面向混合并行计算系统编程环境的研究与实现

武华北[1] 孙济洲[1] 王文义[2]
(天津大学计算机科学与技术学院IBM中心 天津 300072)[1]
(中原工学院并行处理技术研究所 郑州 450007)[2]

摘 要 分析了基于CMP节点的混合并行计算系统并行处理模式,基于分层次的自顶向下、逐步细化的思想,设计并实现了面向此类混合并行计算系统的编程环境,从而大大降低程了序设计人员在此类环境下编写并行应用程序的复杂度,减少了程序编码错误,提高了编程人员的效率。

关键词 多核微处理器,共享缓存,消息传递,混合并行计算系统,并行程序设计
中图法分类号 TP311 **文献标识码** A

Research and Implementation of Parallel Programming Environment for Hybrid Parallel Computing System

WU Hua-bei[1] SUN Ji-zhou[1] WANG Wen-yi[2]
(School of Computer Science and Technology, Tianjin University, Tianjin 300072, China)[1]
(Institute of Parallel Processing Technology, Zhongyuan Institute of Technology, Zhengzhou 450007, China)[2]

Abstract This paper analyzed the parallel processing pattern in the hybrid parallel system of CMP-Cluster and implemented a parallel programming environment based on hierarchy top-down approach. This system can simplify the parallel programming and improve the efficiency and correctness.

Keywords Multicore, Share cache, Massage passing, Hybrid parallel computing system, Parallel programming

1 引言

基于CMP多核微处理器的PC集群系统能够以非常低廉的硬件和软件成本,为用户提供大规模粗粒度与小规模细粒度相结合、多层次且强大而灵活的并行处理能力。然而,在得到这种并行处理能力之前,必须首先将待解决的问题进行并行化划分,然后才能设计并行算法,最终编写程序,以使其能够在特定的并行硬件环境中执行。其中,最繁琐、最容易出现错误和最耗费时间的步骤是并行编程实现并行算法,它也是阻碍并行计算技术被广泛普及应用的主要因素之一。因此,对并行编程环境和工具的研究就成为当前并行计算领域中极其重要的研究课题。

OpenMP[1]是由Sun,HP,IBM和Intel等多家计算机生产巨头和软件开发商联手推出的一种工业标准,其目的是要为软件开发人员提供一种通用的规范,使得各行各业都可以很方便地设计出新的并行应用程序或尽可能地为对其修改或并行化现有串行应用程序提供方便。

MPI(Message Passing Interface)[2]提供了一个实际可用的、可移植的、高效的和灵活的消息传递接口标准,其中不包含任何专用于某个特别的制造商、操作系统或硬件的特性。虽然MPI不是一门语言,但是它可以将FORTRAN+MPI或C+MPI看作是一种在原来串行语言基础之上扩展之后得到的并行语言。

目前OpenMP和MPI已被广泛应用,而且为并行应用程序提供了良好的可移植性。我们说这些环境本质上还只是为用户提供了并行计算机的可编程性及并行程序之间的通信和协同手段,但归根结底它们并没有真正解决如何进行通信和协同等更高一级的复杂性问题,而是把这些棘手的问题留给了进行并行程序设计和实现的人员,所以它们对于如何进一步减轻程序设计人员的负担和提高并行应用程序的编程效率[3]以及减少程序中人为的错误发生并无任何帮助。正是由于这些原因,现在人们已开始把目光投向了对更高一级的编程模型和环境的研究上,譬如目前比较典型的代表是CODE 2.0,CO2P3S和Intel TBB。

CODE 2.0是一个可视化的并行编程环境[4,5]。使用CODE2.0设计的并行程序由一组图例(Graph Instance)组成,通过调用节点(Call Nodes)进行交互。图中有向边表示进入或离开节点的数据流通道。图例类似于传统程序设计中的子程序。

