非定常 Monte Carlo 输运问题的并行算法

刘 杰[1] 邓 力[2] 胡庆丰[1] 袁国兴[2] 李晓梅[3]

[1](国防科学技术大学计算机学院 长沙 410073)

[2](北京应用物理与计算数学研究所 北京 100088)

[3](指挥技术学院 北京 101416)

摘 要 文中给出了非定常 Monte Carlo(下文简写为 MC)输运问题的并行算法,对并行程序的加载运行模式进行了讨论和优化设计.针对 MC 并行计算设计了一种理想情况下无通信的并行随机数发生器算法.动态 MC 输运问题有大量的 I/O 操作,特别是读取剩余粒子数据文件需要大量的 I/O 时间,文中针对 I/O 问题,提出了三种并行I/O 算法.最后给出了并行算法的性能测试结果,对比串行计算时间,使用 64 台处理机时的并行计算时间缩短了 30 倍.

关键词 动态 MC 输运;并行算法;并行随机数发生器;并行 I/O 算法;MPI

中图法分类号 TP301

Parallel Algorithm of Time-dependent Monte Carlo Transport

LIU Jie[1] DENG Li[2] HU Qing-Feng[1] YUAN Guo-Xing[2] LI Xiao-Mei[3]

[1](*School of Computer*, *National University of Defense Technology*, *Changsha* 410073)

[2](*Institute of Applied Physics and Computation Mathematics*, *Beijing* 100088)

[3](*Institute of Command Technology*, *Beijing* 101416)

Abstract This paper is concerned with parallel computing code for time-dependent MC transport. Two parallel algorithms is given and several loading problem of parallel program is explord. In order to reduce the time of entering and exiting parallel computing environment, the old code is modified to use MC computing code as a subroutine block. A parallel random number generator capable of MC parallel computing is proposed. Then the parallel I/O problems are discussed. The code discussed here is significantly different from the general particle transport MC simulations in the side of I/O requirements because it is designed to deal with the variational source problems. At each calculation step, it will randomly sample from a source file, and randomly write to another source file. These I/O operations consist of accesses to a large number of small, noncontiguous pieces of data. Moreover, the sizes of the source files and the amount of computations vary dynamically in each calculation step. So the I/O performance degrades drastically. To avoid this disadvantage, three parallel I/O algorithms are given. The first parallel I/O algorithm is direct parallelized from sequence code. In the second algorithm, each processor first writes the data to local memory, parallelly writes the local data to a file, then all processors parallelly read the file to local memory, and last all the processors can be able to access the small, noncontiguous pieces of data from local memory. In third algorithm, all the data are accessed in memory and the good performance of I/O is given. The experiments are performed on a 64-processor parallel machine

收稿日期:2002-09-11;修改稿收到日期:2003-07-18.本课题得到计算物理国家重点实验室基金资助(2000JS76.4.1.KG0119)资助.
刘 杰,男,1969 年生,博士研究生,副研究员,主要研究方向为大规模科学与工程并行计算. E-mail: liujie.nudt@163.com. **邓 力**,男,1960 年生,博士,研究员,主要研究方向为反应堆安全分析. **胡庆丰**,男,1958 年生,教授,主要研究方向为并行算法. **袁国兴**,男,1938 年生,研究员,主要研究方向为流体力学. **李晓梅**,女,1938 年生,教授,博士生导师,主要研究方向为并行算法和科学计算可视化.

that provides MPI2.0 and supports parallel file systems. The results indicate that the parallel algorithms proposed in the paper are practical and effective to simulate time-dependent MC transport. The best speedup is up to 30.

Keywords time-dependent Monte Carlo transport; parallel algorithm; parallel random number generator; parallel I/O algorithm; MPI.

1 引 言

MC 方法又称随机抽样方法,是以概率论为基础的一种具有独特特点的数值计算方法,它能够处理复杂的三维几何结构,充分利用各种丰富的截面数据和各种降低方差技巧,对粒子与物质相互作用的各种物理过程进行细致的描述.这些特点使得 MC 方法在辐射探测、反应堆物理、医学物理、地质物理、核技术和半导体设计等领域得到了广泛的应用[1~3].为了达到足够的计算精度,MC 通常需要模拟大量粒子的历史,特别当空间网格细化后,计算时间和存储空间都是单机无法承受的.利用高性能并行计算机或 Cluster 组织进行并行计算是数值模拟的最佳途径之一,与此同时,优良的并行算法是实现这一途径的基本保障.

