面向低功耗共享Cache路适应划分算法研究

方 娟 王 帅 于 璐
(北京工业大学计算机学院 北京 100124)

摘 要 如何提高多核处理器的性能和降低多核处理器中Cache的功耗已经成为下一代多核处理器的研究热点。为了降低片上多核处理器的功耗,基于路适应算法可以采用一种新的动态划分机制,该机制主要由路分配模块和动态功耗控制模块组成。路分配模块在程序运行过程中根据处理器核所运行线程的工作集的大小调整处理器核所分配的Cache路。动态功耗控制模块利用程序运行的局部性原理,将处理器核所运行线程的工作空间控制在少数Cache路中。关闭剩余的Cache路,从而达到降低Cache功耗的目的。该机制使用Simics全系统模拟平台模拟多核处理器,并用SpecOMP测试集测试了系统的性能和功耗。与传统的Cache(Conventional L2 Cache,C-L2)相比,其IPC提高了9.27%,功耗降低了10.95%。

关键词 路自适应,低功耗,动态划分
中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.07.006

Research of Lower Power Oriented Way-adaptive Partition Algorithm in Shared Cache of CMP

FANG Juan WANG Shuai YU Lu
(College of Computer Science, Beijing University of Technology, Beijing 100124, China)

Abstract Improving processor performance and reducing energy consumption of the Cache have become research topic of the next-generation processor. To reduce energy consumption in CMP, a new mechanism based on dynamical way-adaptable Cache can be adopted. The mechanism mainly consists of way reallocate module and dynamic power control module. Way reallocate module reassigns ways between cores based on thread's working set on the execution of the program. Our mechanism implements low power consumption by dynamic power control module. The proposed scheme based on dynamical way-adaptable Cache is implemented and simulated by Simics. We applied several programs selected from SpecOMP as benchmarks. Compared with traditional cache(Conventional L2 Cache, C-L2), its IPC increases by 9.27%, and power consumption reduces by 10.95%.

Keywords Way-adaptable, Low power consumption, Dynamical reallocate

1 引言

近年来多核处理器体系结构,特别是片上多核(Chip Muti-Processors,CMP)[1]体系结构已经被学术界广泛看好和接受。该体系结构已经被学术界认为是处理器体系结构的发展趋势。CMP通过在一块芯片上集成更多的核数来达到线程级的并行,从而真正实现了程序的并发执行。

提高CMP的性能,片上共享缓存机制发挥着非常重要的作用[2-5]。为了充分利用片上共享缓存Cache,CMP的Cache系统必须为各个并行执行在不同处理器核上的线程提供快速的数据访问。目前,片上集成的Cache容量在不断增大,这意味着其功耗所占系统总的功耗的比重也越来越大。如何降低Cache功耗已经成为未来多核处理器设计中的核心问题。其中Cache功耗分为驱动集成电路的动态功耗和漏流导致的静态功耗。随着大规模集成电路技术的发展,静态功耗占整体功耗的比重有所增加,达到80%[6]。另一方面,为了提高空间利用率,CMP的L2 Cache或更高层次的Cache一般会全部或部分地共享,因此不同处理器核的线程可能会竞争地使用Cache资源。当该情况发生时,会造成各个处理器核上所运行线程的性能出现抖动,造成总体性能的下降。解决该问题的一种方法为增大Cache的容量,然而该方法将导致功耗的剧增并且将导致Cache的利用率降低。

由于共享Cache功耗问题限制了CMP的性能的进一步提升,如何降低共享Cache的功耗成为了当前学者的研究热点[7-10],例如路预测Cache技术[11]、基于IPC的Cache划分技术[12]、基于公平性的Cache划分技术[13]。根据各个处理器核运行的线程具有空间局部性的原理,文献[14]提出了一个创新的方案,即关闭掉访问频率低的Cache路并不会导致访问时效率的明显降低。