CO2P3S则是一个基于设计模式的并行编程环境[6]。CO2P3S在提供给用户的编程抽象程度上,从高到低分为3层:模式层、中间代码层和原始代码层。用户在模式层选定特定的并行模式,提供与应用相关的参数来实例化模式模板,然后代码生成器会产生正确的并行框架中间代码;用户必须实

到稿日期:2009-07-15 返修日期:2009-09-10 本文受国家自然科学基金项目(10778623),天津市科技支撑重点项目(09ZCKFGX00400)资助。

武华北(1976-),男,博士生,主要研究方向为并行程序设计方法学;孙济洲(1949-),博士,教授,博士生导师;王文义(1947-),教授,博士生导师。

现应用框架中预定义的 hook 方法，才能最终生成与具体应用相关的特定语言原始代码。

Intel TBB 是 Intel 公司针对其多核微处理器环境而开发的并行应用程序开发库[7]。TBB 提供了一套函数库组件，其中包括通用并行算法、线程安全容器、同步基元、任务调度程序、内存分配和计时等功能。用户利用这些函数库组件可以避免重新编写、重新测试及重新调整通用的并行数据结构与算法，帮助并行程序开发人员从线程处理的繁琐细节中解脱出来。

上述编程环境分别只针对单独的某一类并行计算物理模型，如 CODE 2.0 针对分布式存储、CO2P3S 针对共享存储、Intel TBB 针对多核微处理器。对基于多核微处理器的 PC 集群这种混合并行体系结构，目前还没有比较成熟的编程环境和工具支撑，这也正是本文所要研究的内容。

2 HPCS 并行处理模式

HPCS(Hybrid Parallel Computing System，混合并行计算系统)是指在一个并行计算系统中涉及到多种并行硬件系统和多种并行处理模式。本文研究的是 CMP-Cluster 模式，即基于 CMP(Chip Multiple Processors)节点的集群(Cluster)系统。

集群系统各计算节点之间是通过网络互相连接在一起的，各节点之间通过消息传递进行交互和同步。通过网络进行信息交换的时间 Ts，包括数据打包时间 Tp、网卡传输时间 Tc 和网络传播时间 Tt，可简单表示为

$$Ts=Tp+c+Tt$$

其中，Tp 表示系统把数据打包，形成协议负载包所花费的时间，即在用户原始数据基础上添加了网络传输协议数据之后所形成的、能够在网络中传输的数据包；Tc 为网卡传输时间，即网卡把数据包的每一位数据转换成电信号而发送到网络上所花费的时间；Tt 表示传播时间，即数据包通过网络传播到达接收端所花费的时间。其中，最耗时的部分是数据打包时间 Tp，因为每启动一次数据传输，系统都要调用一些费时的系统调用，以实现对数据进行协议封装。

CMP 系统可以被视为是传统 SMP 系统的一种特例，但二者在体系结构上却又不尽相同。传统 SMP 系统中的每个处理器都是一个单一的处理核心，该处理核心独占处理器所有的计算资源，包括所有的一级、二级缓存。而在 CMP 系统中，由于多个处理核心被整合到一个芯片内部，多个处理核需要共享二级缓存，而且共享缓存的数量也随着不同芯片而变化，如图 1 所示。在这种情况下，如果仍然按照传统的 SMP 方式进行程序设计，由于线程 cache 缺失率的提高，会影响系统的计算性能。例如，传统的 SMP 处理模式一般是把整个问题域的数据集 n 按照处理核心的数量 p 平均地分割为 p 等份，把整个二级缓存的空间 m 平均分配给 p 个处理核心，那么每个处理核心在被分配的缓存空间 m/p 中处理自己的 n/p 个数据集。在这种情况下，如果算法的空间局域性不好，就会造成比较严重的 cache 缺失。即使这时的 cache 缺失率不高，每个线程也要在处理完自己 m/p 个数据以后，竞争共享的前端总线，与主存进行频繁的数据交换，这样势必会造成系统性能上的下降。

