WPP-L2：多核处理器中共享Cache低功耗路预测算法

方 娟 郭 媚 杜文娟
（北京工业大学计算机学院 北京 100124）

摘 要 针对片上多核处理器下的二级共享Cache的能耗问题提出了基于Cache划分的路预测Cache结构WPP-L2，该结构首先对共享Cache进行公平性划分，然后采用路预测的方法降低了预测命中和失效时各自的能耗开销。实验表明，在基本保持多核处理器性能的同时，8核处理器系统下WPP-L2 Cache比基于路预测的L2 Cache的能耗延迟乘积EDP(Energy Delay Product)平均下降24.7%，比传统的L2 Cache的EDP平均下降66.1%，极大地降低了L2 Cache功耗。

关键词 多核处理器，低功耗，路预测
中图法分类号 TP302 **文献标识码** A

WPP-L2：A Way-predicting Algorithm for Shared L2 Cache of Multi-core Processors

FANG Juan GUO Mei DU Wen-juan
(College of Computer Science, Beijing University of Technology, Beijing 100124, China)

Abstract This paper proposed a way-predicting L2 cache mechanism based on cache partitioning for the problem of multi-core processors power, which is WPP-L2. WPP-L2 partitions the shared cache for fairness, and then uses cache way-predicting techniques to reduce the energy consumption when the prediction hits and misses. The results turn out, WPP-L2 cache could reduce EDP(Energy Delay Product) by 24.7% compared with the way-predicting L2cache, and reduce EDP by 66.1% compared with the traditional L2cache under the eight-core processors, which lowers the energy consumption of L2 Cache and maintains the performance at the same time.

Keywords Multi-core processors, Low-power, Way-predicting

1 引言

电子工艺的发展提高了处理器的集成度和速度，而随着片上多核处理器的集成核数不断增加，其功耗也在急剧增加，低功耗已经成为多核领域研究的关键问题。降低功耗主要包括降低动态功耗和降低静态功耗，其中动态功耗包括处理器内部各元件正常工作时所消耗的电能，降低动态功耗的技术主要有多元功能电压技术、动态电压调节、时钟屏蔽等，主要集中在电路级。高性能Cache中电容的充电和放电以及传感器的检测，使得Cache的动态功耗成为整个处理器芯片动态功耗的重要组成部分，其功耗约占处理器总功耗的30%～60%[1]。因此，可以通过降低Cache的动态功耗来降低多核处理器的整体功耗。目前，降低多核处理器的动态功耗主要通过动态电压调节算法的优化[2]、可重构Cache技术[3]和Cache划分技术来实现，但是动态电压调节算法与Cache的可重构技术多运用在嵌入式多核处理器下。

主流的多核处理器大多使用私有一级Cache和共享大容量的二级Cache，处理器核间对共享二级Cache的竞争使用容易导致线程饿死和优先级反转等问题。因此，要充分利用共享Cache多核处理器的性能优势，就必须有效地管理和使用共享Cache。共享Cache的划分技术以提高系统吞吐率、公平性或QoS(Quality of Service)等指标为目的，提出了不同的划分策略，将Cache划分给不同的处理器核来实现共享Cache的访问优化，在降低功耗上也有一定成效[4]。Cache路预测技术在单核处理器时期就被提出，在保证路预测命中率的情况下，既可以保持系统性能又可极大地减少Cache的访问功耗[5]。在Cache路预测和多核处理器下共享二级Cache的划分策略方面，也提出了一些解决的方法[6-10]。路预测Cache，以四路组相联Cache为例，在Cache访问前给出一路预测路，以访问一路Cache代替传统访问中并行地访问Cache的所有路，如果路预测命中，则可以比传统的Cache访问减少75%的能耗。共享二级Cache的划分技术以Cache路为单位，通过不同的划分策略将Cache路划分给各个处理器核。Kim[11]提出一种基于公平性的Cache划分策略。通过实验证明，与Cache的LRU算法相比，基于公平性的Cache划分策略能有效地提高系统的性能。

