]的思想,将一个事务的读写地址集合通过 K 个不同的 Hash 函数映射到一个长度为 m 的位数组中,这个位数组就称为 Signature.该机制利用有限位数的 Signature 表示无限多的读写地址集合,不仅节省了大量的存储空间,而且提高了查找的效率.不难看出,Signature 的高效是以引入误判[5]为代价的,即 TM 系统中本来并没有冲突的两个并发事务可能被误判为冲突并导致其中一个事务被终止(Abort).因此,该类算法误判率的高低对系统性能有较大的影响.

文献[6]提出了多种基于 Signature 的冲突检测算法,按其实现思想主要可分为以下四大类:

True-Bloom:实现简单,直接使用 Bloom Filter.误判率与 Hash 函数的数量 K 密切相关,对于已选定的 K 值,地址量越少其性能越好,地址量越多其性能越差.

Cuckoo-Bloom: Cuckoo Hash 算法[7]和 True-Bloom 算法相结合的一种算法.实现较复杂,存储格式在由 Cuckoo 算法格式向 True-Bloom 算法格式换时延时较大.

Adaptive-Bloom:基于 True-Bloom 算法,增加地址个数预测部件,通过预测结果来动态改变 Hash 函数的数量.主要问题是地址个数预测部件实现复杂,预测结果不精确,并且需要额外的空间存储预测表信息.

Hash-Bloom:存储格式动态转换,实现较 Cuckoo-Bloom 简单,平均性能较优.

Hash-Bloom Signatures[6]算法的巧妙之处在于,当 Signature 中插入的地址数量比较少的时候使用较多的位数来存储地址指纹信息,当插入的地址增多的时候就使用较少的位数来存储,最终将逐渐转化成为 $K=1$ 的 True-Bloom 算法.这种动态变化是在地址插入的过程中逐渐的完成的,避免了 Cuckoo-Bloom 算法中存在的当插入地址个数达到一定阈值的时候才集中对已插入的地址进行重新插入而带来较大延时的问题.虽然在插入地址数量较小的时候,该算法的误判率不是最低的,在上面几种算法中处于居中的位置,但是在插入地址数量较多的时候,该算法能达到较低的误判率.这使得系统在所有情况下都能处于较优的性能,而避免了在遇到一些特殊情况时系统性能急剧下降的问题,提高了系统的稳定性.

Greedy-Hash-Bloom(GHB)算法是一种基于 Hash-Bloom 和 True-Bloom 算法的自适应算法,融合了两种算法的优点,很好的降低了地址量较少时的误判率.GHB 算法在不增加额外存储空间开销的条件下,在地址数量较少的时候利用 Hash-Bloom 算法未使用的存储空间存储 Hash 函数数量为 K 的 True-Bloom 算法的映射信息;在地址量较多时仍采用 Hash-Bloom 算法.GHB 算法的误判率取决于 Hash-Bloom 算法和 True-Bloom 算法的误判率的乘积.在地址数量较少时,Hash 函数的数量为 K 的 True-Bloom 算法误判率较低,在地址量较多时,Hash-Bloom 算法误判率较低,这样以来,GHB 算法在所有条件下都能达到较低的误判率.

3 Parallel-Greedy-Hash-Bloom 算法

GHB 算法虽然性能较优,但该算法实现时需要使用 $K+1$ 端口的 SRAM,图 1 给出了该算法的硬件实现方式.我们知道,SRAM 的端口数量对其硬件实现的复杂度影响较大,端口越多实现起来越复杂,占用硅片面

