一种自适应多核处理器I/O一致性处理方法

郭御风,李 琼,窦 强,罗 莉
(国防科学技术大学计算机学院,湖南长沙 410073)

摘 要: I/O引发的数据一致性问题是多处理器系统数据一致性问题不容忽视的重要因素.多核使得处理器中的I/O一致性问题变得更加突出.另一方面,I/O访问模式的多样性使得单一的I/O一致性处理方法很难满足多样的I/O应用需求.本文首先分析了I/O应用访问模式,提出了I/O数据访问的七种特性;并根据I/O数据的访问特性提出了一种新型的自适应多核处理器I/O一致性处理方法,使得I/O一致性处理可以自适应地根据I/O应用进行动态调整,从而满足不同应用需求;最后,我们对自适应的I/O一致性处理方法进行了性能评测,评测结果表明该方法对不同的I/O应用均能获得很好性能.

关键词: 多核处理器; I/O一致性; I/O访问模式; I/O虚拟化
中图分类号: TP302.1 **文献标识码:** A **文章编号:** 0372-2112 (2011) 05-1194-05

An Adaptive Method for Maintaining I/O Data Coherence of Multi-Core Systems

GUO Yu-feng, LI Qiong, DOU Qiang, LUO Li
(*School of Computer Science, National University of Defense Technology, Changsha, Hunan* 410073, *China*)

Abstract: I/O data consistency problem is an important factor which multiple processors systems can not ignore. Multiple cores make the I/O consistency of processors become more and more severe. On the other hand, diversity of I/O pattern makes it's difficult to find a method which can satisfy different applications. Based on analysis of I/O pattern, we put forward seven characteristics of I/O workloads; Then presented a novel adaptive method for maintaining I/O data coherence in multi-core systems; Finally, the test results of our method would be shown, and we can find the method can satisfy different I/O workloads.

Key words: multi-cores processor; I/O consistency; I/O access pattern; I/O virtualization

1 引言

数据一致性模型是系统设计者与应用程序员之间的一种约定,是一种编程接口.DMA操作具有CPU干预少、数据传输带宽高、软件干预少等优点,是目前大多数I/O设备的主要工作方式.由于DMA操作[1]是绕开处理器核直接和存储器间进行数据传输,因此对于存储器来说,能够发起DMA操作的I/O设备和所有的处理器核一样,既可以作为存储器数据的生产者,也可以作为存储器数据的消费者.DMA可以与处理器核同时访问存储器,因此可能引发数据不一致的情况.I/O引发的数据一致性问题是系统数据一致性问题不容忽视的重要因素,需要得到高效的解决.

多核处理器[13]就像把传统的共享主存多处理器系统(DSM[2,3])浓缩到一个单芯片里.这样带来的问题是传统的共享主存多处理器系统的数据一致性问题在多核处理器中也同样存在,而且还将变得更加突出,主要原因有:(1)多核处理器中I/O和处理器、存储器的关系更加密切、耦合度更高;(2)处理器核间的进程调度将会变得更加频繁;(3)SOC设计使得片上网络的延迟和I/O路径延迟都将大大降低;(4)片内的共享Cache在提高性能的同时,也使得数据不一致的可能性变得更大;(5)I/O为多个处理器核所共享,I/O瓶颈问题变得尤为严峻.

I/O子系统可连接的I/O设备多种多样,I/O访问特性也是千差万别.面对这样复杂、多变的I/O访问特性,采用单一的I/O一致性处理方法,很难满足不同的I/O应用需求.有效改善I/O性能的方法之一就是对不同类型的I/O访问进行区别对待,选择最吻合其数据访问特性、能充分发挥系统性能的I/O一致性处理方法进行处理.