本文实验分析单处理器访问组相联二级Cache的动态行为引用的自适应特性,找出该特性的评价指标并加以验证,研究其在多核低功耗路动态划分中的应用。首先,定义程序工

到稿日期:2013-09-03 返修日期:2013-11-04 本文受国家自然科学基金(61202076),北京市教委科技计划面上项目(KM201210005022)资助。
方 娟(1973-),女,博士,副教授,主要研究领域为计算机系统结构、多核计算,E-mail:fangjuan@bjut.edu.cn;王 帅(1986-),男,硕士生,主要研究领域为计算机系统结构;于 璐(1990-),女,硕士生,主要研究领域为计算机系统结构。

作集大小的度量标准 LOC，它能反映出当前运行线程的所需空间大小。该机制主要由路分配模块和动态功耗控制模块组成。路分配模块在程序运行过程中根据处理器核所运行线程的工作集的大小调整处理器核所分配的 Cache 路。动态功耗控制模块将处理器核所运行线程的工作空间控制在少数 Cache 路中。关闭剩余的 Cache 路，从而达到降低 Cache 功耗的目的。本文使用基于 Simics 的全系统模拟平台模拟多核处理器，并使用 SpecOMP 测试集测试系统的性能和功耗，与传统的 Cache(Conventional L2 Cache，C-L2)相比，其 IPC 提高了9.27%，功耗降低了 10.95%。

2 线程工作集的度量

2.1 划分粒度

目前可以把共享高速缓存分配给各个处理器核共享使用的分配粒度分为路划分、页划分及组划分。前者的划分粒度为 Cache 路，后者的划分粒度为 Cache 组。页划分的划分粒度为 Cache 页。由于 SDH[15]可获得采用 LRU 路替换算法时各个处理器核在分配不同 Cache 时的失效率，因此本文选择的划分粒度为 Cache 路。在路分配模块中设置一个向量将特定线程的数据限制在指定路中，位向量的一位代表一路，不同线程的数据可映射到不同的 Cache 路中。

2.2 度量标准

本文首先讨论路分配模块和动态功耗控制模块工作所用到的线程工作集大小的度量依据。线程在执行时，存在程序段的行为，程序段内的失效率相对稳定，各段之间失效率变化较大，线下监测系统无法反映失效率的动态变化。因此我们采用在线监测系统 SDH(Stack Distacnce Histogram)。SDH 可获得采用 LRU 路替换算法时各个处理器核运行线程独占 Cache 时的命中率情况。根据 SDH 可以推算出各个线程在所有可能 Cache 路分配选项下线程的失效率。SDH 为 N-路组相联 Cache 各路增加一个失效率计数器 $C_1, C_2, \cdots, C_N, C_i$ 记录线程运行时第 i 路 Cache 的访问计数，C_N 记录总的失效次数。因此，C_1 和 C_N 分别记录访问频率最高和最低的 Cache 路。

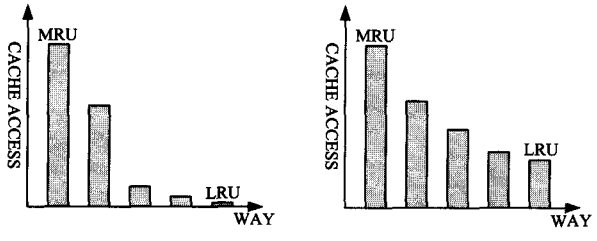

图 1 线程工作集空间模型

图 1 中的线程工作空间模型分别代表两种类型的 Cache 访问类型，线程在执行时工作集小的线程访问 Cache 路主要集中在 MRU 位置，线程工作集大的线程对 Cache 路的访问分布比较平均。本文定义线程工作集度量如下[16]。

$$LOC=\frac{LRU\ COUNT}{MRU\ COUNT} \tag{1}$$

LRU COUNT 和 *MRU COUNT* 分别代表处理器核所运行线程在一个时间段内对 LRU 和 MRU 位置 Cache 路的引用计数，因此 LOC 能够实时性地反映出线程的工作集的大小。

