第 34 卷 第 4 期 计 算 机 工 程 2008 年 2 月
Vol.34 No.4 Computer Engineering February 2008
·工程应用技术与实现· 文章编号：1000—3428(2008)04—0247—03 文献标识码：A 中图分类号：TP391

片上多处理器中的 Cache 压缩和接口压缩

肖俊华[1,2]，冯子军[1,2]，章隆兵[1]
(1. 中国科学院计算技术研究所，北京 100080；2. 中国科学院研究生院，北京 100039)

摘 要：提出一种简单的基于频繁值和频繁模式的压缩方法，给出结合 Cache 压缩技术和接口压缩技术的片上多处理器结构。全系统的模拟结果表明 Cache 压缩技术和接口压缩技术能提高片上多处理器中 Cache 的有效容量和 pin 的有效带宽，从而提高系统的性能。实验表明只采用 Cache 压缩技术平均能提高 10%的性能，只采用接口压缩技术平均能提高 5.5%的性能，同时采用 Cache 压缩技术和接口压缩技术平均能提高 12%的性能。

关键词：片上多处理器；Cache 压缩；接口压缩

Cache Compression and Interface Compression in Chip Multiprocessor

XIAO Jun-hua[1,2], FENG Zi-jun[1,2], ZHANG Long-bing[1]
(1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080;
2. School of Graduate, Chinese Academy of Sciences, Beijing 100039)

【Abstract】 This paper proposes a simple frequent value and frequent pattern based compression method, and a chip-multiprocessor design combining cache compression and interface compression techniques. The full system simulation shows that Cache compression and interface compression techniques can increase the effective Cache capacity and effective pin bandwidth, and then improve the performance of system. Experimental results show that only using cache compression can improve performance by 10%, only using interface compression can improve performance by 5.5%, combining cache compression and link compression techniques can improve performance by 12%.

【Key words】 chip multiprocessor; cache compression; interface compression

1 概述

随着大规模集成电路和深亚微米技术的发展，处理器和内存之间性能的间距越来越大，越来越长的访存延迟成为提高处理器性能的主要障碍。大容量的片上 Cache 或激进的延迟容忍技术可用于隐藏访存的长时延，但是简单地增加片上 Cache 容量会增加芯片面积和制造成本，而激进的延迟容忍技术会使得有限的 pin 带宽成为另外一个瓶颈。

指令级并行性的有限性、设计的复杂度以及功耗问题，使得片上多处理器(CMP)成为一种趋势，大量的处理器厂商采用了这种构架，如 IBM 的 Power5，SUN 的 Niagara，Intel 的 Montecito 等。但是越来越多的处理器核集成在单个片上会增加对 Cache 容量和 pin 带宽等共享资源的压力。压缩技术是片外存储系统常用的一种技术，用来增加有效的存储容量，片上 Cache 和接口压缩是一个新的研究方向，用于缓解 Cache 容量和 pin 带宽所带来的瓶颈问题。

Cache压缩技术就是存储压缩的Cache行到Cache中，增加Cache的有效容量，减少片外的失效，从而提高性能。传统的Cache压缩方法[1]是基于字典的压缩，需要构建一个大的字典表，并且压缩和解压的开销比较大，本文提出了一种简单的基于频繁值和频繁模式的压缩方法，易于用硬件实现，又达到了和传统方法可比的压缩率。

接口压缩技术就是在处理器接口部分对通信的地址和数据进行压缩，提高 pin 的有效带宽。大部分接口压缩的研究是保存地址或者数据的高位，减少重复的传输，关注点在于减少功耗，本文采用了和 Cache 压缩类似的压缩方法，关注点在于提高性能。

本文阐述了基于频繁值和频繁模式的压缩方法，提出了集成 Cache 压缩技术和接口压缩技术的 CMP 结构，并在实验环境中模拟测试了采用这 2 种技术对性能的影响。