本文针对多核处理器下的共享二级Cache提出基于Cache划分的路预测算法。Cache划分策略将共享二级

到稿日期：2012-10-12 返修日期：2013-01-13 本文受国家自然科学基金项目(61202076)，北京市教委科技计划面上项目(KM201210005022)，北京市中青年骨干人才培养项目(00700543112512)资助。

方 娟(1973－)，女，博士，副教授，主要研究方向为计算机系统结构、多核计算，E-mail：fangjuan@bjut.edu.cn；郭 媚(1988－)，女，硕士，主要研究方向为多核计算；杜文娟(1986－)，女，硕士，主要研究方向为多核计算。

Cache 以路为单位划分给各个处理器核，使得处理器核可以拥有类似私有的二级 Cache 空间；进行二级 Cache 访问前，由路预测算法预先提供一路预测路，处理器直接访问该路数据而无需同时访问 Cache 的所有路数据，因此，当预测命中时处理器能耗降低，如果预测失效，也只需要访问划分给该处理器核的 Cache 路中的剩余路而不是所有剩余 Cache 路，这样就降低了由预测失效时导致的能耗开销。实验证明，采用该算法的 L2 Cache 每次的访问功耗比未采用该算法的 L2 Cache 增加了 0.2%，但该算法带来的 L2 Cache 动态功耗的降低却达到了 60%以上。

本文首先介绍相关的路预测技术和共享 Cache 的划分策略；然后介绍相关算法，即基于 Cache 划分的路预测算法；最后对算法进行验证并分析实验结果。

2 相关技术

2.1 传统的路预测 Cache

在当前处理器的缓存结构中，普遍采用组相联的 Cache 结构，根据组相联 Cache 的地址结构，Cache 访问时需要访问 Tag 阵列和 Data 阵列。因此，组相联 Cache 的动态能耗(ECache)可以用下述公式给出[5]：

$$E_{Cache} \approx E_{Memory} = N_{Tag} \times E_{Data} + N_{Data} \times E_{Data} \quad (1)$$

式中，N_{Tag} 和 N_{Data} 分别表示 Cache 访问到的 Tag 和 Data 的路数，E_{Tag} 和 E_{Data} 则分别表示访问一路 Cache 中的 Tag 和 Data 时所需的能耗。

以一个采用 8 路组相联结构的二级 Cache 为例，并假设 Cache 访问时间为一个时钟周期。在没有采用路预测技术之前，发出一个 L2 Cache 访问时，需要并行读取 Cache 中的 8 路 Data 阵列和 Tag 阵列，依次将 8 路 Tag 与所需要的数据地址的 Tag 进行比较，从中选取与所需访问数据地址的 Tag 相等的一路，使用从该路中读出的数据。式(2)给出了未采用路预测技术的 8 路组相联 Cache 的动态能耗和平均访问时间：

$$\begin{aligned} E_{8SACache} &= 8 \times E_{Tag} + 8 \times E_{Data} \\ T_{8SACache} &= 1 \end{aligned} \quad (2)$$

从整个 Cache 访问过程可知，一共读取的 8 路 Data 和 Tag 实际只需要使用 1 路，而对未被使用的 7 路 Data 和 Tag 的读取所耗能量成了 Cache 访问的额外开销。采用路预测技术的 8 路组相联 Cache，在对 Cache 进行访问之前，路预测算法预测性地给出一路 Cache。如果预测命中，Cache 只需访问一路数据就可完成此次访问，否则，Cache 并行访问剩余的 7 路 Cache，如果在剩余路中找到所需数据则完成访问，没找到则产生 Cache 替换。在路预测命中的情况下，整个 Cache 访问过程只需消耗读取一路 Cache 的能量，同时访问时间只需一个时钟周期；如果预测失效，Cache 访问需要增加一个时钟周期来并行地访问剩余的 Cache 路，此时路预测 Cache 并不能节省能耗。式(3)[5]给出了采用路预测技术的 8 路组相联 Cache 的能耗和平均访问时间：

$$\begin{aligned} E_{WP8SACache} &= (E_{Tag} + E_{Data}) + (1 - PHR) \times (7 \times T_{Tag} + 7 \times E_{Data}) \\ T_{WP8SACache} &= 1 + (1 - PHR) \times 1 \end{aligned} \quad (3)$$

