第43卷 第2期 西 安 交 通 大 学 学 报 Vol. 43 №2
2009年2月 JOURNAL OF XI′AN JIAOTONG UNIVERSITY Feb. 2009

面向Cell宽带引擎架构的异构多核访存技术

冯国富，董小社，丁彦飞，王旭昊
（西安交通大学计算机科学与技术系，710049，西安）

摘要：针对Cell宽带引擎架构(CBEA)多核高性能处理器要求软件显式地对分层存储结构进行管理，带来架构的可编程性及性能等问题，提出了一种基于CBEA的异构多核访存技术.将CBEA访存分为批量访存和按需访存；通过合理部署数据缓冲区来减小批量访存计算中的片内访存开销，利用支持粗粒度访问的软件管理cache及数据预取来降低按需访存的片外访存开销；以访存接口库的方式来改善软件的可编程性.实验结果表明，所提技术的访存接口库在批量访存方式下的性能比ALF和CellSs提高了30%～50%，按需访存中软件管理cache性能比CBE软件开发工具包提高了20%～30%，4路数据预取访存比单路缓存的性能提高约50%.

关键词：异构多核；访存技术；分层存储结构；Cell宽带引擎架构
中图分类号：TP393 **文献标志码**：A **文章编号**：0253-987X(2009)02-0001-05

A Memory Access Technology of Heterogeneous Multi-Core System Based on Cell Broadband Engine Architecture

FENG Guofu, DONG Xiaoshe, DING Yanfei, WANG Xuhao
(Department of Computer Science and Technology, Xi′an Jiaotong University, Xi′an 710049, China)

Abstract: A memory access technology is proposed to solve the problems of programming and performance caused by explicit management for multi-level memory hierarchies of Cell broadband engine architecture (CBEA). The access to main memory on CBEA is classified into two major types: bulk data transfer and on-demand data transfer. For bulk data transfer, the overhead of on-chip memory access is reduced by proper organization of data buffer. Meanwhile, a software-managed cache supporting coarse-grained access and data pre-fetch are adopted to enhance the performance of on-demand off-chip data transfer. In order to facilitate programming, a memory access library based on the proposed methods is implemented. Experimental results show that the performance of applications based on the proposed bulk data transfer technology is about 30%-50% higher than that of applications based on ALF and CellSs. For on-demand data transfer, the proposed software-managed cache performs about 20%-30% better than that of CBE SDK does, and the data prefetch technology based on four-buffer gets about 50% performance increase compared to single-buffer cache.

Keywords: heterogeneous multi-core; memory access technology; multi-level memory hierarchy; Cell broadband engine architecture

目前，多核及异构多核已成为流行架构，基于Cell宽带引擎架构(CBEA)的Cell处理器[1]提供了很高的理论计算峰值，但Cell要求显式地管理其多层次的存储结构，这给应用开发和移植带来了不便.

基于Cell的编程模型，如ALF、MPI Microtask[2]及CellSs[3]等，一般是直接将整个计算子任

收稿日期：2008-04-20. 作者简介：冯国富(1971—)，男，博士生；董小社(联系人)，男，教授，博士生导师. 基金项目：国家高技术研究发展计划资助项目(2006AA01A109，2004AA111110)；国家自然科学基金资助项目(60773118).

务置于计算核的本地存储器,以提高计算性能.这些编程模型的计算任务受限于本地存储容量,另外ALF还与传统开发模型有较大的差别,不便于应用的开发和移植.

由于科学计算的可扩展性,所以支持不受限的可访内存空间更符合人类的思维与编程习惯.但是,在异构多核架构中,协处理单元对大容量的主存层次的访问要通过直接存储器访问(DMA),易用性和性能均受到限制.软件管理cache从用户和软件角度扩展了对主内存的访问,IBM公司的软件开发工具包CBE SDK及参考文献[4]的技术均支持该项技术,但访存效率不高.

本文针对CBEA分层存储结构,将访存分为批量访存和按需访存两种模式,分析了其特性及工程科学计算的访存特征,并通过数据缓冲区的合理组织、数据预取和支持粗粒度访问的软件管理cache,达到对主存的高效访问,实现了相应的CBEA访存接口库.