西方发达国家,如日本、德国、法国、英国和美国对高性能计算技术都有重点发展计划.尤其是美国政府,非常重视高性能计算技术的研究,将其提到保持美国在科学、工程和技术上世界领先地位的战略高度优先发展.20 世纪 90 年代中后期美国能源部提出的"加速战略计算创新计划"(ASCI 计划),就是依靠高性能计算机进行三维、高精度全物理、全系统的模拟,来保证美国的核储备.为了充分发挥高性能并行计算机的性能,必须研究适应高性能并行计算机结构特点的并行算法及其关键技术,充分发挥高性能并行计算机的巨大潜力,推动学科发展.

MC 方法具有数据独立、循环粒度大等优点,其计算结果的精度与粒子的轨迹数的平方根成正比,而每个粒子的历史是相对独立的,特别适合于并行计算,目前 MC 方法已成为并行计算研究的主要对象之一.国内外学者对定常 MC 方法设计了很多并行算法,美国 Los Alamos 实验室 1997 年推出了 MCNP 4B 中子—光子—电子耦合输运 MC 程序[4],4B 版程序为 PVM 并行版本程序,其程序有许多缺陷,需要对其修改才能正常运行(我们对其进行了修改和测试,另文发表).王义等人[5]设计了 MC 方法的并行计算程序,对中子输运问题进行了研究.邓力等人[6,7]对 MCNP 3B 程序[8]进行了并行化.上述关于 MC 方法的并行计算问题均是处理静态问题,没有涉及到动态问题,因此没有涉及到 I/O 问题.

对于本文要讨论的非定常 MC 粒子输运并行计算问题相关的文献很少,该方法的最大特点是时间步多,每步计算粒度有限,剩余粒子信息的存储与访问给 I/O 带来巨大的压力,这些 I/O 操作是影响并行性能的瓶颈.动态 MC 方法的并行算法的运行模式也决定了并行算法性能的好坏.文献[9]利用 MPI I/O 设计了 PTMC 程序的并行 I/O 算法,但我们认为还有 4 个问题没有解决:(1)程序的重启动问题,文献[9]中的每个迭代步都需要进入和退出 MPI 系统,很费时,特别是当处理机台数多时;另外,多用户时,可能申请不到所需处理机,造成程序无法运行;(2)将每台处理机产生的数据 sb2_xx.dat 合成 sb1.dat,和将 sb1.dat 拷贝为多个 sb1_yy.dat($yy=1,2,\cdots,Q$)都是串行执行,而非并行 I/O;(3)每个迭代步中的每次 I/O 的数据量很小,为 36 字节的 1~3 倍,严重影响 I/O 性能,且在其运行模式下没有办法解决;(4)文中测试数据仅考虑前 105 迭代步,而实际应用中需要成千上万步,所得加速比在前 50 迭代步的加速比为 70 左右,而 50 迭代步以后的加速比仅大约为 10,迭代步数多时算法性能下降.

本文第 2 节给出了非定常 MC 方法的模型与并行算法,讨论了并行程序的加载运行模式,并对通信过程进行了优化;第 3 节给出了三种并行 I/O 算法;性能测试在第 4 节中给出;第 5 节给出了结论和下一步的工作.

2 问题模型与并行算法

2.1 问题模型

所要求解的问题是一个多时间步输运过程,串行计算的流程图由图 1 给出,图 2 给出了 MC 输运计算流程图.

MC 输运计算部分是一个独立的主程序,在程

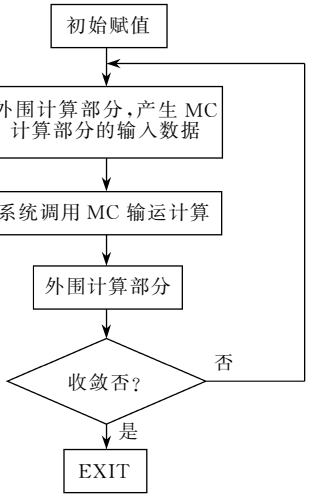

图 1　主控计算流程

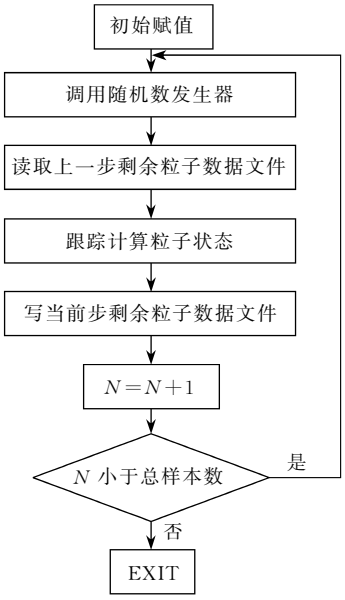

图 2　MC 输运计算流程

序的调用中采用 Fortran 语言调用系统命令(call system('mc'))的方式来完成,也就是说 MC 输运程序不是子程序.这种加载过程的好处是不需要对 MC 程序做任何改动,可以和静态 MC 输运计算使用同一个程序.

