分布式计算环境下的并行体绘制算法

余盛明 李 华 刘慎权

(中国科学院计算技术研究所CAD开放实验室 北京 100080)

摘要 分布式计算环境中基于消息传递机制的分布式共享缓冲区中,Cache效率是算法性能的"瓶颈".本文在分布式共享缓冲区上实现了一个并行体绘制算法.在数据空间,八叉树快速分类改善了Cache的空间相关性;在图象空间,Hilbert象素遍历方式改善了Cache的时间相关性.在曙光1000和SGI工作站网络上的实验结果都表明,算法的网络数据传送量大大减少,Cache效率明显提高,绘制时间大大缩短.

关键词 并行体绘制,八叉树,象素遍历,光线追踪.

快速体绘制是可视化应用的重要目标.为了交互地分析、探索科学数据,可视化应用程序必须提供灵活的用户接口和快速的图象更新速率.在追求快速体绘制的过程中,人们提出了4种主要的方法.第1种方法以图象质量为代价来换取速度.[1]第2种方法依赖特定的硬件设备,它所实现的算法缺少灵活性.[2]第3种方法是在SIMD上并行化简单体绘制算法[3,4],当绘制算法任务间的通信模式规则划一时,它可以取得较好效果.第4种方法是在MIMD上并行化快速体绘制方法,通过算法的优化来减少计算代价[5],它的缺点是大多数算法优化都有数据依赖性,常常需要预处理.但是,与其它方法相比,这个方法不需要过于复杂的硬件环境,相应的性能价格比要高一些.

我们采用第4种方法,开发出分布式计算环境中的并行光线追踪方法.分布式计算环境,是指具有分布式存储、基于消息传递的多处理机(器)系统.在分布式环境中的一个重要问题是当计算所需的数据不在本地存储器时的处理策略.

很自然的办法是找到一种任务分配方案,让每个任务只访问本地数据[4,5],或者当任务要访问远程数据时,把该任务发送到数据所在的节点.[6]在光线追踪方法中,只访问本地数据几乎是不可能的.

我们在分布式计算环境下实现了一个虚拟存储器—分布式共享缓冲区.这个用软件实现的共享缓冲区(或称Cache)是本地存储器中的一个缓冲区,它存放最近用到的非本地数据,避免远程数据访问.我们将体数据分布在物理位置不同的计算节点中,通过共享缓冲区

• 本文研究得到国家自然科学基金资助.作者余盛明,1970年生,硕士生,主要研究领域为科学计算可视化和并行计算.李华,1957年生,博士,研究员,主要研究领域为计算机图形学,计算机辅助几何设计,工程CAD技术等.刘慎权,1930年生,研究员,博士导师,主要研究领域为计算机图形学,科学计算可视化,计算机动画,CAD/CAM等.

本文通讯联系人:余盛明,北京 100080,中国科学院计算技术研究所CAD开放实验室

本文1996-03-04收到修改稿

接口,光线追踪算法可以把整个数据体都看成是局部的.

Cache效率是分布式共享缓冲区性能好坏的关键,考虑Cache本身的设计外,体数据的访问方式对Cache效率也产生影响.在光线追踪过程中,象素遍历方式对Cache相关性有较大影响.利用数据集的相关性,对体数据进行八叉树分类,减少不必要的远程数据访问,可以改变数据访问模式,从而改善Cache的性能,下节将着重讨论这个问题.

1 八叉树分类快速绘制算法

本节介绍一种基于八叉树的分类体绘制算法,对数据体进行简单的分类后,可改进传统的光线追踪方法,减少不必要的数据访问,加快绘制过程.

1.1 最大最小八叉树

利用数据集的相关性,在不影响图象质量的前提下,加快绘制过程,人们提出了许多方法.这些方法依赖于空间数据结构,对体元的透明度进行编码,避免数据体透明部分的计算.这些数据结构有八叉树[1,7]和k—d树[8]等.

