软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(Suppl.(2)):80−89

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

面向动态异构多核处理器的公平调度算法*

王 涛, 安 虹, 孙 涛, 高晓川, 张海博, 程亦超, 彭 毅

(中国科学技术大学 计算机科学与技术学院,安徽 合肥 230027)

通讯作者: 王涛, E-mail: tao36@mail.ustc.edu.cn

摘 要: 动态异构多核处理器的处理器核可动态调整的特征给操作系统调度算法带来了新的机遇和挑战.利用处理器核动态可调整的特征能更好地适应不同任务的运行需求,带来巨大的性能优化空间.然而也带来新的代价和更复杂的公平性的计算.为了解决面向动态异构多核处理器结构上的公平性调度问题,提出了一个基于集中式运行队列的调度模型,以降低调度算法在动态处理器核变化所带来的维护开销.并重新思考在动态异构处理器结构下公平性的定义,基于原有 CFS 调度算法提出新的 HFS 调度算法.HFS 调度算法不仅能简单而有效地利用动态异构多核处理器的性能优势,而且能提供在动态异构多核处理器上的公平性调度.通过模拟 SCMP,ACMP,DHCMP 平台,证明了提出的 HFS 调度算法能够很好地发挥 DHCMP 结构的性能特征,比运行目前主流调度算法的 SCMP 和 ACMP 结构提升 10.55%的用户级性能(ANTT),14.24%的系统吞吐率(WSU).

关键词: 动态异构多核;集中式运行队列;任务调度;资源分配;公平性算法

中文引用格式: 王涛,安虹,孙涛,高晓川,张海博,程亦超,彭毅.面向动态异构多核处理器的公平调度算法.软件学报,2014,25(Suppl.(2)):80−89. http://www.jos.org.cn/1000-9825/14026.htm

英文引用格式: Wang T, An H, Sun T, Gao XC, Zhang HB, Cheng YC, Peng Y. Fair scheduling on dynamic heterogeneous chip multiprocessor. Ruan Jian Xue Bao/Journal of Software, 2014,25(Suppl.(2)):80−89 (in Chinese). http://www.jos.org.cn/1000-9825/14026.htm

Fair Scheduling on Dynamic Heterogeneous Chip Multiprocessor

WANG Tao, AN Hong, SUN Tao, GAO Xiao-Chuan, ZHANG Hai-Bo, CHENG Yi-Chao, PENG Yi

(School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)

Corresponding author: WANG Tao, E-mail: tao36@mail.ustc.edu.cn

Abstract: Dynamic Heterogeneous CMPs (DHCMP), which provide the capability to configure different number and types of processing cores at system runtime, dramatically improve energy- and power-efficiency by scheduling workloads on the most appropriate core type. A significant body of recent work has focused on improving system throughput through scheduling on asymmetric CMPs (ACMP). However, none of the prior work has looked into fairness. In this work, centralized run queue is introduced and a heterogeneity-aware fair scheduler (HFS) is proposed to address the fair scheduling problem on DHCMP. HFS algorithm can not only gain the capability of DHCMP to configure the types of processing cores to match the granularities of parallelism in the tasks, but also keep the fairness when tasks running simultaneously. Experimental results demonstrate that HFS on DHCMP outperforms the best performing fair scheduler on SCMP and ACMP by 10.55% in user-oriented performance (ANTT), and 14.24% in system throughput (WSU).

Key words: dynamic heterogeneous; centralized run queue; task scheduler; resource allocation; fair algorithm

随着工艺的迅猛发展,同构多核结构(symmetric CMPs,简称 SCMPs)来到了新的叉路口:核数少但每个核结构复杂,提供高的单线程性能、低的并行吞吐率;核数多但每个核结构简单,提供高的线程级并行性、低的串行

* 基金项目: 国家自然科学基金(60970023); 国家重点基础研究发展计划(973)(2011CB302501); 国家高技术研究发展计划(863)(2012AA010902, 2012AA010901)

收稿时间: 2013-08-05; 定稿时间: 2014-03-13

代码性能.于是在芯片上放置不同粒度的处理器核,在使用乱序超标量核开发串行代码性能的同时,仍然能够使用大量的结构简单的处理器核开发线程级并行性的异构多核处理器便应运而生.

实际上,异构多核处理器只有当芯片上处理器核的粒度与任务负载的并行特征相匹配时,才能达到最大的性能/功耗比.然而,不仅不同的应用程序具有不同的并行性特征,而且同一程序的不同执行阶段也可能具有不同的运行特征[1],因此任务负载的并行特征和资源需求不是固定的.这导致静态异构多核处理器(asymmetric CMPs,简称 ACMPs)由于其固定核的计算资源,从而无法实时满足不同的任务负载的不同需求而严重影响其异构设计所带来的计算效率的收益.

