文章编号:1001 - 9081(2006)12 - 2843 - 05

基于绘制历史的 sort-first 集群绘制负载平衡方法

沈兵虎,金哲凡,潘瑞芳
(浙江传媒学院 传播科技系,浙江 杭州 310018)
(pan_rf@126.com)

摘 要:负载平衡对 sort-first 集群并行绘制系统的性能有很大影响,但静态负载平衡方法适应性较差。而动态方法中,基于几何数据遍历的方法不具备实用性,基于粗粒度几何数据组织的方法在实际中得到应用,但依赖于对几何数据的预处理。为此,提出了一种实用的、基于绘制历史的负载平衡方法:它不依赖于几何数据,用以前帧的绘制时间和负载分布信息作为未来帧绘制负载分配计算的输入。在集群绘制平台上的测试和比较表明,该方法有很好的效果。

关键词:集群绘制;sort-first;负载平衡;绘制历史
中图分类号: TP391.41 **文献标识码**:A

Load balancing method based on render history in cluster rendering

SHEN Bing-hu, JIN Zhe-fan, PAN Rui-fang
(*Department of Broadcasting Technology, Zhejiang University of Media and Communication, Hangzhou Zhejiang* 310018, *China*)

Abstract: Performance of sort-first cluster parallel rendering system is greatly influenced by load balance. Static load balancing method is poorly adaptive. As to dynamical methods, those based on geometry data traversing are not practical, and those based on coarse geometry data granularity can be put into practice but depend on the pretreatment of geometry data. Therefore, a practical load balancing method based on render history was proposed. This method does not depend on geometry data, but utilizes render time of previous frame and its information of load distribution as the input of load balancing for the next frame's rendering. The method is tested on cluster rendering platform and the test results show that it is very effective.

Key words: cluster rendering; sort-first; load balance; render history

0 引言

面对要求越来越高的应用需求,孤立的图形系统的性能仍然在以下几个方面受到制约[1]:产生几何数据的能力,图形计算能力,几何指令发射能力和显示分辨率。

并行绘制技术突破了孤立图形系统的局限。高性能的图形并行绘制系统可以为海量数据场景绘制、大规模虚拟现实和仿真、超高分辨率科学计算可视化等高端应用提供所需的图形绘制能力。一个并行绘制系统包含多条图形绘制流水线,按照绘制流水线的组织方式,并行绘制系统可分为 sort-first、sort-middle、sort-last 三种[2]。sort-first 系统的输出图像被划分为一些不相交的任务块,每条流水线负责一个或多个任务块的绘制。图元通过预变换确定覆盖的任务块并进入相应的流水线,各条流水线独立工作,输出的子图像拼接为最终图像。sort-first 系统有 WireGL[3]、Display Wall[4]等。sort-middle 系统在图像划分和拼接上与 sort-first 相同,只是图元乱序地(不通过预变换)进入流水线的几何变换单元,几何变换之后再根据屏幕座标传送到正确的光栅化单元。sort-middle 系统有 InfiniteReality[5]、PGL[6]等。sort-last 系统的图元乱序进入多条流水线,每条流水线独立完成几何变换和扫描转换,生成一副尺寸完整、包含部分图元的深度图像,多幅图像作深度合成得到最终图像。sort-last 系统有 Pixel-Flow[7]、Parallel-Mesa[8]、Sepia[9]等。

并行绘制系统的实现方式有三种[10]:1)基于 ASIC(Application Specific Integrate Circuit)硬件实现,如 InfiniteReality 和 PixelFlow;2)基于大型机实现,如 Parallel Mesa 和 PGL;3)基于图形工作站集群实现,如 WireGL、AnyGL[11]、Display Wall、Sepia 和 Pomegranate[12]。由于低端图形硬件性能的提高和百兆、千兆网络设备的应用,集群并行绘制系统逐渐凸现出高性能、低成本、可扩展性好的优点,成为并行绘制系统的主流。集群并行绘制系统的应用包括三方面:1)利用其低成本构造仿真系统,对新的硬件体系结构进行研究,如 Pomegranate;2)在人机交互领域,用集群驱动多台投影仪,构筑大尺寸、高分辨率的投影墙,如 Display Wall 系统;3)通过对图形任务的并行处理,提供对包含大量数据和复杂纹理的场景的快速绘制,应用于沉浸式 VR、科学计算可视化等领域,如 AnyGL。

三种体系结构中,sort-first 最大程度保持了图形流水线的完整性,因此集群并行绘制系统大都采用 sort-first 结构。绘制集群中每一台图形工作站称为一个绘制节点,sort-first 系统通过剖分屏幕在多个绘制节点间分配计算任务,当几何图元在屏幕上分布不均匀时,容易出现节点间的负载不平衡:一些绘制节点因处理大量图元而处于忙碌状态,同时另一些节点因分配到稀疏的图元而处于空闲。由于一帧的绘制时间取决

收稿日期:2006 - 06 - 23;**修订日期**:2006 - 08 - 29
基金项目:浙江省自然科学基金资助项目(Y105324);浙江省科技厅资助项目(2006C31065)
作者简介:沈兵虎(1954 -),男,浙江人,副教授,硕士研究生,主要研究方向:图像处理、电子技术; 金哲凡(1974 -),男,浙江杭州人,讲师,博士,主要研究方向:计算机图形学、并行绘制、软件工程; 潘瑞芳(1959 -),女,江西南昌人,教授,硕士,主要研究方向:数据挖掘、图形图像处理.