1 Cell硬件架构

Cell处理器内置1个主处理单元(PPE)和8个协处理单元(SPE),它们通过片内单元内连总线(EIB)与主存和IO相连.Cell处理器的计算能力主要来自SPE,其存储系统由寄存文件(Register File)、片上SPE本地存储(SRAM)和片外存储(DRAM)组成.SPE没有硬件cache,每个SPE由256 kB的可直接寻址的本地存储(LS)和支持单指令多数据流(SIMD)的矢量计算引擎(SPU)组成.SPU只能直接访问LS中的数据和代码,当SPE上运行的代码和数据量超过256 kB时,数据和代码则需要借助DMA操作从片外主存中获取.

2 Cell访存

2.1 访存分类

如果计算任务代码大小相对固定并可忽略,那么根据计算任务的访存范围,可将计算任务划分为访存范围小于LS、大于LS及访存范围不可确定的非规则访存3种情况.当访存范围小于LS时,可以使用批量访存(bule data transfer)完成数据传输,即在计算前将访存范围内数据成批地由主存移动到LS的数据缓冲区中,在计算完成后再成批写入主存.批量访存可充分利用架构提供的高带宽,同时计算过程因无需考虑数据传输而充分发挥SPU的计算效率,整体性能表现出色.但是,这种访存要求应用数据易分割,并提供相关分割信息,对应用程序编写和移植要求较高.

当访存范围大于LS或不可确定时,若将代码和数据按批量访存模式置入LS可能会导致任务崩溃.对于这两种情况,只能在计算任务中通过DMA操作按需将数据写入LS中较小的数据缓冲区内并加以使用.本文将根据计算需要访问主存的方式定义为按需访存.

按需访存对计算任务访问的数据区域不做限制,对应用数据分割要求不高,易于应用程序编写和移植.这种访存虽然使用了软件管理cache或类似方法来提高数据复用率,但在每次使用数据前需检查数据是否在LS之中,所以仍存在可编程性和效率不高等问题.

本文根据LS容量与计算任务对LS的需求,将CBEA下的访存分为批量访存和按需访存,批量访存用于计算子任务代码及其访存范围之和小于LS容量时的情况,按需访存用于计算子任务代码及其访存范围之和大于LS容量或为不确定时的情况.

2.2 不同访存模式的性能差异及原因分析

在单个SPE上,本文以批量访存、按需访存执行分块矩阵乘法 $\boldsymbol{C}=\boldsymbol{A}\times\boldsymbol{B}$ 为例,来分析两种访存方式的性能差异,其中 $\boldsymbol{A}$、$\boldsymbol{B}$ 均为512×512的单精度浮点矩阵.通过调整分块大小来模拟具有不同访存范围的计算任务,当分块为32×32时,$\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$ 的3个分块分别占用了32×32×4 B=4 096 B存储空间,3个分块可以全部载入LS;当分块为256×256时,$\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$ 的3个分块分别占用了262 114 B本地存储空间,且每个分块均大于LS容量.

使用ALF和CellSs基于批量访存实现了512×512矩阵乘法,其执行时间如表1所示.使用CBE SDK软件管理cache模拟单缓冲区按需访存,实现了分块为32×32和256×256的512×512矩阵乘法,在2种分块情况下每个矩阵的cache行分别为8 192 B和12 288 B,执行时间见表1.

表1 分块512×512矩阵乘法的执行时间比较

分块	执行时间/s		
	ALF	CellSs	单缓冲区按需访存
32×32	1.13	1.64	4.93
256×256	—	—	19.49

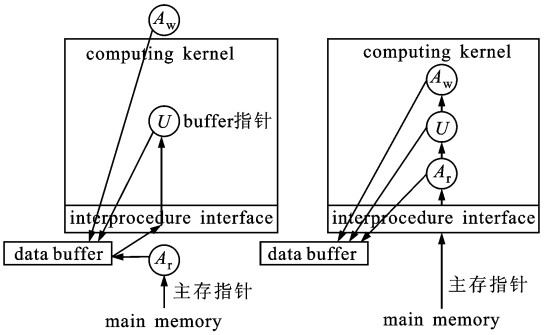
由表1知:对于同样规模的计算任务,批量访存性能最好;小数据的按需访存次之,执行时间大约为批量访存的4倍左右;大数据的按需访存最差,执行

时间远大于批量访存. 性能下降的主要原因是缓冲区管理开销及 DMA 操作引入的通信迟延，前者来自处理器片内访存，后者主要与 SPE 访问片外主存相关.