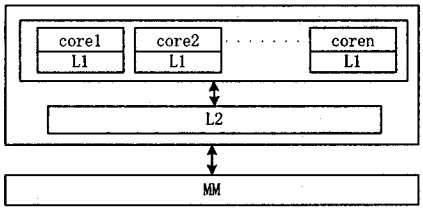

图 1 CMP 体系结构

一种处理模式是以共享的 cache 大小为基础进行数据划分，即把 n 个数据划分成 n/m 个大小为 m 的数据块；然后所有 p 个处理核心并行地对装入 cache 中的 m 个数据进行处理；最后需要同步所有处理核心，以保证全部的 m 个数据都被处理完成，然后再进行 cache 和主存之间的整个 m 个数据交换。这样既可以提高共享 cache 的利用率[3]和命中率，又可以提高系统前端总线的利用率。

通过以上的分析可知，集群系统各节点之间不太适合频繁地发送小的琐碎数据，而应该在某些特定时刻发送较大规模的数据包，这样可以有效减少数据打包的时间损耗。此外，还应该尽可能让通信操作和计算操作能够重叠进行。CMP 系统更适合在局部范围内本地数据集上进行细粒度的数据并行操作，尽量减少并发任务(线程)间的交互，以提高系统并行效率。

HPCS 系统处理模式可以分为 4 步：任务分配、数据分配与交换、本地加速计算和结果归约。

(1) 任务分配：通过对问题域的分析，进行任务分解，把整个任务分解成可以相对独立并行运行的子任务，确定各子任务之间的数据依赖关系，并自动进行任务的分配及节点映射。

(2) 数据分配与交换：根据各子任务的需求，对应用数据进行划分，通过消息传递的方式把数据传输到各子任务运行的节点上，并在需要的时候进行节点间数据交换。

(3) 本地加速计算：系统中每个节点获得自己的任务和数据以后，在节点内部进行多线程的数据并行运算。节点中的每个处理核心可以运行相同的指令，也可以分别运行不同的指令，处理不同线程的任务。

(4) 结果归约：各节点完成运算后，把自己的局部计算结果发送给相应的节点，由该节点对所有局部结果进行归约操作，得到整个任务的最终结果。

本文针对 HPCS 系统的并行处理模式，设计并实现了 PBHPCS(Pattern-based Hybrid Parallel Computing System)并行编程环境。

3 PBHPCS 并行编程环境

PBHPCS 采用分层结构对编程过程进行建模，如图 2 所示，包括任务分配模式层、任务计算模式层、设计(通信)模式层、框架代码层和任务映射调度模式层。

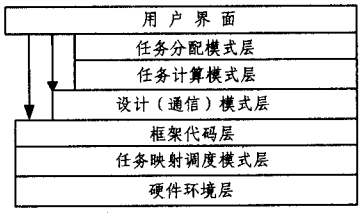

图 2 PBHPCS 并行编程环境逻辑层次

3.1 任务分配模式层

在任务分配模式层对整个应用任务的划分和分配进行建模。建立应用系统的任务分配与交互模式模型图，用来表示整个系统的进程结构和交互方式，给用户一个系统的最高层次模型图。该模型如图 3 所示。

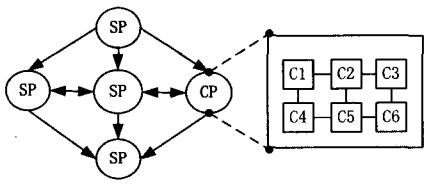

图 3 任务分配模式图

图 3 中每一个节点表示一个可以独立运行的任务(进程)，节点之间的连接线表示进程间的通信方式和数据交互，箭头表示数据传递的方向。用户可以利用这样的任务分配模式图进行任意的任务划分和数据分配，指定整个系统的通信模式。每个节点可以是简单的计算模式 SP(Single Pattern)，也可以是复杂的组合计算模式 CP(Combinatorial Pattern)。简单计算模式代表单个进程，进程内部不再涉及更复杂的并行模式，其中只有简单的通信语句；组合计算模式在单个进程内部又涉及到多个线程并行的情形，而且多个线程之间的数据通信具有特定的模式。

