# 粒子跟踪算法的并行设计

A Parallel Scheme of Particle Tracing Algorithm

赵晓玲　孙济洲

(天津大学计算机科学与技术系　天津300072)

**Abstract** This paper presents a parallel implement scheme for particle tracing algorithm and the equipollent parallel model. The scheme is designed and implemented by programs. In addition, implement is accomplished and results are shown.

**Keywords** Particle tracing, PVM, Parallel implement, Test and result

任何算法的有效并行要求深入了解计算过程的细节,掌握参与计算的各个数据部分之间的相互依赖关系,针对计算的类型和应用的约束条件提出合理的任务划分和算法分解方案。因此,本文首先介绍了粒子跟踪算法的计算过程,并分析了该算法并行设计的可能性。从计算过程的数据相关性出发,我们给出了具体的并行绘制模型,设计出可行的并行划分策略。最后,对该并行策略进行了测试,验证了该设计的正确性和可行性。

## 1. 粒子跟踪算法并行设计的可行性分析

粒子跟踪算法是近年来逐渐成为最受瞩目的一种整体光照算法,它是一种视点无关算法,可以同时很好地表现漫反射表面和镜面反射表面,而且这种算法具有良好的可并行性。

粒子追踪算法的本质是利用 Monte Carlo 方法对绘制方程进行直接求解。在该算法中,主要的计算量是粒子跟踪任务。粒子跟踪的任务是,对于场景中的每一个光源,根据其能量 $\Phi$ 的大小,由光源向场景中发射一定数目的粒子并分别对这些粒子进行跟踪,每一个粒子具有波长等光学特征并携带一定能量 $\phi$(所有的粒子的能量总和为光源的能量,即 $n\times\phi=\Phi$),粒子从光源出发时其方向以及其它的光学特性(如波长)由建模时指定的光源特性决定。进入场景后,当粒子遇到场景中的表面时,一方面在这个表面上记录一个撞击点(具体地说,记录这个撞击点的坐标、波长以及能量),另一方面根据该面的双向反射分布方程 BRDF 概率地决定是对粒子进行反射、折射还是透射、吸收,经过与多个表面进行撞击之后,粒子的能量逐渐减小,当它的能量小于预先设定的域值时,认为再跟踪该粒子已经不能提高该场景图像的效果,结束对这个粒子的跟踪过程。当对所有的粒子的跟踪完成之后,粒子跟踪计算结束,此时在场景中的每一个表面上都生成了一系列撞击点,由表面撞击点的多少可以确定该表面的亮度。

从粒子跟踪的过程可以看出,粒子跟踪算法本身具有很好的可并行性,粒子从光源出发进入场景后,它的整个跟踪过程便与其他的粒子无关,只和场景的分布情况有关。因此将粒子跟踪算法实现并行是可行的。但是,由于粒子计算的复杂程度和场景的复杂程度成正比,即场景越是复杂,粒子跟踪的计算量就越大,因此只简单地考虑粒子并不能有效地实现粒子跟踪算法的并行化。

粒子跟踪过程的密集程度与空间位置有直接的关系。在离光源距离近的局部空间内,粒子撞击的数目多,该空间表面亮度强一些;离光源远的局部空间内,粒子撞击的数目少,表面亮度弱一些。因此粒子撞击的密集程度,也就是粒子跟踪过程的计算量与空间位置有直接的关系。因此任何基于 Particle Tracing 的绘制算法为了加速光线的相交运算,都需要对空间的划分,在本算法中采用的是具有良好加速性能的空间自适应八叉树的划分策略。空间自适应八叉树是首先建立物体空间的包围盒,然后,使用通过包围盒中点,且互相垂直的三个平面把包围盒等分成八个互不相交的子包围盒。如果与子包围盒相交的物体个数超过预先规定的个数(如四个),则把该子包围盒再等分成八个更小的包围盒。这样递归地处理下去,直至得到一个叶子可能在不同深度的八叉树,称作自适应八叉树。与树中每个叶子结点相交的物体个数不超过预先规定的个数。离光源近的子空间计算量相对大一些,而离光源远的子空间计算量相对就小一些。因此粒子跟踪算法的并行实现可以在划分空间上入手。

## 2. 并行绘制模型

任何一个计算问题的求解都要在具体的计算环境中实现,计算环境中处理机数目、网络带宽及互连结构等各参数的不同将对计算的性能产生不同的影响。根据可用的实验环境对各计算参数的具体约束条件,可以得出并行计算的运行约束模型,为并行划

分和算法设计提供验证的理论依据。

### 2.1 分布式网络运行环境

我们将粒子跟踪算法进行并行划分的目标运行系统即为目前普遍使用的这一类分布式并行环境，一般是由多台 PC 机通过局域网连接在一起，运行 Linux/Windows 系列操作系统，通过 PVM/MPI 完成进程之间的消息传递。

