基于保留模式的 In-the-Core 并行超大数据量图形绘制*

彭浩宇+, 金哲凡, 石教英

(浙江大学 CAD&CG 国家重点实验室,浙江 杭州 310027)

An In-the-Core Parallel Graphics Rendering System for Extreme Large Data Sets Based on Retained-Mode

PENG Hao-Yu+, JIN Zhe-Fan, SHI Jiao-Ying

(State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou 310027, China)

+ Corresponding author: Phn: +86-571-87951045, Fax: +86-571-87951045, E-mail: phy@cad.zju.edu.cn, http://www.zju.edu.cn

Received 2004-04-05; Accepted 2004-07-05

Peng HY, Jin ZF, Shi JY. An in-the-core parallel graphics rendering system for extreme large data sets based on retained-mode. *Journal of Software*, 2004,15(Suppl.):222~229.

Abstract: We present a hybrid sort-first and sort-last parallel rendering system based on retained-mode, called In-the-Core, which can render extreme large models on PC clusters. Traditional parallel systems have to issue all primitive-instructions when draw a new frame, or to copy whole scene on each node of cluster in advance. The approach can utilize full main memory capacity of cluster and lower the traffic of network. The main method is to partition a large model into smaller structures of elementary particle and distributes them into the main memory of rendering nodes; meanwhile the load-balance is kept by view-depend partition algorithm with overhead feedback. Experiments on a cluster of 8 nodes show that the In-the-Core system can render models about 30~40M triangles at interactive frame-rate. It shows that a cluster of inexpensive PCs is an attractive alternative to those high-end graphics workstations.

Key words: parallel rendering; In-the-Core; elementary particle; overhead feedback; view-depend redistribution

摘 要: 立即模式图形并行绘制系统多采用 sort-first 算法,每绘制一帧都要求重发所有图元绘制指令,容易造成网络堵塞.保留模式并行绘制系统多要求集群机每个节点保留一份完整的数据拷贝,能改善网络堵塞,它可采用 sort-first 和 sort-last 并行绘制算法.保留模式并行绘制系统每个节点保留一份完整数据拷贝,使模型规模受单个节点内存容量限制.本文设计的称作 In-the-Core 的系统是混合 sort-first 和 sort-last 型保留模式并行绘制系统,将模型数据剖分并分布到绘制节点以充分利用集群机内存总量,用时间反馈的视点相关重分布算法保持负载平衡,有效减少了网络通信量.经验证此系统在绘制超大数据量模型时表现优秀.

关键词: 并行绘制;In-the-Core;基本粒子;时间反馈;视点相关重分布

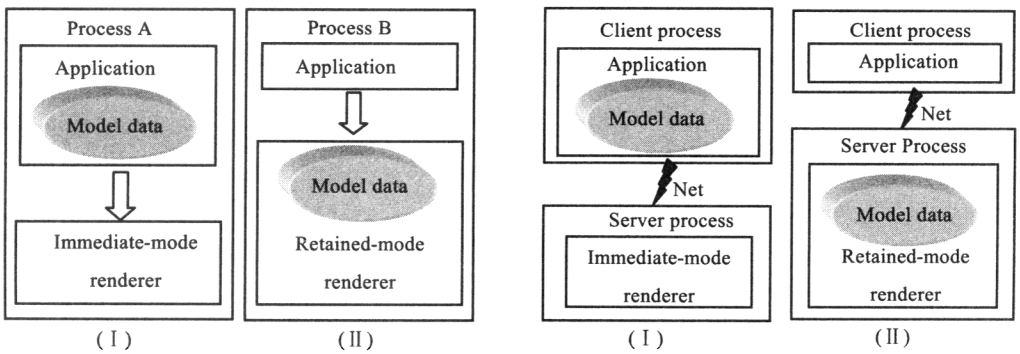
* Supported by the National Grand Fundamental Research 973 Program of China under Grant No.2002CB312105 (国家重点基础研究发展规划(973))

作者简介: 彭浩宇(1978－),男,湖南益阳人,博士生,主要研究领域为计算机图形分布式绘制;金哲凡(1974－),男,博士,讲师,主要研究领域为计算机图形分布式绘制;石教英(1937－),男,教授,博士生导师,主要研究领域为计算机图形学,计算机辅助设计.

