MPI + TBB 混合并行编程模型在分子动力学中的应用*

白明泽[1,2], 赵文辉[2], 豆育升[2,3], 孙世新[1], 温 迪[2]
(1. 电子科技大学 计算机科学与工程学院, 成都 611731; 2. 重庆邮电大学 高性能计算与应用研究所,重庆 400065; 3. Department of Physical Sciences, Nicholls State University, LA 70310, USA)

摘 要: 为了提高分子动力学模拟在对称多处理(SMP)集群上的计算速度,在分子动力学并行方法中引入 MPI + TBB 的混合并行编程模型。基于该模型,在分子动力学软件 LAMMPS 中设计并实现混合并行算法,在节点间采用 MPI 及空间分解技术实施进程级并行,节点内采用 TBB 及临界区技术实施线程级并行。在 SMP 集群中的测试表明,该方法在体系较大以及节点数较多时可以明显减少通信时间,使加速比在纯 MPI 模型上提高 45%。结果表明,MPI + TBB 混合并行编程模型可促进分子动力学并行模拟且效率明显提升。

关键词: 分子动力学;混合并行编程模型;SMP 集群; MPI; 线程构建块
中图分类号: TP399 **文献标志码**: A **文章编号**: 1001-3695(2012)05-1772-03
doi:10.3969/j.issn.1001-3695.2012.05.045

Application of hybrid MPI + TBB parallel programming model for molecular dynamics simulation

BAI Ming-ze[1,2], ZHAO Wen-hui[2], DOU Yu-sheng[2,3], SUN Shi-xin[1], WEN Di[2]
(1. *School of Computer Science & Engineering, University of Electronic Science & Technology of China, Chengdu* 611731, *China*; 2. *Institute of High Performance Computing & Application, Chongqing University of Posts & Telecommunications, Chongqing* 400065, *China*; 3. *Dept. of Physical Sciences, Nicholls State University, LA* 70310, *USA*)

Abstract: This paper introduced an MPI + TBB hybrid parallel programming model in parallelization of molecular dynamics simulation on SMP clusters. Designed the hybrid algorithm and applied on a molecular dynamics simulation program named LAMMPS. MPI and spatial decomposition method were employed to achieve process-level parallelism, while TBB and critical section method were adopted to achieve thread-level parallelism. The performance tests indicate that the speedup of hybrid model is 45% over that of the MPI model when the simulation system and number of node are relatively large. The performance boost comes from that the hybrid model can significantly reduces the communication time. Results prove that the hybrid parallel algorithm can be feasible and efficient for molecular dynamics simulations.

Key words: molecular dynamics (MD); hybrid parallel programming model; symmetrical multi-processing (SMP) cluster; message passing interface (MPI); threading building blocks (TBB)

0 引言

随着高性能计算机的飞速发展,基于多核处理器(chip multi-processor,CMP)的 SMP 集群逐渐成为高性能计算领域的主流[1]。其节点配备的计算核心越来越多,而普遍应用的基于消息传递的 MPI 并行编程模型却将共享存储的多个计算核心简化为相互通信的节点,掩盖了多核计算集群的特点。为避免因简化而带来的性能损失,一种混合并行编程模型(以下简称混合模型)近年来被提出[2,3],它在节点间采用 MPI 等消息传递机制进行通信,在节点内部采用基于共享存储模式进行并行算法设计。该模型在任务切分时增大了子任务的粒度,从而有效降低了在集群节点间的通信量;共享存储并行模式下可直接读写变量,其速度比消息传递模型更快,从而可能加快线程级并行计算速度。研究结果表明,混合模型可以提高并行计算性能[2,4]。

共享存储模型主要有 TBB[5]、OpenMP[6]、Pthreads[7] 等。其中,Pthreads 需要相关人员掌握操作系统等底层知识;OpenMP 相对较简单但需要编译器的专门支持。TBB 是由 Intel 公司开发的一套 C ++ 模板库,只要操作系统中有 C ++ 编译器,就能编译运行 TBB 程序。同时 TBB 抽象程度较高,相关人员不需要掌握太多的并行计算知识就可以较快地实现 C ++ 串行程序的并行化。

分子动力学(MD)是高性能计算领域的重要应用之一。MD 是一套应用广泛的分子模拟方法,可用来研究物质的微观动力学过程[8]。MD 模拟可以帮助人们获取在实验中很难得到的原子尺度动力学信息,因而被广泛应用于多种科学研究领

收稿日期: 2011-10-26; **修回日期**: 2011-11-30 **基金项目**: 国家自然科学基金资助项目(21073242)

作者简介: 白明泽(1982-),男,重庆酉阳人,讲师,博士研究生,主要研究方向为并行算法设计、分子动力学模拟(baimz@cqupt.edu.cn);赵文辉(1985-),男,山西忻州人,硕士研究生,主要研究方向为并行算法设计;豆育升(1953-),男,陕西礼泉人,教授,博导,主要研究方向为计算化学、高性能计算;孙世新(1940-),男, 湖北孝感人,教授,博导,主要研究方向为并行与分布式计算.