### 2.2 并行划分的要求

任务划分策略的好坏，将直接影响并行执行的效率和性能，也对负载平衡策略等的选择起着重要的影响。一个好的并行划分策略，不仅要将计算任务均衡分割，还要同时考虑分割后子任务之间的通讯量、子任务计算所需的存储量等问题，具体要求有分解计算量、减少通讯量、降低存储量。

粒子跟踪算法最基本的计算单位是空间中粒子的一次碰撞，这是可以实现的最细粒度的计算估计。在各个层次的循环和迭代之间，存在着复杂的数据关系和计算关系，由于基本计算的数量巨大，所需要的源数据和生成的结果数据累积起来的数据量是相当庞大的，在分布式的网络计算环境，所造成的传输延迟和等待对于任何计算而言都是无法忍受的。因此，减少通讯量将成为影响并行性能的决定性因素。随着节点机能力的不断增强，存储量已不再成为制约性能发展的主要瓶颈，在必要时，可以牺牲存储量来获得通讯量的降低，这对并行系统整体性能的提升将起到很大的促进作用。

### 2.3 并行绘制计算的主从模型

按照 Flynn 的分类，粒子跟踪计算属于以指令并行为特征的 SPMD 计算，在不同的数据空间上执行相同的程序指令，得到相应的运算结果，每个计算节点完成的是同样的工作，运行同样的计算程序。一般的计算方案通常为主-从(Master-Slave)方式：有一个专门的主进程(称之为 Master 进程)负责任务的分发和调整，其余的多个从进程(为 Slave 进程)负责接受任务调度并完成计算，将计算结果向主进程提交。主从方式的任务并行可以使用静态或动态负载平衡方法，允许主进程对从进程进行实时任务调度和迁移，使应用程序可以根据系统资源的变化自动进行调整。主从模型能够获得很高的计算速度，同时也具有一定的可扩展性。对于数量众多的从进程的情况，主进程集中控制的方法可能会成为应用程序的瓶颈，这时可以将主进程扩展为一个主进程集，分别控制不同的子进程集合，从而提高了该模型的可扩展能力。

对于粒子跟踪计算，应用主从模型可以实现多种并行划分策略。如下图所示，任务的划分和调度都可以在主进程中实现，从进程完成分配的任务后可以主动向主进程申请任务，由主进程按照其执行情况进行分配。任务划分的粒度可以有所不同，由主进程根据总体任务执行情况和各从进程的负载情况实时进行调整。此处进程为计算的逻辑组成部分，一台处理机上可以有一个或多个进程。根据具体的网络连接，各进程之间可以为不同的互连方式和拓扑结构。

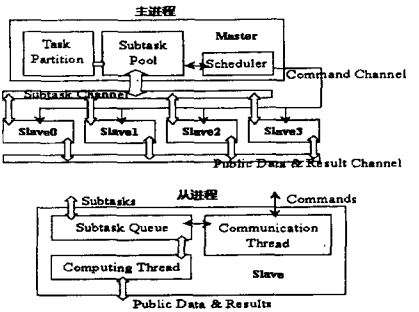



图1 并行绘制计算的主从模型

## 3. 任务分解策略及评价

本节我们介绍基于图像空间的划分方法。结合粒子跟踪算法，可以得出相应的负载时间一空间分布，所需要进行的负载平衡策略，相应的通讯开销等。通过分析可以发现，基于图像(像素)空间的划分策略由于其各个计算部分基本不相关，因此产生的通讯量极小，对于我们的目标实验环境非常适合。

### 3.1 基于对象空间的划分策略

由于场景文件一般都比较复杂，每台从进程上都对粒子进行整个场景中的跟踪是比较耗费资源的。因此我们考虑按场景空间进行分配。场景空间存在很大的不均衡性。有的子空间很简单，计算量很小，而有的子空间很复杂，计算量因而就比较大。所以按场景空间向从进程分配时，考虑到负载平衡的问题，我们按八叉树的结构进行分配。八叉树结构是按照场景的复杂度建成的，该树的层次反映了场景的复杂程度。但是它没有考虑到光源对粒子跟踪计算量的影响。因此我们进一步提出一种新的八叉树结构，即加权八叉树。它的主要思想是：第一步，首先建立物体空间的八叉树；第二步，给八叉树的各个结点加权。例如每个非空叶结点的权值是1，空叶结点的权值是0，父结点的权值由其子结点的权值决定；第三步，根据光源的位置对八叉树的各个结点加权。例如光源所在的最大的子空间增加权值80，那么它的每一个孩子空间的权值增10(假设其孩子空间都不为空)。这样在向各台从进程分配任务时，只分配该从进程负责的子空间的范围和信息及该子空间内的粒子就可以了。