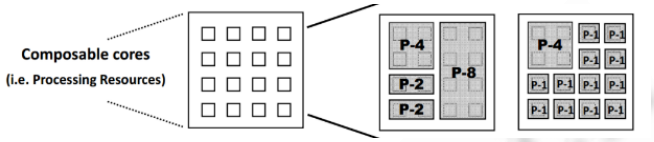


Fig.1 Example 16-base core DHCMP and its 2 runtime configurations

图 1 具有 16 个基本物理核的 DHCMP 处理器示例及其在运行时的两种配置

动态异构多核处理器(dynamic heterogeneous CMPs,简称 DHCMPs)[2−5]与静态异构多核处理器不同的是,DHCMP 是由许多同构的、性能较弱的"基本核"组成,在运行过程中可以实时地组合不同数量的基本核来构成临时的相对性能强大的"逻辑核".这个特性使得 DHCMP 能适应不同任务对资源的不同需求以实现资源的最优分配.图 1 给出了一个 16 个基本核的 DHCMP 的典型例子以及两种动态配置.每个基本核用 *P*–1 表示,一个由 *n* 个基本核构成的逻辑核用 *P*–*n* 表示,一般来说,*P*–*n* 逻辑核拥有 *n* 倍于基本核的发射宽度、指令窗口大小、1 级缓存大小等.当 $x \neq y$ 时,称 *P*–*x* 逻辑核与 *P*–*y* 逻辑核是不同类型的逻辑核.DHCMP 能够在运行过程中实时地释放原有的逻辑核,创建新的逻辑核,以适应系统当前任务负载特征,从而达到效率的最大化.

然而,操作系统任务调度要利用这种动态异构多核处理器实现任务公平性调度却面临着两个主要问题.首先,目前操作系统假设处理器核的数目在运行时是固定的,采用分布式队列将系统当前运行任务分别保存在核各自的运行队列中.但是,动态异构处理器上逻辑核的数量往往随着任务负载的变化而频繁地发生改变,当一个逻辑核释放时,在其运行队列上的任务不得不分发到其他的逻辑核上继续运行;当一个逻辑核被创建时,由于负载不均,其他逻辑核又需要转移部分任务到新的逻辑核的运行队列中.正是这种频繁的改变导致在动态异构处理器上维护当前普遍使用的每处理器核分布式任务队列的开销巨大,见表 1.

表 1 处理器核创建/释放时 Linux 内核数据结构更新的时间开销[6]

处理器核的变化	释放 1 个 *P*–2→2 个 *P*–1	释放 1 个 *P*–44→个 *P*–1	创建 2 个 *P*–1→1 个 *P*–2	创建 4 个 *P*–1→1 个 *P*–4
内核更新开销(ms)	220	640	150	430

另一方面,公平性的重新定义.在"完全理想的多任务处理器"下,每个进程都能同时获得 CPU 的执行时间.当系统中有两个进程时,CPU 的计算时间被分成两份,每个进程获得 50%.然而在实际的硬件上,当一个进程占用 CPU 时,其他进程就必须等待.这就产生了不公平.目前主流操作系统 Linux 中使用的完全公平调度算法(completely fair scheduler,简称 CFS)将处理器使用时间按任务优先级平均分配给任务,在同构多核平台下达到了很好的公平性效果.然而,为了维护 DHCMP 上任务调度的公平性,不仅要考虑处理器的使用时间,还应该考虑不同处理器间性能上的差异.

本文中,我们将实现面向动态异构多核结构下的公平性调度算法 HFS(heterogeneity-aware fair scheduler).首先,我们将采用集中式运行队列来代替 CFS 原有的分布式运行队列,以实现新的 CCFS(CRQ-based CFS)调度算法.集中式运行队列能使调度算法简单、有效地适应动态异构处理器的逻辑核实时变化的特征,降低调度的开销.然后,我们将采用考虑异构核性能的公平性机制提出新的动态异构多核结构下的调度算法 HFS.HFS 将大幅提高 DHCMP 上不同优先级的任务的公平性.最后,为了评估我们的设计,本文采用一个时钟精确的可重构模

拟器 TFlex[2]构造了一个由 32 个基本核构成的 DHCMP、一个由 8 个 *P*–4 性能逻辑核构成的 SCMP 和一个由 4 个 *P*–4 逻辑核和 8 个 *P*–2 逻辑核构成的 ACMP.我们将选取 HFS,CCFS 调度算法实现在 DHCMP 平台上;主流公平性调度算法 CFS,RR,DWRR[7]实现在 SCMP 平台上;ADWRR[8]实现在 ACMP 平台上.由此可以看到 DHCMP 相比于 SCMP 和 ACMP 多核结构的性能优势,以及 HFS 在 DHCMP 平台上实现公平性调度的优化效果.

本文第 1 节主要讨论研究背景和相关工作.第 2 节提出我们采用的集中式调度队列机制,并讨论面向动态异构多核处理器结构下的公平性调度算法.第 3 节给出实验方法设计等内容.第 4 节则是对实验结果的分析与讨论.第 5 节是本文总结和未来工作展望.

1 研究背景与相关工作

从编程的角度,可以将异构多核处理器分为两大类:功能异构多核和性能异构多核.前者是指芯片上不同的处理器核采用不同的指令集,例如,片上一部分核面向通用计算,同时一部分核面向专用应用的加速,例如图形应用、加解密应用等.后者是指芯片上不同的处理器核都采用相同的指令集,处理器核之间的异构只体现在性能上,不同核可以采用不同的微结构设计和制造工艺.与功能异构多核相比,性能异构多核能克服前者由于不同处理器核采用的指令集不同而导致的可编程性、编译、运行时系统设计的困难,而且限制了程序在不同核之间的迁移.本文研究的关注点正是这种性能异构的异构多核处理器.