2.3　SPE 片内访存性能及片内缓冲区管理分析

2.3.1　*批量访存*　数据使用前，计算任务由访问点 A_r 移动到 LS 的数据缓冲区中，并在数据使用点 U 直接使用缓冲区中的数据；数据使用后，再在访问点 A_w 将脏数据写回主存. 在批量访存中，影响性能的主要因素是数据缓冲区的引用，而数据缓冲区的位置对性能的影响并不大.

如图 1a 所示，当 A_r 和 A_w 在计算核心函数的外部时，数据缓冲区指针通过过程间调用传递给计算核心函数. 由于编译器对指针的处理是非常谨慎的，所以计算任务在编译时通常无法进行深度优化. 如图 1b 所示，若将 A_r 和 A_w 植入计算核心函数的内部，此时通过过程参数传递的是主存指针. SPU 不能直接访问主存，对编译器而言主存地址仅是长整型数，而不再是一个地址指针，从而打消了编译器优化时的一些限制，可以显著提高访存性能.



(a)数据访问点外置　　(b)数据访问点植入

图 1　批量访存缓冲区引用

表 2 是以 512×512 分块矩阵乘法为例的缓冲区引用对访存的影响. 由表 2 可知，数据访问点的植入可使编译器充分发挥优化作用，使基于批量访存的计算性能提高约 45%.

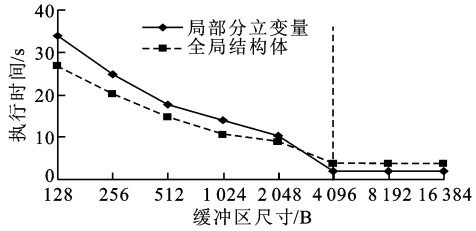
表 2　数据缓冲区引用对批量访存的影响

分块大小	执行时间/s	
	植入数据访问点	数据访问点外置
32×32	0.64	1.11
64×64	0.59	1.03
128×128	0.56	1.02

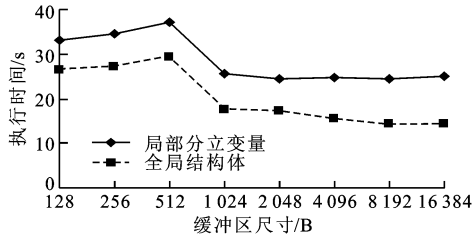
2.3.2　*按需访存*　数据缓冲区及其管理数据在 LS 中的位置和组织方式变成性能敏感因素. 以 512×512 分块矩阵乘法，分块为 32×32 为例，来模拟访存范围不可确定但实际小于 LS 情况的按需访存. 如图 2a 所示，当数据缓冲区小于访存范围 (4 096 B)，即数据缓冲区＜访存范围＜LS 容量时，使用全局结构的组织方式性能较好；当数据缓冲区大于访存范围，即访存范围＜数据缓冲区＜LS 容量时，局部定义的分立变量性能较全局数据结构体可提高 45%左右.

如图 2b 所示，当实际访存范围大于 LS 时，如分块大小为 256×256，即数据缓冲区 ＜LS 容量＜访存范围，使用全局结构体的性能总是优于局部分立变量的组织方式.

由以上实验及分析可以总结得出：对于缓冲区及其管理数据，当数据缓冲区小于访存范围时，由于要经过多次的 DMA 操作，所以全局结构体定义要优于局部分立变量定义；当访存范围小于数据缓冲区时，计算只需要进行较少的 DMA 操作，开销主要为 cache 的 lookup 操作，此时局部分立变量定义要优于全局结构体定义.



(a)小范围访存 32×32 分块矩阵乘法



(b)大范围访存 256×256 分块矩阵乘法

图 2　按需访存中数据缓冲区对性能的影响

2.4　SPE 访问片外主存及优化技术

2.4.1　*多路数据预取*　多路数据预取技术通过异步 DMA 操作将计算与通信重叠来隐藏访存延迟，相对于组相联 cache，数据预取仅占用了少量的 LS 空间. 在缓冲区命中率不高的情况下，数据预取效果明显，但其需要程序员提供数据访问规则及数据界限信息. 在原本缓冲区命中率高的应用中，由于需要额外的管理开销，数据预取反而会引起性能恶化.