3.2 任务计算模式层

任务计算模式层是对任务分配模式图中每个计算任务节点具体的实现，对其中单个任务节点进行多线程并行化程序建模。这里侧重于对更细粒度的多线程数据并行的模式建模。在这一层，我们在建模的参数中引入了共享同一级 cache 的处理内核数和被共享 cache 大小的参数，让用户根据具体应用的特点，选择用传统的 SMP 处理方式还是用 CMP 的 cache 可见处理方式。通过可视化用户建模界面来建立的任务分配模式图和任务计算模式图都被保存为 xml 类型的文件。

3.3 设计(通信)模式层

通过对典型的并行问题解决和通信方式的建模，我们预定义实现了一些常用的并行设计模式框架模板，其中包括数据并行模式(一维网格、二维网格、三维网格)、流水线模式、主从模式、分治模式、分治界定模式和动态规划模式等[8]。这样就给用户提供了一个充足的选择空间：如果用户问题的模式匹配了上述模式之一，那么就不需要用户自己再花力气去建立进程模式模型图，只需直接选定相应的模式框架模板，对其进行参数化配置即可。之后，系统就会根据参数实例化一个具体模板，最终生成并行框架代码。

3.4 框架代码层

系统的框架代码生成器解析任务分配模式图和任务计算模式图，自动生成整个应用的框架代码。这些框架代码已经包含所有与并行处理细节相关的语句，如消息传递、进程同步、线程创建和管理等，用户只要在对应位置添加具体的串行代码，就可以实现整个应用功能。用户也可以屏蔽掉本层上面的 3 层，而在框架代码层直接从底层编写自己的应用代码，完全可以自己控制应用实现的模式，而这恰恰是普通用户所希望的。

3.5 任务映射调度模式层

任务映射调度模式层是把用户建立的逻辑任务分配模式图根据特定的策略映射到特定的硬件节点上。选定一个主节点，在该节点上维护一张系统任务分配模式图，并建立任务节点池。当任务节点池中有空闲节点时，主节点就会根据任务分配图中各节点的依赖关系和优先级，扫描系统任务分配模式图，找到可以运行的任务调度到空闲节点上运行，并动态更新系统任务分配模式图的拓扑，直到系统任务分配模式图变成空图为止。

3.6 系统实现

针对 PBHPCS 编程模式，我们开发了一套并行应用辅助开发平台 EasyPAB[9]，如图 4 所示。该系统基于可视化建模工具、设计模式、结构骨架等方法为用户简化并行程序设计提供了帮助。图 5 至图 7 则分别对应于系统建模到框架代码生成的过程。在开发过程中，基于阵列式数据并行应用原理，提出了一种新的数据并行设计模式 FJRR，其在实际应用和实验比对中获得了良好的效果。