从 MC 输运计算流程图中可看出,MC 计算需要调用随机数发生器,并频繁地读上次迭代产生的粒子数据文件和写新的粒子数据文件,且总量很大,例如 20 万样本数,数据大小为 10MB,且数据大小随样本数增加线性增长.每步存取的数据量大小差异较大,I/O 性能差,影响程序的运行性能.

2.2　并行算法

2.2.1　模式 1

模式 1 是对原有动态 MC 输运程序的直接并行化,即直接对 MC 输运过程进行并行算法设计,其它计算部分作为串行程序不作任何改动,并行计算主控流程如图 1 保持不变,基于 MPI 并行环境的 MC 输运并行计算流程如图 3 所示.图 3 中 MPI_ Init( )表示启动进入 MPI 并行运行环境,MPI _Finalize( )表示退出 MPI 并行运行环境.结合图 1 可以看出,利用这种程序加载运行模式,每迭代一次,就必须进入和退出 MPI 并行运行环境一次,而这一过程受计算机延时的影响,利用的处理机越多,其加载与退出 MPI 并行运行环境的时间越长.当处理机台数为 32 时,1 次加载与退出 MPI 并行运行环境的时间大约为 1s,如果要计算的物理问题需要 1 万次迭代,那么加载与退出 MPI 并行运行环境的时间约为 1 万秒,它严重降低了并行效率.

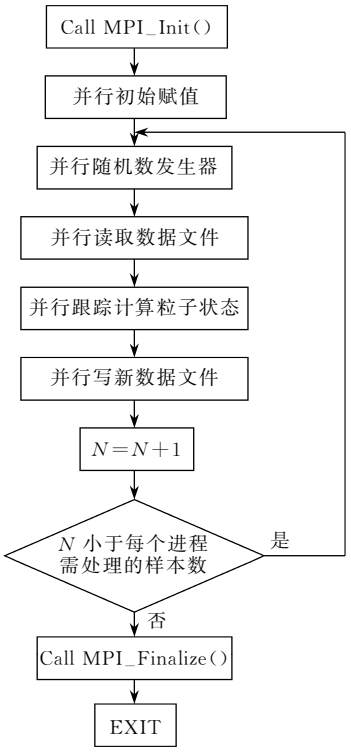

图 3　模式1:MC 输运并行算法流程

这种并行运行模式的好处是程序独立,和串行程序计算过程完全一致,不需要对程序作很大改动.

2.2.2　模式 2

模式 1 的每次迭代进入和退出 MPI 并行运行环境,耗时多,浪费了资源,另外,多用户时,可能申请不到所需处理机,造成程序无法运行.我们考虑的模式 2 是将 MC 输运过程变成总控程序的一个子程序,将进入和退出 MPI 并行环境放到主控程序中,这样进入和退出 MPI 并行运行环境只进行一次,效率显著改善.图 4 和图 5 给出了相应的并行算法流

程,图中进程和处理机概念相同.