首先由主机将数据文件读入并进行场景分析，用空间网格、八叉树场景层次结构树来对整个场景进行划分，每一个场景由一独立的处理单元来表示，存放该场景块的信息，并与周围的场景块相连接。粒子由光源（也为场景块）发出，按照空间划分进入不同的场景块，产生撞击，并在该场景块中留下撞击点。此后，该场景块又成为新的粒子发射单元。考虑到通信量的问题，我们在所有的处理机上复制全部的场景数据，而把要计算的空间划分成 N 个子空间，分配给各个处理机，每台处理机处理各自子空间的粒子。

·所有的处理机上都存有自适应八叉树结构、场景内容。

·并行粒子跟踪任务：每一台从进程的并行任务有三步：

1. 收主进程的分配任务，包括子空间的范围及属于该子空间内的粒子起始位置、方向、波长、能量等信息；

2. 进行粒子跟踪；

3. 处理过的粒子如果不在该子空间内，则保存在一个结构中，然后统一向主进程发送。

·主从机器的任务分配：

1. 主进程的任务是划分八叉树结构，处理光源，将粒子分配到各个子空间中去，等待向从进程分配任务。并处理从进程返回的信息，将处理过的粒子按其返回信息放入所属子空间中。

2. 从进程的任务是接收子空间的范围，进行粒子跟踪，然后返回处理过的粒子信息。

·主从机器之间的通信量：

每分配一次任务，都要传送分配子空间的起始位置、终止位置，及该空间内粒子的起始位置、方向、波长、能量等信息。返回要传送处理过的粒子的各项信息及将要所属的子空间的起始位置、终止位置。

·优缺点

这种做法可以使处理机之间的通信量减到最少，仅限于传送任务分配信息和回送计算结果。能达到较好的加速比。

它的缺点就是要在所有的处理机上复制全部的数据，如果场景非常复杂且所用的数据结构又需要大量存储空间的话，则这种图像空间划分的方法不够有效。

### 3.2 动态空间分配策略

各进程处理能力与各子任务计算量之间的不一致性，是导致静态划分策略出现困难的根源。结合空间划分策略的优点，可以得到动态空间分配策略。具体实现如下：

1. 主进程负责任务的分配和调度，从进程在计算空闲时向主进程请求任务，由主进程根据总体任务完成情况进行分配；

2. 每个从进程都有一份场景数据和光源信息的拷贝，该从进程获得的所有子任务共用该拷贝；

3. 主进程将光源粒子划分为大量的粒子集合，每个粒子集合包含若干个粒子，以此作为计算的基本分配单位；

4. 粒子集合的数目要远大于参与计算的从进程数目，以保证宏观上计算的近似平均分配。

动态空间分配策略所要求的条件是粒子集合数要远大于计算进程数，这样才能保证在整个计算进行的过程中，每个计算进程所分配到的计算量是近似平衡的，使得所有进程可以在近似相等的时间内结束计算。对于一般的计算场景来说，粒子数 $N_p$ 的数量级为100×100，而分布式计算的处理节点数目一般不超过40，而粒子集合的粒度可以为每个集合含1000个粒子，这样该条件是可以满足的。同时对于数量巨大的节点机，可以用来绘制更大规模的图像，其数量级同样是可以扩展的。实际运行中，粒子集合中的粒子数要根据运行情况进行调整，以得到最好的整体并行效果。

## 4. 并行测试及结果分析

### 4.1 测试环境

我们在实验室的4节点集群系统上完成了渐进势能跟踪算法的并行测试。该系统的基本情况如下：

硬件系统：4台 IBM-PC 节点机，75MHz，64MB RAM

网络连接：10Mbps 以太网卡，通过10/100M 自适应交换式集线器连接

软件系统：RedHat Linux 6.2，TCP/IP 网络协议，GCC 3.5.1

并行开发及支持环境：PVM 3.4（RedHat 6.2 自带）

窗口及图形绘制系统：RenderPark 3.2.4，Mesa 3.0，Lesstif 2.0.5

测试场景的主要特征是：曲面数是17个，物体个数是3个，光源2个。在该场景中，光源位于上侧和右侧，物体经过镜面反射在镜子里形成了自己的像。

在实验过程中我们记录了总体计算时间（从程序开始到整个程序结束为止的时间）和纯粹计算时间（总体计算时间除去初始化时间、打包、解包以及各台 slave 建立输出文件的时间）随机器数量的增加而发生的变化。另外我们分别测试了不同计算规模的系统性能。