2 I/O数据访问特性

不同的I/O数据访问特性对I/O一致性处理方法的要求不一样,要想很好地解决I/O一致性处理问题,

收稿日期:2010-08-16;修回日期:2010-09-29
基金项目:核高基重大专项-面向大规模科学计算的高性能多核CPU项目

必须先准确地分析 I/O 数据访问特性.W.Hsu 等[4,5]对存储系统的 I/O 数据负载特性进行了一定的分析和研究,但都只是分析了 I/O 数据负载的部分特性,缺乏对 I/O 数据负载特性全面的刻画和描述.我们从下面七个方面刻画了 I/O 数据的访问特性.

(1)I/O 与计算亲和度:I/O 与计算亲和度刻画了 I/O 数据被处理器核访问的可能性,或者处理器核处理后数据被 I/O 访问的可能性.从一个侧面反映了 I/O 和处理器核对数据的共享度;

(2)I/O 访问时间聚合度:I/O 访问时间聚合度是从时间轴上观察 I/O 访问的分布特性,反映了 I/O 访问的突发特性;

(3)I/O 访问空间聚合度:I/O 访问空间聚合度是从空间轴上观察 I/O 访问的分布特性,描述了 I/O 访问的数据大小分布,反映了 I/O 访问粒度大小;

(4)I/O 同步特性:I/O 同步特性是指 I/O 访问所采用的同步方法;

(5)I/O 访问数据写分布:I/O 访问数据写分布特性是描述 I/O 写在 I/O 访问流中的分布和比例;

(6)I/O 数据为处理器核的共享度:I/O 数据为处理器核的共享度描述了 I/O 访问的数据被多个处理器核所共享的情况;

(7)I/O 数据的生命周期:I/O 数据的生命周期描述了 I/O 数据有效的生命窗口.

以上从七个方面刻画了 I/O 数据的访问特性,对 I/O 数据访问的时间、空间、共享度等特性进行了限定和描述.这些特性都对 I/O 一致性处理方法产生了重要影响,要想高效地实现 I/O 一致性处理,必须充分考虑 I/O 数据访问的这些特性,做到区分对待,如果不同特性之间出现冲突,还必须进行权衡和折中,使得性能/开销比最大,以尽可能小的代价获得尽可能大的性能.

3　自适应混合 I/O 一致性方法-AHIOCM

一致性模型[10,11]为程序员定义了硬件允许的行为,是软硬件接口的体系结构视图的概念.I/O 一致性保证了 I/O 参与操作的数据与处理器核和存储器视图数据的一致性.I/O 一致性处理的效率将直接影响 I/O 性能,另一方面 I/O 数据的访问特性也将极大影响 I/O 一致性处理的效率.我们在充分考虑 I/O 数据访问特性的基础上,提出了一种自适应多核 I/O 一致性处理方法-AHIOCM(Adaptive Hybrid I/O Consistency Method).其实现思想是:系统中支持多种 I/O 一致性处理方法,不同的 I/O 一致性处理方法对某些特定的 I/O 数据访问是较优的;对于不同的 I/O 应用,系统可以静态配置或者动态自适应地选择适合的 I/O 一致性处理方法.I/O 虚拟化[6,7,12]使得不同的 I/O 域可以单独地进行虚实地址转换,地址转换的关键是 IOMMU 表,我们可以通过在 IOMMU 表中增加 I/O 一致性处理方法选择位,选择位的不同值将决定 I/O 控制单元选择哪种 I/O 一致性处理方法;I/O 一致性处理方法的静态配置是指操作系统或者设备驱动根据设备特性和工作模式直接静态确定,自适应的动态选择是根据系统的统计信息和性能反馈自动分析 I/O 数据访问特性,并动态地选择 I/O 一致性处理方法.

3.1　实现算法

图 1 说明了 AHIOCM 方法的处理流程,从图中我们可以看出对于特定的 I/O 应用,首先必须进行通过静态或动态方式进行 I/O 一致性处理方法选择;确定了 I/O 一致性处理方法后,所有该应用的 I/O 访问报文都进行相应一致性处理后,送往存储器;最后还要对硬件处理的一些信息进行收集和统计,并将这些信息反馈给动态一致性确定模块,实现对 I/O 一致性处理器方法的自适应动态调整.