2 CMP 中 Cache 压缩和接口压缩

2.1 Cache 压缩

对程序行为的观察发现在Cache中一些值或模式频繁出现[2-3]，具有一定的规律性，并且这些数据可以压缩为很少的几位，例如在Cache中频繁出现的 0 值，以及其他只需要 4 位、8 位或 16 位就能存下的小数值。如果将这些数据存放在压缩的格式中，就能腾出空间增加Cache容量，本文的Cache压缩方法就是针对这一现象提出的。

在本文中，Cache 压缩粒度为一个 Cache 行，每个 Cache 行被划分为若干个 32 位的字，每个 32 位的字经压缩之后由 3 位的前缀和压缩后的数据位构成，对应的编码见表 1。与前 7 种情况中的任意一种相匹配的字为可压缩的字，如果 Cache 行中的字是可压缩的字，则被编码为压缩的格式。

基金项目：国家自然科学基金资助项目(60673146)；国家杰出青年基金资助项目(60325205)
作者简介：肖俊华(1975－)，男，博士研究生，主研方向：高性能微处理器设计；冯子军，博士研究生；章隆兵，副研究员
收稿日期：2007-03-20 **E-mail：**xiao_jh@ict.ac.cn

表 1　数据压缩编码

数据值或类型	编码	压缩后的大小
0	000	不用保存
-1	001	不用保存
1	010	不用保存
4 位数的符号扩展	011	4 位
8 位数的符号扩展	100	8 位
16 位数的符号扩展	101	16 位
前 16 位数后面补零	110	16 位
不能压缩的字	111	32 位(原值)

采用上述方法编码，理论上一个 Cache 行能被压缩到任意位，但这样的设计会增加 Cache 管理的复杂性，为了简化设计，本文采用了固定长空间管理变长压缩块和分离的 Cache 访问 2 种技术。固定长空间管理变长压缩块就是分配相同大小的 Cache 空间用于存储压缩块，分离的 Cache 访问就是在地址 tag 比较和数据访问之间添加一层间接访问。以下是 Cache 压缩的具体设计，假定 Cache 是 4 路组相连，Cache 行为 64 B。定义一组为固定长空间，大小为 4×64 B，把一组切分为 32 个 8 B 的段，每个段为 8 B，每个 Cache 行压缩后存储到 1~8 个段中。Cache 中的每个组原先包含 4 个 tag(tag0~tag3)，在此基础上再增加 4 个 tag(tag4~tag7)，这样一组最少能存放 4 个没有压缩的 Cache 行，最多能存放 8 个压缩后的 Cache 行，最多能使得有效容量翻一倍。

Cache 行压缩。检查 Cache 行中的每个字是否为可压缩的字，如果是可压缩的字，编码这个字到更紧缩的格式，如果不是可压缩的字，在该字前面加前缀“111”。Cache 行压缩发生在把没有压缩的 Cache 行填充到 L2 Cache 以及数据从 L1 Cache 写回到 L2 Cache 的时候。其中，L1 Cache 写回到 L2 Cache 的情况比较复杂，分为 4 种情况：(1)原来的数据是非压缩的，写的数据也是不能压缩的，则写入原来的位置；(2)原来的数据是非压缩的，写的数据是可以压缩的，则需要往前调整；(3)原来的数据是压缩的，写入的数据也是可压缩的，并且压缩后的大小不大于原来的段数，则写入原来的位置；(4)原来的数据是压缩的，解压、修改、重新压缩，新压缩的块不能存放在原来的位置，或者它的总长度比原来的压缩块的长度长，称为“数据扩展和胖写”问题，需要后移调整 tags 和数据段，如果空间不够，需要根据 LRU 算法替换若干行。

Cache 行解压缩。首先确定该行在组中的偏移，计算公式为 $seg\_offset(k)=sum(actual\_size(i))\ \{i=1,2,\cdots,k-1\}$。其中，$i$ 为组中第 k 行之前的行；$actual\_size$ 为行 i 的实际大小。然后，并行的前缀解码器计算所压缩字的长度，用来计算每个字起始的位置，最后，根据压缩的格式和前缀编码重新构建解压后的字。解压缩发生在数据从 L2 Cache 读到 L1 Cache 的时候。