文献[9]中提出了最大最小八叉树的一种结构.数据体对应八叉树,八叉树的根结点存放整个数据体中分类参数(如密度、梯度)的最大、最小值.在X,Y,Z3个方向分别对数据体二分,得到8个数据块,对应于根结点的8个子结点,在每个子结点中存放相应数据块的分类参数最大、最小值.继续对数据块二分,可以得到八叉树的下一级结点.最大最小八叉树的层次数由叶子结点的大小来决定.由于最大最小八叉树不依赖于视见参数和透明度转换函数,它可以在体数据被装入的同时进行预计算.而且,对一个数据体而言,八叉树的计算是一次性的,以后可以重复使用,这正是八叉树分类法的最大优点.

1.2 区域积分表

当确定透明度转换函数后,在绘制的前一刻,我们计算数据体分类的辅助数据结构:区域积分表.假设$f(p,q)$是在矩形区域内的离散网格点上定义的函数,那么函数f的区域积分表是一个二维数组S,数组元素$S(p,q)$由下式计算:

$$S(p,q)=\sum_{p'=0}^{p}\sum_{q'}^{q}f(p',q')$$

在文献[10]中给出了多维区域积分表的计算方法,它表明这个计算的时间开销是常量阶的.假设透明度转换函数是从多维标量场到一维标量场的映射,例如透明度可以是体数据密度值和梯度的函数.函数f把多维特征空间分成透明和非透明的区域,我们感兴趣的是非透明区域.如图1所示的区域积分,在区域R上对函数f积分,如果积分结果为零,则区域R为透明;如果积分结果等于区域R的面积,则R为不透明;否则,区域R为半透明.

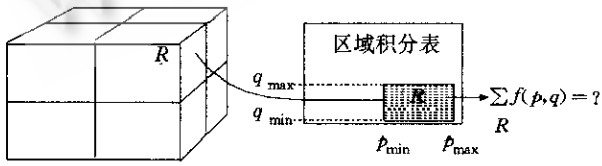

图1 最大最小八叉树与区域积分表

1.3 八叉树分类光线追踪算法

利用区域积分表,以深度优先的次序遍历最大最小八叉树,可以把八叉树的每个结点归结为3种状态之一:透明、不透明、半透明.在确定光线与数据体相交后,判断相交的光线段所跨越的数据体部分的状态,如果是透明的,则跳过这一部分.如果不透明,就按传统的光线投射方法积分这一段数据体.如果为半透明,则将光线段分成2段,分别进行类似的判断.由于光线贡献量的合成运算"over"具有结合性,因此分成2段分别计算后,再将部分结果合成,结果没有差别.

采用这种办法,以牺牲较小的存储(八叉树占用)来减少不必要的计算和数据访问,特别是那些对光线毫无贡献的体元.当光线离开数据体时,如果此时积累的不透明度尚未达到阈值,还可以对环境中的几何景物进行光线追踪的计算.

2 并行算法

在分布式计算环境下设计并行算法,要考虑计算任务分配、负荷平衡、数据分布等问题.

2.1 图象分割与对象分割

并行体绘制中常见的任务分割是对象分割与图象分割.在对象分割中,每个处理器分得体数据的一个子集,在这个子集上进行重采样.[5,11]在图象分割时,每个处理器负责计算整个图象的一部分[12],体数据必须在不同的处理器间"移动",尤其是当视点发生变化的时候.

Neumann[5]指出,对象分割比图象分割的数据通信量要少.在消息传递的多机系统中,由于通信需要操作系统介入,通信的开销大,此时对象分割比较有吸引力.但是,对象分割的第1个缺点是无法有效地提前中止光线的计算,第2个缺点是需要更复杂的同步机制.我们选择图象分割方式,在最终图象合成之前,没有同步要求,这适合于分布式环境的特点.

2.2 任务形状与分配单位

在分布式共享缓冲区机制下,要提高数据访问的相关性,我们有必要分析象素遍历的顺序,使得在Cache中的数据,在不久的将来被再度访问的可能性最大,减少远程数据的网络传输时间.

根据文献[13]的结果可知,象素扫描顺序为空间填充曲线方式时,由于空间填充曲线上的连续点列具有相关性,连续遍历的2个象素访问相同体数据的可能性大大增加.我们采用最简单的空间填充曲线—Hilbert曲线,如图2所示.因此任务形状就是图中所示的曲线,任务分配单位则是曲线上的折线段.