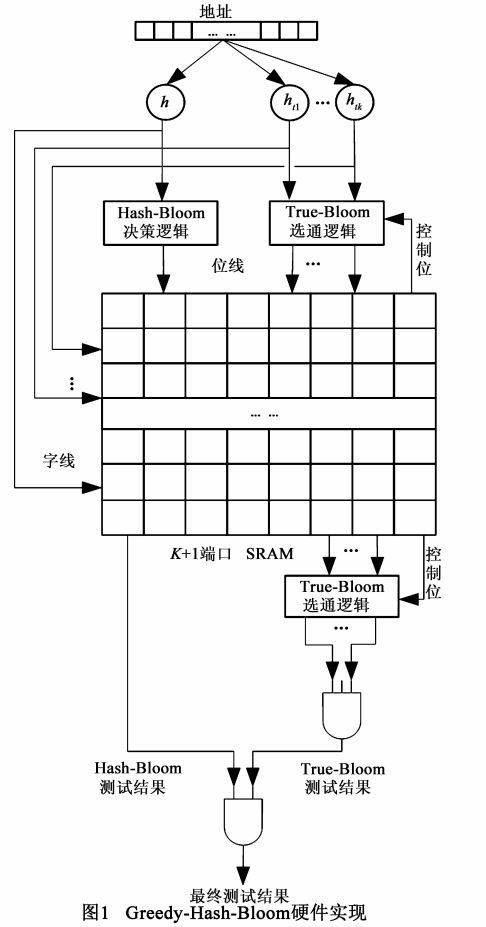

图1 Greedy-Hash-Bloom硬件实现

积也将越大.为了降低硬件开销,文献[8]改进了该算法,采用并行的方式实现 GHB,并把这种算法称之为 Parallel-Greedy-Hash-Bloom,简称为 PGHB 算法.

PGHB 算法实现时使用 K 个 2 端口的 SRAM,每个 SRAM 大小为原 SRAM 的 $1/K$,每个 True-Bloom 的映射范围限制到一个固定的 SRAM 中去,也就是说 K 个 True-Bloom 的值并行的映射到 K 个 SRAM 中去.每个 SRAM 需使用一个端口来读写 True-Bloom 的映射信息,同时,还需要一个端口来读写 Hash-Bloom 的映射信息.由于 Hash-Bloom 同一时刻只会映射到一个 SRAM 中,所以,K 个 SRAM 中同一时刻只有一个 SRAM 需要共享的同时读写 Hash-Bloom 和 True-Bloom 的映射信息.虽然同一时刻只有一个 SRAM 需要两个端口,但是每个 SRAM 都有读写 Has-Bloom 信息的可能,所以每个 SRAM 都需要两个端口.图 2 给出了 PGHB 算法的硬件实现方式.