2.2　接口压缩

接口压缩也称为互连压缩，就是压缩处理器和片外部件之间的通信数据，使得同样宽度、同样频率的互连，在相同时间内能传递更多的数据，从而增加系统中 pin 的有效带宽。接口压缩需要在处理器和片外部件之间增加少量硬件支持数据的压缩和解压缩。

本文的接口压缩设计只对通信中的数据部分进行压缩，而不对传递的地址进行压缩。如果数据是没有压缩的，则采用与 Cache 压缩同样的压缩算法来压缩，使用的消息格式和 Cache 压缩使用的格式类似，每个数据消息被转化为 1 到 8 个 8 B 的消息段，消息的头包含长度域，表示这个行中段的数目。如果主存是非压缩的，片外内存控制器在从主存到处理器的通路上压缩该行，解压缩是由片外内存控制器在处理器到主存的通路上进行的。

2.3　集成 Cache 压缩和接口压缩的 CMP 结构

本文综合 Cache 压缩和接口压缩技术提出了一种 CMP 结构，见图 1。其包含 8 个单线程的处理器核、共享的 L2 Cache、用于处理器核和 L2 Cache 互连的交叉开关、4 个片上的内存通道以及接口。每个处理器核的结构和 MIPS R10000 类似，4 发射，动态调度，频率为 2 GHz，重排序队列的大小为 128 项，私有的 L1 指令和数据 Cache，各为 32 KB，访问时延为 1 拍。L2 Cache 在处理器核间共享，4 MB，访问时延为 10 拍，切分为 4 个，使用 Cache 行物理地址的低位进行交错。每个内存通道提供 3.2 GB/s 的带宽，pin 的总带宽是 12.8 GB/s，访问内存的时延是 200 拍。

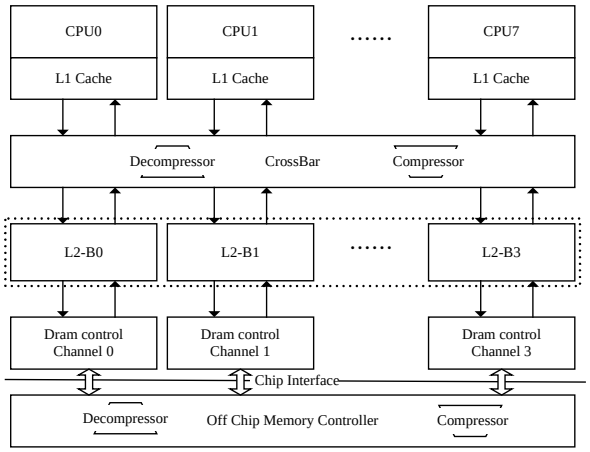

图 1　集成 Cache 压缩和接口压缩的 CMP 结构

在该结构中 L1 Cache 是非压缩的，L2 Cache 是压缩的，在 L1 Cache 和 L2 Cache 之间有硬件实现的压缩器和解压器。一个 L1 Cache 失效的访存操作如果命中一个压缩的 L2 Cache 行，这个行被解压缩，然后进入 L1 Cache，如果命中的是一个非压缩的 L2 Cache 行，则不需要解压直接送到 L1 Cache，如果 L2 Cache 也失效，所请求的行则从内存中取出。在芯片接口处增加了硬件的压缩器和解压器，存放在片外内存控制器模块中，来自内存的数据如果是可以压缩并且压缩率高于一定门槛，则压缩该数据，并通过接口部分送入处理器内部，来自芯片内部的数据如果是压缩的，数据传出接口后需要解压缩后送入到内存。