式中，PHR(Prediction Hit Ratio)表示路预测的命中率，它在很大程度上决定了路预测 Cache 的性能和能量消耗。

从式(3)可得出，当路预测命中时，同样的访问时间下 Cache 的动态能耗可以节省 87.5%。

2.2 Cache 划分策略

共享二级 Cache 以其高效的使用和灵活的动态分配成为目前片上多核处理器上普遍采用的 Cache 结构。传统的 LRU 算法在提高共享 Cache 性能上效果甚微，而且还可能导致线程饿死。因此，提出了提高公平性和服务质量(QoS)等方案来提高共享二级 Cache 的性能[10]。公平性是提高系统性能的关键所在，因为系统的线程调度效率是通过硬件给所有同步线程提供的公平性来体现的[12]。所谓公平性就是所有的线程有相同的机会获得其运行所要求的资源，并保证没有线程饿死[13]。实际上，目前二级 Cache 上普遍采用的 LRU 替换策略会根据每个线程的需要来划分 Cache 空间，这种对 Cache 的划分被称为隐式划分，但隐式划分并没有把公平性考虑在内，因此会导致线程饿死和优先级反转等问题[14]。

DFC(Dynamic Fair Caching)算法是由 Kim[11]等人提出的基于公平性的 Cache 划分算法，该算法的核心思想主要包含 3 个步骤：初始化、回溯和重划分。在初始化阶段，将 L2 Cache 以路为单位均匀地划分给各个线程；在回溯阶段，DFC 选择 5 个值作为公平性的衡量指标，它们分别表示为：

$$M_1^{ij} = |X_i - X_j|, \text{where } X_i = \frac{Miss\_shr_i}{Miss\_ded_i} \quad (4)$$

$$M_2^{ij} = |X_i - X_j|, \text{where } X_i = Miss\_shr_i \quad (5)$$

$$M_3^{ij} = |X_i - X_j|, \text{where } X_i = \frac{Missr\_shr_i}{Missr\_ded_i} \quad (6)$$

$$M_4^{ij} = |X_i - X_j|, \text{where } X_i = Missr\_shr_i \quad (7)$$

$$M_5^{ij} = |X_i - X_j|, \text{where } X_i = Missr\_shr_i - Missr\_ded_i \quad (8)$$

其中，*Missr* 是 Cache 的缺失率，*Missr_shr* 是共享 Cache 缺失率，*Missr_ded* 是独占 Cache 缺失率；*Miss* 是 Cache 的缺失数，*Miss_shr* 是共享 Cache 缺失数，*Miss_ded* 是独占 Cache 缺失数。Kim 提出 Cache 缺失数和 Cache 缺失率与公平性紧密相关并且易于测量，因此选择它们作为衡量公平性的指标。当一个时间片结束时，计算 *Mxt*(*Mx* 是 *M*0～*M*5 其中之一)，如果 *Mxt* 大于上一时间片中计算所得的 *Mxt* −1，并且它们的差值大于某一个阈值，则撤销此次 Cache 重划分方案并恢复上一时间片的划分方案，否则转入重划分阶段；在重划分阶段中，没有回溯的线程将会获得更多或者更少的 Cache 路空间，以保证在下一个时间周期内线程间有更好的公平性。评测结果表明，相对于传统的 LRU 算法，基于公平性的 Cache 划分通常会提高系统的性能[11]。本文提出的基于 Cache 划分的路预测算法采用 DFC 划分策略对多核下的共享二级 Cache 进行划分，在此基础上结合已经在单核处理器上被采用的路预测算法，从而达到降低多核处理器共享二级 Cache 的动态功耗的目的。

3 基于 Cache 划分的路预测 L2 Cache

3.1 WPP- L2 Cache 结构

本文提出一种新的共享二级 Cache 结构 WPP-L2 Cache，即基于 Cache 划分的路预测 Cache，用于降低共享二级 Cache 的动态能耗。如图 1 所示，基于 Cache 划分的路预测 Cache