3 共享 L2Cache 划分机制

3.1 划分框架

在本文中，我们假设 4 个处理器核共享 L2Cache。由于大容量的 Cache 并不是随时都能充分利用的，因此增加 Cache 容量并不一定会带来性能的显著提升，所以动态功耗控制的目的就是希望在满足程序运行所需 Cache 路的基础上尽量关闭程序用不到的 Cache 路，从而达到降低系统功耗的目的。本文是通过在线监测系统收集线程工作集信息，当系统的工作集变大时开启关闭的 Cache 路来满足线程对 Cache 空间的需求。

在本文中有下面基本参数：运行时间片 t，是指线程对 Cache 的访问次数(例如 100000～200000 次)，图 2 中的流程会每隔一个时间片执行一次，直至程序结束。Loc_i 代表各个处理器核所运行线程对 Cache 的需求量。$Inact_i$ 表示分配给处理器核 i 的 Cache 路中关闭的数量。

图 2 显示了整个机制的工作框架。首先进行 L2 Cache 路分配，本文各个处理器核线程开始运行时将得到相同的 Cache 容量。处理器核可以私有地访问所分配的 L2 Cache 路。划分框架每隔一个时间片执行一次访问采样操作，该操作收集各个处理器核增加的在线监测系统 SDH 的统计数据。根据该统计数据计算出各个处理器核的 *LOC*。Loc_i 用于路重新分配模块的输入参数。

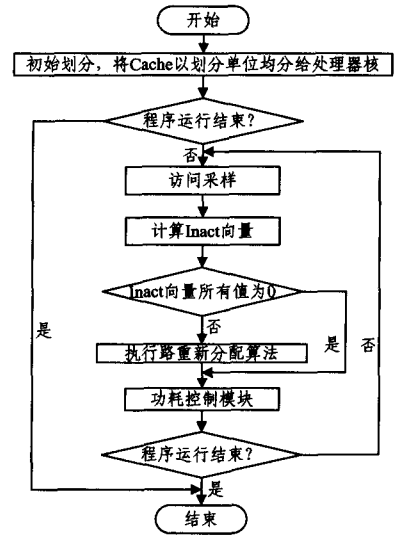

图 2 划分机制框架

由图 2 可知，$Inact_i>0(1,2,3,4)$表示各个处理器核中线程在运行过程中为了达到降低功耗的目的都有 Cache 路关闭，当且仅当这种情况下路重新分配模块被跳过不执行。在这样的情况下，重新配置不利于性能的提升，而且会导致系列开销增加。路重新分配模块和功耗控制模块将在下面讨论。

3.2 路分配模块

假设 n 个线程同时运行在 n 核处理器上，共享 C 路组相联 L2 Cache，L2 Cache 路动态分配算法如下。

路动态分配算法：

输入：

1. LOC 统计向量 $LOC=(Loc_1, Loc_2, \cdots, Loc_n)$。n 为处理器核数。
2. Cache 上次划分完成时的划分向量 $C=(\cdots, core_i, \cdots, core_j, \cdots)$。向量 C 元素个数为 Cache 的路数，$core_i$ 表示该路分配给核 i。
3. Cache 上次划分完成时的状态向量 T(…，O，…，N，…)。向量 T

元素个数为 Cache 的路数，O 表示该路开启，N 表示该路关闭。

输出：本次 Cache 划分结果向量 C。

初始化阶段：

划分阶段：

1. 一个时间片 t 后，判断是否到划分周期，是则跳转到步骤 2，否则跳转步骤 10。
2. 初始化 C、T 向量：将上次划分完成时的划分向量 C 赋值给本次 Cache 划分向量 C，T 等于上次划分完成时的状态向量 T。
3. 设置临时向量 I=(…，$Inac_i$=0，…)(1≤i≤n)，$Inac_i$ 表示分配给核 i 的 Cache 中关闭的路数，扫描 C、T 向量计算向量 I 的各个元素：