1.1 动态异构多核结构

近年来,许多研究小组都开始研究设计这种动态异构多核处理器.例如:TFlex[2],Core Fusion[3],Voltron[4],WiDGET[5].这些动态异构多核处理器能让系统软件动态地调整当前逻辑处理器核的数目和大小,实现性能异构的实时变化,使系统获得更好的实时能效比.目前的动态异构多核处理器中的单个逻辑核还不能支持同时多线程(SMT),但多个任务仍然能像普通的多核处理器一样同时运行在不同的逻辑核上.

系统软件可以通过底层提供的硬件原语(创建、释放、改变大小等)来实现逻辑核的重新配置.一般来说,不同的 DHCMP 有不同的硬件原语实现,本文选用的是具有代表性的 TFlex,下面进行简单的介绍.

TFlex 使用显式数据流图(explicit data graph execution,简称 EDGE)指令集以支持逻辑核进行更灵活的粒度调整.相比于传统控制流指令集,EDGE 主要有两个特征:第一,指令超块(hyperblock)的原子执行.编译器负责将程序编译成至多 128 条指令大小的超块,指令的取指、提交都以超块为单位,超块之间按照控制流推测执行.第二,显式数据流编码指令.一个超块内的每条指令都显示编码它的消费者指令的通信地址,超块内的指令以显式数据流驱动的方式执行.

TFlex 在处理器硬件衬底上放置了同构的基本核.在系统运行过程中,单个基本核可以构成粒度最小的逻辑核;如果需要大粒度的逻辑核以开发指令级并行性,则 TFlex 支持以 2 的指数倍调整逻辑核粒度,即一个逻辑核可以由 1/2/4/8/16/32 个基本核组成.粒度为 *n* 的逻辑核具有 *n* 倍于单个基本核的指令发射宽度、指令窗口、计算部件、分支预测器和一级指令/数据缓存大小.

1.2 在SCMP,ACMP,DHCMP上的调度算法

在同构多核处理器上,公平性调度算法的研究十分成熟.*O*(1)调度算法[9],又称为动态优先级算法,是 Linux 内核 2.6.23 之前版本使用的调度算法.而在 Linux 2.6.23 以后的版本中,采用的是新的完全公平调度算法(CFS)[10].DWRR 也是在同构多核平台上提出的公平性调度算法.这 3 种调度算法将会在实验中搭建的 SCMP 平台上进行测评.

在静态异构多核处理器(ACMP)上,已经有多种调度方案来研究如何有效地利用 ACMP 上的异构资源达到优化性能的目的.HASS[11]通过离线的评估来获得实际应用在不同核上的运行效率,之后将评估结果应用在实际的运行过程中.CAMP[12]通过实时采样末级缓存(LLC)的失效率来决定调度执行.HASS,CAMP 都是以最大化系统资源利用率为目标来进行调度的,所以在公平性的表现上将会有所欠缺.ADWRR 是在 ACMP 上在考虑系

统性能的基础上仍然考虑公平性的调度算法.我们将选取 ADWRR 作为 ACMP 平台上的代表性调度算法进行评测.

在动态异构多处理器上的调度算法的研究较少,大部分的研究集中在如何实现资源的有效分配上,如 EQUI,PDPA 分配算法.其原因是这种动态异构多处理器的逻辑核的频繁变化给操作系统调度算法带来巨大的困难所致.

2 面向 DHCMP 的调度算法 HFS

2.1 基于集中式队列的CCFS

由于 Linux 内核采用的是分布式运行队列,在每个处理器核上都拥有一个本地的运行队列,一旦原有处理器核消失或者有新的处理器加入时,内核将会通过 CPU-hotplug 过程对内核数据结构进行更新维护.表 1 列出了这种核的变化所带来的开销.从表中可以看出这种核变化所带来的代价还是比较高的,但是,由于在 SCMP 处理器中,处理器核变化的情况较为少见,这种开销还是可以接受的.然而,在 DHCMP 处理器结构下,逻辑核的变化将会非常频繁,基本可以达到毫秒级别,这说明每处理器核分布式任务队列模型并不能有效地支持动态异构处理器.

我们将使用集中式任务队列(centralized run queue,简称 CRQ)以支持逻辑核数目的快速调整,图 2 是基于集中式任务队列调度模型的示意图.系统中准备好且可以加载到处理器核上的任务都在一个集中式运行队列中进行维护.当一个时钟中断到来时,之前正在处理器核上运行的任务将会按原来分布式运行队列的处理过程更新其相应的数据参数,维护运行队列;然后从集中式运行队列中挑选优先级最高的几个任务,构成执行队列(是 CRQ 的子集);执行队列中的任务将根据底层的分配算法(如 PDPA)计算出最优逻辑核,利用硬件原语创建相应大小的新的逻辑核,加载到对应的逻辑核上运行.这样,既能避免使用分布式运行队列时逻辑核变化给内核数据结构带来的更新维护开销,又能利用底层分配算法实现资源的最优分配,达到更高的性能/效率比.