在传统的 Cache 结构上增加了 3 个逻辑模块，分别是路预测器、路预测表和 Cache 划分结果表。其中，路预测表(Way-predicting Table，WPT)存放被访问过的 Cache 块的路编号，根据程序运行时的局部性原理，刚刚访问过的数据很可能在下一次访问时再次用到，通过存放在路预测表中的内容来预测下一次访问时数据所在的路编号。每次 L2 Cache 访问结束后，将该次访问 Cache 块所在的路编号存入路预测表中的 way 表项下，Cache 块的地址 Tag 部分存放在路预测表的 tag 表项下，路预测表使用 MRU(Most Recently Used)替换策略；路划分表(Partition Table，PT)存放 Cache 划分的结果，如图 1 所示。每个处理器核在 Cache 划分后都独自占有一定数量的 Cache 路，编号为 0 的处理器核占有 L2 Cache 的第 0 路和第 1 路，编号为 1 的处理器核占有第 2 路和第 3 路，编号为 2 和 3 的处理器核则分别占有 L2 Cache 的第 4、5 路和第 6、7 路；路预测器比较 L2 Cache 访问提供的地址 Tag 与路预测表中存放的 Tag，并将比较结果传给多路选择器(MUX)。

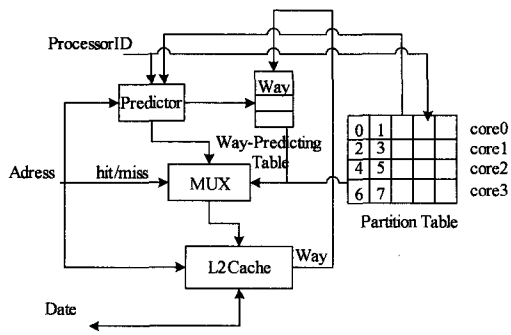


图 1　基于 Cache 划分的路预测 Cache 结构

3.2　WPP-L2 Cache 工作原理

在 L2 Cache 访问前，发出 L2 Cache 访问的处理器核的编号(processor ID)和访问地址分别与路预测表和路划分表中的表项进行匹配，其中，L2 Cache 访问地址与路预测表中的 tag 项进行比较，如果相等，则读取该 tag 项对应的 way 表项中的路编号，将其作为路预测结果发送到 MUX 中。得到预测路之后，L2 Cache 只需访问预测路的数据，如果预测命中则 L2 Cache 访问只需一个时钟周期便可完成。如果预测失效，即需要访问的数据并不在预测路中，则通过处理器核的编号检索路划分表，得到该处理器核在 L2 Cache 划分中所得的 L2 Cache 路的数量和编号，访问其中除预测失效路之外的其它 L2 Cache 路。

以图 1 中 0 编号处理器核为例，如果处理器核 0 发出的 L2 Cache 访问经过路预测后，发现预测失效，如图 2 所示，预测失效路是第 1 路 L2 Cache，而处理器核 0 在 L2 Cache 划分中划分所得的 Cache 一共有两路，分别为第 0 路和第 1 路，即处理器核 0 所需访问的数据若在 L2 Cache 内则一定存放在这两路 Cache 中。此时，处理器核 0 只需要在第二个时钟周期中访问第 0 路 L2 Cache，而传统的 Cache 路预测失效则需要访问 L2 Cache 中剩余的 7 路 Cache，相比之下，基于 Cache 划分的路预测 Cache 结构减少了在路预测失效时的能耗开销。如果在访问的剩余 Cache 路中找到了需要访问的数据，则 L2 Cache 访问完成，同时通过 MRU 算法将此次 Cache 访问的路编号及地址更新到路预测表中。