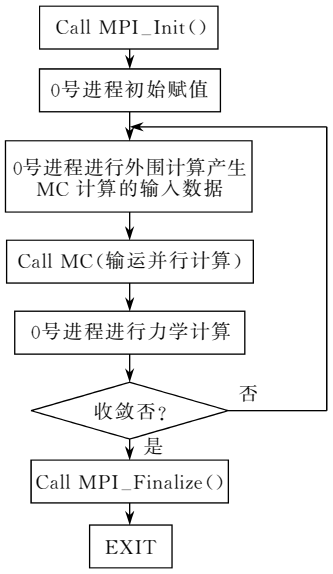

图 4 模式2:主控计算并行算法流程

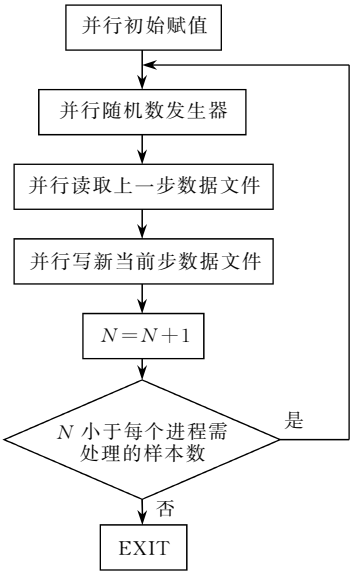

图 5 模式2:MC 输运并行算法流程

这种并行程序运行模式,总控部分的工作主要由 0 号进程来完成,此时相当于串行计算,因为现在的计算问题外围计算部分所占的时间很小,可以不考虑其并行计算. MC 输运计算部分不必每次都进入和退出 MPI 并行环境,但需要对程序的初始化工作进行仔细处理.

3 并行随机数发生器

串行随机数发生器有很多好的方法,杨自强等人[10]对其进行了综述. 要把在串行机上使用的 Monte Carlo 方法移植到并行计算机上运行,首先就需要解决随机数发生器的并行化问题,针对不同的并行计算机结构,可以构成不同的并行随机数发生器算法,文献[11~13]中讨论了并行随机数发生器的若干算法,但对分段种子如何产生未给出相关算法,需要加以研究. 乘同余发生器是应用最广泛的随机数发生器之一,设自然数 M 为模,α 为乘子,给定初始种子 x_0,乘同余发生器的串行算法使用的递推公式为

$$x_{n+1} = \alpha x_n \bmod M, \quad n = 0,1,\cdots \tag{1}$$

$$r_{n+1} = r_n / M \tag{2}$$

此时,$\{r_n\}$ 为(0,1)上均匀分布的伪随机数序列.

利用 Monte Carlo 方法在分布式并行计算机上进行计算,通常要把整个任务分解为多个子任务在各处理机中并行计算,所以对并行随机数发生器算法的要求是:(1)并行计算所使用的随机数序列和串行计算完全一致,这样才能保证并行计算结果的正确性;(2)在各个处理机中产生自己所需要的随机数序列;(3)处理机间通信量小,理想情况是没有通信的算法. 基于上述要求我们考虑在分布式存储环境下设计乘同余发生器的并行算法.

设分布式系统处理机台数为 P,需要产生的伪随机数序列的总长度为 N,将这个原始的伪随机数序列分割为 P 段,每段长度为 $L=N/P$(假设 N 可被 P 整除),将产生 P 段随机数的任务依次分配给 P 台处理机,第 i 台处理机需要计算的伪随机数序列为

$$r_{iL}, r_{iL+1}, \cdots, r_{iL+(L-1)}, \quad i = 0,1,\cdots,P-1 \tag{3}$$

上述计算随机数的过程,在各处理机上只要知道第一个随机数,其计算随机数的过程就是独立的,处理机间不需要通信,此时是最佳的并行算法,但难点是如何确定每台处理机上的第 1 个随机数.

利用公式(1),x_k 可以表示为 $x_k = a^k x_0 \bmod M$. 设 m 是满足 $2^m > k$ 的最小整数,令 $b=x_0$,$t=\alpha$,k 从 1 增加到 m,重复做下述操作:

$$\begin{cases} j \leftarrow k/2 \\ b \leftarrow tb \bmod M \quad \text{if } 2j \neq k \\ t \leftarrow t^2 \bmod M \\ k \leftarrow j \end{cases} \tag{4}$$

利用数学归纳法容易证明过程(4)最后所得的 b 值就是 x_k,且利用过程(4)迭代求出随机数序列中的第 k 个随机数的计算复杂性为 $\log k$. 每台处理机利用过程(4)迭代求出每台处理机上的第一个随机数,并将其作为种子,利用分段并行随机数算法,每

台处理机可求出相应的随机数列.

4　并行 I/O

近年来,并行 I/O 的研究在国际上极受重视,是当前高性能计算研究的重点之一,文献[14]讨论了 MPI I/O 的非连续数据块存取问题,文献[15]讨论了并行 I/O 问题,文献[16,17]讨论了应用问题的并行 I/O 问题. 因为实际问题的复杂性,对 I/O 提出了不同的要求,要根据实际问题的需要,设计并行 I/O 算法,充分发挥计算机系统的 I/O 性能.

MC 输运过程追踪每个粒子状态的计算过程都要随机访问上次迭代产生的截面数据文件的某些行的元素(每一行有 9 个元素),计算完成后又要将新产生的数据写到截面数据文件中,因为 I/O 量比较大,必须对其进行仔细处理,设计好的并行算法.