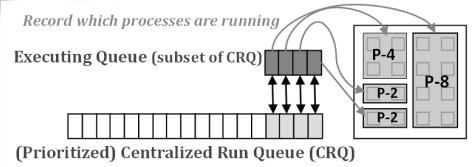


Fig.2 CRQ-Based schedule model on DHCMP

图 2 DHCMP 结构上的基于集中式任务队列调度模型

CFS 算法,即红黑树调度算法,其调度目标是让所有进程获得和其优先级成比例的运行时间,完全公平的调度思想.它是 Linux 2.6.23 以后的版本中采用的调度算法.它没有了时间片的概念,不再跟踪进程的睡眠时间,也不再企图区分交互式进程,它将所有的进程都统一对待.CFS 的算法和实现都相当简单,众多的测试表明其性能也非常优越.其一大亮点就是采用红黑树数据结构来存放运行队列.

为了能够利用 DHCMP 处理器的特点和性能优势,我们提出了基于集中式调度队列的 CFS 调度算法(CRQ-based CFS,简称 CCFS).在 CCFS 算法中,系统中的可执行任务采用上面描述的集中式运行队列模型进行维护,因此,所有的可执行任务将会集中到一起通过红黑树的数据结构进行维护.红黑树的结构如图 3 所示.相比较于分布式的运行队列机制,集中式运行队列调度机制势必会带来负载偏高的问题.但是,由于采用红黑树数据结构,利用红黑树的特点仍然有良好的运行效果.

- 红黑树可以始终保持平衡.
- 对于大多数操作,红黑树的执行时间为 $O(\log n)$.$O(\log n)$行为具有可测量的延迟,但是对于较大的任务数无关紧要.
- 红黑树可通过内部存储实现——即不需要使用外部分配即可对数据结构进行维护.

由于红黑树是二叉树,查找操作的时间复杂度为对数.在分布式队列中,只需要查找最左叶子节点为

pick_next 操作所选取的下一个要加载到处理器的进程.但集中式队列模型中,我们就需要查找出二叉树的中序遍历的前缀子序列,如图 3 中虚线部分所示,找出 4 个任务分配到 DHCMP 上运行.由于查找子序列的过程无需完成整棵红黑树的中序遍历,而是查找到 m(m 为 DHCMP 最多可运行进程数,一般为较小的常数)个节点就可以停止,所以时间复杂度仍为 $O(\log n)$.因此 CCFS 调度算法仍然可以在 DHCMP 结构下保持良好的运行效果.

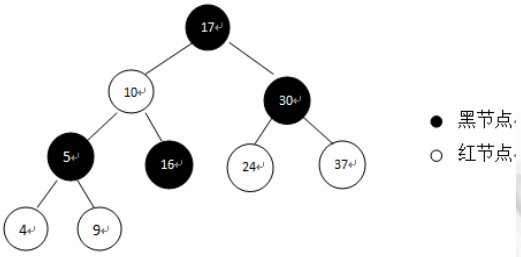


Fig.3 RB-Tree and the choice of scheduler

图 3 红黑树结构以及 CCFS 的执行队列的选取

2.2 考虑异构性能的公平调度算法

调度公平性是任何多任务(进程)系统能够提供稳定的性能和服务质量的关键.在多任务系统中,用户通过对任务设置优先级来反映对任务所能获得性能的期望.在传统的单核处理器上,由于多个任务采用时分复用的方法共享处理器资源,所以调度程序只要按照优先级比例分配时间片,就可以保证基于优先级的调度公平性.因此,传统的调度公平性围绕在 CPU 的使用时间这一概念上,以 CFS 为例,其红黑树键值由 3 个因子计算而得:一是进程已经占用的 CPU 时间;二是当前进程的 nice 值;三是当前的 CPU 负载.其中,进程已经占用的 CPU 时间对键值的影响最大,实际上,很大程度上我们在理解 CFS 时可以简单地认为键值就等于进程已占用的 CPU 时间.在传统的同构多核处理器上,这种假定能够获得很好的公平性效果.

然而,在异构平台下,这种只考虑时间的假定是不够的.简单的例子:同样优先级的两个任务,任务 1 在 P–4 上运行 1ms,任务 2 在 P–1 上运行 1ms,CFS 中将认为这两个任务获得了同样的处理器性能,因为原来的同构多核处理器的每个核的性能相同.但实际情况是,任务 1 获得的计算性能是任务 2 的 4 倍(假设 P–4 的性能是 P–1 的 4 倍).于是就产生了不公平.

因此,我们需要重新定义异构平台下任务调度的公平性.从之前这个简单的例子可以看出,传统公平性的计算只考虑时间的原因正是因为同构多核处理器每个核的性能是完全相同的,没有必要进行区别对待,而在异构多核处理器中,不同的任务运行在不同的处理器核上,任务获得的计算性能是不同的,如果不考虑这个区别,将会导致公平性的缺失.特别是,同一个任务在运行的过程中也有可能由于迁移而转移到不同的处理器核上运行,单一记录 CPU 运行时间不论对于不同的任务之间还是对于同一个任务自身都是不公平的.于是,在异构平台下的公平性将从单一的时间决定转变为由时间和核的性能共同决定.在处理器性能资源占用的计算上,我们以 P–1 为单位性能,于是运行在 P–n 逻辑处理器上的任务占用的性能资源为 n.