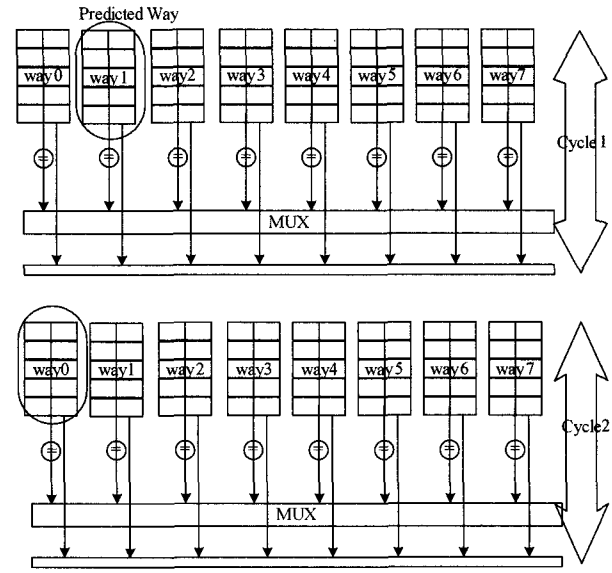


图 2　WPP-L2 路预测失效 Cache 访问图

利用本文 2.1 节中给出的能耗计算式(1)和 Cache 访问时间的假设，则图 1 所示的基于 Cache 划分的路预测 L2 Cache 的动态能耗和平均访问时间可由式(9)计算得出：

$$E_{WPPL2Cache}=(E_{Tag}+E_{Data})+(1-PHR)\times\{(way\_count-1)\times E_{Tag}+(way\_count-1)\times E_{Data}\} \quad (9)$$

$$T_{WPPL2Cache}=1+(1-PHR)\times 1$$

其中，PHR 是 Cache 预测的命中率，way_count 表示处理器核通过 L2 Cache 划分策略所得到的 L2 Cache 路的数量。从式(9)中可以看出，当路预测命中时，L2 Cache 访问只需在一个时钟周期内访问一路 Cache，而当预测失效时，由失效导致的 L2 Cache 能耗开销通过减少访问的 L2 Cache 剩余路的路数来降低。

4　实验结果及分析

4.1　实验环境

本文采用的 Simics3.0.31 全系统模拟器及其组件 General Execution-driver Multi-core Processor(GEMS)可以灵活详细地模拟片上多核处理器系统(CMP)。实验中分别模拟了片上 4 核和 8 核处理器系统，系统中每个处理器核私有 L1 I-Cache 和 D-Cache，共享 L2 Cache，Cache 采用组相联结构，Cache 块大小为 64B，替换策略为 LRU，CMP 模拟器的具体配置如表 1 所列。

表 1　CMP 模拟器 Simics 的配置

CMP	Simics
System	1 Chip，Sparc V9，Solaris9
Processor	4cores/8cores/16cores，L1Cache，L2 Cache，4 issues，out-of-order
L1 Cache	Private I-Cache and D-Cache for each core，3 cycle response latency，2 cycle request latency，I-Cache：64kB，64Bline size，4-way/8-way/8-way，LRU，D-Cache：64kB，64Bline size，4-way-/8-way/8-way，LRU
WPP-L2 Cache	Shared Cache：512kB/1MB/2MB，64B line size，8-way/16-way/32-way，LRU，6 cycle response latency，4 cycle request latency，3kB/4kB/5kB Way-predicting Table，10B/21B/136B Partition Table
memory	4GB，158 cycle access latency

本文算法的提出是基于 L2 Cache 访问的顺序性，尤其当

L2 Cache 访问的数据块大于 L2 Cache 的容量时，位于相同路的 L2 Cache 组的数据将被顺序访问，因为它们都源自于相同的数据块，L2 Cache 访问的顺序性有利于路预测的准确性。因此实验选取了 6 组 SPEC2000 基准测试程序，分别为：gzip，vortex，vpr，parser 和 bzip2，其中 gzip，vpr，mcf 和 bzip2 都是访存敏感的[15]，vortex 与 parser 是非访存敏感的。