3　实验方法及结论

本文使用基于 Simics 的全系统模拟器，并使用 GEMS 工具集进行扩展，GEMS 包括一个详细的存储系统时序模型和乱序超标量处理器时序模型。为了评价 Cache 压缩和接口压缩技术的有效性，选用商业测试程序和科学计算测试程序，包括 Apache, OLTP(TPC-C), SPECjbb2000 和 Zeus，以及来自 SPEComp2001 的 OMPfma3d, OMPmgrid, OMPart。

3.1　对 Cache 失效率的影响

图 2 标示出使用 L2 Cache 压缩方法后不同测试程序 Cache 失效率减少的程度，测试结果表明 L2 Cache 压缩会减少大多数测试程序的 Cache 失效率，其中，商业测试程序 Cache 失效率减少 9%~20%，平均减少 13%，SPEComp 的测试程序 Cache 失效率减少相对较小，为 2%~8%，平均减少 4%，主要原因在于大多数浮点程序的 Cache 压缩率比较低，另外对于工作集本身不是很大的程序，如 OMPfma3d，Cache

压缩对其失效率影响很小。

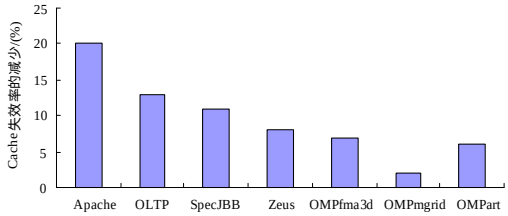

图 2　采用 L2 Cache 压缩方法后 Cache 失效率的减少

3.2　对 pin 带宽需求的影响

测试程序在一个无限可用 pin 带宽的系统中所使用的带宽在本文中被定义为 pin 带宽需求(以下简称带宽需求)，其与每次失效请求的字节数(*bytes/miss*)、每条指令平均失效数(*misses/instr*)、每拍完成指令数(*instr/cycle*)以及处理器频率相关，通过下面的公式计算：

BandwidthDemand(*bytes/sec*)=
(*bytes/miss*)×(*misses/instr*)×(*instr/cycle*)×(*cycle/sec*)

图 3 列出了测试程序在未压缩、L2 Cache 压缩、接口压缩以及 L2 Cache 和接口同时压缩 4 种情况下的带宽需求。在未压缩的情况下，商业测试程序的平均带宽需求为 7.8 GB/s，其中 Apache 的带宽需求为 10 GB/s ;SPEComp 测试程序的平均带宽需求为 16.1 GB/s，其中 OMPfma3d 的带宽需求为 26.4 GB/s。由此可见，在未压缩的情况下商业测试程序不会受到带宽的限制，而多数 SPEComp 测试程序是带宽受限的。从图 3 中还可看出 L2 Cache 压缩后与未压缩的带宽需求几乎一致，可见 L2 Cache 压缩对带宽需求影响甚微，原因在于 Cache 压缩减少了每条指令平均失效数，同时增加了每拍完成指令数。在接口压缩的情况下，测试程序的带宽需求大幅下降，4 个商业测试程序带宽需求减少了 28%~40%，SPEComp 程序带宽需求减少了 15%~30%，原因在于接口压缩减少了每次失效请求的字节数。L2 Cache 和接口同时压缩时，测试程序带宽需求与接口压缩时几乎一致。