我们采用 CFS 调度算法的思想,基于集中式调度队列,提出面向动态异构多核处理器的调度算法 HFS(heterogeneity-aware fair scheduler).HFS 为每个进程都维护两个重要变量:*fair_perf* 和 *wait_runtime*.与 CFS 调度算法中的 *fair_clock* 不同,*fair_perf* 是一个进程已经获得的 CPU 性能统计.具体如公式(1)所示.其中,CPU_i_perf 为处理器核 i 对应的性能,$runtime_i$ 为进程在处理器 i 上的运行时间,*priority* 表示进程的优先级.

$$fair\_perf = \left(\sum_{i=0}^{n} CPU_i\_perf \times runtime_i\right) \Big/ priority \tag{1}$$

wait_runtime 是进程的等待时间.进程插入红黑树的键值即为 *fair_perf* 和 *wait_runtime* 的差值.它们的差值代表了一个进程的公平程度.该值越大,代表当前进程相对于其他进程越不公平.

假设当前一个进程运行在处理器核 i 上,由于 *fair_perf* 是进程不断更新的统计量,虽然公式(1)中需要累积进程从产生到目前所有的占用性能统计,但之前时钟周期(上一次调度)的历史值 $fair\_perf_{last}$ 是记录在进程中

的,因此只需再加上本次时钟周期占用处理器 i 所获得的性能 $CPU_i\_perf \times runtime_i$ 就可以更新 $fair\_perf_{now}$ 值,如公式(2)所示,因此计算开销很小.

$$fair\_perf_{now} = fair\_perf_{last} + (CPU_i\_perf \times runtime_i)/priority \quad (2)$$

下面我们将通过第 3 节的实验部分比较 HFS 与 CCFS 调度算法在公平性上表现的差距,并可以看出 HFS 调度算法能够很好地适应动态异构多核处理器的特征,在保持最优系统性能的基础上保证任务的公平性调度.

3 实验方法设计

3.1 模拟SCMP,ACMP,DHCMP

本次实验在一个时钟精确的 DHCMP 结构模拟器 TFlex[1]上展开.我们使用 TFlex 配置具有相等计算资源总数的 SCMP,ACMP 和 DHCMP 处理器.表 2 列出了 TFlex 结构上每个基本核的参数配置.由于实验配置的每种处理器结构的计算资源总数都相等(32 个基本核),因此负载在各处理器平台上的执行性能是可以相互比较的.

表 3 列出模拟 SCMP,ACMP,DHCMP 这 3 种处理器结构的具体配置及在相应结构平台上实现的调度算法.

表 2 TFlex 上基本核的具体配置参数

Instruction supply	L1 I-cache (8KB, 1-cycle hit); Local/Gshare Tournament predictor (8K+256 bits, 3 cycle latency), Local: 64(L1)+128(L2), Global: 512, Choice: 512, RAS: 16, CTB: 16, BTB: 128, Btype: 256
Execution	Dual-issue & Out-of-order execution; RAM structured 128-entry instr window; INT-ALU: 2. FP-ALU: 1
Data supply	L1 D-cache (8KB, 2-cycle hit, 2-way set-associative, 1-read+1-write port); 44-entry LSQ bank

表 3 实验配置的处理器结构及实现的调度算法

处理器类型	SCMP	ACMP	DHCMP
处理器配置	8×(P–4)	4×(P–4)+8×(P–2)	32×基本核
调度算法	RR/CFS/DWRR	ADWRR	CCFS/HFS

3.2 任务负载的构建

我们将从 SPEC2K 和 EEMBC 中选取部分具有代表性的程序来组成我们的测试任务负载.为了涵盖大部分程序对处理器核资源的需求的不同特征,将程序分成 3 类进行挑选.

- (L)类:该类程序主要具有访存限制(memory-bound)特征,对计算资源需求低,如 mcf 和 tblook.
- (M)类:该类程序有一定程度的计算密集特征,如 art;或者可以在大核上建立比较高的访存并行度(MLP),如 applu.该类程序在小核、大核上皆可运行.
- (H)类:该类程序通常包含大量浮点计算,具有典型的计算密集(computing-intensive)特征,对计算资源的需求高,适合在大核上运行.

经过这样的分类,我们从每种类别中选取部分程序来组成我们总体的任务负载,使得任务负载更具有代表性和说服力,具体的任务负载组成见表 4.任务负载 W1,W2 是由相同优先级的同一程序构成;任务负载 W3,W4 是由不同优先级的同一程序构成,30 个程序中有 15 个程序的优先级是另外 15 个程序的 2 倍;任务负载 W5, W6 是由相同优先级的不同程序构成.