4.1　方法 1

因为每台处理机追踪每个粒子状态的计算过程是不相关的,因此对截面数据文件的读与写也是不相关的. 对于读可以采用并发读取同一个文件的方式进行;对于写可以让每台处理机各写一个局部文件,MC 输运过程完成以后,再将局部文件合成一个完整的文件,供下次迭代使用,这个合成过程必须串行完成,图 6 给出上述过程的示意图.

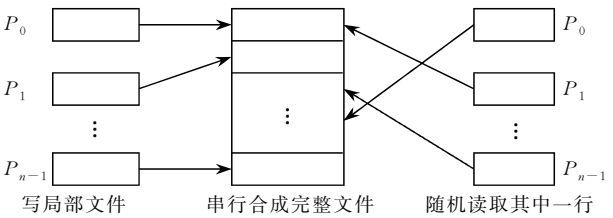

图 6　并行 I/O 方法1示意图

MPI 2.0 提供了 MPI 并行 I/O 函数,可以分为两类:第一类是每个过程并行访问不同的文件;第二类是各个进程并行访问同一文件. 因此方法 1 可以直接调用 MPI 2.0 提供的并行 I/O 函数完成,调用的函数如下:MPI_File_Open, MPI_File_Read, MPI_File_Write 和 MPI_File_Close.

这种方法程序设计简单,只需将原来的 open, read, write 和 close 与 MPI 2.0 提供的并行 I/O 函数进行相应的修改即可. 这种并行 I/O 存取模式由大量的数据量很小、且非连续的小块数据组成,每次存取 36～72 字节数据,而一个应用问题如果需要很多小而频繁的 I/O 请求,就会导致 I/O 性能的戏剧性下降,另一方面这些 I/O 操作依赖于每个粒子的追踪过程,很难利用 MPI 2.0 提供的并行 I/O 函数获得高性能.

为了进一步深入了解方法 1 的性能,找到瓶颈问题,我们编写了 3 个测试程序:测试程序 1 对应方法 1 的写局部文件过程,各进程每次写的数据量大小为截面数据文件的两行,共 72 字节;程序 2 将各进程写的局部数据文件串行合并为一个完整的数据文件;程序 3 对应方法 1 的读局部文件过程,每个进程根据随机数产生的行号读取数据文件. 表 1 给出了测试结果,表中完整文件的大小为 10MB,设 P 为处理机台数,每个进程写或读的文件大小为(10/P)MB.

表 1　方法 1 性能测试　(时间单位:s)

处理机台数	测试程序 1	测试程序 2	测试程序 3	合计
2	11.9	1.2	19.3	32.4
4	6.3	1.2	11.8	19.3
8	3.6	1.3	8.3	13.2
16	2.2	1.3	7.6	11.1
32	1.4	1.4	8.3	10.9
64	0.8	1.4	8.4	10.6

从表 1 的测试数据可以看出测试程序 2 将局部数据文件串行合并为一个完整的数据文件的时间基本保持不变,测试程序 1 的计算时间基本上是线性减少,而测试程序 3 在处理机台数大于 8 时测试时间基本保持不变,是方法 1 的瓶颈.

4.2　方法 2

注意到,当前并行机系统从磁盘中存取文件的速度比从内存中存取数据的速度慢;另一方面,假设存取的数据量一样,那么存取大块连续数据的时间,显然要比存取小块且不连续数据的时间快得多.

基于上述考虑,我们首先将整个数据连续读到内存中,这样追踪每个粒子所需的截面数据只需到内存中访问即可;写截面数据文件时先将小块数据写到内存中,待输运过程完成,再将生成的大块数据一起写到同一文件中(图 7 给出示意图). 将小块数据合成大块数据一起并行写,将少量、不连续地随机

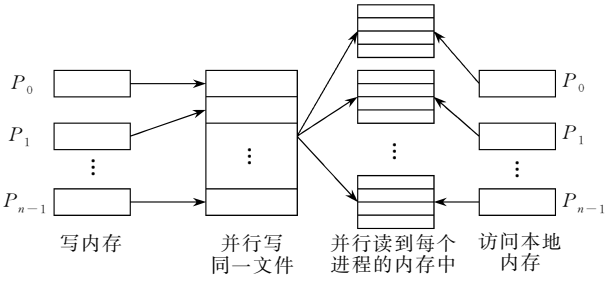

图 7　并行 I/O 方法2示意图

访问截面数据文件转化为访问内存.