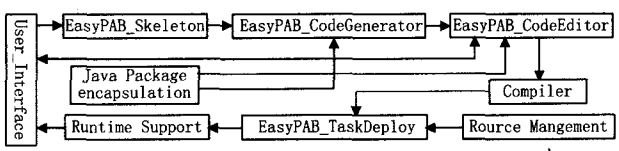

图 4 EasyPAB 体系结构图

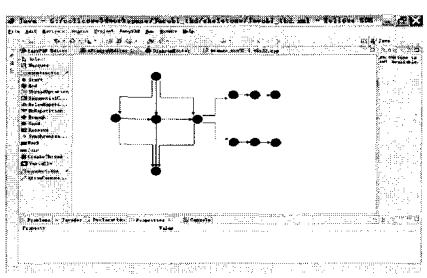

图 5 任务分配建模图

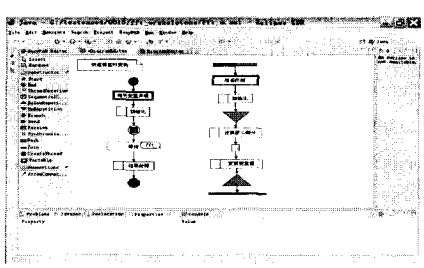

图 6 任务计算建模图

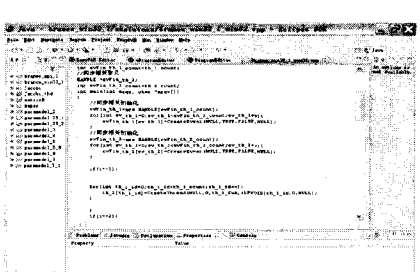

图 7 自动生成的框架代码

结束语 本文提出了一种 PBHPCS 并行编程环境，它通过使用层次化、可视化建模和预定义模式代码自动生成的手段来简化 HPCS 环境的并行程序设计，可以明显降低程序设计人员在此类环境下编写并行应用程序的复杂度，减少程序

(下转第 178 页)

声数值依次设定为 $Q=10^{-4},10^{-3},10^{-2},10^{-1},10^{0},10^{1},10^{2},10^{3},10^{4}$。对比 SFFN,FFSN,CCF 和 OF 4 种融合算法在不同过程噪声水平下的均方根误差。当给定一个过程噪声参数值时,其仿真结果的均方根误差定义与式(10)相同。

实验结果如图 4(a)和图 4(b)所示,当过程噪声范围设定为 $Q<10^{4}$ 时,本文算法融合性能优于 CCF 和 OF,当过程噪声水平高于 10^{4} 时,本文算法性能开始退化。也就是说,当过程噪声数量级别和各个局部传感器量测噪声级别相差 1000 倍以上时,新算法性能开始退化。但是此时过程噪声相对于量测噪声来说,其取值级别已经不再具有应用价值。

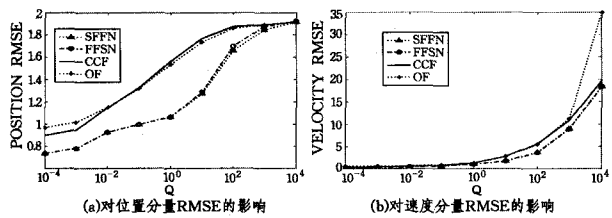

图 4 本文算法对过程噪声 Q 敏感程度分析

结束语 在航迹融合过程中,采用分段 RTS 平滑过程的凸组合航迹融合算法能够明显地改善融合性能,且能够兼顾融合系统的实时性。本文的贡献在于:提出了分段 RTS 算法,并将其应用在航迹融合过程中;针对局部节点有无额外计算能力,提出了两种基于分段 RTS 平滑的凸组合航迹融合方法来应对不同的情况,从而改善了融合效果。

本文采用的 RTS 算法是经典的基于 Kalman 滤波条件下的平滑算法。实际上,新的滤波算法和相应的 RTS 平滑算法也具有相同特征。这些平滑算法包括 UKF RTS 平滑算法以及粒子滤波 RTS 平滑算法等。对新型滤波平滑算法在航迹融合过程中的应用值得进一步研究。

参 考 文 献

[1] 韩崇昭,朱洪艳,段战胜. 多源信息融合[M]. 北京:清华大学出版社,2006

[2] Chong C Y, Mori S, Chang K C, et al. Architectures and algorithms for track association and fusion [J]. IEEE Transactions on Aerospace and Electronic Systems, 2000, 15(1): 5-13

[3] Chang K C, Saha R K, Bar-Shalom Y. On optimal track-to-track algorithm [J]. IEEE Transactions on Aerospace and Electronic Systems, 1997, 33(4): 1271-1276

[4] Chen Xiaohui, Zhang Yang. A fuzzy fusion in multitarget tracking with multisensor [C]// International Conference on Computational Intelligence and Security. Harbin, China, 2007: 152-156