由于是在模拟器上进行测评,我们使用 simpoint[13]工具来处理每个程序,压缩整个实验评测过程同时保证实验结果的可靠性.为了保证每个选取出的程序在 P–1 处理器核上运行时间相同,我们对每个程序都截取 200M 时钟周期的间隔程序段进行评测.

表 4 任务负载的构建

负载编号	负载的构成
W1	30 个相同优先级的 tblook(L)
W2	30 个相同优先级的 fft(H)
W3	同 W1,但优先级不同
W4	同 W2,但优先级不同
W5	(15L)3×mcf,6×tblook,6×cacheb+(15H)6×rspeed,3×mgrid,6×fft
W6	(10L) 2×mcf,4×tblook,4×cacheb+(10M)4×canrdr,2×applu,4×matrix+(10H)4×rspeed,2×mgrid,4×fft

3.3 性能指标

本实验中,所有任务负载中的程序都是同时进入系统.我们使用完成所有程序的平均使用时间(mean_time)来表征各个处理器平台的性能表现.

为了评估公平性,我们采用 StrictFairness[14]来衡量各个调度算法在公平性上的表现.为了比较调度算法在用户层和系统层的性能表现,分别使用平均标准化周转时间 ANTT(average normalized turnaround time)[15]和权重加速比 WSU(weighted speedup)[14,15]两个测量指标.各个指标由如下公式定义:

$$StrictF = \frac{task\_finish\_time_{best}}{task\_finish\_time_{worst}} \tag{3}$$

$$ANTT = \frac{1}{n}\sum_{i=1}^{n}\frac{C_i^{MP}}{C_i^{SP}} \tag{4}$$

$$WSU = \sum_{i=1}^{n}\frac{C_i^{SP}}{C_i^{MP}} \tag{5}$$

其中, C_i^{MP} , C_i^{SP} 分别代表任务 i 在多任务环境下执行完成所需时间和自己单独运行时执行完成所需时间.

4 实验结果分析

4.1 同构程序组成的负载的公平性分析

W1,W2 是有相同优先级的相同任务,其运行结果如图 4 所示.从图 4 中可以看出,每个调度算法的 StrictF 都很高,除了 ADWRR 外都超过 0.9,表明在这种同构同优先级的负载中,公平性的维持是没有问题的.虽然在 ADWRR 算法中考虑到了异构核所带来的性能不同,但并没有考虑同一个程序在不同核上的加速比也是不同的,因此公平性上略有下降.

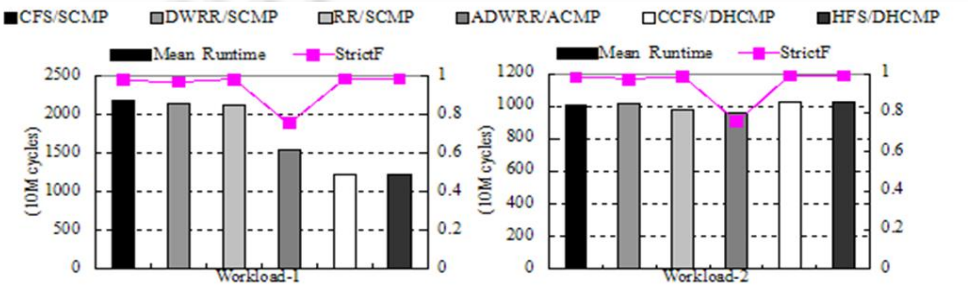


Fig.4 Results of W1 and W2, homo-tasks and same priority

图 4 W1 和 W2 负载的运行结果,同构同优先级任务

从整体结构性能表现上来看,DHCMP 的性能要优于 SCMP 和 ACMP.图 4 中 W1 和 W2 在性能上差异的不同恰好体现了动态异构多核处理器能适应不同任务负载的能力.W1 中的 tblook 只需要小核.而 W2 中的 fft 就需要性能更高的大核,而 DHCMP 在运行 fft 时会为其分配 *P*–4 的逻辑核,因此就退化成了实验中的 SCMP 结构,因此两者性能差异不大.

W3,W4 唯一与 W1,W2 不同的地方在于将 30 个任务平均分成两组,一组设定为低优先级,另一组为高优先级,两者优先级比为 2:1.如果调度算法完全公平的话,两组程序运行结束所用时间应该也是 2:1.实验结果见表 5.从表中可以看出,在各个平台下调度算法的表现良好,结果都接近于 2.

