第45卷 第6期 西 安 交 通 大 学 学 报 Vol. 45 No. 6
2011年6月 JOURNAL OF XI′AN JIAOTONG UNIVERSITY Jun. 2011

多核编程模型运行时环境的自适应性研究

曹仰杰[1]，杨海兵[1]，钱德沛[1,2]，伍卫国[1]
(1. 西安交通大学电子与信息工程学院，710049，西安；2. 北京航空航天大学计算机学院，100191，北京)

摘要：针对多核编程模型运行时环境易造成处理器核资源竞争加剧以及可扩展性较差等弊端，基于动态反馈控制思想，将资源分配、运行时控制、任务执行视为有机整体，提出了自适应协同调度模型ACSM. ACSM采用集中式与分布式相结合的协同机制，动态调节处理器核资源在不同应用负载间及其内部的分配与管理. ACSM的优势在于充分体现了多核编程模型良好的可编程性和可移植性，消除了传统多核运行时环境显式指定核数的弊端，增强了处理器核资源分配的高效性和自适应性. 实验结果表明，ACSM在提高多核编程模型易用性的同时，减少了系统处理器核资源的不良竞争，提升了系统的整体性能和资源利用率. 与仅依赖多核编程模型运行时环境的调度算法相比，ACSM使应用程序的运行时间缩短了近50%，并且随着应用程序数量的增加效果更加显著.

关键词：多核；编程模型；运行时环境；协同调度
中图分类号：TP301 **文献标志码**：A **文章编号**：0253-987X(2011)06-0130-05

On Adaptability of the Runtime Environment for Emerging Multi-Core Programming Models

CAO Yangjie[1], YANG Haibing[1], QIAN Depei[1,2], WU Weiguo[1]
(1. School of Electronics and Information Engineering, Xi′an Jiaotong University, Xi′an 710049, China;
2. School of Computer Science and Engineering, Beihang University, Beijing 100191, China)

Abstract: The adaptability and collaboration of the multi-core runtime system is studied to address the problems that the current multi-core runtime can easily lead to intensified competition for processor resources and the system scalability is inferior. An adaptive and collaborative scheduling model, named ACSM, is presented based upon the dynamic feedback-control principle by taking resource allocation, runtime control, and task execution as a holistic system. The ACSM dynamically reallocates and manages processor resources among and within workloads in both centralized and distributed manners. The superiorities of ACSM over the current multi-core runtime system are as follows. The ACSM maintains good programmability and portability, enhances efficiency and adaptability in processor resources allocation, and eliminates the need of explicitly specifying the number of cores. The experiment results show that ACSM greatly reduces the competition of processor resources and improves both the overall system performance and the usability of the current multi-core programming models. Comparisons with the scheduling algorithm that relies only on the original multi-core runtime show that applications of ACSM reduce the run time by about 50% or even more, especially when the system load increases.

Keywords: multi-core; programming model; runtime system; collaborative scheduling

---

收稿日期：2011-03-23. 作者简介：曹仰杰(1976—)，男，博士生；钱德沛(联系人)，男，教授，博士生导师. 基金项目：国家自然科学基金资助项目(61073011)；国家高技术研究发展计划资助项目(2009AA01A135，2009AA01A13)；中意国际合作项目(2009DFA12110).
网络出版时间：2011-04-22 网络出版地址：http://www.cnki.net/kcms/detail/61.1069.T.20110422.1045.001.html

近年来,处理器芯片制造受到功耗、散热、集成度与频率等诸多壁垒的束缚,促使多核、多线程成为处理器技术发展的新动力[1-2]. 当前,增加单片处理器内部执行核心的规模已成为提升处理器性能的必然趋势. 为了充分发挥片内大规模并行处理部件的性能,针对多核处理器系统的编程模型如 OpenMP[3]、TBB[4] 和 Cilk[5-6] 等发展迅速,应用日趋广泛. 与传统并行编程模型如 PThread 和 MPI 等相比,多核编程模型具有易编程性、运行时环境支持、可移植性强等优势. 然而,目前多核编程模型在实际应用中仍存在以下弊端.