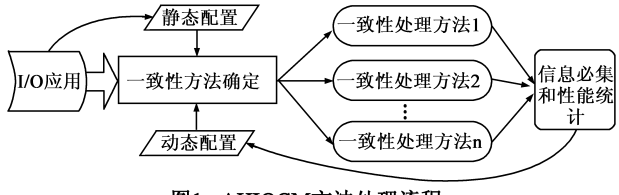

图1　AHIOCM方法处理流程

3.2　关键技术

3.2.1　I/O 一致性处理方法

由于系统实现开销的限制,不可能实现太多种 I/O 一致性处理方法,只能支持对几种典型 I/O 应用表现较好的方法.我们根据前面对 I/O 数据访问特性的分析,进行综合考虑后,选择支持三种 I/O 一致性处理方法:软件实现的 I/O 一致性方法(SIOCM-Software I/O Consistency Method)、硬件支持的写失效处理方法(HIOICM-Hardware I/O Invalidation Consistency Method)和硬件支持的写失效加读共享方法(HIOISCM-Hardware I/O Invalidation and Shared Consistency Method).

(1)SIOCM:软件实现的 I/O 一致性方法[8]是由软件负责 I/O 数据的一致性维护,硬件不进行任何 I/O 一致性处理,数据直接写到存储器中或者直接从存储器中返回.这种方法主要是针对 I/O 和处理器核亲和度比较低,数据传输粒度比较大或生命周期比较短的应用.SIOCM 方法的硬件实现开销比较小,软件处理开销比较大,性能比较差.

(2)HIOICM:硬件支持的写失效处理方法是由硬件维护 I/O 数据的一致性,一致性处理主要由硬件完成,写失效是指当有 I/O 发起的 DMA 写时,目的存储器位置对应的其它数据副本将被失效,且写数据不进共享 Cache,避免大量 I/O 数据对 Cache 污染造成的 Cache 性

能急剧下降的情况.这种方法主要是针对 I/O 和处理器核亲和度比较高,但数据传输粒度大的应用.HIOICM 硬件实现开销比 SIOCM 大,但性能优势比较明显,且对共享 Cache 影响较小.

(3)HIOISCM:硬件支持的读共享方法是在写失效方法的基础上允许 I/O 数据进共享 Cache,对于 I/O 的 DMA 写数据除了更新到存储器外,还将在共享 Cache 中保留副本,从而减小共享读延迟.这种方法主要是针对 I/O 和处理器核亲和度高,但数据量不是太大的情况,允许 I/O 数据进共享 Cache 将大大降低处理器核对 I/O 数据访问的延迟和开销.HIOISCM 方法硬件实现开销和 HIOCM 相当,相对于 SIOCM 方法性能优势明显,相对于 HIOCM 方法,利用共享 Cache 资源换取共享读性能的提高.

3.2.2 I/O 定序方法

I/O 定序功能实现 I/O 请求的定序.I/O 定序子模块维护了 Order 队列和 ByPass 队列两个 I/O 队列,用于地址比较,记录报文相关性,其中所有需要严格定序的报文进入 Order 队列,可以乱序发送的报文则进入 ByPass 队列.每接收一个新的请求后,都要更新上述两个队列,更新规则是:

(1)新到一个写请求进入 Order 队列时,依赖标记根据 Bypass 队列中最新的写和 Bypass 队列中的最新的地址匹配项向 Bypass 队列项中写入一个依赖指针并且设置依赖指针有效;

(2)新到一个读请求进入到 Order 队列时,依赖标记根据 Bypass 队列中的最新的地址匹配项向 Bypass 队列项中写入一个依赖指针并且设置依赖指针有效;

(3)新到一个请求进入到 Bypass 队列时,依赖标记根据 Order 队列中的最新的地址匹配项向 Order 队列项中写入一个依赖指针并且设置依赖指针有效.