### 4.2 结果分析

·加速比　由实验所记录的数据我们可以得到该并行方案对纯粹计算时间和总体计算时间的加速比。我们对两种计算规模进行分析，一种是光源发射

的粒子个数为205000个，另一种粒子个数为25000个。从图2和图3可以看出该并行方案对纯粹计算时间能得到理想的加速比，而且粒子数目较多时更理想，也就是计算规模越大，取得的效果越好。总体计算时间相对于纯粹计算时间来说，加速比差一点，这是因为通信延迟带来的不利因素。每次通信都要进行初始化、打包、解包等处理工作，在这种情况下每次打包发送的粒子个数对通信延迟有直接的影响。在多次实验后，我们发现每次通信时发送的粒子个数在数千个左右时比较理想。

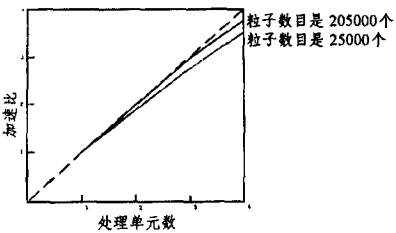
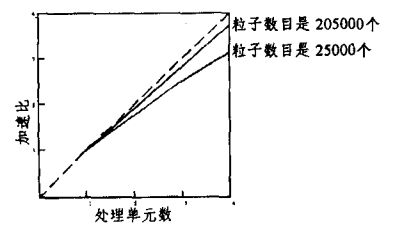



图2 两种计算规模对纯粹计算时间的加速比




3 两种计算规模对总体计算时间的加速比

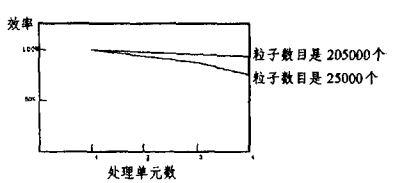



图4 效率曲线

·效率曲线 并行系统的效率是用来度量系统中的所有处理单元用于有效计算时间占整个计算时间的比率。同样是对两种计算规模进行比较。图4给出的就是两种计算任务下的效率曲线。从图中可以看出这种并行策略对较大的计算规模比较理想，而粒子跟踪算法本来就是计算量很大的整体光照算法，因此并行策略能够取得很好的效果。

·可扩展性问题 由实验得到的数据可知，当处理单元数目增加时，加速比成线性增长，也就是说该算法能得到很好的扩放性，可以有效地利用大量的处理器，可以移植到大规模处理机上运行。

**结论和展望** 本文对粒子跟踪提出了并行算法，并在此基础上设计了并行方案。经过单机和多机实验，我们可以得出结论，本文提出的并行算法能够取得比较理想的加速作用，确实是可行的，并且具有以下优点：

·主从机器之间的通信量减到最少，仅限于传送任务分配信息和回送计算结果。因此网络传输带来的延迟对整个计算时间的影响不大；

·由于采用了动态分配方案，每台 slave 动态申请计算某一个空间的粒子。这样负载平衡效果较好；

·采用了限制子任务大小的方法，加速比不会受场景中各个空间计算量分配的影响；

·该并行算法更适合于多光源，或是光源强度比较大的场景文件的处理。这样得到的加速比会更理想。

但是实验也出现了一些问题，对总体计算时间不能得到线性的加速比。由于实验条件的影响，每台处理单元的性能不是很理想，而且网络传输也有延迟。对于运行环境来说，我们采用的系统是 LINUX 平台下的 PVM 环境，受软件的影响，通信策略不得不适合该软件的通信格式，也受到不同程度的影响。最关键的因素还是数据收发和调度策略的问题。因此我们还需要进一步的改进，进一步完善该并行算法。我们继续研究的方向包括以下几点：

1. 进一步改进任务划分的方法，减少主从机器之间的通信量。继续划分场景文件，在划分场景文件中，要考虑到光源位置对该空间计算量的影响，即用到我们前面讨论的加权八叉树的内容。在本次实验中我们可以看到光源的位置对该空间计算量起着主要作用。

2. 进一步改进调度策略和负载平衡的方法。

3. 改进硬件和软件环境，使网络传输延迟小一点，通信机制更灵活一些，让我们的调度策略不受影响。

## 参考文献


1 张林波，等. 网络并行计算与分布式编程环境. 科学出版社
2 Arvo J. Analytic Methods for Simulated Light Transport: [Ph. D. Dissertation]. Yale University, Dec. 1995
3 Xu Q, Sun J, Wei Z. Zero Variance Importance Sampling Driven Potential Tracing Algorithm for Global Illumination. In: Proc. of Ninth Intl. Conf. in Central Europe on Computer Graphics and Visualization (Winter School on Computer Graphics), Plzen, Czech Republic, 05-09 Feb. 2001
4 黄凯，高等计算机系统结构：并行性可扩展性可编程性. 北京：清华大学出版社，1995
5 彭群生，鲍虎军，金小刚. 计算机真实感图形的算法基础. 北京：科学出版社，1999