表 5 W3,W4 任务负载高低优先级两组任务完成时间的统计

结构类型	SCMP		ACMP	DHCMP	
调度算法	CFS	DWRR	ADWRR	CCFS	HFS
W3	2.00	1.98	2.06	2.00	2.00
W4	1.90	2.14	1.97	1.95	1.91

4.2 异构程序组成的负载公平性分析

W5,W6 是由相同优先级的不同程序组成的负载,更接近于真实环境下的负载状况,由于所有程序的理想运行时间都是相同的,并且程序的优先级相同,理想状态下所有程序的完成所用的时间应该是相同的.结果如图 5 所示.从图中的 StrictF 指标可以看出,即使是之前公平性表现很稳定的调度算法在这种情况下的表现也都不是很好,特别是在 SCMP 结构下仍然无法达到很好的效果,最好的也只有 0.31.根本原因是,同样的处理器核,不同的任务在其上运行获得的性能是不同的.即(H)类的程序比(L)类的程序在 SCMP 核上运行获得的 IPC 加速比(相比较于 P–1)要高.DHCMP 结构底层的分配算法 PDPA 能保持(H)类程序和(L)类程序获得的 IPC 加速比相同,但 CCFS 仍然表现不佳,只有 0.30,原因正是之前提到的,即没有将处理器核的性能加以考虑.于是,可以看到,采用新的公平性定义的 HFS 调度算法在公平性上的表现为 0.603,是 CCFS 的 2 倍.而且这种公平性的优化并没有带来性能上的损失,反而有一定提升,因为更好的公平性能使系统负载更为均衡,减少 workload 在运行结束前出现仅剩的几个任务无法充分利用系统资源的现象.

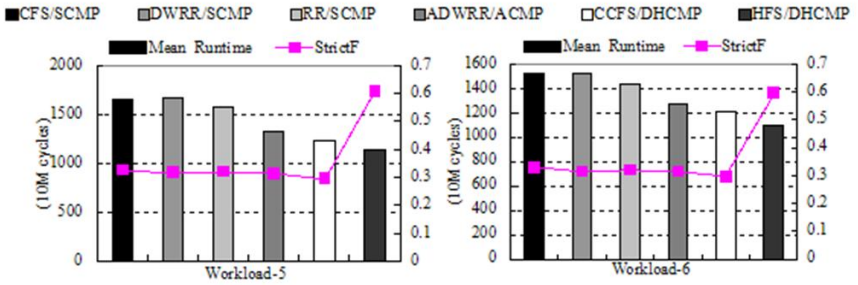


Fig.5 Results of W5 and W6, heter-tasks and same priority

图 5 W5 和 W6 负载的运行结果,异构同优先级的任务

图 6 给出各种调度算法搭配不同平台所取得的用户级性能(ANTT)和系统吞吐率(WSU).从图中我们可以总结出,HFS 调度算法能够很好地发挥 DHCMP 结构的性能特征,比运行目前主流调度算法的 SCMP 可提升 20.96%的用户级性能(ANTT),31.44%的系统吞吐率(WSU);比 ACMP 结构提升 10.55%的用户级性能(ANTT),14.24%的系统吞吐率(WSU).这个结果从一个侧面也反映出,在真实的运行环境中,系统的任务负载往往具有异构性和随机性,而传统的 SCMP 处理器结构无法根据任务负载的特征做出变化.因此,ACMP 结构在异构任务负载情况下比起 SCMP 结构有较大优势.但是,ACMP 结构虽然是异构结构,但其仍然保留了静态的缺点,在灵活性上仍然不如 DHCMP 结构.HFS 调度算法能够动态识别出当前任务的资源需求特征,并充分利用 DHCMP 结构的计算资源的可动态分配的特性,使处理器的资源利用率最大化,从而提高了系统的整体性能.

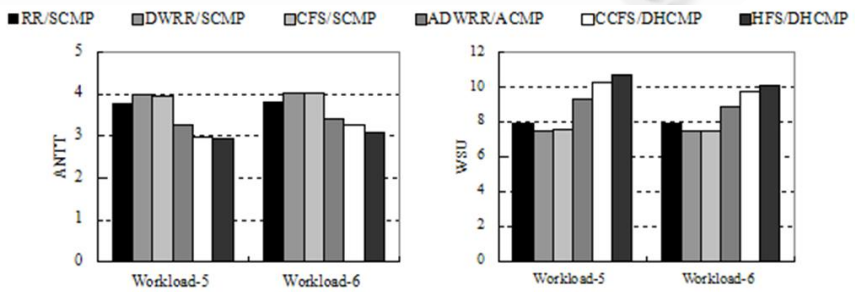


Fig.6 Results of ANTT and WSU in W5 and W6

图 6 运行 W5 和 W6 统计的用户级性能以及系统吞吐率

5 总结和未来工作展望

动态异构多核处理器的处理器核可动态调整的特征给操作系统调度算法带来了新的机遇和挑战.利用处理器核动态可调整的特征能够更好地适应不同的任务运行环境,带来巨大的性能优化空间.然而也带来新的代价和更为复杂的公平性的计算.为了解决面向动态异构多核处理器结构上的公平性调度问题,本文提出了一个基于集中式运行队列的调度模型,降低调度算法由于动态处理器核变化所带来的维护开销,并重新思考定义在动态异构处理器结构下公平性的定义,基于原有 CFS 调度算法提出 HFS 调度算法,使得公平性在动态异构多核处理器中也有良好的表现.通过实验,其结果表明我们提出的 HFS 调度算法能够很好地发挥 DHCMP 结构的优势,比运行目前主流调度算法的 SCMP 和 ACMP 结构可提升 10.55%的用户级性能(ANTT),14.24%的系统吞吐率(WSU).