   For i=1，2，…，n do

   If(T[i]=N) I[C[i]]++；
4. 根据线监测系统 SDH 统计的 MRU、LRU 的访问次数计算 LOC 向量的各个元素值，并按非递减顺序赋值给 LOC 向量。
5. 令 Prior，Rear 指针分别指向 LOC 向量的首尾
6. 若 Prior<Rear，跳转步骤 7，否则跳转步骤 10。
7. 查找临时向量 I，若第 Rear 个处理器核所拥有的 Cache 路中存在关闭的 Cache 路(I(Rear)>0)，则跳转到步骤 8，否则跳转到步骤 9。
8. 从第 Rear 个处理器核关闭的 Cache 路中选择一路 w 分配给第 Prior 个处理器核。将划分向量 C 中 w 路更新为 Prior，将状态向量 T 中 w 开启状态更新为 O(开启)状态，Prior 后移一个位置。跳转到步骤 7。
9. 选择第 Rear 个处理器核 LRU 位置的 Cache 路分配给第 Prior 个处理器核，数据写回内存，将划分向量 C 中 w 路更新为 Prior。Prior 后移一个位置，Rear 前移一个位置。跳转到步骤 7。
10. 本次划分结束，输出本次的划分向量 C。

路自适应划分算法的流程如图 3 所示。

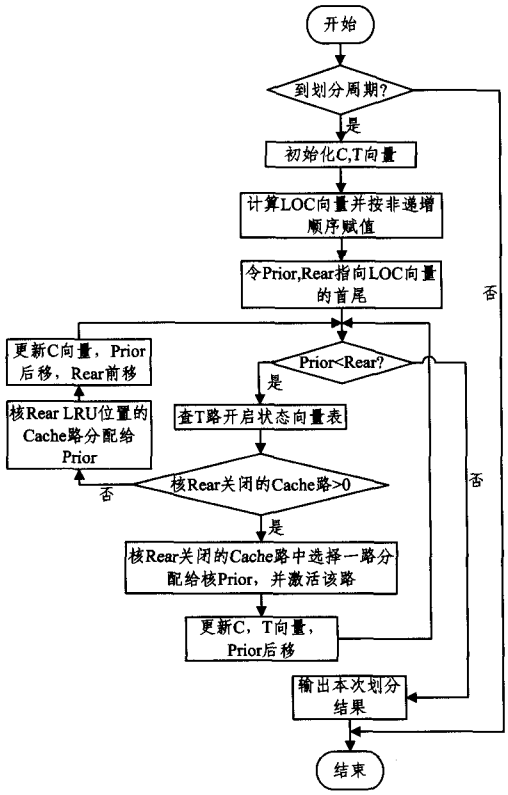

图 3　重划分流程

3.3　功耗控制模块

本文中提到的功耗控制的理论主要基于路自适应 Cache[16]。线程存在程序段的行为，各个时刻线程对 Cache 的访问需求量不同，Cache 并不是随时都能充分利用的，因此动态功耗控制的目的就是在满足程序运行所需 Cache 路的基础上尽量关闭程序用不到的 Cache 路，从而达到降低功耗的目的。

在路分配模块中已经计算出 Loc_i(1≤i≤n)，每个时间片 Loc_i 会被重新计算，本文使 Loc_i 在每个内时间片内保持在两个阈值(T_1<T_2)之内来达到降低功耗同时保持性能的目的[17]。Loc_i 小于 T_1 时表明该线程的工作集主要集中在 MRU 位置，此时关闭 LRU 位置的 Cache。反之，Loc_i 大于 T_2 表明该线程的工作集较大，此时开启分配给该线程的已关闭的 Cache 路。

以上的论述通过量化对 Cache 资源的需求来控制功耗。在线程执行过程中获得 Loc_i 值，将 Loc_i 与两个阈值 T_1 和 T_2 进行比较：若 Loc_i 值大于 T_2，则该程序可以被认为具有较低的局部性，因此需要更多的 Cache 资源。在该种情况下，功耗控制模块的输出信号为 Inc(向上调整大小请求)，以增加开启的 Cache 资源。另一方面，功耗控制模块输出 Dec(向下调整大小请求)信号来关闭更多的 Cache 资源。若 Loc_i 值介于 T_1 与 T_2 之间，功耗控制模块输出 keep 信号来保持当前的状态。