域[8,9]。MD 模拟体系通常由大量的(有时多达数以百万计)原子组成,并且计算的时间步数较多,因此 MD 模拟计算需要耗费大量的计算资源,加速 MD 模拟的方法主要靠多台计算机并行完成计算任务的计算。

在并行 MD 领域应用较广泛的就是基于消息传递的 MPI 模型[10],许多著名的 MD 软件如 LAMMPS[11]、NAMD[12]多是基于 MPI 的,可在 MPICH2[13]、Intel MPI[14]等 MPI 标准的实现支持下并行运行。用于 MD 的混合模型主要是 MPI + OpenMP 混合模型[15],但 OpenMP 需要专门的编译器支持。本文提出将 MPI + TBB 的混合模型应用到 MD 模拟中以提高 MD 模拟的并行性能,并以应用广泛的 MD 程序 LAMMPS 为例,设计并实现了混合模型在 MD 程序中的并行方法。测试结果表明,MPI + TBB 的混合模型在节点数较多时可以明显提高 LAMMPS 的并行性能。

1　分子动力学模拟算法与程序

1.1　分子动力学模拟算法

采用近邻表方法的 MD 模拟一般遵循如下步骤:a)给定系统初始位形后开始 N 步循环;b)循环体中每隔一定时间步数更新一次粒子的近邻表;c)每一个时间步里根据近邻表计算粒子之间的作用力,以及积分牛顿方程获得粒子新坐标。在整个模拟过程中,粒子的受力计算最费时,近邻表的构建也比较费时,这两部分往往成为并行化的主要目标。

1.2　LAMMPS 模拟软件

LAMMPS 是一个应用广泛的 MD 软件,它遵循开源协议且预留有添加新功能模块的接口,可定制性强;同时它采用空间分解并行算法,并行性能较好[16];此外,它还支持多数常见的势能函数,能胜任不同的模拟需求。LAMMPS 从 2005 年停止发布 Fortran 版本而只发布 C ++ 版本,C ++ 语言的面向对象特性使得针对修改软件的开发管理和添加新功能都更容易。

LAMMPS 软件的核心类如图 1 所示,该图只是示意性地描述了与一个特例(LJcut 势能,对应命令“pair_style lj/cut”)有关的核心类,其他势能类所处位置与特例类似。当系统启动 LAMMPS 程序过程中,首先创建一个 LAMMPS 对象,该对象随即创建如图 1 所示第二行所列的一系列类的实例,这些对象将担负起 MD 模拟的主要计算任务。

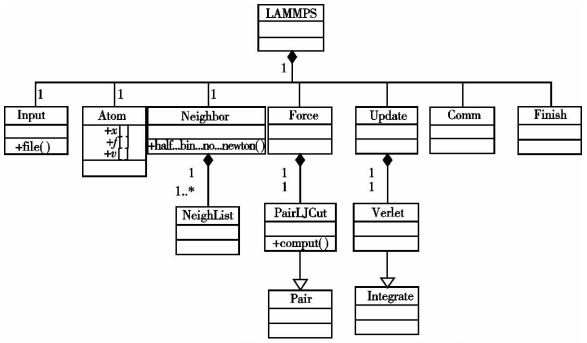

图1　以LJcut势能为例的LAMMPS核心类图

LAMMPS 中的 LJcut 模拟计算主要由图 1 中所列类的实例对象实施,其中的程序流程如图 2 所示。该图示意性地描述了各核心对象执行一个简单 MD 模拟的计算流程。