计算机科技的持续发展带来了硬件性能的飞速提高,图形卡的处理速度每六个月就提高一倍,发展速度突破了摩尔定律[1].但不管图形硬件如何发展,绘制超大规模复杂模型或场景的应用需求如医学模型、沉浸式虚拟环境、气象预报、数字地图等总是超出了单个计算机的能力.通常人们采用昂贵的高端专业图形并行工作站来满足要求,如 SGI 的 Infinite Reality[2,3]和 UNC 的 PixelFlow[4].这些设备的主要缺点是价格昂贵,难以扩展和升级.近年来,人们开始研究采用便宜但是功能日益强大的个人计算机,依靠高速网络组建计算机集群机来代替专业图形工作站,取得了良好的成效.高性价比、升级方便、使用灵活且易于扩展是集群机的主要优点.

图形绘制流水线一般包括两个基本的处理过程:几何处理和光栅化.几何图元需要按照一定的任务划分策略分配到不同的绘制节点,这个过程称为归属判断(sort). Molnar[5]等人根据几何图元归属判断发生的时机,将并行算法归为三类:sort-first,sort-middle,sort-last.sort-first 类并行算法的图元归属判断在几何处理之前进行,sort-middle 的归属判断发生在几何处理之后、光栅化之前,而 sort-last 是发生在光栅化之后.在 sort-first 类并行系统中,最终的显示屏幕被划分为多个区域分配给绘制节点,绘制每一帧前先根据几何图元在显示屏幕上的投影确认图元将由哪一个绘制节点负责处理,将图元发送到绘制节点,绘制完成后将所有绘制节点的结果拼接成最终显示图像.sort-first 的优点是各条流水线完整而独立,且相对其他两种方式需要的通信带宽较小,故特别有利于构建集群式并行绘制系统.图元归属判断是此类并行系统的主要开销和瓶颈所在.sort-middle 类并行算法要求得到绘制流水线几何处理后、光栅化之前的中间结果进行重分布,而计算机集群机所采用的商业图形卡一般将几何处理和光栅化功能集成在一起,很难得到流水线的中间结果,这限制了 sort-middle 算法在集群机上的应用.sort-last 类并行系统中的绘制节点得到整个场景的部分图元,每个绘制节点都输出整个显示屏幕大小的图像,然后根据他们的深度信息进行全屏幕深度合成得到最终的显示结果.sort-last 最大的优点是简明,图元的不均匀分布引起的负载失衡较小,像素的传输和深度合成是 sort-last 并行系统的主要开销和瓶颈所在.

计算机上的应用程序通过 API 操纵图形绘制器进行绘制功能.单机上的图形绘制 API 可以分为立即模式和保留模式.立即模式的 API 如 OpenGL,图形绘制器不保存任何图元数据的信息,绘制新帧时需要客户程序重新发送图元数据.而保留模式的 API,如 Direct3d-Retain Mode,绘制器保留了图元的信息,绘制时应用程序只需发出绘制命令而无需发送图元数据.应用程序、图元数据和绘制器的关系如图 1(a).



(Ⅰ) Immediate-mode APIs on single computer

(Ⅱ) Retained-mode APIs on single computer

(Ⅰ) Immediate-mode parallel architecture

(Ⅱ) Retained-mode parallel architecture

(a) (b)

图 1 立即模式与保留模式

在并行绘制系统中,应用程序通过网络操纵远程计算机上的图形绘制器进行绘制工作,单机上立即模式和保留模式 API 可引申为并行系统中两类不同的体系结构.如图 1(b)所示.立即模式体系结构的并行系统图元数据由应用程序控制,每绘制一帧,都需要应用程序重新发出所有的数据指令;而保留模式是由图形绘制器控制,可以避免数据频繁传送造成网络堵塞,并可以有效的利用帧间连贯性(frame-to-frame coherence)提高绘制效率.