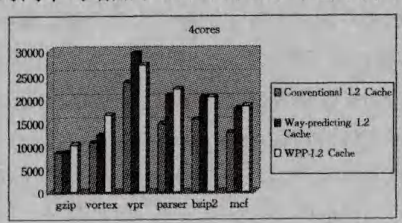
4.2 实验结果

图 3 显示了 SPEC2000 测试程序在分别采用传统的 L2 Cache 结构、基于路预测的 L2 Cache 结构和基于 DFC 划分的路预测 L2 Cache 结构时的 4 核处理器系统下的程序执行时间和 L2 Cache 的能耗延迟乘积；图 4 和图 5 则显示了 8 核与 16 核处理器系统下的程序执行时间和 L2 Cache 的能耗延迟乘积。能耗延迟乘积 EDP(Energy Delay Product=Ecache * Tcache)是目前被普遍采用的综合评价算法在性能和功耗之间平衡能力的评价指标，EDP 越小表明算法的能量效率越高。

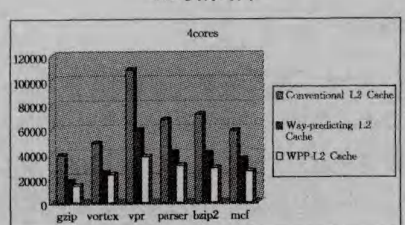
通过修改功耗评估软件 CACTI6.5 的配置参数，模拟 Cache 的工作过程，可以获得本文提出的基于 WPP-L2 的 Cache 访问功耗。在模拟的 4 核、8 核及 16 核处理器中，增加了路预测表和路划分表的 WPP-L2 Cache 的存储空间，比传统的 L2 Cache 存储空间平均增加了 0.053%，且通过模拟得出每次访问 WPP-L2 Cache 时由查找路预测表和路划分表而导致的额外功耗开销比传统的 L2 Cache 平均增加了 0.2%。然而由 WPP-L2 节省的 Cache 功耗要远大于路预测表和路划分表所增加的额外功耗。

图 3—图 5 的实验结果显示，在 4 核处理器下系统路预测准确率为 65%时，基于 Cache 划分的路预测 L2 Cache 结构系统的程序执行时间比基于路预测的 L2 Cache 结构的程序执行时间平均增加了 7.1%，EDP 平均下降了 30.6%；比传统的 L2 Cache 结构的程序执行时间平均增加了 27%，EDP 平均下降了 62.7%。

在 8 核处理器下系统下路预测准确率为 73%时，基于 Cache 划分的路预测 L2 Cache 结构系统的程序执行时间比基于路预测的 L2 Cache 结构的程序执行时间平均增加了 6.1%，EDP 平均下降了 24.7%；比传统的 L2 Cache 结构的程序执行时间平均增加了 29.7%，EDP 平均下降了 66.1%。