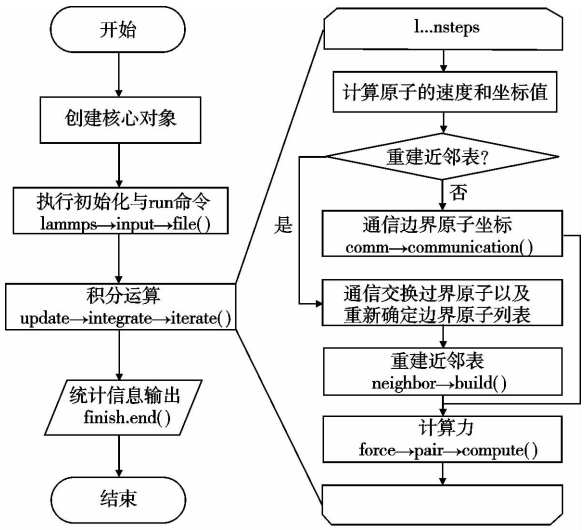

图2　LAMMPS执行LJcut模拟计算的核心程序流程

2　基于 MPI + TBB 的混合模型

2.1　TBB 并行编程模型

TBB 是 2007 年 Intel 公司发布的适用于多核计算平台的 C ++ 的共享内存并行编程模,它是一套 C ++ 模板库并且采用“任务”的形式实现并行,“任务”是比线程的并行抽象级别更高的并行基本单位。相比其他共享内存并行模型,TBB 具备如下优点:

a)适用平台广。TBB 采用标准的 C ++ 语法和模板库,可广泛应用于配备 C ++ 编译器的各种平台,而不像 OpenMP 那样需要编译器的专门支持。

b)并行设计简单。TBB 是面向任务编程的,它需要设计可以并行执行的各种任务,而不需要关注线程细节,如负载均衡、优化调度等。

c)可有效利用计算资源。TBB 可自动根据平台处理器情况通过任务密取技术实现负载均衡。

TBB 常见的一项应用是借助模板函数 TBB:parallel_for 将一个普通 for 循环进行并行化。TBB 通过该函数能负载均衡地并行执行固定数量的独立循环迭代体。具体实现方法是将范围从 $0 \sim n-1$ 的迭代区间分成许多小空间(chunk),并为每个空间创建一个独立的工作任务;之后通过任务调度器分配这些任务到具体的线程上进行计算。实现 for 循环并行化的第一步是将该循环体转换为在小空间上进行操作的形式。在 TBB 中,由体对象(body object)代表该小空间。体对象是一种标准模板库风格的函数对象,其中包含一个 operator( )函数,并由该函数负责处理对应的迭代小空间。

2.2　MPI + TBB 混合模型

Balaji 等人[17]提出了基于 MPI + TBB 的混合模型。但具体的应用实例还较少见,将 MPI + TBB 混合模型应用到 MD 领域则未见文献报道。图 3 演示了基于 MPI + TBB 的混合模型计算框架,该图仅示意性地表示了两个节点时的情况。图中每个虚线框代表运行在节点上的一个进程,框内流程即进程的计算流程。在进程中进行 TBB 并行的步骤如下:首先执行 TBB:task_scheduler 初始化任务调度器,然后由 TBB 隐式地初始化线程簇,并由调度器将任务分配到各个线程进行计算,各线程计算完毕后由 terminate( )处理善后。

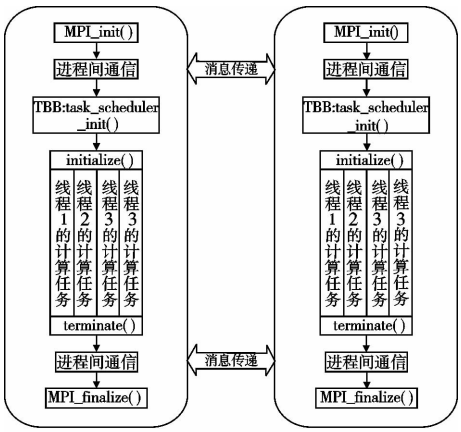

图3　MPI+TBB混合模型图

2.3　MD 混合并行算法设计

LAMMPS 是一种应用广泛的采用 MPI 进行消息传递的 MD 程序,本文以它为例设计并实现基于 MPI + TBB 的 MD 混合并行算法。由于 MPI 已经配置,本文工作主要集中于如何在 LAMMPS 程序中增加 TBB 并行上。本文选取典型的势能函数——带截断半径的 LJ 势能(LJcut)进行 MPI + TBB 的混合并行算法设计,其他势能函数可相应地作类似的设计与实现。

LJcut 势能函数在 LAMMPS 中由 Pairljcut 类的成员函数 compute( )进行力计算,由 Neighbor 类中的 half_bin_no_newton( )构建该势能函数的近邻表。根据 2.1 节所述的 TBB 并行化方法可进一步设计适用于 LAMMPS 的 MPI + TBB 实现方法:首先在 Pairljcut 类和 Neighbor 类中分别添加体对象 Opfor_ljcut 和 Opfor_nb。体对象 Opfor_ljcut 中需设置的构造函数和 operator( )函数内容如下:构造函数主要负责获取 compute( )函数传递过来的变量;operator( )函数负责实现新构造的迭代小空间计算内容,此迭代小空间的循环控制条件由原(i = 1 … nlocal)转变为(i = rr. begin( ) … rr. end( )),循环体内计算逻辑与原 compute( ) 函数中的核心计算部分相同,原 compute( )函数只负责调用 TTB 任务。Neighbor 类以及体对象 Opfor_nb 作类似的改动。