当前,还没有一个并行绘制系统在普通集群机上完美的绘制超大数据量模型,立即模式的 sort-first 系统需要太大的网络带宽,保留模式的 sort-first 和 sort-last 系统不能充分利用节点内存总量.本文提出了 In-the-Core

系统是一种混合 sort-first、sort-last 的保留模式并行绘制系统构架,它通过将模型剖分到绘制节点充分利用集群机整体内存总量以容纳大数据量模型,通过视点相关的模型剖分算法和基于时间反馈的数据调整方案,达到较好负载平衡的同时减少网络通讯量.原型系统可以实时高效地绘制超大规模模型.

1 相关工作

Humphreys 等人开发的 WireGL[6]是较早的采用 sort-first 算法的立即模式集群机并行绘制系统.在 WireGL 的基础上 Humphreys 等人继续开发出了 Chromium[7] 立即模式系统,它将诸如数据剖分、绘制处理、深度合成等基本图形 API 集成到独立的模块里,称为 SPU(Stream Processing Unit).SPU 之间是相互可替代的并行关系,或者是功能前后连续的串行关系,根据需要可灵活地构建包括 sort-first 和 sort-last 在内的多种并行体系结构.WireGL 和 Chromium 将整个模型数据放在客户机上,使模型大小受到客户机主存大小的限制,而且频繁的图元传送给网络带来了沉重的压力,影响了性能的提升.

Samanta[8]等人先开发了基于集群机的保留模式 sort-first 并行系统,设计了开销预测算法预测几何数据在本地和远程节点上的绘制开销,以预测结果为基础将屏幕动态的划分和分配以获得最优化的绘制效率.该预测算法是个复杂度大而且容易失准.为了减少网络开销,整个模型数据被完整的拷贝到每一个绘制节点,这限制了模型的规模.

随后 Samanta[9]等人设计了保留模式的混合 sort-first 和 sort-last 并行算法,根据 3 维模型在 2 维屏幕上的投影进行视点相关的动态剖分,在维持负载平衡的同时减少因绘制节点所属屏幕区域间重叠造成的像素传输,减轻像素合成阶段的负荷.视点相关的动态剖分算法是不错的思路,但是 Samanta 设计得过于繁琐,使剖分过程容易成为系统瓶颈.本文对它做了简化改进,提高了运行效率.这个混合系统同样在每个节点上复制完全的数据拷贝,限制了模型规模.

在最近的工作中,Samanta[10]开始着手研究数据重复拷贝的问题.为了减少因动态分配造成的网络通信,同时又不将整个模型拷贝到每个节点,先离线地将模型生成多分辨率的层次结构对象,然后将每个对象复制到 K 个节点上,K 远小于整个集群机节点数目 N.在绘制阶段,通过视点相关的剖分算法确定绘制哪些对象,然后从保存这些对象的 K 个节点中选出一个节点进行绘制,绘制完成后的图像以 sort-last 算法深度合成最终显示的图像.他们的原型系统可以每秒绘制 30M~40M 三角面片,帧速达到交互性要求.这个方案要求前处理,负载容易不平衡,并且仍然不能充分利用集群机的内存总量.

Wagner 和 Klosowaki[11]等人的 Out-of-Core 集群并行绘制系统有一个预处理模型的过程,将模型数据按照八叉树剖分并以 1M 大小的文件组织保存.除了叶节点文件保存模型数据外,上级节点包括根节点文件都只保存下一层次节点包含的空间状态.系统运行时有 3 个线程工作:绘制线程,读取线程,前瞻线程.绘制线程根据当前视点确定所要绘制的八叉树节点,请求读取线程从磁盘上装载相应的文件获得几何数据.为了防止 I/O 堵塞,前瞻线程预测下一帧可能要绘制的节点,将相应文件的数据预先读入缓冲区供读取线程使用.通过以上方式,Wagner 宣称 Out-of-Core 并行系统理论上可以绘制接近集群机硬盘总容量大小的模型.原型系统在绘制 13M 三角面片的模型时帧速可以达到每秒 10.8 帧.Out-of-Core 系统在绘制超大数据量模型的能力上表现突出,只是因为频繁的 I/O 处理,速度上受到限制.当前的计算机主存容量一般在 512M 以上,一个集群机内存总量可以达到数十个 G,可以满足绘制超大规模场景的要求,将模型保存在硬盘上并非必要.