通过 D 与两个阈值的比较给出调整高速缓存大小的请求：Inc、Dec、或 Keep。开启或关闭 Cache 路的判断依据为程序的局部性行为。然而，在访存行为存在高度不规则和不稳定的情况下，短时间内 Cache 路的开启和关闭性行为也极不规则，这可能导致系统的性能急剧下降，同时功耗并没有降低。因此，功耗控制应能在程序访存的行为不规则的情况下也能给出合理的表现。为此，本文在功耗控制模块中加入了状态机机制。状态机会接收并保存 Dec 的信号，该信号只有在一直连续请求一段时间后，最终达到满载状态，且达到此状态后继续收到 Dec 信号，状态机才会发出减小 Cache 资源大小的请求。无论状态机处于何种状态，只要收到 Inc 信号就恢复清零状态的最初始状态，若恢复清零状态后继续收到 Inc 信号，则状态机发出增大 Cache 资源的请求。本文设置的 3bit 状态机如图 4 所示。

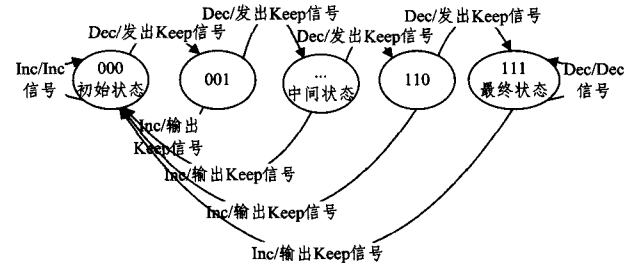

图 4　状态机机制

4　实验结果

4.1　实验环境

本文使用 Virtutech Simics[18]全系统模拟平台模拟 CMP 多核处理器系统，在该系统上安装 solaris 操作系统，最后在 solaris 系统上运行 specOMP 基准测试程序测试系统的性能和功耗。4 核 CMP 配置见表 1，8/16 核的配置见表 2。

表 1　实验配置

CMP	Simics
Processor	4cores, Four issues, out-of-order
L1 Cache	Private Instruction Cache and data Cache for each core I-Cache: 32kB, 64byte line size, 4-way LRU, 3 cycle hit latency D-Cache: 32kB, 64byte line size, 4-way LRU, 3 cycle hit latency
L2 Cache	Shared: 2MB, 64byte line size, 16 way LRU, 6 cycle hit latency
Memory	2GB, 158 cycle access latency

表 2　8/16core CMP 配置

CMP	Simics
Processor	8/16cores, Four issues, out-of-order
L1 Cache	Private Instruction Cache and data Cache for each core I-Cache: 32kB, 64byte line size, 4-way LRU, 3 cycle hit latency D-Cache: 32kB, 64byte line size, 4-way LRU, 3 cycle hit latency
L2 Cache	Shared: 4MB, 64byte line size, 32 way LRU, 6 cycle hit latency
Memory	4GB, 158 cycle access latency

4.2　实验结果

本文使用 specOMP 基准测试程序测试改进前后的处理器模型的执行效率。ave 是各处理器核静态均分 L2 Cache 的情况，realloc 是本文提出的机制。在应用中，平均每周期执行的指令数(Instruction Per Cycle，简称 IPC)是衡量系统性能的重要指标，4 核 CMP 的 IPC 对比如图 5 所示，10 个测试程序的 IPC 平均提升了 9.41%。