由于多线程之间进行共享变量的写操作时可能会遇到数据竞争,可采用一种被称为临界区的方法[18]以避免数据竞争。因此,在 Opfor_ljcut 对象的 operator( )函数里需要增加一个 ftmp[ ]数组以存储临时的力,并且在函数处理完迭代小空间计算内容后,设置一个临界区并在其中将各线程的 ftmp[ ]数组汇总到 f[ ]数组。为了减少此临界区的计算时间,在 compute( )函数调用 TBB 时将粒度也就是迭代空间的大小设置到最大,使得一个线程只接收一次迭代空间,实现一个线程只处理一个迭代空间的目的。原 compute( )函数中一个需读写共享变量的、用来更新压力、能量等值的函数 ev_tally( )因为可能遇到数据竞争被注释掉。为正确评价混合模型的并行效率,作为参考的串行和 MPI 并行的 LAMMPS 代码也作同样的改动。

3　测试与分析

为验证混合并行算法的正确性和有效性,本文在一个 SMP 集群架构的刀片中心上进行了测试。刀片中心型号为 IBM BladeCenter H,配置 14 个刀片服务器作为计算节点,每个节点配备一颗至强 E5405 (2.0 GHz,四核)和 4 GB 内存。操作系统为 Red Hat Enterprise 5.3,内核版本 2.6.28,gcc 版本 4.1.2,MPI 环境为 MPICH2 1.1 版。

本文选取 N = 10 976(14 × 14 × 14)、108 000(30 × 30 × 30)、500 000(50 × 50 × 50)三个体系(N 为原子数,括号内为晶格数)进行测试,以分析混合模型在各种情况下的并行性能。所有体系的计算时间步数为 500 步,其余参数设置同文献[11]。

MPI + TBB 混合模型与原有的 MPI 模型的运算时间如表 1 所示。

表 1　MPI + TBB 模型与 MPI 模型的 LAMMPS 模拟时间

节点 × 核心数	N = 10 976		N = 108 000		N = 500 000	
	MPI + TBB/s	MPI/s	MPI + TBB/s	MPI/s	MPI + TBB/s	MPI/s
1 × 4	3.21	2.65	38.62	26.32	177.5	132.24
2 × 4	2.38	1.73	20.97	13.80	95.56	69.32
4 × 4	1.67	1.18	12.04	7.97	55.18	40.04
8 × 4	1.73	0.84	6.40	4.61	28.78	28.23
14 × 4	1.35	0.89	5.14	4.46	18.45	26.71

横向地看,当节点数为 1 ~ 8 时,MPI + TBB 版所需的时间均比 MPI 版所需的时间多,主要是因为 TBB 并行化时采用临界区的方法,较大地影响了并行性能[17],且在节点数较小时仍然受到 TBB 开销的牵制;当节点数增加到 14 时,出现了两种不同的情况:a)当 N≤108 000 时,MPI + TBB 版的计算时间比 MPI 版的计算时间长;b)当 N = 500 000 时,MPI + TBB 版的计算时间为 18.45 s,MPI 版的计算时间为 26.71 s,混合模型明显优于 MPI 模型。

纵向地看,当节点数从 8 增加到 14 时可以看到 MPI 模型的模拟时间并没有明显减少。为分析加速比不增长的原因,获取了两种模型的通信时间占总时间比例(N = 500 000),如图 4 所示。从图 4 中可看出,随着节点数的增加,MPI 模型的通信时间占比快速增加;当节点数为 14 时,通信时间占比竟达到约 70%。与之相对应的是,混合版本的通信时间比例虽然也随着节点数而增加,在 14 个节点时却只有约 28%,远小于 MPI 版本的时间占比。

为了能更直接地比较两个模型的并行性能,计算了两种模型的加速比,如图 5 所示。从图 5 中可看出,在计算核心总数从 32(8 节点 ×4 核)增加到 56(14 节点 ×4 核)时,加速比只有微弱的增长,这是因为通信时间占比过高导致的。反观混合版本的加速比曲线,由于通信量的相对大幅减少,它体现出良好的可扩展性。在计算核心总数为 56 时,混合版本的加速比为 22.6,较 MPI 版本的 15.6 多 45%,从而证明混合模型可以有效提高 MD 模拟在 SMP 集群上的计算效率。