2 模型的读取与剖分

客户节点首先读取模型文件头信息,将模型的空间范围沿坐标轴分为 N 等份(N 为绘制节点数),每个绘制节点负责接收一个空间范围的数据.系统构造 N 个对应的三角面片数据缓冲区.然后顺序读取三角面片,每读取一个就根据它的顶点位置放入其所在空间对应的缓冲区.若某个缓冲区满了就通过一个后台线程发送到对应的绘制节点,并清空此缓冲区.当某个绘制节点收到的三角面片特别多可能超出内存容量时,将其对应空间的三角面片转到其他数据较少的节点上.图 2 是这个过程的示意图.

这样模型就被剖分为空间上紧凑的子集分送到 N 个绘制节点.绘制节点收到所有三角面片后,以一定的粒度构造基本粒子.基本粒子是系统构造的包含部分图元的微粒,是系统所有图元操作的基本单位.基本粒子也包括信息头与数据,如图 3 所示.客户节点维护所有绘制节点上基本粒子的信息头,为后续阶段进行视点相关的动态重分布计算做准备.系统对数据的操作以基本粒子而非单个三角面片为单位,可以减少处理的时间.

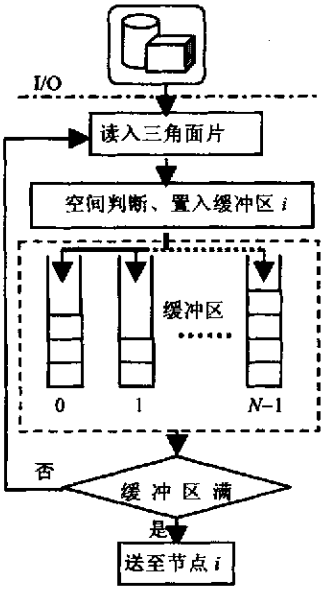


图 2 模型剖分流程

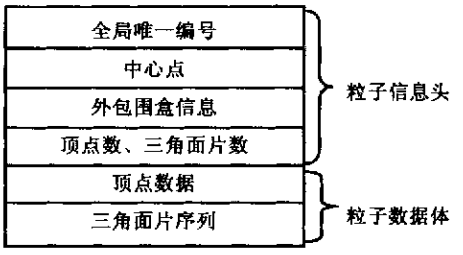


图 3 基本粒子结构

3 系统结构与流程

系统有 4 种功能节点:客户节点,绘制节点,合成节点,显示节点.他们的逻辑关系见图 4.

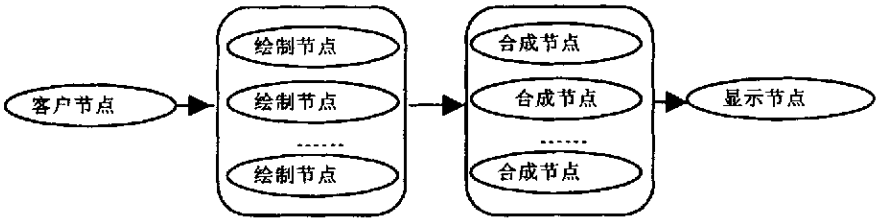


图 4 系统结构图

其功能描述如下:

客户节点:分发模型数据,并跟踪基本粒子在绘制节点的分布状态;根据绘制时间反馈进行动态视点相关的重分布计算,决定基本粒子的绘制节点间如何调整;发出其他操作指令.

绘制节点:根据保存的模型几何数据构造基本粒子并将粒子信息反馈给客户节点,同时根据客户节点指令进行基本粒子的调整;完成绘制并输出像素到合成节点.

合成节点:完成图像的拼接与深度合成,输出最终结果到显示节点.

显示节点:显示最终图像.

其工作流程可简单描述如下:

Step 1: 客户节点读取模型文件,将模型剖分为 N 部分,发送到 N 个绘制节点.

Step 2: 绘制节点获取模型数据,按一定粒度构造基本粒子,构造完成后将所有基本粒子信息返回客户节点.