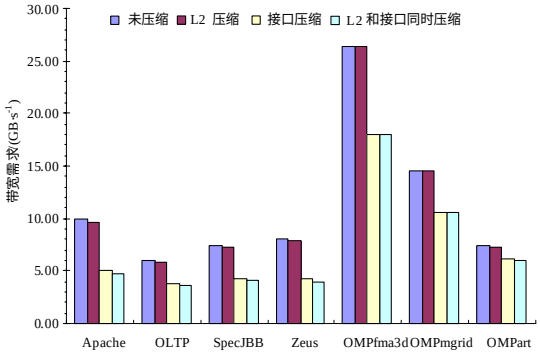

图 3　测试程序带宽需求的影响

3.3　对性能的加速

图 4 列出了 L2 Cache 压缩、接口压缩以及 L2 Cache 和接口同时压缩 3 种情况下，测试程序性能的加速。在只有 L2 Cache 压缩的情况下性能平均提高了 10%，商业测试程序的性能平均提高了 14%，其中 Apache 提高最大为 28%，SPEComp 程序的性能平均提高了 6%。在只有接口压缩的情况下性能平均提高了 5.5%，商业测试程序性能平均提高了 3%，SPEComp 程序的性能平均提高了 8%，其中 OMPfma3d 提高了 18%。在 L2 Cache 和接口同时压缩的情况下性能平均提高了 12%，商业测试程序的性能平均提高了 15%，SPEComp 程序的性能平均提高了 9%。图 4 表明与 SPEComp 程序相比，L2 Cache 压缩技术对商业测试程序的性能提高比较大，其原因在于商业测试程序的工作集较大，压缩率较高。由于商业测试程序本身对带宽的需求不是很大，因此接口压缩技术对性能的加速比 L2 Cache 压缩技术对性能的加速要小。对于诸如 OMPfma3d 等带宽受限的科学计算程序，接口压缩技术可以大幅度提高性能。实验结果同时也表明，Cache 压缩技术和接口压缩技术是正交的关系，可以同时采用这两种技术来提高系统的性能。

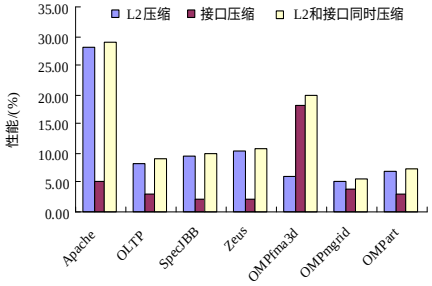

图 4　压缩对性能的加速

4　结束语

片上 Cache 容量和 pin 带宽是未来 CMP 设计中的主要瓶颈，本文对 CMP 中的 Cache 压缩技术和接口压缩技术进行了研究，研究结果表明 Cache 压缩技术能增加 Cache 的有效容量，减少 Cache 的失效率，接口压缩技术能增加 pin 的有效带宽，减少应用程序竞争带宽资源的可能性，综合使用 Cache 压缩和接口压缩技术能大幅提高系统的性能。

参考文献

[1] Hallnor E G, Reinhardt S K. A Compressed Memory Hierarchy Using an Indirect Index Cache[R]. University of Michigan, Technique Report: CSE-TR-488-04, 2004.

[2] Yang Jun, Zhang Youtao, Gupta R. Frequent Value Compression in Data Caches[C]//Proceedings of the 33rd Annual IEEE/ACM International Symposium on Microarchitecture. Monterey, CA, USA: [s. n.], 2000.

[3] Alameldeen A R, Wood D A. Frequent Pattern Compression: A Significance-based Compression Scheme for L2 Caches[Z]. UW-Madision, 2004-04.

(上接第 232 页)

参考文献

[1] Potter M, Jong K D. A Cooperative Coevolutionary Approach to Function Optimization[C]//Proc. of the 3rd Conference on Parallel Problem Solving from Nature. Jerusalem, Israel: Springer Press. 1994: 249-257.

[2] Potter M, Jong K D. Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents[J]. Evolutionary Computation, 2000, 8(1): 1-29.

[3] Popovici E, Jong K D. Understanding Competitive Co-evolutionary Dynamics via Fitness Landscapes[M]. [S. l.]: AAAI Press, 2004.

[4] 康　卓, 李　艳, 刘　溥, 等. 求解函数优化问题的两种异步并行算法[J]. 武汉大学学报, 2002, 48(1): 34.

[5] 詹　炜, 戴光明, 龚文引. 求解函数优化问题的一种高效混合演化算法[J]. 计算机工程与应用, 2006, 42(2): 70-72.

[6] 史彦军. 复杂布局的协同差异演化方法与应用研究[D]. 大连：大连理工大学, 2005-06: 36-41.