[5] Tafti A D, Sadati N. Novel adaptive Kalman filtering and fuzzy track fusion approach for real time applications [C]// 3rd IEEE Conference on Industrial Electronics and Applications. Singapore, 2008: 120-125

[6] 潘泉,张磊,崔培玲,等. 动态多尺度系统估计理论与应用[M]. 北京:科学出版社,2007

[7] 杨艳娟,金志华,田蔚风,等. RTS 平滑算法在捷联惯性导航系统初始对准精度事后评估中的应用 [J]. 上海交通大学学报,2004,38(10):1743-1747

[8] Chou K C, Willsky A S, Benveniste A. Multiscale recursive estimation, data fusion, and regularization [J]. IEEE Transactions on Automatic Control, 1994, 39(3): 464-478

[9] Sarkka S. Unscented Rauch-Tung-Striebel smoother [J]. IEEE Transactions on Automatic Control, 2008, 53(3): 845-849

[10] Rauch H E, Tung F, Striebel C T. Maximum likelihood estimates of linear dynamic systems [J]. Journal of American Institute of Aeronautics and Astronautics, 1965, 3(8): 1445-1450

[11] Godsill S J, Doucet A, West M. Monte Carlo smoothing for nonlinear time series [J]. Journal of the American Statistical Association, 2004, 99(465): 156-168

[12] Klaas M, Briers M, de Freitas N, et al. Fast particle smoothing: if I had a million particles [C]// Proceedings of ICML Pittsburgh. PA, 2006: 481-488

[13] Bar-Shalom Y, Li X R, Kirubarajan T. Estimation with Applications to Tracking and Navigation [M]. New York: Wiley-Interscience, 2001

(上接第 145 页)

编码错误,提高编程人员的效率。

当然,我们只是做了一些初步的工作,该模型在许多方面还有待于进一步完善。首先,对集群节点间的任务划分和分配模式尚需要进一步研究,以便实现让任务的划分自动化和动态化,从而最大程度地减轻程序设计人员的负担,并使系统达到动态负载平衡;其次,更多的设计模式也需要被逐步地发掘,同时需要对它们做进一步的调试和优化,以使现有的设计模式模板库更加丰富,让用户具有更大的选择空间和获得更好的系统性能。

参 考 文 献

[1] Dagum L, Menon R. OpenMP: an industry standard API for shared-memory programming [J]. IEEE Computational Science & Engineering, 1998, 5(1): 46-55

[2] Gropp W, Lusk E, Skjellum A. Using MPI: portable parallel programming with the message-passing interface[M]. 2rd ed. Cambridge, Mass: MIT Press, 1999

[3] 王文义,董绍静. 基于并行程序效率和通用性的实践与研究[J]. 计算机科学,2009,36(6):290-293

[4] Newton P, Browne J C. The CODE 2.0 graphical parallel programming language[C]// Proceedings of the 6th ACM International Conference on Supercomputing. New York: ACM, 1992: 167-177

[5] Szafron D, Schaeffer J. An experiment to measure the usability of parallel programming systems[J]. Concurrency: Practice and Experience, 1996, 8(2): 146-166

[6] Macdonald S, Anvik J, Bromling S, et al. From patterns to frameworks to parallel programs[J]. Parallel Computing, 2002, 28(12): 1663-1683

[7] Reinders J. Intel Threading Building Blocks [M]. Sebastopol CA: O' Reilly, 2007

[8] Wu Huabei. Design-pattern based parallel programming model and system implementation[C]// Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing. Piscataway: IEEE, 2008, 11(3): 1-5

[9] Yu Ce, Sun Jizhou, Wu Huabei, et al. EasyPAB: An Extensible IDE Framework for Parallel Applications[C]// Proceedings of Advanced Parallel Process Technologies. LNCS4847 Germany Heidelberg: Springer, 2007: 666-673