3.2.3 一致性方法选择

I/O 一致性方法的选择是本方法实现的一个难点,我们支持静态和动态两种方式,如果采用静态方式,由软件根据 I/O 设备特性静态确定的,在这里我们不详细讨论;对于动态方法,由硬件根据系统的执行状态和统计信息动态决定采用哪种一致性处理方法.

系统的执行状态和统计信息通过性能计数器采集.硬件周期性地根据收集到的信息动态调整一致性处理方法.

性能计数器采集的信息主要有:

(1) SCM(Shared Cache Miss Rate):共享 Cache 的平均失效率; $SCM = \sum_{i=1}^{i=n} CM_i/n$,其中 n 为共享 Cache 的 bank 数, CM_i 为第 i 个 Cache 的失效率;

(2) SCU(Shared Cache Usage Rate):共享 Cache 的平均占用率; $SCU = \sum_{i=1}^{i=n} CU_i/n$,其中 n 为共享 Cache 的 bank 数, CU_i 为第 i 个 Cache 的占用率;

(3) IOB_I(I/O Bandwidth):单位时间内从发出的来自 I/O 域 I 的 DMA 写的平均数据大小(B/s); $IOB_I = \sum_{s=0}^{s=n} D_s/T$,其中 n 为时间 T 内收到来自 I/O 域 I 的 DMA 写报文数量, D_s 为每次 DMA 写的大小;

(4) ROW_I(Read Percent of Window):统计时间窗口内收到的来自 I/O 域 I 的 DMA 读请求比率; $ROW_I = Num_r/(Num_w + Num_r)$;

为了计算选择哪种处理方法,我们还设置了以下参数:

γ_{scm}: SCM 的影响系数($0 \leqslant \gamma_{scm} \leqslant 1$);

γ_{scu}: SCU 的影响系数($0 \leqslant \gamma_{scu} \leqslant 1$);

γ_{iob}: IOB_I 的影响系数($0 \leqslant \gamma_{iob} \leqslant 1$);

γ_{row}: ROW_I 的影响系数($0 \leqslant \gamma_{row} \leqslant 1$);

α_{siocm}, α_{hioicm}, $\alpha_{hioiscm}$:为三种一致性处理方法的补偿因子,可以由操作系统或驱动程序配置;

$B_{\max}$:I/O 控制单元流出的理论最大带宽;

设 S_{siocm}, S_{hioicm}, S_{hioscm} 为每种一致性处理方法对应的选择因子,选取选择因子最大的一致性处理方法进行处理,选择因子的计算公式如下:

$$S_{siocm} = (1 - SCM) * \gamma_{scm} + (1 - SCU) * \gamma_{scu} + IOB_I * \gamma_{iob}/B_{\max} + \alpha_{siocm}$$

$$S_{hioicm} = (1 - SCM) * \gamma_{scm} + SCU * \gamma_{scu} + \gamma_{iob} * IOB_I/B_{\max} + (1 - ROW_I) * \gamma_{row} + \alpha_{hioicm}$$

$$S_{hioiscm} = SCM * \gamma_{scm} + (1 - SCU) * \gamma_{scu} + (1 - IOB_I/B_{\max}) * \gamma_{iob} + ROW_I * \gamma_{row} + \alpha_{hioiscm}$$

$$S = MAX\{S_{siocm}, S_{hioicm}, S_{hioiscm}\}$$

我们通过实验选择了最合适的影响系数.软件也可以根据应用特性进行配置.

4 性能测试

本节将基于我们自主设计的高性能多核处理器芯片对自适应多核处理器 I/O 一致性处理方法(AHIOCM)进行性能测试和分析.我们设计的多核处理器芯片采用 SoC 多核系统结构,片内集成 8 个多线程处理器核,每个处理器核支持 8 个线程,全芯片支持 64 个线程.每个处理器核有私有的一级指令和数据 Cache,所有处理器核共享 4MB 二级 Cache.芯片采用 65nm 工艺,处理器主频为 800MHz,Die 大小为 17 × 20,全芯片 3.2 亿个晶体管,采用 HFCBGA 封装、总引腿数为 1517,功耗为 50W 左右.支持四个 DDR3 通道和一个 PCI Express2.0 接口.下面将对 AHIOCM 方法进行性能测试,并分析其实现开

销.