(1)资源竞争加剧. 虽然多核编程模型运行时环境通常都包含针对其内部线程的负载均衡策略,如 OpenMP 分别支持静态、动态和运行时等调度策略[3],Cilk、TBB 支持 Work-Stealing 调度策略[4-5],但是这些调度策略仅局限于单一应用程序内部工作线程的负载均衡,在很大程度上忽视了系统整体计算资源的协调分配,易造成不必要的资源竞争,导致系统吞吐率降低.

如图 1 所示,在浪潮英信 NF5220 8 核(16 线程)服务器上同时运行基于 Cilk Strassen 的并行程序[5]以及基于 OpenMP 和 TBB 的 Blackscholes 并行程序[7],并以非抢占式先到先服务(FCFS)作业调度策略的最短完成时间(Makespan)为参照,结果表明目前的多核运行时环境造成了比较大的资源竞争,应用程序的运行时间随着系统负载的增加而显著上升.

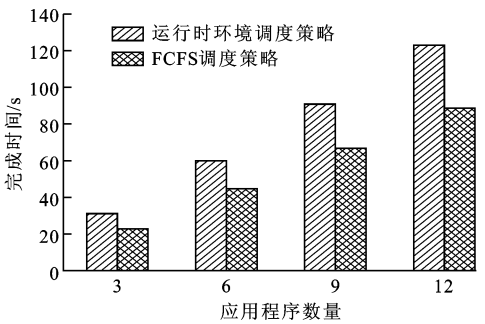

图 1　多个并行应用程序同时运行的情况

(2)系统性能优化困难. 目前支持多核编程模型的各类运行时环境相互独立,彼此之间不能通信,当多种编程模型混合使用时,系统整体性能难以优化,并且目前的操作系统也缺乏针对不同多核编程模型运行时环境的性能优化方法,使得新型编程模型在实际应用中存在较大性能瓶颈.

(3)可扩展性较差. 多核编程模型运行时环境在程序运行时通常需要显式或隐式指定系统所包含的核数,以达到较高的系统利用率. 然而,随着片内处理器核数的快速增长,未来单个芯片内将包含数十或上百个处理核心,显式指定核数的运行方式将严重制约系统的可扩展性和可移植性,最终导致系统整体利用率降低,应用程序性能很难达到与处理器核数成正比提高.

针对多核编程模型目前面临的问题与不足,本文研究了多核运行时环境的自适应协同问题,基于动态反馈控制思想,将资源分配、运行时控制、任务执行视为有机整体,提出了自适应协同调度模型 ACSM. ACSM 采用集中式与分布式相结合的协同机制,动态调节处理器核资源在不同应用负载间及其内部的分配与管理. 集中式表现为处理器核资源由资源控制器统一向运行时环境进行分配,确保资源分配的公平性和有效性,而处理器核资源则由运行时环境自主、分布式地管理与调度,确保资源使用的灵活性. ACSM 的优势在于充分体现了多核编程模型良好的可编程性和可移植性,消除了传统多核运行时环境显式指定核数的弊端,增强了处理器核资源分配的高效性和自适应性. 实际原型系统的验证结果表明,ACSM 在提高多核编程模型易用性的同时,减少了系统处理器核资源的不良竞争,提升了系统的整体性能和资源利用率. 与仅依赖多核编程模型运行时环境的调度算法相比,ACSM 使应用程序的运行时间缩短了近 50%,并且随着应用程序数量的增加效果更加显著.

1　自适应协同调度模型

针对目前的多核编程模型运行时环境,自适应协同调度模型 ACSM 从以下几个方面对其进行了改进. ①增强多核编程模型的易用性. 去除当前多核编程模型运行时环境需要显式或隐式(通过函数调用获取)指定系统所包含的处理器核数的限制,由 ACSM 根据系统当前的负载状况自动为应用程序分配处理器核资源,消除手动设定易引发系统性能瓶颈的弊端. ②引入动态反馈控制机制. 多核编程模型由于自身运行时环境的支持,其并行度可灵活划分,反馈控制机制的引入可以有效获取任务负载的运行时信息,用以指导处理器核资源的动态调度与再分配,优化竞争-共享处理器核资源的使用效率,提升系统整体吞吐量. ③增强自适应性和协同能力. 由于多核编程模型的运行时环境都是针对特定编程模型设计的,具有很强的针对性和自治性,为避免对处理器核资源的恶性竞争,ACSM 通过在不同运行

时环境中增加交互控制，动态协调和优化运行时环境配置，提升整个系统的自适应性和扩展能力. 例如，当由于系统升级而使多核处理器核数增加时，通过 ACSM 及运行时环境的动态调节，应用负载不需作任何设置即可获取更多可用的处理器核资源.