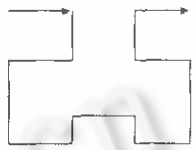

图2 Hilbert曲线

2.3 负荷平衡(Load Balancing)

减为少由负荷不平衡引起的计算损失.下面考虑3种方法:静态连续分配、静态交叉分配、动态分配.

静态连续分配时,整个Hilbert曲线被平均分割成固定的长度,每个处理器分得一段.静态交叉分配时,Hilbert曲线被分成数目较多的曲线段(多于处理器数),每个曲线段按Round—Robin的次序分配给处理器.在动态分配中,任务可以从1个处理器重新分配到另1个处理器上,从而达到较好的负荷平衡效果.我们采用的动态分配方法是分布式的任务队列和动态的任务窃取,实现了静态交叉分配和动态任务窃取相结合的方法,取得的效果最好.

2.4 体数据分布

采用图象分割方式时,每个处理器所需的体数据依赖于视点的选择.我们采用的数据分布策略基于分布式共享缓冲区.每个处理器的主存分为2部分,①局部数据区,②Cache.在初始化阶段,整个体数据被分成许多块,向处理器分配时可以按Round—Robin方式,每个处理器把分到的体数据块置于局部数据区中.在光线追踪过程中,当处理器(设为A)所需的数据不在局部数据区时,它先到Cache中查找,如果找到(称为命中),则返回该数据.如果未找到(称为失效),处理器A要到远地处理器取来数据,这个过程和操作系统中的缺页处理类似.我们采用的是LRU最近最少使用算法.Cache的效率是决定分布式共享缓冲区性能

的关键.

2.5 并行算法概要

并行算法中每个处理器的流程控制大致是:首先,处理器接收公用数据,建立必要的数据结构(如任务队列、八叉树等).然后,处理器从任务队列中取来任务,开始计算.如果队列为空,它要向其它处理器请求任务.当取到任务后,处理器开始光线追踪的计算,在计算过程中可能要通过分布式共享缓冲区访问远程数据.每个处理器在向其它处理器发出请求的同时,还必须响应其它处理器的请求.当所有的任务队列均为空时,等待同步.然后合成并显示生成的图象.

3 实现与结果分析

我们分别在曙光1000系统和工作站网络上实现了上述算法.曙光1000系统是分布式存储MPP多处理机,结点通信采用消息传递,通信用NXLIB库实现.工作站网络是由最多8台SGI Indigo构成,通信用PVM实现.

我们用2个数据集来测试上述算法.体数据分别为CT人脑(大小为256×256×132)和CT胸骨(大小为128×128×197),测试时所生成的图象分别如图3、图4所示,图象大小为512×512.

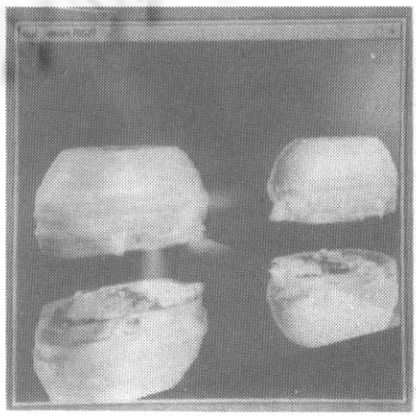

图3 CT人脑图象

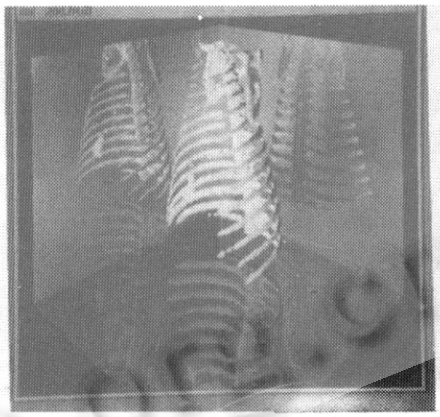

图4 CT胸骨图象

3.1 八叉树分类法对绘制时间的影响

表1是在1台SGI Indigo上绘制图3、图4所花的时间.由表中可见,随着八叉树叶子结点的大小发生变化,绘制时间在作相应的变化.在八叉树叶子结点为8×8×8时,绘制速度最快.叶子结点为256×256×256所对应的情况是未经八叉树分类的结果.当叶子结点为8×8×8时,每个结点对应512个体元,在递归深度和分类精确性之间达到了较好的平衡.