4.1 AHIOCM 性能

下面我们对处理器芯片进行 I/O 性能测试,评测 AHIOCM 方法的实际工作效果.目前 AHIOCM 方法的影响因子和补偿因子还不能智能获得,我们根据实验经验设置补偿因子 α_{hioicm} 为 1/3、$\alpha_{hiosicm}$ 为 1/3、α_{siocm} 为 1/4,设置影响因子 γ_{scm} 为 0.5、γ_{scu} 为 0.7、γ_{iob} 为 0.9、γ_{row} 为 0.8.

测试程序主要采用磁盘测试程序和高速通信接口测试程序两种,磁盘 I/O 和高性能通信接口 I/O 代表了两种典型的 I/O 应用特性,能够反映 AHIOCM 算法对典型 I/O 应用的加速效果.高速通信接口测试测试环境采用两块多核处理器芯片通过 PCI Express 接口分别连接一块自主设计的高性能通信接口卡,高性能通信接口卡,支持用户级通信,主机接口为 16 通道的 PCI Express2.0 接口,网络接口为 10GB/s 的光纤接口,单向通信峰值带宽为 6.3GB/s,双向通信峰值带宽为 9.8GB/s.测试采用经过一级路由点到点连接.分别测试了采用 AHIOCM 方法、SIOCM 方法、HIOICM 方法和 HIOSICM 方法时,点到点通信的单向带宽,数据传输粒度从 8 字节到 8M 字节.

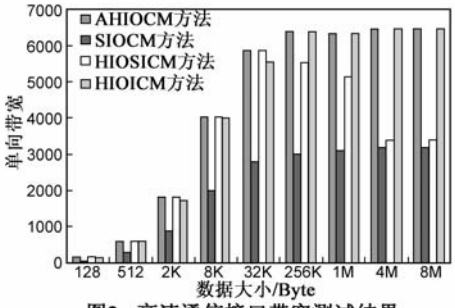

图2　高速通信接口带宽测试结果

从图 2 测试结果我们可以看出,通信接口的带宽随着传输数据大小增加呈线性递增,当传输数据大小达到 256KB 左右,达到峰值带宽,峰值单向传输带宽达到 6.3GB/s.而且对于所有不同粒度的数据传输 AHIOCM 方法获得传输带宽都是最高的.

磁盘测试中,处理器通过 PCI Express 总线外接一个 LSI1068e SAS 卡,再连接 SATA 磁盘.磁盘测试采用 iozone[9]测试程序.总测试数据集大小为 16GB,测试划分的数据块从 32KB 到 1MB,分别测试顺序读写性能和随机读写性能,最后取平均值.图 3 为 iozone 测试结果,从测试结果我们可以看出对顺序读、顺序和随机写,AHIOCM 方法比其它方法获得了更高的平均 I/O 性能,但对于随机读,由于参数不是最佳,因此没有选择性能最优的一致性处理方法进行一致性处理,性能表现不佳,这个问题可以通过优化选择因子和加强算法的训练来解决.