ACSM 的逻辑结构如图 2 所示，整个系统的任务负载被划分为受控任务和非受控任务. 受控任务是 ACSM 的主要控制对象，是由多核编程模型如 Cilk、OpenMP 等的运行时环境支持的并行程序. 非受控任务是指传统的串行程序及无多核运行时环境支持的并行程序，如 PThread 等. ACSM 对于非受控任务采用简单控制策略，即在程序执行的初始时刻，依据系统负载情况分配相应的处理器核资源，运行过程中不再对此类任务进行控制，任务结束后进行资源回收. ACSM 对非受控任务的统一管理减少了受控任务和非受控任务间对处理器核资源的不良竞争，可有效提升和优化系统的整体效能. 受控任务的动态管理和调度是 ACSM 的核心部分，为此基于动态反馈控制思想，通过将资源分配、运行时控制、任务执行视为有机整体，研究了适用于多核运行时环境的自适应协同控制算法.

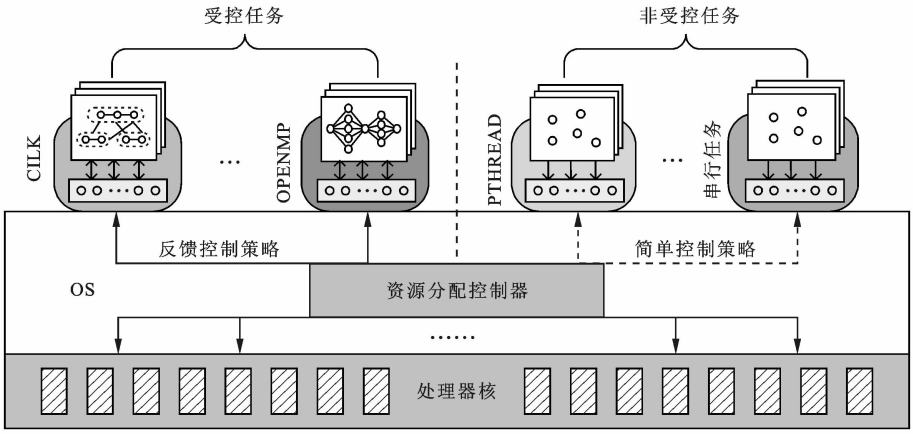

图 2 ACSM 逻辑结构

2 基于 Work-Stealing 的自适应控制算法

Work-Stealing 是目前多核运行时环境如 Cilk、TBB 等广泛支持的调度算法[8]. 基于动态反馈控制思想，本文提出一种适宜于自适应协同模型 ACSM 的 Work-Stealing 调度算法——A-WS（adaptive Work-Stealing），并在多核编程模型 Cilk 的运行时环境中进行了实现与验证.

A-WS 主要包括控制处理器核运行状态和反馈处理器核使用状况两大基本功能，其基本工作原理如下：在初始状态，A-WS 针对每个处理器核创建与之相对应的工作线程（Worker）；在运行过程中，A-WS 依据系统资源分配器对处理器核资源的周期性调节与分配，动态地调整工作线程的状态使其与实际分配保持一致，并周期性地反馈处理器核的使用状况，以指导系统资源分配器下一步的分配方案. 为实现对工作线程的动态管理，A-WS 赋予每个 Worker 4 种状态：Working，Stealing，Mugging，Sleeping. Working 表示工作线程处于工作状态，即本地工作队列栈非空；Stealing 表示工作线程为窃取状态，即本地工作队列栈为空，依据设定的窃取协议窃取其他工作线程的工作负载；Mugging 是一种特殊的 Stealing，即当被窃取的工作线程处于 Sleeping 状态时，迁移被窃取对象的整个工作负载；Sleeping 表示工作线程处于休眠状态. A-WS 算法中工作线程的状态转换图如图 3 所示.