Step 3: 绘制节点根据当前视点将基本粒子中心点投影到显示屏幕上,若投影点在屏幕范围内或者离边界

不超过一定距离,则判定此基本粒子将被绘制.客户节点得到所有绘制节点的基本粒子中心点投影情况将显示屏幕分为 N 部分(每部分包含有相同数目的基本粒子),然后根据 step 2 得到的粒子分布信息通知各个绘制节点进行数据调整.

Step 4: 绘制节点根据指令从其他节点得到属于自己的基本粒子,同时将需由其他节点进行绘制的基本粒子发送到相应节点,这个过程称为数据调整.完成调整后各个绘制节点只拥有自己需要绘制的那部分基本粒子,根据这部分数据构造其拥有粒子的外包围盒.最后通知客户节点调整已经完成.

Step 5: 客户节点发出绘制命令,绘制节点完成绘制并将绘制时间反馈给客户节点.随即绘制节点,根据外包围盒信息得到图像在显示屏上的覆盖区域,从帧缓存中读取此有效区域内像素的颜色和深度信息,压缩后发送给合成节点.

Step 6: 客户节点比较绘制节点返回的绘制开销,若差值小于允许范围 t,则判断负载平衡.若超出允许范围 t,则重复 Step 3 和 Step 4.同时,合成节点解压缩像素信息包完成拼接合成最终显示图像并将之发送到显示节点.

Step 7: 显示节点显示图像.转 step 5.

前四步是系统绘制第一帧前必须要完成的步骤,包括模型剖分、重分布计算和数据调整.在开始正常绘制后,帧间连贯性可以保证在大多数情况下系统无需进行动态重分布计算和数据调整,系统可以在第 5 步和第 7 步之间流畅的运行.

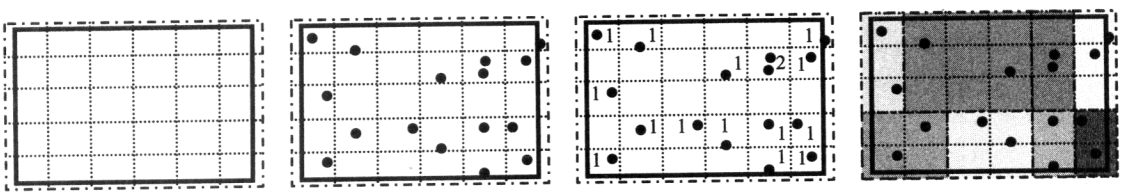
当进行图像缩放或视点变化等必须进行数据调整的操作时,由于重分布计算和图像的拼接合成是在不同的功能节点上进行,能够同时处理以节约时间.数据调整是在绘制节点间通过点对点通讯进行,不再由客户节点单独完成,减小了客户节点成为系统瓶颈的可能性.

为了减少重分布计算和数据调整算法的开销,流程的实现上做了细节处理,如用基本粒子代替单个图元,用粒子中心点代表整个粒子,这些细节都可以有效的降低算法的复杂度.

4 基于时间反馈的动态重分布算法

已有的一些采用动态视点相关的屏幕划分算法的系统,每绘制一帧就进行一次重分布计算调整数据.但帧间连贯性的存在表明节点在帧与帧之间需绘制的数据基本保持不变,为了微小的数据调整而频繁的进行重分布并不划算.因此本文采用时间开销反馈来确定重分布计算的时机.绘制节点绘制每一帧时都会记录绘制的时间开销,客户节点分析所有绘制节点的时间开销来判断负载平衡情况,确定是否需要重分布计算.

本文将模型剖分为小的基本粒子,不过为了简化计算,以粒子中心点代表整个粒子.当中心点投影在屏幕边界内或者在边界外但不超过一定的距离(此距离由最大的粒子半径决定)时,均认为此粒子落入有效区域而将被绘制.将有效区域分成 $n\times n$ 小块,逐个确定中心点投影位置,落入第 i 个小块则将 i 小块的权值加 1.计算完后根据各个小块的权值将屏幕划分为 N 部分,使每部分包含小块权值总和基本相等.以八个绘制节点为例,划分过程如图 5 所示,其中粗实线方框表示屏幕空间,外层的虚线方框表示有效区域,粗黑点表示基本粒子中心点投影数字表示小块的权值,不同颜色区分各个节点负责的屏幕区域.