如矩阵乘法 $C=A\times B$，分块 B 进行预取可大大提高性能，而对于顺序访问的分块 A 和 C 则不然. 表3是以512×512分块矩阵乘法为例的性能差异分析结果.

表3 多路数据预取对性能的影响

分块大小	执行时间/s		
	情况1	情况2	情况3
32×32	4.93	4.12	4.99
256×256	19.49	9.06	9.22

情况1：A、B、C 均为单缓冲区；情况2：A、C 为单缓冲区，B 为4路缓冲区；情况3：A、B、C 均为4路缓冲区.

2.4.2 粗粒度cache管理 在传统工程与科学计算中，数组是常用的重要数据结构，其占用的内存空间常随问题规模动态变化，访存模式也往往是非规则的，需要使用效率较低的按需访存.

应用对数据的访问有一定的局部连续性，如在分子动力学模拟中，最耗时的计算是粒子间的非键合力计算. 以Gromacs[5] kernel 010为例，描述一个粒子位置与受力的pos和faction数组需要 X、Y、Z 3个方向上的分量元素，所以在单精度浮点数的情况下，它们均需要访问3×4共12 B的连续存储单元，而在kernel 112中，会连续对9个pos和faction的相邻元素，即9×4共36 B的连续存储单元进行访问. 这种现象在科学计算中普遍存在，甚至在有些程序设计中会利用聚合分散方法将要访问的数据刻意连续存放，以达到优化DMA访存的目的. 通常，软件管理cache对每一个主存地址的访问均要调用一次cache管理逻辑，这将增加调用管理逻辑和访存的次数，若将一段连续的数据定义为粗粒度的cache管理单元，并由一次单独的cache调用来管理，不仅可减少cache管理逻辑带来的开销和访问主存次数，同时也可充分地利用架构提供的高带宽访存总线.

3 访存接口库设计与调用

基于以上分析，本文设计了不同情况下的基于CBEA的访存策略，并在一个组相联软件管理cache的基础上设计了支持数据预取、粗粒度cache及批量访存的访存接口库.

如图3所示，使用该接口库，不同的变量可以根据对主存的访问模式申请建立自己的数据缓冲区并选用合适的访存策略. 批量访存主存访问点 A_r 和 A_w 可直接植入到如图3所示的计算核心函数内部. 对于按需访存，在数据访问点 A_c 通过软件管理cache访问主存，而对于可写数据，在访问点 A_w 将缓冲区的数据写回主存. 指针cur_buf根据用户选择的接口及数据预取的情况指向相应的数据缓冲区，在数据使用点 U，通过cur_buf使用缓冲区中的数据.

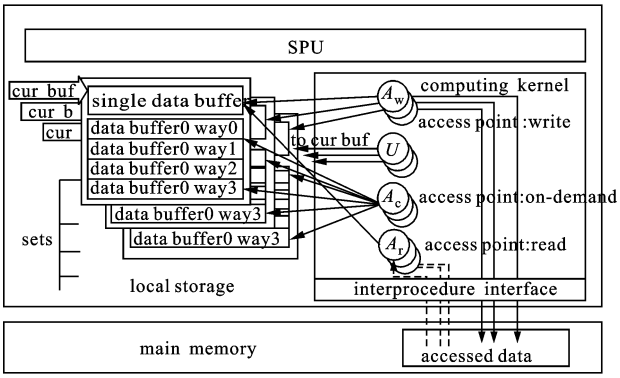


图3 访存接口及调用

用户可在cache接口中设置读取数据单元数，对一段连续的内存数据进行粗粒度管理.

4 验证与测试

基于本文的访存接口库，在PlayStation 3 (PS 3)上对本文访存技术进行了验证. 软件环境为Fedora core 6，CBE SDK 2.1. 由于本文侧重于访存技术研究，所以测试用例仅依赖于编译器自动优化(编译选项-O3)，未采用SIMD等额外优化措施.

4.1 Micro-Benchmark 测试

以512×512分块矩阵乘法和1024×1024稀疏矩阵(LU)分解为例进行测试，使用的矩阵均为单精度浮点矩阵，并取32×32分块矩阵模拟小访存范围的计算任务，取256×256分块矩阵模拟大访存范围的计算任务.

4.1.1 批量访存测试 图4显示了分别使用CellSs、ALF和本文方法的实验结果. 由图4可知，在同等条件下本文方法运行时间分别是ALF和CellSs的70%和45%左右.