为了进一步深入了解方法 2 的性能,找到瓶颈问题,我们编写了 4 个测试程序:测试程序 1 对应方法 2 的写内存;程序 2 各进程并行写同一数据文件;程序 3 将数据文件并行读到本地内存中;测试程序 4 访问本地内存. 表 2 给出了测试结果,表中完整文件的大小为 10MB,程序 1 和程序 4 每个进程写或读的文件大小为(10/P)MB.

表 2 方法 2 性能测试 (时间单位:s)

处理机台数	程序 1 的计算时间	程序 2 的计算时间	程序 3 的计算时间	程序 4 的计算时间	合计
2	0.05	2.1	2.6	0.08	4.8
4	0.03	1.5	3.1	0.04	4.7
8	0.02	1.2	3.3	0.03	4.5
16	0.01	1.2	3.3	0.02	4.5
32	0.003	1.1	3.9	0.01	5.0
64	0.002	1.1	4.2	0.005	5.3

从表 2 的测试结果可看出:测试程序 1 和测试程序 4 的计算时间很小,计算时间主要集中在测试程序 3 和测试程序 2.

4.3 方法 3

方法 1 和 2 都是将上次迭代产生的数据文件写到磁盘上,然后在下次迭代中再读取这一数据文件,这两种方法适合于第 2 节中介绍的两种模式的并行算法.

下面要讨论的方法 3 只适用于模式 2 的并行算法流程. 从方法 2 中可以看出,如果不将文件写到硬盘上,而是直接在内存中将每台处理机中的数据合成一个大的数据文件,每台处理机都拥有这个数据文件,这样将大大缩短 I/O 时间,示意图由图 8 给出.

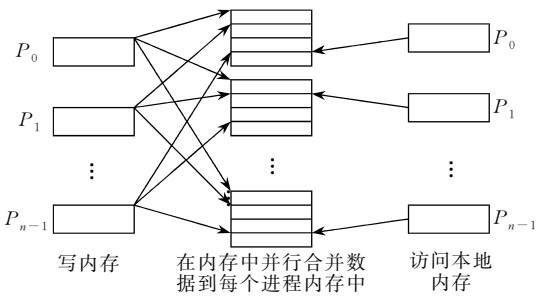

图 8 并行 I/O 方法3示意图

当然方法 3 的实现前提是这个截面数据文件不能超内存,如果超内存就会导致性能下降. 方法 3 中的写内存与访问本地内存和方法 2 相同,不同点在于方法 3 不是将数据写到硬盘的文件中,实际上用 MPI 编写并行程序时只需用 MPI_ALLGATHER 函数来完成数据收集和合并的过程. 表 3 给出了 MPI_ALLGATHER 函数的性能测试结果,数据大小为 10MB. 从测试结果可以看出方法 3 的性能要比方法 2 好.

表 3 MPI_ALLGATHER 函数性能测试结果

(时间单位:s)

处理机台数	时间
2	2.1
4	2.2
8	2.3
16	2.5
32	2.6
64	2.8

5 数值测试

测试的机器环境为提供并行文件系统和 MPI 2.0 并行计算环境的高性能并行计算机系统. 物理模型为非定常中子输运问题,总样本数为 200000,收敛需要的迭代次数为 1089 步.

我们首先对并行随机数发生器的性能进行了测试,因为随机数发生器并行计算过程中不需要通信,并行性能很好,只需测试其正确性,我们将串行计算结果和并行随机数发生器的计算结果进行了对比,计算结果完全一样,这表明这种并行随机数发生器算法不但正确而且高效,是理想的并行随机数产生算法.

接着我们对并行算法的两种运行模式进行了对比,图 9 给出了测试结果. 两种运行模式均采用文中给出的并行随机数发生器算法,并行 I/O 方法采用方法 2. 从图 9 中可以很清楚地看出,模式 2 好于模式 1,特别是当处理机台数很多时,更是如此,这主要是因为模式 2 节省了大量的进入和退出 MPI 并行环境的时间. 但模式 1 保持串行程序不变,使得外围部分的工作与输运部分的工作成为相对独立的两个部分,保证了研究人员能够相对独立地从事自己