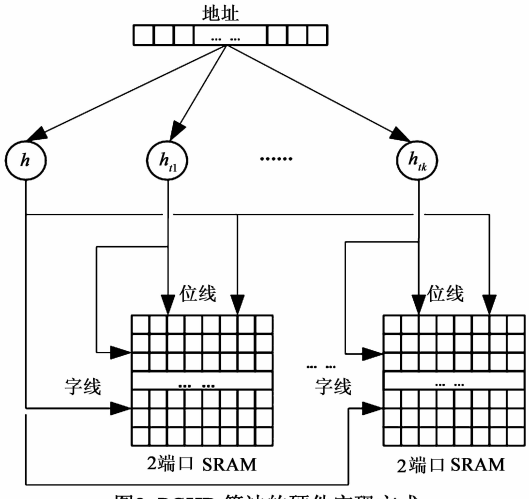

图2 $PGHB_s$算法的硬件实现方式

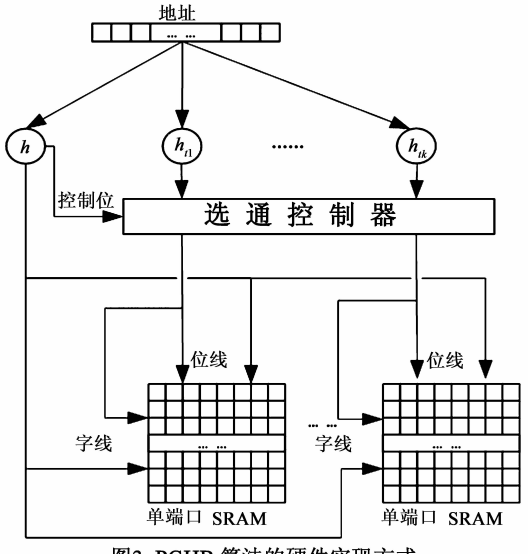

图3 $PGHB_e$算法的硬件实现方式

由于 Hash-Bloom 是在 K 个 SRAM 的范围内映射,所以同一时刻必然会存在 Hash-Bloom 和一个 True-Bloom 映射到同一个 SRAM 中的情况,文献[8]采用两端口的 SRAM 来实现 PGHB 算法.本文采用单端口的 SRAM 来实现 PGHB 算法,从而进一步降低硬件开销.我们把原有的 PGHB 算法称之为 $PGHB_s$(share),新的算法称之为 $PGHB_e$(exclude).

$PGHB_e$ 算法实现时使用 K 个单端口的 SRAM.与 $PGHB_s$ 算法不同的是,当 True-Bloom 和 Hash-Bloom 映射到同一个 SRAM 中时,Hash-Bloom 具有高优先级,进行信息的读写,True-Bloom 将被屏蔽,不进行信息的读写(需要增加一个选通控制器),这样每个 SRAM 只需要一个端口即可.图 3 给出了 $PGHB_e$ 算法的硬件实现方式.

4　性能评估

由于采用了并行的实现方式,每个 True-Bloom 映射范围相对独立,这样避免了不同的 True-Bloom 映射到同一个位置时产生的冲突(内部冲突),但每个 True-Bloom 的映射范围变小,这增加了不同地址被同一个 True-Bloom 映射到同一位置的可能(外部冲突).内部冲突和外部冲突的改变将对系统的误判率产生一定的影响.

尽管 $PGHB_e$ 算法并不复杂,但是要推导出一个数学公式来表示其误判率将是一项极其复杂的工作,所以,我们采用蒙特卡罗的方法来对其误判率进行分析.测试方法如下:

(1)模拟地址插入:随机产生 a 个地址,然后使用 $PGHB_e$ 算法将这 a 个地址插入到 Signature 中,并将已插入的地址记录到已插入地址表中;

(2)模拟地址测试:随机产生 t 个地址,对每个地址使用 $PGHB_e$ 算法进行测试,如果命中但该地址不在已插入地址表中,则该地址为误判.累加误判的地址个数 m_1,求得误判率 $f_{a1}=m_1/t$;

(3)重复测试:记录求得的误判率并将 Signature 复位,重复(1)、(2)N 遍,求得 N 个误判率;

(4)取平均值:将所得到的 N 个误判率取平均值,从而得到在 Signature 中已插入 a 个地址时的误判率

$$f_a=\frac{\sum_{n=1}^{N} f_{an}}{N} \tag{1}$$

为了尽量减小计算过程中小数舍位所产生的误差,避免因误差累积而降低模拟精度,我们对式(1)进行推导:

$$f_a=\frac{\sum_{n=1}^{N} f_{an}}{N}=\frac{\sum_{n=1}^{N}\frac{m_n}{t}}{N}=\frac{\frac{1}{t}\sum_{n=1}^{N} m_n}{N}=\frac{\sum_{n=1}^{N} m_n}{t\cdot N} \tag{2}$$

根据式(2)我们就可以先累加误判的地址的个数,

在模拟 N 次后再求误判率.

图 4 和图 5 给出了 Signature 大小为 4KB, $K=8$ 时的模拟结果,其中图 5 是图 4 的局部放大.两图中我们同时对比了不同 K 值的 True-Bloom、GHB 和 $PGHB_s$ 算法的误判率.从两个图中我们可以看出,$PGHB_e$ 算法的误判率较 $PGHB_s$ 算法的误判率并没有增加,两算法在图中的曲线基本重合.