4.1.2 按需访存测试 以32×32分块模拟访存范围大小不确定但实际小于LS的访存范围计算任务，结果如图5a所示，本文方法的执行时间大约为CBE SDK的70%. 以256×256分块模拟大访存范围计算任务，分别实现单数据缓冲区、4路数据缓冲

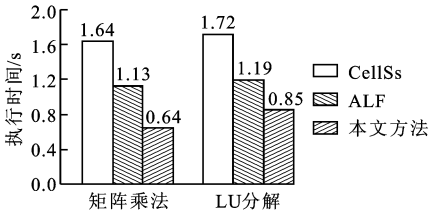


图 4　批量访存实验结果

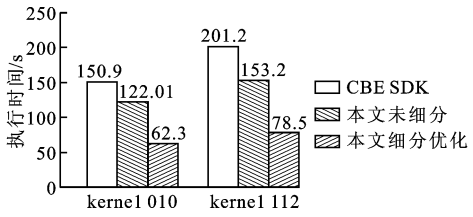
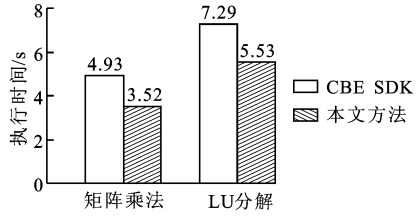
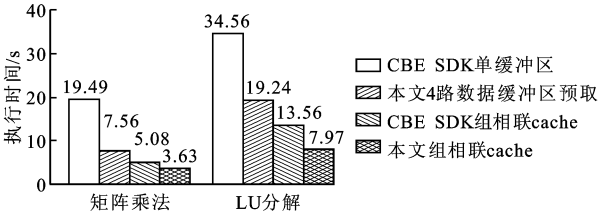


图 6　Gromacs kernel 测试

区预取及组相联 cache 的实验结果如图 5b 所示. 由图可知，基于本文方法在 4 路数据缓冲区预取下的执行时间仅为 CBE SDK 单缓冲区的 40%～60%左右. 在组相联 cache 中，本文方法的执行时间为 CBE SDK 的 60%～70%左右.



(a)小范围访存 32×32 分块



(b)大范围访存 256×256 分块

图 5　按需访存实验结果

4.2　实际应用测试

以 Gromacs Benchmark d. poly 运行 kernel 010 及 Benchmark d. villin 运行 kernel 112 各 1 000 步的执行时间为例进行了测试，结果如图 6 所示. 可见，基于本文按需访存库的执行时间约为 CBE SDK cache 的 80%. 若针对不同数组的访问模式使用不同的库调用，如小数组采用批量访存进行手工细分优化，则执行时间可进一步优化到 CBE SDK 执行时间的 30%左右.

5　结束语

本文基于 CBEA，将访存分类为批量访存和按需访存，通过合理地组织数据缓冲区、数据预取，利用支持粗粒度访问软件管理 cache，可达到对主存的高效访问，并实现相应的 CBEA 访存接口库. 实验结果表明，在两种访存模式下，本文提出的访存技术性能显著超过当前其他相关技术.

参考文献：

[1] GSCHWIND M. Chip multiprocessing and the cell broadband engine [C]// Proceedings of ACM Computing Frontiers. New York, USA: ACM Press, 2006: 1-8.

[2] OHARA M, INOUE H, SOHDA Y, et al. MPI microtask for programming the cell broadband engine processor [J]. IBM Systems Journal, 2006, 45(1): 85-102.

[3] BELLENS P, PEREZ J M, BADIA R M, et al. CellSs: a programming model for the cell BE architecture [C]// Proceedings of the ACM/IEEE SC 2006 Conference on High Performance Networking and Computing. New York, USA: ACM Press, 2006: 86.

[4] EICHENBERGER A E, O'BRIEN J K, O'BRIEN K M, et al. Using advanced compiler technology to exploit the performance of the cell broadband engine [trademark] architecture [J]. IBM Systems Journal, 2006, 45(1): 59-84.

[5] VAN DER SPOEL D, LINDABL E, HESS B, et al. Gromacs: fast, flexible, and free [J]. Journal of Computational Chemistry, 2005, 26(16): 1701-1718.

（编辑　苗凌）