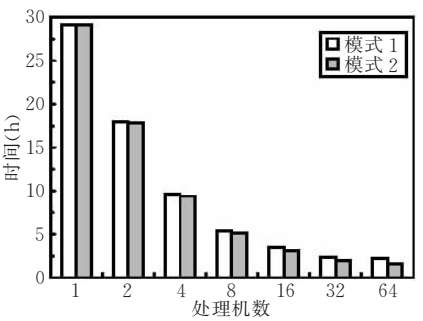

图 9 并行算法模式 1 和 2 的测试时间
(并行 I/O 采用方法 2)

的研究工作;另外,对于有的计算模型,每步迭代的时间差异很大,此时为了节省计算机资源和缩短计算时间,要根据计算时间的长短动态地调整处理机的台数,从而得到最佳的并行计算效果,此时模式 1 比较适用,而对于模式 2,处理机台数固定,不能自动调整.

下面是采用不同并行 I/O 方法对并行算法的性能进行的测试,图 10 给出了测试结果.测试采用文中给出的并行随机数发生器算法和并行算法的运行模式 2.从测试结果可以看出,方法 1 最差,方法 3 最好.同上面的分析,因为模式 1 存在的必要性,使得方法 1 和方法 2 同样是必要的,对于数据文件不超内存时,模式 1 可结合方法 2 使用;对于截面数据文件超内存时,方法 1 是适用的.目前我们求解的问题规模还无法扩得太大,无法给出测试对比的结果.图 11 给出了加速比,对比单机串行计算结果,64 台处理机最大加速比为 30,达到了预期的目的.

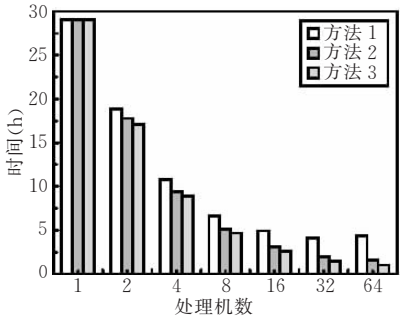

图 10　并行算法模式 2 与并行 I/O 采用不同方法的测试时间

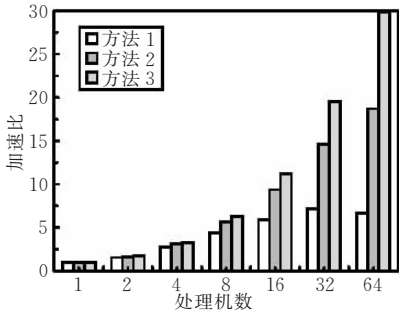

图 11　并行算法模式 2 与并行 I/O 采用不同方法的加速比

6　结论和下一步的工作

本文给出了依赖于时间的动态 Monte Carlo 输运问题的并行算法,设计了一种并行随机数发生器算法,提出了三种并行 I/O 算法,性能测试结果表明,我们所提出的并行计算方法是可行的,获得了很好的并行计算效果.

近期内,我们将开展下面两个方面的工作:

(1)算法中所要存取的截面数据文件的大小是随样本数的增加而线性增长的,特别是在读取数据文件时要随机访问全局数据,需要对每台处理机产生的数据进行合并,需要大量的时间,严重影响算法的性能.下一步我们将从物理模型出发,研究数据局部化的方法,即增加样本数,在不影响计算精度的前提下,访问局部的截面数据文件;

(2)目前的并行算法对外围计算部分还没有并行化,下一步将完成外围计算部分的并行化工作.

致　谢　感谢北京应用物理与计算数学研究所的莫则尧博士、黄正丰研究员、国防科学技术大学的黄春、周恩强、张文勇等同志与作者的有益讨论和宝贵意见.

参　考　文　献

1　Jiang Xin-Biao, Chen Da, Xie Zhong-Sheng, Zhang Ying. Monte Carlo method for reactor duct shielding calculation. Chinese Journal of Computational Physics, 2001, 18(3): 285～288(in Chinese)
(江新标,陈　达,谢仲生,张　颖. 反应堆孔道屏蔽计算的蒙特卡罗方法. 计算物理, 2001, 18(3): 285～288)

2　Jiang C. Z., Rosenberg N., Morin P.. The spatial distribution and resolution of coaxial backscattered electron in SEM calculated by Monte Carlo simulations. Journal of Micoscopy, 2000, 198(1): 17～23

3　Matsuzawa K., Uchida K., Nishiyama A.. Monte Carlo simulation of sub-0.1 μm devices with Schottky contact model. Ieicet Electron, 2001, E83C(8): 1212～1217

4　Briesmeister J. F.. MCNP——A general Monte Carlo for N-particle transport code. Los Alamos National Lab: Technical Report 4B LA-12625-M, March, 1997