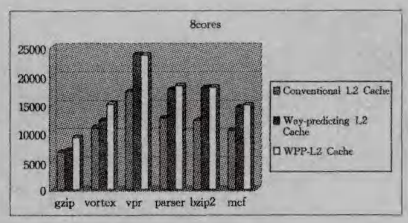


(a) 执行时间

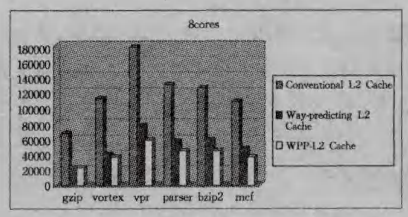


(b) EDP

图 3 4 核处理器下测试程序在 3 种 Cache 结构下的执行时间和 L2 Cache 的 EDP

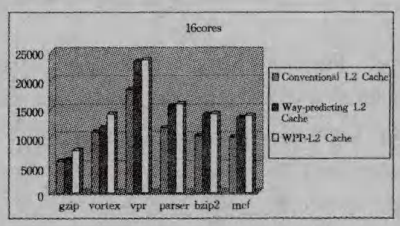


(a) 执行时间

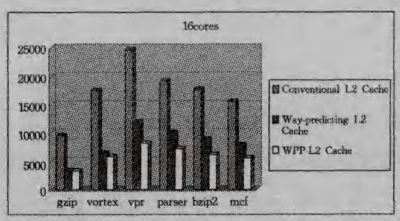


(b) EDP

图 4 8 核处理器下测试程序在 3 种 Cache 结构下的执行时间和 L2 Cache 的 EDP



(a) 执行时间



(b) EDP

图 5 16 核处理器下测试程序在 3 种 Cache 结构下的执行时间和 L2 Cache 的 EDP

在 16 处理器下系统路预测准确率为 70%时，基于 Cache 划分的路预测 L2 Cache 结构系统的程序执行时间比基于路预测的 L2 Cache 结构的程序执行时间平均增加 7.5%，EDP 平均下降 21.6%；比传统的 L2 Cache 结构的程序执行时间平均增加 25.7%，EDP 平均下降 65.1%。由实验结果可知，基于 Cache 划分的路预测 L2 Cache 算法比基于路预测的 L2 Cache 算法和传统的 L2 Cache 结构算法在能耗节省幅度上是程序执行时间增加幅度的两倍，体现了低功耗的效果；其中访存敏感的测试程序在降低功耗的效果上要比非访存敏感的程序更明显。

结束语 多核处理器的动态功耗一直在处理器的总体功耗中占据着主要的地位，降低 CMP 片上各模块的功耗有助于提高整个处理器系统的能量效率。在片上各模块的功耗开销中，缓存系统的功耗开销所占比重最大。因此，降低片上 L2 Cache 的动态功耗对降低处理器的动态功耗具有重要的意义。多核处理器下基于公平性的共享 L2 Cache 划分策略将线程间使用共享 Cache 资源的公平性考虑在内，从而提高

(下转第 42 页)

几乎所有的私钥组件和相关的密文组件转移到云端，大大降低了 DO 的计算量，达到高效性；并可以灵活地对用户进行权限授予和权限撤销；同时，由于 RMIA 的引入，也减少了云服务端的计算量，但仍没有解决计算量随属性数量线性增长的问题，同时，KP-ABE 在判断访问权限上也存在很大问题，当文件的属性范围少于用户属性范围时（即用户在满足文件属性的前提下，增加其它属性），用户却不能访问该文件。由于 DO 计算代价和 CSP 的计算代价是相关的，要想使 DO 的计算代价降低则需在保证安全的前提下将部分的计算转移到云端，这样势必会增加云端的计算量，所以在以后的研究中，在降低 DO 的计算量的同时，也使云端的计算量降低，这些都是亟需要解决的问题。

参 考 文 献

[1] Cachin C, Keidar I, Shraer A. Trusting the cloud [J]. ACM SIGACT News, 2009, 40(2): 81-86

[2] Sahai A, Waters B. Fuzzy identity based encryption [C]// Proceedings of the Advances in Cryptology (EUROCRYPT). Aarhus, Denmark, Berlin, Heidelberg: Springer-Verlag, 2005: 457-473

[3] Goyal V, Pandey O, Sahai A, et al. Attribute based encryption for fine-grained access control of encrypted data [C] // Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06). New York, NY, USA: ACM, 2006: 89-98

[4] Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption [C] // Proceedings of the 2007 IEEE Symposium on Security and Privacy. Oakland, California, USA, Washington, DC, USA: IEEE Computer Society, 2007: 321-334

[5] Yu S, Wang C, Ren K, et al. Attribute based data sharing with attribute revocation [C] // Proceedings of the 5th International Symposium on Information, Computer and Communications Security (ASIACCS 2010). New York, NY, USA: ACM, 2010: 261-270

[6] Yu S, Wang C, Ren K, et al. Achieving secure, scalable, and fine-grained data access control in cloud computing [A] // Proceedings of IEEE INFOCOM 2010 [C]. San Diego, CA, 2010

[7] 洪澄，张敏，冯登国. AB-ACCS：一种云存储密文访问控制方法 [J]. 计算机研究与发展，2010，47（增刊）：259-265

[8] 吕志泉，张敏，冯登国. 云存储密文访问控制方案 [J]. 计算机科学与探索，2011(09)

[9] 洪澄，张敏，冯登国. 面向云存储的高效动态密文访问控制方案 [J]. 通信学报，2011(07)