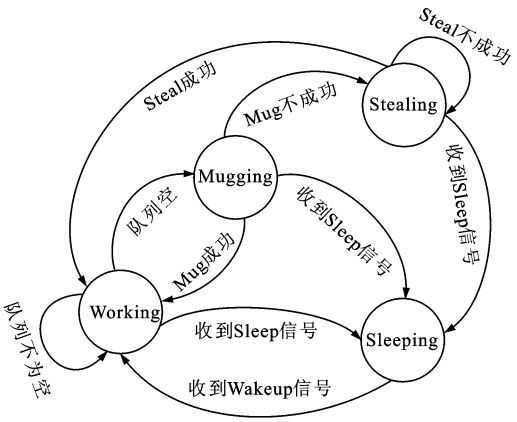

图 3 工作线程(Worker)运行时的状态转换图

在每个调度周期末，A-WS 依据工作线程的平均利用率，采用乘性增乘性减的启发式策略[9]反馈下一周期处理器核资源的需求量，计算公式如下

$$d(q+1)=\begin{cases} d(q)\cdot\rho, & u(q)\geqslant\eta \\ d(q)/\rho, & \text{其他} \end{cases} \quad (1)$$

式中：q 为当前调度周期；$d(q)$、$d(q+1)$分别表示当前和下一个周期处理器核的期望量；$u(q)$表示周期 q 内的处理器核平均利用率；ρ、η 分别表示资源调节响应系数(通常为 2)和核资源利用率阈值(通常为 80%).

动态反馈机制不可避免地会引发系统开销，但是在多核处理器系统中，可以设置独立的处理器核来负责反馈信息的收集、处理和控制，通过并行处理有效屏蔽由此而带来的系统开销. 实际测试结果表明，并行处理方式引发的系统开销较低，在调度周期稍大(>10 ms)时系统开销可以忽略不计. 此外，随着多核处理器核数的增多，设置独立处理器核负责反馈信息收集、处理及控制是可行的.

3　实验验证

基于 Linux 2.6.18 内核，以共享存储、信号量等为进程通信机制，采用 Linux 后台守护进程方式实现处理器核资源分配控制进程. 基于 POSIX Threads 重新改写 Cilk 运行时环境使其支持 A-WS 算法，实现运行时处理器核资源的动态调节与重新分配，并命名为 ACilk (adaptive Cilk). 实验环境为浪潮英信 NF5220 8 核(支持 16 线程)服务器，8 GB 内存；采用 GCC 编译器，版本为 4.1.2，程序编译时的优化级别选项为-O2. 采用经典 CK、Heat、LU、Strassen 为测试程序，具体说明见表 1. 实验过程：对于每一种测试程序，分别生成 $n(2\leqslant n\leqslant 5)$种测试用例，每个测试用例批量提交与其编号数相同的测试程序，以产生不同情况的系统负载，例如 CK 的测试用例 2 表示同时提交 2 个 CK 测试程序. 使用原始 Cilk 和 ACilk 分别执行上述测试用例的完成时间以及这 2 种时间之比($R=t_{\mathrm{ACilk}}/t_{\mathrm{Cilk}}$)如表 1 所示.

表 1　Cilk 与 ACilk 的性能测试结果对比

测试程序	测试用例 2			测试用例 3			测试用例 4			测试用例 5		
	t_{Cilk}/s	t_{ACilk}/s	R/%	t_{Cilk}/s	t_{ACilk}/s	R/%	t_{Cilk}/s	t_{ACilk}/s	R/%	t_{Cilk}/s	t_{ACilk}/s	R/%
CK	22.1	13.9	0.63	30.0	17.9	0.60	41.1	20.8	0.51	57.5	25.3	0.44
Heat	21.6	19.2	0.89	34.9	27.9	0.80	45.3	36.3	0.80	56.1	46.6	0.83
LU	75.2	73.0	0.97	118.8	108.7	0.91	246.9	144.3	0.58	309.6	180.0	0.58
Strassen	8.2	7.6	0.93	14.1	10.9	0.77	49.0	14.1	0.29	66.0	17.4	0.26
平均值			0.85			0.77			0.54			0.53

CK：西洋跳棋模拟，黑白双方搜索深度分别为 10 和 13；Heat：基于 Jacobi 迭代的热扩散方程求解，规模为 4 096×2 048；LU：LU 矩阵分解，规模为 8 192×8 192；Strassen：基于 Strassen 算法的矩阵相乘，规模为 4 096×4 096.