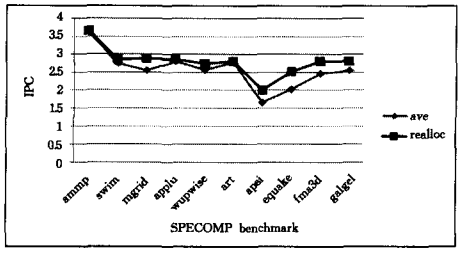

图 5　4 核 CMP 的 IPC 对比图

实验得到的 4 核 CMP 处理器访问 L2cahce 时每条指令的平均功耗如图 6 所示。ammp(计算化学)程序每条指令的平均功耗从 168.75 降低到了 142.59，降低了 18.3%。实验中测试程序的功耗平均降低了 11.3%。

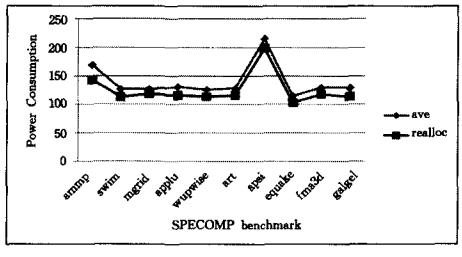

图 6　4 核 CMP 的功耗对比图

实验所得的 8 核 CMP 处理器的 IPC 如图 7 所示(L2Cache 配置 32 路)，从图中可以看出，applu 和 equake 的 IPC 提升最显著，分别为 31.8%和 22.9%，平均提升了 9.13%。

8 核 CMP(L2Cache 配置 32 路)处理器的功耗情况如图 8 所示，实验中测试程序的 IPC 平均提升了 9.13%，平均功耗降低了 10.54%。

图 9 为 16 核 CMP 的两种 Cache 分配策略下的程序总执行时间对比图。本文提出的 L2Cache 分配策略能够在程序运行过程中根据程序的工作空间动态地将 L2Cache 资源分配给各个处理器核，较静态分配策略有明显的优势，specOMP 中所选测试程序的总运行时间平均降低了 9.89%。

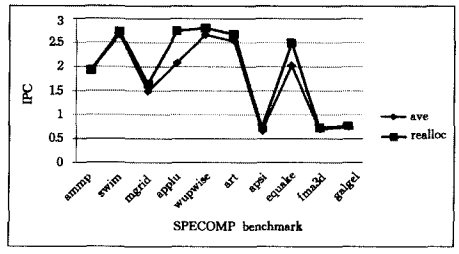

图 7　8 核 CMP 的 IPC 对比图

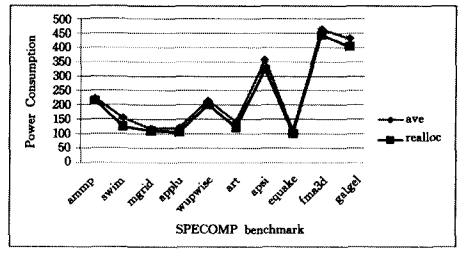

图 8　8 核 CMP 的功耗对比图

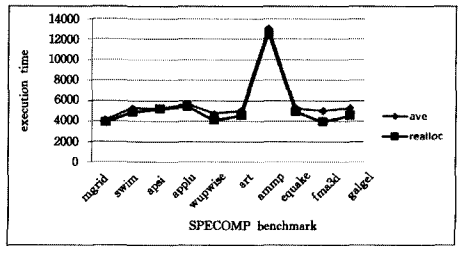

图 9　16 核 CMP 运行时间对比图

16 核 CMP 处理器的功耗情况如图 10 所示，实验中选取 specOMP 测试程序测试系统的功耗，与静态均分 L2Cache 相比，本文提出的策略平均功耗降低了 11.03%。