(1) 有效区域划分为 n×n 个小块　(2) 确定基本粒子中心点在区域内的投影　(3) 根据中心点投影计算小块的权值　(4) 根据权值将有效区域分为 8 部分

图 5 屏幕划分过程

屏幕划分之后,根据基本粒子中心点在各个区域的分布情况,客户节点可以得到新的所有绘制节点需要绘制的基本粒子总表,以此新表替换旧表,同时将它发送到所有绘制节点.绘制节点根据此表对照自己拥有的基本

粒子序列,即可判断该从哪些节点得到哪些基本粒子,将不属于自己的基本粒子向哪些节点发送.绘制节点间以点对点通讯方式完成数据调整后,再进行后续帧的绘制.

5 有效区域的压缩合成

原型系统的屏幕像素合成可以用二叉树回溯的方式进行,这可以使合成过程不再集中于一个合成节点,过程如图 6 所示.每次屏幕重划分数据调整之后,绘制节点负责屏幕上某个区域内的模型绘制,虽然这个区域的分布状态会随着模型或视点的变化(如旋转)而变动,但在下一次屏幕重分布以前,绘制节点拥有的数据是不变化的,所以图示中用屏幕划分后的初始状态表示节点负责区域.图左为屏幕划分图,粗实线方框表示屏幕空间,外层的虚线方框表示有效区域,数字表示绘制节点负责区域.图右表示图像合成过程,带数字的圆框代表合成节点,粗实箭头代表像素传输方向,虚箭头表示过程进行方向,显示节点完成最终的拼接合成并显示.

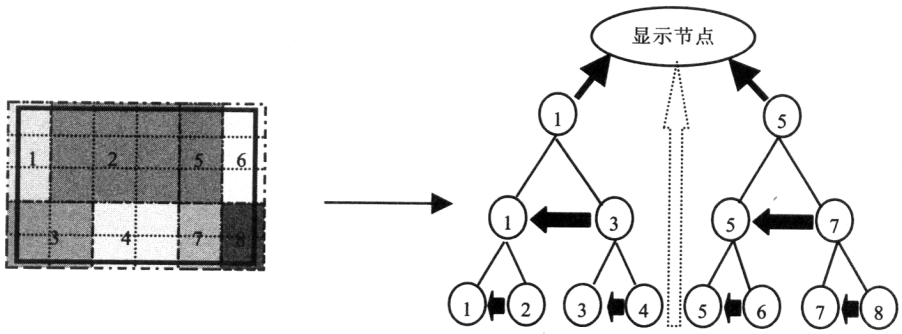


图 6 图像拼接合成流程示意图

像素传输采取了特殊处理以减少数据传输量,以缓解网络带宽的压力.在系统的模型剖分与基本粒子构造阶段,每个绘制节点都根据其拥有的基本粒子得到它们的外包围盒信息,以后每次数据调整后绘制节点都会根据新的数据更新此外包围盒.在合成阶段,绘制节点根据此外包围盒的投影得到一个有效区域,模型图像必集中于此区域内.因此只需发送帧缓存中有效区域内的像素点的颜色和深度信息,减少了发送的数据量.

6 试验结果分析

运行 In-the-core 系统的集群机有八个节点,操作系统是 Windows2000 Professional,采用 Intel 奔腾 4CPU 2.4GHz,512M 内存,节点间以千兆网络连接.测试用的模型见表 1.

表 1 模型对照

Models	Amount of triangles	Memory needed
Cow	5804	417K
Hand	654666	41.136M
Buddha	1087716	78.316M
Blade	1765388	127.108M
Dragons	7165246	515.898M
Lucy	28055742	2020.01M

可以看到 Dragons 和 Lucy 已经超出了单个节点内存容量.以模型 Lucy 为例,它有 28M 三角面片,为了剖分基本粒子和数据调整的方便,模型在内存中都是以顶点序列表示.一个面片 3 个顶点,每个顶点包括位置和法向量 6 个浮点数,浮点数占用 4 个字节,计算出 Lucy 占用内存 M 为 28M×3×6×4=2020M,无法在单个绘制节点完成绘制.In-the-core 并行系统可以将它分到 8 个节点进行绘制,结果如图 7 所示.左图是八个绘制节点绘制的图像,右图是显示节点显示的合成图像.