实验结果表明，自适应协同调度模型更好地体现了多核编程模型的优势，系统效能提升显著. 当系统内的任务负载数增大到 4 时，与原始 Cilk 相比，ACilk 使应用程序的执行时间缩短了近 50%，并且随着系统负载的增加，效果更加显著. 由于 ACSM 消除了多核运行时环境需要显示指定处理器核数的弊端，通过运行时的自适应协同调节，改善和优化了处理器核资源的分配与使用，减少了不同应用程序对处理器核资源的恶性竞争，因此使并行应用程序的执行效率得到明显提升，大大缩短了程序的执行时间.

4　结论与展望

随着多核处理器技术的快速发展，开发具有良好的可编程性、灵活性和可移植性的新型多核编程模型已成为提升计算机效能的重要途径. 然而，目前多核编程模型的运行时环境还不能很好地发挥多核处理器的资源优势，交互能力差，严重制约了系统的可扩展性，因此，研究多核编程模型运行时环境的动态交互性和协同性，对于提升未来多核处理器的性能有着非常重要的意义. 本文提出的自适应协同调度模型 ACSM 为解决上述问题提供了一种研究思路和借鉴，在充分体现现有多核编程模型优势的前

提下,通过增强多核编程模型的易用性、交互性及相互协同性,优化和提升了多核处理器的性能和扩展能力.

进一步的研究内容包括:①操作系统内核级支持.现有操作系统与多核编程模型运行时环境的交互能力有限,严重影响了系统资源分配的公平性和高效性,而在操作系统中引入针对不同多核编程模型的自适应协同机制,可以优化和提高系统资源分配的灵活性和高效性,降低系统性能瓶颈.②增强系统的动态优化能力,即研究多核编程模型并行负载的内部特性,在操作系统与多核运行时环境之间设计更为灵活、高效的反馈机制,提升系统的动态优化性能.

参考文献:

[1] HILL M, MARTY M. Amdahl's law in the multicore era [J]. Computer, 2008, 41(7): 33-38.

[2] 易会战,刘永鹏. 改善系统能量效率的体系结构方法:并行处理 [J]. 计算机学报, 2009, 32(12): 2475-2481.
YI Hui-Zhan, LIU Yong-Peng. An efficient architecture method for improving energy efficiency: parallel processing [J]. Chinese Journal of Computers, 2009, 32(12): 2475-2481.

[3] CHAPMAN B, HUANG Lei. Enhancing OpenMP and its implementation for programming multicore systems [M]// Parallel Computing: Architectures, Algorithms, and Applications. Amsterdam, Netherlands: IOS Press, 2008:3-18.

[4] REINDERS J. Intel threading building blocks: outfitting C++ for multi-core processor parallelism [M]. Sebastopol, CA, USA: O'Reilly Media, 2007: 133-168.

[5] FRIGO M, LEISERSON C E, RANDALL K H. The implementation of the Cilk-5 multithreaded language [C]// Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation. New York, USA: ACM, 1998: 212-223.

[6] 龙国平,张军超,范东睿. 众核体系结构对 Cilk 语言的硬件支持及评测研究[J]. 计算机学报, 2008, 31(11): 1975-1985.
LONG Guo-Ping, ZHANG Jun-Chao, FAN Dong-Rui. Architectural support and evaluation of Cilk language on many-core architectures [J]. Chinese Journal of Computers, 2008, 31(11): 1975-1985.

[7] BIENIA C, KUMAR S, SINGH J P, et al. The PARSEC benchmark suite: characterization and architectural implications [C]// Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques. New York, USA: ACM, 2008: 72-81.

[8] AGRAWAL K, LEISERSON C E, SUKHA J. Executing task graphs using Work-Stealing [C]// Proceedings of 24th IEEE International Parallel and Distributed Processing Symposium (IPDPS). Piscataway, NJ, USA: IEEE, 2010: 1-12.

[9] AGRAWAL K, HE Y, LEISERSON C E. Adaptive work stealing with parallelism feedback [C]// Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP). New York, USA: ACM, 2007: 112-120.