表1 绘制时间随八叉树叶子结点大小变化表

叶子结点大小(K=)	2	4	8	16	32	64	128	256
CT人脑绘制时间(s)	309	294	286	287	300	383	584	568
CT胸骨绘制时间(s)	349	337	334	413	454	508	500	488

3.2 并行绘制时间

表2是我们在曙光1000上绘制图3时,随着结点数的增多,绘制时间发生相应的变化.其它主要参数:Cache大小/数据大小=20%,八叉树叶子结点大小为8×8×8.

表3是我们在SGI工作站网络上绘制图4时,随着工作站数目的增多,绘制时间的变化情况,其它主要参数:Cache大小/数据大小=20%,八叉树叶子结点大小为8×8×8.

表 2　曙光 1000 上的绘制时间

计算结点数目	1	2	4	6	8	10	12	14	16	18	20
绘制时间(s)	2 027	1 053	545	371	287	233	200	173	151	129	120

表 3　SGI 工作站网络上的绘制时间

工作站数目	1	2	3	4	5	6	7	8
绘制时间(s)	345	193	134	104	92	80	73	66

图 5、图 6 是我们依据表 2、表 3 的结果画出的加速比曲线.算法的可扩展性是比较理想的.但是,当结点数目继续增加时,加速比开始出现非线性的变化.究其原因,由于我们采用体数据的均匀分布方式,结点数目越多,每个处理器内的局部数据越少,远程数据访问的概率增大,导致网络传送量的增加,此时网络传输在整个计算中占重要部分,Cache 已不能完全屏蔽掉远程访问的延迟,因而速度的加快不再和处理器的数目成比例.

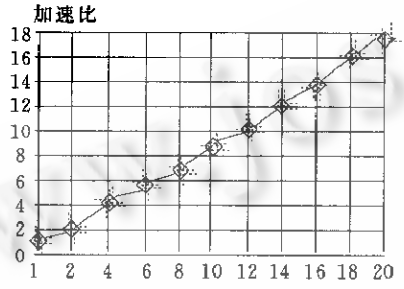

图5 曙光1000加速比曲线

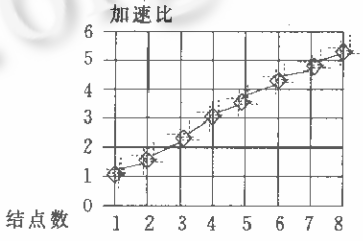

图6 工作站网络加速比曲线

3.3　Cache 大小对算法的影响

Cache 的效率可以通过由 Cache 失效而引起的体数据网络传送量(字节数)来表示.改变局部存储器中 Cache 的大小,数据网络传送量会发生明显的变化,对绘制时间也有明显的影响.表 4 是我们在曙光 1000 上对 CT 人脑测定的数据传送量及相应的绘制时间,用到的计算结点为 12 个,八叉树叶子大小为 8×8×8.

表 4　曙光 1000 上 Cache 大小对绘制时间的影响

Cache 大小/数据大小	2%	3%	6%	10%	20%	30%	40%	50%
数据传送量(MB)	3 935	211.7	62.8	53.1	44.2	41.7	41.1	40.7
绘制时间(s)	814	237	207	205	203	201	199	199

我们注意到,当 Cache 太小(例如不到数据集大小的 3%)时,数据传送量急剧上升,绘制时间明显增长,这就是由于 Cache 缺页替换 LRU 算法所引起的"抖动"现象.

3.4　八叉树分类对 Cache 效率的影响

当 Cache 容量(相对于体数据的大小)越小时,八叉树分类的优越性越明显.也就是说,Cache 的容量(受机器内存的限制)只能占整个体数据的一小部分,如 10%甚至更少.下面我们在 Cache 大小远远小于数据大小时,改变八叉树叶子结点的大小,观测网络数据通信量以及绘制时间的变化.表 5 是用 8 台工作站对 CT 胸骨测定的结果.