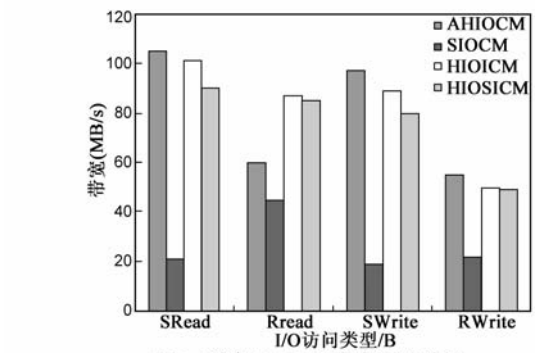

图3　磁盘IOZONE带宽测试结果

4.2 实现开销

下面我们将讨论一下 AHIOCM 方法带来的性能开销和逻辑实现开销.AHIOCM 方法由于需要通过计算确定工作的一致性处理方法,由于计算过程都采用硬件逻辑实现,计算和选择带来了大约 3 个时钟周期(约 6ns)的性能开销,由于从 I/O 设备到存储器的路径延迟通常大于 200ns,因此性能开销可以忽略不计.另外,由于芯片实现了多种算法,并且需要性能统计和选择判断逻辑来保证针对特定的 I/O 应用选择最优的一致性处理方法进行处理,造成芯片的逻辑门数增加了大约几 K 左右,这相对于全芯片几千万门来说影响非常小.因此 AHIOCM 虽然带来了一定的性能和逻辑开销,但相对于所获得的性能提升来说可以忽略不计.

5 结束语

高效的 I/O 一致性处理方法是改善 I/O 系统性能的关键技术之一.然而单一的 I/O 一致性处理方法很难满足多样的 I/O 应用需求.本文在深入 I/O 数据访问特性的基础上,提出了一种自适应的 I/O 一致性处理方法,很好地满足多种 I/O 应用的需求.该方法已经用到多核处理器芯片的研制中,实际芯片测试表明能够获得很好的性能加速效果.下一步将采用更智能的方法选择影响因子,使得影响因子选择更加合理;并将引入机器学习的方法,提高 AHIOCM 方法的智能性,减少人工的干预,从而获得更好的性能和易用性.

参考文献

[1] Mark Smotherman. A Sequencing-Based Taxonomy of I/O Systems and Review of Historical Machines[M]. San Francisco: Morgan Kaufmann Publishers Inc, 1989. 10 – 15.

[2] B Nitzberg, V Lo. Distributed shared memory: a survey of issues and algorithms[J]. IEEE Computer, 1991: 24(8): 52 – 60.

[3] A Forin, J Barrera, R Sanzi. The shared memory server[A]. USENIX Winter Conference[C]. Baltimore: USENIX Press, 1989. 229 – 243.

[4] W Hsu, A Smith. Characterization of I/O traffic in personal and

server workloads[J]. IBM Systems Journal, 2003, 42(4): 347 – 372.
[5] B Pasquale et al. A static analysis of I/O characteristics of scientific applications in a production workload[A]. Conference on High Performance Networking and Computing archive[C]. New York: ACM Press, 1993. 388 – 397.
[6] AMD Corp. AMD I/O Virtualization Technology (IOMMU) Specfication[R]. 2006. 17 – 21.
[7] Intel Corp. Intel Virtualization Technology for Directed I/O Architecture Specfication[R]. 2006. 13 – 16.
[8] David E. Culler, Jaswinder Pal Singh, Anoop Gupt. 并行计算机体系结构:硬件与软件结合的设计与分析[M]. 北京: 机械工业出版社, 2002. 612 – 668.
[9] IOzone Filesystem Benchmark[Z]. http://www.iozone.org.
[10] Goodman J. R. Cache consistency and Sequential Consistency [R]. SCI Committee, 1991. 1 – 4.
[11] Strenstrom, P, A survey of cache coherence schemes for multiprocessors[J]. IEEE Computer 1990, 23(6): 12 – 24.
[12] Sun Microsystems, Inc. UltraSPARC Virtual Machine Specification[R]. Santa Clara, 2008. 5 – 293.
[13] 邓让钰,陈海燕,等.一种异构多核处理器的并行流存储结构[J].电子学报,2009,37(2):312 – 317.
Deng Rangyu, Chen Haiyan, et al. A parallel stream memory architecture for heterogeneous multi-core processor[J]. Chinese Journal of Electronics, 2009, 37(2): 312 – 317. (in Chinese)

作者简介

郭御风 男,1979 年生于江西九江.国防科学计算大学计算机学院讲师.研究方向为高性能计算、微处理器设计和海量信息存储.
E-mail: yfguo21@yahoo.com.cn

李 琼 女,1967 年生于湖南湘潭.国防科学计算大学计算机学院教授,研究方向为高性能计算和海量信息存储.