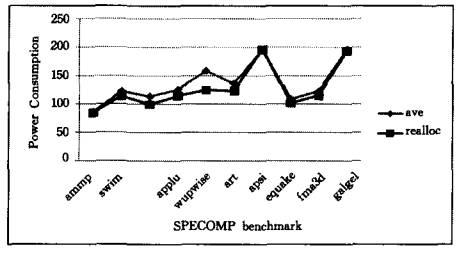

图 10　16 核 CMP 功耗对比图

结束语　本文通过实验分析单处理器访问组相联二级 Cache 的动态行为的自适应特性，提出了该特性的评价指标并加以验证。依据该特性提出了多核低功耗路动态划分算法，并在多核模拟平台 Gems 下模拟了本文提出的机制。实验结果表明 4 核 CMP 的 IPC 平均提高了 9.41%，系统平均功耗降低了 11.3%，8 核 CMP 的 IPC 平均提高了 9.13%，系统平均功耗降低了 10.54%，16 核 CMP 的相对运行时间减少了 9.89%，功耗降低了 11.03%。实验结果表明，本文提出的多核低功耗自适应划分算法能够有效地提高 Cache 资源的利用率，降低系统的功耗。而在核数增多的同时要维持系统的性能，则须为处理器核配置更多的 Cache 资源，否则，Cache 资源的不足将会导致线程间冲突的增多，从而导致系统性能的下降。

(下转第 73 页)

对其做出各种反应，事物以属性及其变化过程反映其自身状态、运动和变化过程，人类的认知过程与这些接收到的属性信息紧密相关。在哲学观点上，属性不仅表现着它要表现的质特征，而且还拥有需要它界定和规范的量特征。本文基于这样的观点，结合粒计算理论和定性基准变换的理论给出了认知的基本数学模型及其 Petri 网形式定义。然而，人类的认知过程是一个非常复杂的问题，这里只是给出一个基本的认知模型框架，其他的工作需要进一步深入研究，例如更深层的认知与识别、认知与决策等。

参 考 文 献

[1] 冯康. 基于一元事件的认知模型 [J]. 模式识别与人工智能，2012,25(1):173-180

[2] 张文修，徐伟华. 基于粒计算的认知模型[J]. 工程数学学报，2007,24(6):957-971

[3] 钟义信. 机制主义：人工智能的统一理论[J]. 电子学报，2006,34(2):317-321

[4] Badgaiyan R D. Neuroanatomical Organization of Perceptual Memory: An fMRI Study of Picture Priming[J]. Human Brain Mapping,2000,10:197-203

[5] 北大生物智能技术研究组. 第五代计算机及其认知逻辑[J]. 前沿科学，2007,1(1):18-23

[6] 冯嘉礼，董占球. 基于属性整合的知觉模式生成与识别模型[J]. 计算机研究与发展，1997,34(7):487-491

[7] Zadeh L A. Towards a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic[J]. Fuzzy Sets and Systems,1997(19):111-127

[8] Feng Jia-li. Attribute network computing based on qualitative mapping and its applications in pattern recognition[J]. Journal of Intelligent & Fuzzy Systems,2008,19(2):1-16

[9] 冯嘉礼. 判断基准的可变性与面向判断的性质逻辑[J]. 广西师范大学学报：自然科学版，1994,12(1):28-35

[10] 冯嘉礼. 基于定性映射的模式识别方法(II)[J]. 广西师范大学学报：自然科学版，2004,22(2):14-18

[11] 张铃，张钹. 问题求解理论及应用[M]. 北京：清华大学出版社，2007

[12] 李鸿. 粒集理论：粒计算的新模型[J]. 重庆邮电大学学报：自然科学版，2007,19(4):397-404

[13] 黄林鹏，孙永强. 性质继承的线性逻辑表示和推理[J]. 计算机工程，1993,19(3):1-6

[14] 陈文伟，黄金才，毕季明. 适应变化环境的元知识的研究[J]. 智能系统学报，2009,4(4):331-334

[15] 孟宪刚，严洪森. 基于多属性模糊 Petri 网的知识化制造系统产品需求预测[J]. 系统工程理论与实践，2012,32(4):790-798

[16] Wang Ying-xu, Wang Ying. Cognitive Informatics Models of the Brain[J]. IEEE Transactions on Systems, Man and Cybernetics, 2006,36(2):203-207