表 5　工作站网络上数据传送量随八叉树叶子结点大小变化表(Cache/Data=10%)

叶子结点 K×K×K(K=)	2	4	8	16	32	64	128	256
网络数据传送量(MB)	1 756	1 729	1 715	1 719	1 924	1 987	2 027	2 050
绘制时间(s)	249	246	237	249	307	321	324	329

从上表中可以看出,当 Cache 容量相对于体数据较小时,随着八叉树分类的精度不同,网络传输的数据量有明显的不同,但这都比未采用八叉树分类时传送的数据量要小.而且当叶子结点大小取 8×8×8 时,Cache 效率最佳.再细分下去,网络传送量的变化不大,但递归

深度的增加，使绘制时间也略微增加.

4 结束语

本文利用分布式共享缓冲区概念，在消息传递的分布式环境中，实现了并行体绘制的算法. 这个算法在对数据体光线投射的同时，还可以对数据体周围的简单几何体进行光线追踪. 测试结果表明，八叉树分类与 Hilbert 象素遍历方式相结合，简化了并行任务的分配与管理，明显地提高了 Cache 的效率，使绘制时间大大缩短.

参考文献

1 Laur D, Hanrahan P. Hierarchical splatting: a progressive refinement algorithm for volume rendering. In: Computer Graphics (SIGGRAPH'91), July 1991, **25**: 285～288.

2 Terry S Yoo, Neumann U, Henry Fuchs *et al*. Direct visualization of volume data. IEEE Computer Graphics & Applications, July 1992, **12(4)**: 63～71.

3 Craig M Wittenbrink, Arun K Somani. Permutation warping for data parallel volume rendering. In: Proceedings of the 1993 Parallel Rendering Symposium, San Jose, Oct. 1993. 57～60.

4 William M Hsu. Segmented ray casting for data parallel volume rendering. In: Proceedings of the 1993 Parallel Rendering Symposium, San Jose, Oct. 1993. 7～14.

5 Neumann U. Parallel volume rendering algorithm performance on mesh—connected multicomputers. In: Proceedings of the 1993 Parallel Rendering Symposium, San Jose, Oct. 1993. 97～104.

6 Badouel D, Bouatouch, Priol T. Distributing data and control for ray tracing in parallel. IEEE Computer, July 1994. 55～59.

7 Levoy M. Efficient ray tracing of volume data. ACM Transactions on Graphics, 1990, **9(3)**: 245～261.

8 Subramanian, Fussel D S. Applying space subdivision techniques to volume rendering. In: Proceedings of Visualization '90. San Francisco, Oct. 1990. 150～159.

9 Wilhelms, Jane, Gelder. Octree for faster isoface generation. Computer Graphics, 1990, **24(5)**: 57～62.

10 Crow, Franklin. Summed—area tables for texture mapping. Computer Graphics, 1984, **18(3)**: 207～212.

11 Kwan—Liu Ma, Painter J S, Hansen *et al*. A data distributed, parallel algorithm for ray—traced volume rendering. In: Proceedings of the 1993 Parallel Rendering Symposium, San Jose, Oct. 1993. 15～22.

12 Brain Corrie, Mackerras P. Parallel volume rendering and data coherence. In: Proceedings of the 1993 Parallel Rendering Symposium, San Jose, Oct. 1993. 23～26.

13 Zhang Hansong, Liu Shenquan. Order of pixel traversal and parallel volume ray—tracing on the distributed shared volume buffer. In: Eurographics Workshop on Visualization '95, 1995.

PARALLEL VOLUME RENDERING ALGORITHM FOR DISTRIBUTED COMPUTING ENVIRONMENT

Yu Shengming Li Hua Liu Shenquan

(*CAD Laboratory Institute of Computing Technology The Chinese Academy of Sciences Beijing* 100080)

Abstract This paper presents a distributed shared buffer paradigm based on message passing in distributed computing environment. According to this paradigm, this paper implements a parallel volume rendering algorithm. In order to improve the efficiency of the cache, the parallel algorithm take advantage of octree classification to increase local coherence of the cache in data space. In image space, the parallel algorithm use the Hilbert pixel traversal order to increase the temporal coherence of the cache. The experiments demonstrate good speedup and salability of this approach.

Key words Parallel volume rendering, octree, pixel traversal, ray tracing.