HFS 调度算法调用 DHCMP 底层的 PDPA 分配算法来实现任务与逻辑处理器的分配,但 PDPA 算法本身的性能还有待提高,可以通过优化 PDPA 算法来实现更好的调度性能.另外,由于不同特征的程序对计算资源的需求是不同的,在同样的核上获得的性能也有一定的差异,考虑程序的特征将有助于 HFS 调度算法实现更好的公平性.

本文提出的 HFS 调度算法没有将系统资源按计算资源和存储资源分开,而将二者进行综合考虑,但实际中的程序仍然可以很明显地区分出计算密集型还是访存密集型,因此加以区分将会使任务的资源需求特征更加精确.另外,HFS 调度算法并没有考虑到多任务负载在末级缓存(LLC)上的竞争问题,但在我们的实验过程中发现,一些(M)类的应用程序(如:canrdr,matrix)对共享缓存是较为敏感的.未来的工作中,我们将会继续对上述两方面作更深入的考量,从而更精确地衡量一个程序的资源需求并实现更智能的调度策略.

致谢 在此,我们向对本文的工作给予支持和建议的同行,尤其是 Doug Burger、孙荪、陈俊仕等人表示感谢.

References:

[1] Sherwood T, Sair S, Calder B. Phase tracking and prediction. In: Proc. of the 30th Annual Int'l Symp. on Computer Architecture. San Diego: ACM, 2003. 336−349.

[2] Kim C, Sethumadhavan S, Govindan M, Ranganathan N, Gulati D, Keckler SW, Burger D. Composable lightweight processors. In: Proc. of the 40th Int'l Symp. on Microarchitecture. 2007. 281−294.

[3] Ipek E, Kirman M, Kirman N, Martínez JF. Core fusion: Accommodating software diversity in chip multiprocessors. In: Proc. of the ISCA 2007. 2007. 186−197.

[4] Zhong H, Lieberman SA, Mahlke SA. Extending multicore architectures to exploit hybrid parallelism in single-thread applications. In: Proc. of the HPCA2007. 2007. 25−36.

[5] Watanabe Y, Davis JD, Wood DA. WiDGET: Wisconsin decoupled grid execution tiles. In: Proc. of the ISCA 2010. 2010.

[6] Panneerselvam S, Swift MM. Dynamic processors demand dynamic operating systems. In: Proc. of the 2nd USENIX Conf. on Hot Topics in Parallelism. Berkeley: USENIX Association, 2010. 9.

[7] Li T, Baumberger D, Hahn S. Efficient and scalable multiprocessor fair scheduling using distributed weighted round-robin. In: Proc. of the PPoPP 2009. 2009. 65−74.

[8] Li T, Brett P, Knauerhase R, Koufaty D, Reddy D, Hahn S. Operating system support for overlapping-ISA heterogeneous multi-core architectures. In: Proc. of the HPCA 2010. 2010. 1−12.

[9] IBM Developer Works. Inside the Linux scheduler. 2006. http://www.ibm.com/developerworks/linux/library/l-scheduler

[10] Molnar I. Modular scheduler core and completely fair scheduler [CFS]. 2008. http://lwn.net/Articles/230501

[11] Shelepov D, Saez JC, Jeffery S, Fedorova A, Perez N, Huang ZF, Blagodurov S, Kumar V. HASS: A scheduler for heterogeneous multicore systems. ACM OS Rev., 2009,43(2):66−75.

[12] Saez JC, Prieto M, Fedorova A, Blagodurov S. A comprehensive scheduler for asymmetric multicore systems. In: Proc. of the EuroSys 2010. 2010. 139−152.

[13] Sherwood T, Perelman E, Calder B. Basic block distribution analysis to find periodic behavior and simulation points in applications. In: Proc. of the PACT 2001. 2001. 3−14.

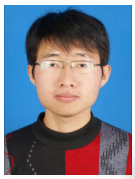
[14] Vandierendonck H, Seznec A. Fairness metrics for multi-threaded processors. IEEE Comp. Arch. Letter, 2011,Issue 1.

[15] Eyerman S, Eeckhout L. System-Level performance metric for multi-program workloads. IEEE Micro, 2008,28(3):42−53.

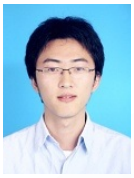
王涛(1988－),男,福建长乐人,硕士,CCF 学生会员,主要研究领域为多核众核芯片体系结构,异构平台任务调度.
E-mail: tao36@mail.ustc.edu.cn

安虹(1963－),女,教授,CCF 高级会员,主要研究领域为计算机系统结构,并行处理.
E-mail: han@ustc.edu.cn

孙涛(1987－),男,博士,主要研究领域为众核结构系统的性能优化.
E-mail: suntaos@mail.ustc.edu.cn



高晓川(1989－),男,硕士生,主要研究领域为计算机系统结构.
E-mail: gxc0805@mail.ustc.edu.cn



张海博(1989－),男,硕士,CCF 学生会员,主要研究领域为计算机系统结构.
E-mail: kopcarl@mail.ustc.edu.cn

程亦超(1989－),男,硕士生,主要研究领域为并行程序设计.
E-mail: yichao@mail.ustc.edu.cn

彭毅(1990－),男,硕士生,主要研究领域为计算机系统结构.
E-mail: peng1990@mail.ustc.edu.cn