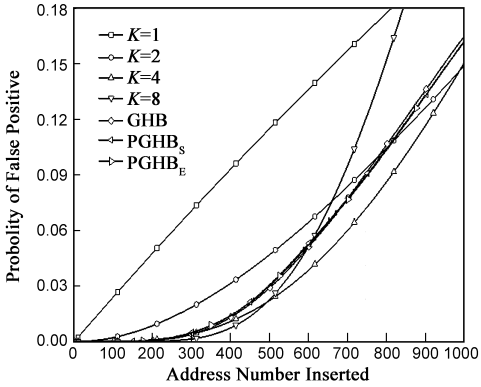

图4 Signature大小为4KB时的误判率对比1

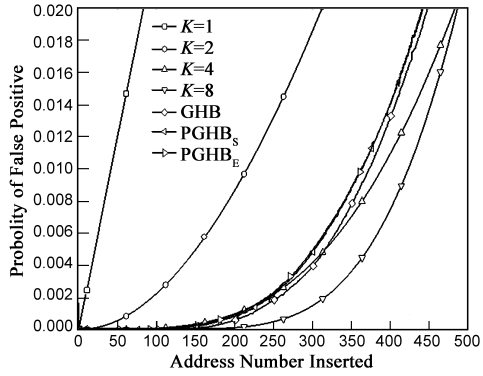

图5 Signature大小为4KB时的误判率对比2

表 1 三种算法 SRAM 的硅片占用面积对比

参数	9PortsSRAM (GHB)	2PortsSRAM ($PGHB_s$)	1PortSRAM ($PGHB_e$)
SRAM Size(bytes)	512	64	64
Size of each Entry(bytes)	8	8	8
Read Ports	9	2	1
Write Ports	9	2	1
Single Ended Read Ports	0	0	0
Nr. of Bits Read Out	8	8	8
Technology Node (nm)	65	65	65
Total area One bank(mm^2)	2.0375	0.0212	0.0062
Total area(mm^2)	2.0375	0.1695	0.0493

我们使用 HP 的 CACTI4.2[9] 对 GHB 算法和两种 PGHB 算法的硅片占用面积进行了评估,结果如表 1 所示.从表中我们看出,$PGHB_s$ 算法实现时 SRAM 所占用的硅片面积仅是 GHB 算法占用面积的 8.32%,而 $PGHB_e$ 算法实现时 SRAM 的硅片占用面积更少,仅是 GHB 算法的 2.42%.同时,$PGHB_e$ 算法较 $PGHB_s$ 算法在实现时可节约 71% 的硅片占用面积.

通过以上的对比我们看出 $PGHB_e$ 算法不仅在性能上有出色的表现,而且实现时还可以节约大量的硬件成本,从而在误判率和硬件开销之间取得了相对于 $PGHB_s$ 算法更好的折衷.

5 结束语

基于 Signature 的冲突检测算法利用有限位数的 Signature 来表示无限多的读写地址集合,是 TM 系统中一种很有前景的冲突检测设计方案.PGHB 算法通过动态变化 Hash 函数的数量,使得该算法在绝大多数情况下能达到较低的误判率.本文对 PGHB 算法进行进一步改进,提出了 $PGHB_e$ 算法,与原算法相比,$PGHB_e$ 算法在硬件开销和误判率两者之间取得了更好的折衷.

作者简介:

窦 强 男,1973 年生,博士,国防科技大学计算机学院副研究员,硕士生导师,研究方向为高性能计算机体系结构和高性能微处理器设计.

E-mail:douq@vip.sina.com

王 勇 男,1984 年生,国防科技大学计算机学院硕士研究生,研究方向为高性能计算机体系结构.

参考文献:

[1] Herlihy M, Moss J E. Transactional Memory: Architectural Support for Lock-Free Data Structures[M]. Proceedings of the 20th Annual International Symposium on Computer Architecture, 1993. 289 – 300.

[2] Larus J R, Rajwar R. Transactional Memory[M]. Morgan & Claypool, 2006.

[3] Yen L, Bobba J, Marty M M, et al. LogTM-SE: Decoupling Hardware Transactional Memory from Caches[M]. Proceedings of the 13th International Symposium on High-Performance Computer Architecture(HPCA), 2007.

[4] 王勇,窦强,万轶,刘超.一种改进的事务存储系统冲突检测算法[J].计算机工程与科学,2008,30(A1):190 – 193.

(下转第 212 页)