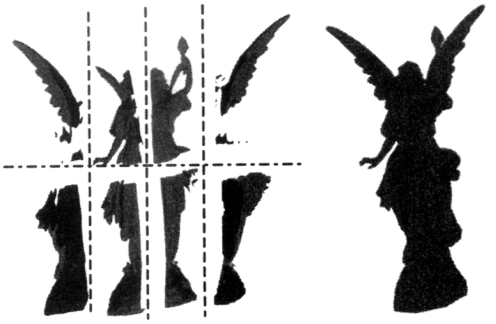


图 7 Lucy 绘制效果

让上述 6 个模型在 1024×768 的分辨率下绕 y 轴旋转 360 度,每转一度绘制一帧,计算系统的绘制时间(单位:秒)和加速比(单机/8pc),结果见表 2.加速比是一台机器与 8 个节点绘制时间比.其中 Dragons 和 Lucy 的单机测试是在更高配置的 pc 上完成的.

表 2 模型加速比

Models	1 pc	2 pc	4pc	6 pc	8pc	Speedup-ratio
Cow	6.12	12.31	16.11	18.14	25.31	0.2418
Hand	41.041	51.32	58.684	60.02	62.21	0.6597
Buddha	66.67	60.71	65.991	66.44	68.516	0.973
Blade	102.4	98.41	103.96	104.53	106.64	0.96
Dragons	783.14	601.89	321.825	289.14	231.345	3.385
Lucy	1786.85	1153.69	481.189	402.31	304.263	5.872

从表 2 发现,系统在绘制小模型的时候不但不能提高速度,反而会增加绘制时间,而在绘制大模型如 Dragons 和 Lucy 的时候,能明显提高绘制速度.这是因为数据调整和图像合成等算法本身需要占用一定的开销.随着模型增大,这个开销占模型绘制总开销的比重越小,加速比就越大.

以上是在模型充满整个屏幕的情况下旋转 360 度的测试结果,如果将模型缩小一半,最终显示的图像变小,合成所需的像素减少,像素传输和合成的速度加快,系统绘制速度应该会有提高.依照表 2 在 8 个节点旋转 360 度的实验过程,只将模型缩小一半,测试结果见表 3.

表 3 完整尺寸与缩小尺寸的模型绘制时间对比

Models	Whole-size	Shrunken-size	Speedup-ratio
Cow	25.31	11.31	2.23
Hand	62.21	30.14	2.064
Buddha	68.516	35.5	1.93
Blade	106.64	38.1554	2.79
Dragons	231.345	138.874	1.665
Lucy	304.263	231.63	1.313

从表 3 可以看出,模型缩小后都能明显提高绘制速度,这反映了 In-the-core 并行系统中像素传输和合成算法有很大的改进空间.

前面提到根据时间反馈的重分布计算在绘制节点间进行基本粒子调整有利于达到负载平衡,从而提高系统性能.在八个绘制节点下将模型从屏幕中心平移出屏幕范围,再平移回屏幕中心,测试系统在进行数据调整和不进行数据调整的绘制开销对比,结果见表 4.

可以看出,当模型越大基本粒子数目越多时,负载不平衡对系统性能影响更大,相应的数据调整的效果就越明显.

表 4 数据调整与不调整的比较

Models	Amount of particles	Adjust-ratio	Data-adjust (s)	No data-adjust (s)	Speedup-ratio
Cow	64	7.1%	6.32	5.36	0.84
Hand	1000	11.3%	17.31	14.24	0.82
Buddha	8000	18.53%	20.83	18.39	0.88
Blade	8000	19.6%	34.2	31.24	0.916
Dragons	27000	24.43%	79.14	91.46	1.15
Lucy	125000	57.56%	103.06	158.42	1.537

7 结 论

In-the-core 系统是基于保留模式的混合 sort-first 和 sort-last 的并行绘制系统,它将模型数据按照空间上最紧凑的原则剖分并且保存到集群机绘制节点的内存中,使系统能够充分利用集群机内存总量以满足超大数据量模型绘制的要求.将模型进一步细剖分为基本粒子作为算法处理的基本单位以简化计算,用绘制时间反馈决定重分布计算的时机能够避免不必要的数据调整,降低网络流量.客户节点只发出操作命令,模型数据调整和大部分算法计算分散在绘制节点进行,减低了客户端成为瓶颈的风险并有利于系统性能的提高.经实验验证表明,在 8 个绘制节点的集群机上系统每秒种可以绘制 30M~50M 三角面片,加速比达到 6 倍以上.