[10] Blaze M, Bleumer G, Strauss M. Divertible protocols and atomic proxy cryptography [C] // Proc. of EUROCRYPT '98. 1998

（上接第 37 页）

系统的性能；而路预测算法在单核处理器中已被证明能够很大程度地降低系统功耗。本文将基于公平性的 Cache 划分策略与路预测算法相结合，运用于多核处理器下的共享 L2 Cache 中，在保证系统性能的同时，很好地降低了 L2 Cache 的动态能耗。

众核处理器是未来处理器发展的趋势，随着片上集成的核数增多，对低功耗的要求也会越来越显著。在未来的工作中，将会进一步优化基于 Cache 划分的路预测 L2 Cache 算法，使之在核数更多的处理器系统下仍保持良好的低功耗效果。

参 考 文 献

[1] Ijaykr V, Ishman N, Kandemir M, et al. Energy-driven integrated hardware-software optimizations using simple-power [A] // Proceedings of the 27th Annual International Symposium on Computer Architecture, 2000 [C]. California: IEEE, 2000: 95-106

[2] Lu Jun-yang, Guo Yao. Energy-Aware Fixed-Priority Multi-core Scheduling for Real-Time Systems [A] // The 17th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2011 [C]. California: IEEE, 2011: 277-281

[3] Wang Wei-xun, Mishra P, Ranka S. Dynamic Cache Reconfiguration and Partitioning for Energy Optimization in Real-Time Multi-core Systems [A] // Design Automation Conference. New Jersey, 2011 [C]. IEEE, 2011: 948-953

[4] Reddy R, Petrov P. Cache partitioning for energy-efficient and interference-free embedded multitasking [J]. ACM TECS, 2010, 9(3): 1-35

[5] Inoue K, Ishihara T, Murakami K. Way-Predicting Set-associative Cache for High Performance and Low Energy Consumption [A] // Proceedings on Low Power Electronics and Design. New Jersey, 1999 [C]. IEEE, 1999: 273-275

[6] Chen H-C, Chiang J-S. Low-power Way-predicting Cache Using Valid-bit Predecision for Parallel Architectures [A] // Proceedings of the 19th International Conference on Advanced Information Networking and Applications, 2005 [C]. California: IEEE, 2005: 28-30

[7] Chung C-M, Kim J. Low-power L2 cache design for multi-core processors [J]. Electronics Letters 29th, 2010, 46(9): 618-120

[8] Suh G E, Rudolph L, Devadas S. Dynamic Partitioning of Shared Cache Memory [J]. Journal of Supercomputing, 2004, 28(1): 7-26

[9] 所光，杨学军. 面向多线程多道程序的加权共享 Cache 划分 [J]. 计算机学报，2008，31(11)：1938-1947

[10] Muralidhara S P, Kandemir M, Raghavan P. Intra-Application Cache Partitioning [A] // IEEE International Parallel & Distributed Processing Symposium, 2010 [C]. New Jersey: IEEE, 2010: 1-12

[11] Kim S, Chandra D, Solihin Y. Fair cache sharing and partitioning in a chip multiprocessor architecture [A] // Proceedings of 13th International Conference on PACT, 2004 [C]. California: IEEE, 2004: 111-122

[12] Hsu L R, Reinhard S K, Iyer R. Communist utilitarian, and capitalist cache policies on CMPs: Caches as a shared resource [A] // Proc of PACT-15, 2006 [C]. California: IEEE, 2006: 13-22

[13] Gabor R, Weiss S, Mendelson A. Fairness and throughput in switch on event multithreading [A] // 39th IEEE/ACM International Symposium on Microarchitecture, 2006 [C]. California: IEEE, 2006: 149-160

[14] Iyer, Ravi. A framework for enabling QoS in shared caches of CMP platforms [A] // Proceedings of ICS-18, 2004 [C]. USA: ACM, 2004: 257-266

[15] 隋秀峰，吴俊敏，等. ARP：同时多线程处理器中共享 Cache 自适应运行划分机制 [J]. 计算机研究与发展，2008，45(7)：1269-1277