[17] 王延江，齐玉娟. 基于记忆机制的视觉信息处理认知建模[J]. 模式识别与人工智能，2013,24(2):144-150

[18] 董占球，冯嘉礼. 按模式记忆理论的数学描述(I)—记忆模式的属性坐标表示法[J]. 计算机研究与发展，1998,35(8):694-698

(上接第 39 页)

参 考 文 献

[1] Herrero E, Gonz'alez J, Canal R. Distributed cooperative caching [C]// Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques. 2008:134-143

[2] Kotera I. Power-Aware Dynamic Cache Partitioning for CMPs H [C]// Transactions on High-Performance Embedded Architectures and Compilers III. 2011:135-153

[3] Sinharoy B, KallaR N, Tendler J M. Power 5 system micro-architecture[M]. IBM J. Res Dev,2005,49:505-521

[4] Kim S, Chandra D, Solihin Y. Fair Cache sharing and partitioning in a chip multiprocessor architecture[C]// Proc. of PACT 2004. Antibes, Juanles-Pins, France,2004:111-122

[5] Sui Xiu-feng, Wu Jun-min, Chen Guo-liang, et al. Augmenting Cache Partitioning with Thread-Aware InsertionIPromotion Policies to Manage Shared Caches[C]// Proceedings of the 7th ACM International Conference on Computing Frontiers. 2004:79-80

[6] Meng Y, Sherwood T, Kastner. Exploring the limits of leakage power reduction in Caches[J]. ACM Trans. Archit. Code Optim. ,2005,2:221-246

[7] Janapsatya A, Parameswaran S, Ignjatovic A. HitME: Low power Hit MEmory buffer for embedded systems[C]// Asia and South Pacific Design Automation Conference. 2009:335-340

[8] Tsai Y-Y, Chen C-H. Energy-Efficient Trace Reuse Cache for Embedded Processors[J]. IEEE Transaction On Very Large Scale Integration(VLSI)System,2010,19:1681-1694

[9] Datta K A M A S. Energy efficient i-Cache using multiple line buffers with prediction[J]. IET Comput. Digit. Tech. , 2008, 2(5):355-362

[10] Ali K, Aboelaze M. Energy efficient I-Cache using multiple line buffers with prediction[J]. Computers and Digital Techniques, IET,2008(2):355-362

[11] Inoue K. Way-Predicting Set-Associative Cache for High Performance and Low Energy Consumption[C]// Proceedings. 1999 International Symposium on Low Power Electronics and Design 1999(Cat. No. 99TH8477). 1999:273-275

[12] Suo Guang, Yang Xue-jun, Liu Guang-hui, et al. IPC-Based Cache Partitioning: An IPC-Orient Dynamic Shared Cache [C]// International Conference on Convergence and Hybrid Information Technology. Washington DC: IEEE Computer Society, 2008:399-406

[13] Kim S, Chandra D, Solin Y. Fair Cache sharing and partitioning in a chip multiprocessor architecture[C]// Proc. of PDCT 2004. Antibes, Juanles-PINS, France IEEE,2004:111-122

[14] Zhang C, Yang J, Vahid F. Low static-power frequent-value data Caches[C]// The Design, Automation and Test in Europe Conference and Exhibition. Paris, France,2004,1:214-219

[15] Qureshi M K, Patt Y N. Utility based Cache partitioning: a low overhead, high performance, runtime mechanism to partition shared Caches[C]// Proc. of the 39th Annual IEEE/ACM Int Symp on Microarchitecture. Orlando, Florida, USA: IEEE, 2006: 423-432

[16] Kobayashi H, Kotera I, Takizawa H. Locality analysis to control dynamicallyway-adaptable Caches[J]. SIGARCH Comput. Archit. News,2005,33(3):25-32

[17] Kobayashi H, Kotera I, Takizawa H. Locality analysis to control dynamically way-adaptable Caches[J]. SIGARCH Comput. Archit. News,2005,33:25-32

[18] Christensson M, Eskilson. Simics: a full system simulation platform[J]. IEEE Computer,2002(3):50-58