下一步的工作包括:进一步完善 In-the-core 系统,使它支持更复杂的包含纹理的数据模型;改进数据调整策略、采用效率更高的像素传输方式比如压缩像素,以减少网络传输量;设计更合理的合成策略来缓解图像合成的瓶颈.

致谢 在此,我们向对本文的工作给予支持和建议的同行,尤其是浙江大学 CAD&CG 实验室的林海副研究员,并行绘制研究小组的李超、王总辉、刘真等同学表示感谢.

References:

[1] Yang J. AnyGL: A lager scale hybrid distributed graphics system [Ph.D. Thesis]. Hangzhou: Zhejiang University, 2002 (in Chinese with English abstract).

[2] Akeley K. Reality engine graphics. In: Computer Graphics Proc., Annual Conf. Series of SIGGRAPH'93. Anaheim: ACM Press/Addison-Wesley Publishing Co., 1993. 109~116.

[3] John SM, Daniel RB, David LD, Christopher JM. InfiniteReality: A real-time graphics system. In: Proc. of SIGGRAPH'97. Los Angeles: ACM Press/Addison-Wesley Publishing Co., 1997. 293~302.

[4] Marc O, Anselmo L. A shading language on graphics hardware: The pixelFlow shading system. In: Computer Graphics Proc., Annual Conf. Series of SIGGRAPH'98. Orlando Florida: ACM Press/Addison-Wesley Publishing Co., 1998. 159~168.

[5] Steven Molnar, Michael C, David E, Henry F. A sorting classification of parallel rendering. IEEE Computer Graphics and Applications, 1994,14(4):23~32.

[6] Greg Humphreys, Matthew E, Ian B, Gordon S, Matthew E, Pat H. WireGL: A scalable graphics system for clusters. In: Computer Graphics Proc., Annual Conf. Series of SIGGRAPH 2001. Los Angeles: ACM Press/Addison-Wesley Publishing Co., 2001. 129~140.

[7] Greg Humphreys, Mike H, Ren Ng, Randall F, Sean A, Peter DK, James TK. Chromium: A stream processing framework for interactive rendering on clusters. In: Computer Graphics Proc., Annual Conf. Series of SIGGRAPH'2002. San Antonio: ACM Press/Addison-Wesley Publishing Co., 2002. 693~702.

[8] Rudrajit Samanta, Zheng JN, Thomas F, Li K, Jaswinder PS. Load balancing for multi-projector rendering systems. In: Proc. of the SIGGRAPH/Eurographics Workshop on Graphics Hardware. Los Angeles: ACM Press/Addison-Wesley Publishing Co., 1999. 107~116

[9] Rudrajit S, Thomas F, Li K, Jaswinder PS. Sort-First parallel rendering with a cluster of PCs. In: Technical Sketch at SIGGRAPH 2000. New Orleans, 2000. 1~3. http://www.cs.princeton.edu/omnimedia/papers/sketch00.pdf

[10] Samanta R, Funkhouser T, Li K. Parallel rendering with k-way replication, In: IEEE Symp. on Parallel and Large-Data Visualization and Graphics. San Diego, 2001. http://www.cs.princeton.edu/omnimedia/papers/pvg01.pdf

[11] Wagner T, James TK, Claudio TS. Out-of-Core sort-first parallel rendering for cluster-based tiled displays. In: Proc. of PGV 2002 (4th Eurographics Workshop on Parallel Graphics and Visualization). Blaubeuren, Germany: ACM Press/Addison-Wesley Publishing Co., 2002. 89~96.

附中文参考文献:

[1] 杨建. AnyGL:一个大规模混合分布图形系统[博士学位论文].杭州:浙江大学,2002.