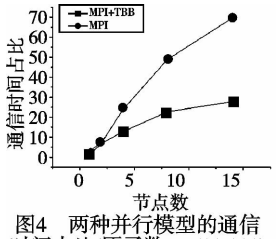

图4　两种并行模型的通信时间占比(原子数N=500 000)

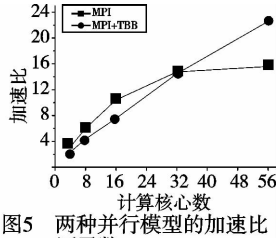

图5　两种并行模型的加速比(原子数N=500 000)

4　结束语

基于 MPI + TBB 的混合并行编程模型,以 LAMMPS 为例设计并实现了 MD 的混合并行算法。实验测试结果表明,应用混合并行技术的 MD 模拟程序在体系较大且节　(下转第 1794 页)

(上接第 1774 页)点数较多时可以明显提高并行性能。分析表明,节点数增加会导致 MPI 模型的通信时间剧增,混合模型则可以有效地减少通信时间,使 MD 模拟程序具备良好的可扩展性。本文是在 SMP 集群上进行的一次 MPI + TBB 混合并行的应用,混合模型的良好性能表现表明它对于提高 MD 并行计算性能的研究工作具有较好的促进作用。

参考文献:

[1] Top 500 supercomputer sites. Architecture share for 11/2010[EB/OL]. [2011-11-20]. http://www.top500.org/stats/list/36/archtype.

[2] 李鸿健,白明泽,唐红,等. 混合并行技术在激光化学反应模拟中的应用[J]. 计算机应用,2010,30(6):1687-1689.

[3] 李鸿健,唐红,豆育升,等. 基于 SMP 集群的激光化学反应模拟效率分析[J]. 计算机应用研究,2011,28(4):1233-1241.

[4] HIPP M, ROSENTIEL W. Parallel hybrid particle simulations using MPI and OpenMP[C]//Lecture Notes in Computer Science, vol 3149. Berlin:Springer-Verlag,2004:189-197.

[5] REINDERS J. Intel threading building blocks 编程指南[M]. 聂雪军,译. 北京: 机械工业出版社, 2009:20-24.

[6] CHAPMAN B M, MASSAIOLI F. OpenMP[J]. Parallel Computing,2005,31(10-12):957-959.

[7] NICHOLS B, BUTTLAR D, FARRELL J P. Pthreads programming[M]. Sebastopol:O'Reilly & Associates,Inc,1996:6-7.

[8] ALLEN M P, TILDESLEY D J. Computer simulation of liquids[M]. New York: Oxford Clarendon Press,1991:2-3.

[9] RAPAPORT D C. The art of molecular dynamics simulation[M]. Cambridge: Cambridge University Press,2004:7-8.

[10] Message Passing Interface Forum. MPI: a message-passing interface standard, version 2.2[EB/OL]. [2011-11-20]. http://mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf.

[11] PLIMPTON S. Fast parallel algorithms for short-range molecular dynamics[J]. Journal of Computational Physics,1995,117(1):1-19.

[12] KALE L, SKEEL R, BHANDARKAR M, *et al*. NAMD2: greater scalability for parallel molecular dynamics[J]. Journal of Computational Physics,1999,151(1):283-312.

[13] Argonne National Laboratory. MPICH2[EB/OL]. [2011-11-20]. http://www.mcs.anl.gov/mpi/mpich2.

[14] Intel. Intel MPI library[EB/OL]. [2011-11-20]. http://software.intel.com/enus/articles/intel-mpi-library.

[15] MARTIN J C, DAVID W W, MARTYN F G. Hybrid message-passing and shared-memory programming in a molecular dynamics application on multicourse clusters[J]. International Journal of High Performance Computing Applications,2009,23(3):196-211.

[16] ALAM S R, AGARWAL P K. On the path to enable multi-scale biomolecular simulations on petaFLOPS supercomputer with multi-core processors[C]//Proc of IEEE International Parallel and Distributed Processing Symposium. 2007:1-8.

[17] BALAJI P, BUNTINAS D, GOODELL D, *et al*. Fine-grained multithreading support for hybrid threaded MPI programming[J]. International Journal of High Performance Computing Applications, 2010,24(1):49-57.

[18] 白明泽,程丽,豆育升,等. 基于 OpenMP 的 MD 并行算法性能分析与优化[J]. 计算机应用,2012,32(1):163-167.