5　Wang Yi, Yang Ping-Li, Zhu Wei-Jie, Xia Shi-Qing. Research in parallel computing of Monte Carlo code. Chinese Journal of Nuclear Electronics and Detection Technology, 2001, 21(1): 52～54(in Chinese)
(王　义,杨平利,朱伟杰,夏四清. 蒙特卡罗程序并行计算研究. 核电子学与探测技术, 2001, 21(1): 52～54)

6　Deng Li, Huang Zheng-Feng, Xu Hai-Yan, Wang Rui-Hong. Parallelization of MCNP Monte Carlo neutron and photon transport code. Chinese Journal of Numerical Mathematics and Applications, 2001, 22(4): 262～265(in Chinese)
(邓　力,黄正丰,许海燕,王瑞宏. 蒙特卡罗中子-光子输运程序 MCNP 的并行化. 数值计算与计算机应用,2001, 22(4): 262～265)

7　Deng Li, Xie Zhong-Sheng, Huang Zheng-Feng, Xu Hai-Yan.

The parallel design of Monte Carlo code and measure of enhance speedup. Chinese Journal of Computational Physics, 2001, 18(2): 177～180(in Chinese)
(邓 力,谢仲生,黄正丰,许海燕. MC 程序并行设计及提高加速比的措施. 计算物理,2001, 18(2): 177～180)

8 Briesmeister J. F.. MCNP——A general Monte Carlo for N-particle transport code. Los Alamos National Lab: Techcical Report 3A LA-7396-M, Rev. 2, March, 1986

9 Mo Ze-Yao, Huang Zheng-Feng, Xu Hai-Yan. Application of MPI-IO in parallel particle transport Monte-Carlo simulation. Annals of Institute of Applied Physics and Computation Mathematics, 2001. 334～342(in English)

10 Yang Zi-Qiang, Wei Gong-Yi. A review on some new methods to generate random numbers. Chinese Journal of Numerical Mathematics and Applications, 2001, 22(3): 201～216(in Chinese)
(杨自强,魏公毅. 综述:产生伪随机数的若干新方法. 数值计算与计算机应用, 2001, 22(3): 201～216)

11 Makino J., Miyamura O.. Parallelize feedback shift register generators of pseudorandom numbers. Parallel Computing, 1995, 21(4): 1015～1028

12 Makino J.. Lagged-fibonacci random number generators on parallel computers. Parallel Computing, 1994, 20(5): 1357～1367

13 Wei Gong-Yi, Yang Zi-Qiang. Some algorithms of parallel random number generators. Chinese Journal of Numerical Mathematics and Applications, 2001, 22(4): 311～320(in Chinese)
(魏公毅, 杨自强. 关于并行随机数发生器的若干算法. 数值计算与计算机应用,2001, 22(4): 311～320)

14 Thakur R., Gropp W., Lusk E.. Optimizing noncontiguous access in MPI I/O. Parallel Computing, 2002, 28(1): 83～105

15 Schikuta E., Wanek H.. Parallel I/O. The International Journal of High Performance Computing Applications, 2001, 15(2): 162～168

16 Boss R., Nurmi D.. A case study in application I/O on Linux clusters. In: Proceeding of the SC'2001, 2001

17 Mackay D., Mahinthakumar G., Azevedo E.. A study of I/O in parallel finite element groundwater transport code. International Journal of High Performance Computing Applications, 1998, 12(3): 307～319

LIU jie, born in 1969, Ph. D. candidate, associate professor. His research interests include parallel algorithms, parallel applications and benchmarks and high performance computing.

DONG Li, born in 1960, Ph. D., professor. His research interests include particle transport and computational physics.

HU Qing-Feng, born in 1958, professor. His main research interests include computer performance evaluations, computational physics and high performance computing.

YUAN Guo-Xing, born in 1938, professor. His research interests include parallel computing, computer performance evaluations and computational physics.

Li Xiao-Mei, born in 1938, professor, Ph. D. supervisor. Her research interests include parallel computing, computer performance evaluations, high performance computing, and computer graphics.

Background

Monte-Carlo (MC) method is a traditional and effective numerical method capable of treating very complex three dimension configurations for particle transport problems. In order to simulate the history of a large number of particle, especially after refined space meshes, the calculation becomes very large, so must researching parallel algorithms of MC method to improve the performance. Authors have designed two parallel codes of MPI and PVM for time-independent MC transport simulations.