于最后完成工作的节点,负载不平衡导致集群并行绘制系统绘制速度下降,sort-first 统的负载平衡办法是影响系统性能的一个重要问题。

1 负载平衡方法简介

按照是否动态修改屏幕的剖分,负载平衡方法可分为静态方法和动态方法两类。静态方法将屏幕固定地剖分为一些相等的矩形任务块,块数大于绘制节点数[13]。如图1,对4个绘制节点,屏幕被划分为16个任务块。这些任务块间隔、固定地分配给各绘制节点,使每个绘制节点既有屏幕中央部分的块,也有屏幕边缘部分的块。对图形应用程序的观察统计发现,图元更易出现在屏幕中央,因此划分越细,各绘制节点越倾向于获得相同的负载。静态方法优点是实现简单,几乎没有什么开销。缺点是其有效性只是统计意义上的,过于粗糙,不能满足高性能图形并行绘制的需要。

动态负载平衡方法根据图形应用负载分布情况动态改变屏幕的剖分和分配。图2(a)为 Roble 方法[14],它先将屏幕划分为相等的任务块,通过计算落在各任务块中的图元数确定其负载,然后将负载轻的块合并,将负载重的块平分为二,该方法比较粗糙。图2(b)为 Whelan 的 median-cut 方法[15],该方法以图元的质心作为计算负载的单位,不断对每个任务块的长边作剖分使两边负载相等,直至任务块数等于单元数。Whitman[16]的自顶向下分解方法改进了 Whelan 方法,它将每个图元的外包围盒与一精细的网格作比较,如果一格子被一图元覆盖,则将其权重加1。之后在网格格子上构建 Huffman 树,从树根向下访问直至经过的块数为处理单元数的10倍,再将这些负载基本相等的任务块分配给处理节点。图2(c)的 MAHD(Mesh-based Adaptive Hierarchical Decomposition)[13]方法同样使用一个精细的网格,当一个图元覆盖 N 个格子时,每个格子的权重增加 $1/N$。之后格子被组织成区域和表,在区域和表上作块的划分要快于 Whitman 方法。

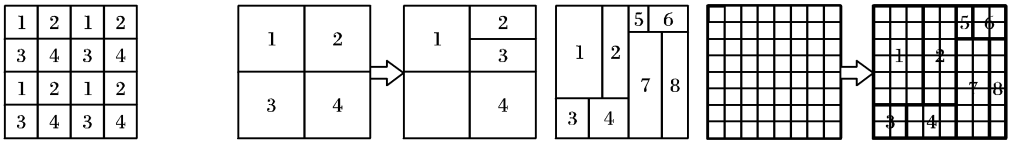


图1 静态负载平衡方法

(a) Roble (b) Whelan (c) WADH

图2 几种主要的动态负载平衡方法

这些方法都通过离线的模拟程序证明了有效性,但没有在直接在集群绘制系统中得到应用。原因在于它们都需要对所有图元做几何变换以确定其屏幕位置,集群绘制系统往往用于处理上百兆、甚至上 G 的几何数据,遍历图元的计算开销太大。理论上,几何变换是图形绘制计算的一部分,其计算结果可以与负载平衡计算共享。但现实中的并行绘制系统是软、硬件高度耦合的系统,将负载平衡计算插入到绘制流水线中去是不现实的。

针对这个问题,文献[4]在 DisplayWall 系统中对几何数据进行预处理,将其组织成粗粒度的单元,负载平衡计算针对这些单元进行,并通过实验对静态方法和几种不同屏幕剖分策略的动态方法的负载平衡效果进行了比较。该方法避免了几何数据遍历的巨大开销,实验证明是有效的。它的缺点是需要对图元进行预处理,影响系统的兼容性。

图形应用程序的一个规律是普遍存在运动连贯性,连续帧之间存在较大的相似。因此,以绘制历史、即以前帧的绘制时间和负载分布信息为参考进行未来帧的负载平衡计算是可行的。根据这个原理,本文提出一种 sort-first 集群绘制系统负载平衡方法:放弃数量巨大的几何数据,以绘制时间作为负载的指标,以绘制历史作为输入,以有效的算法进行屏幕任务块的剖分和重组。实际测试和比较结果证明这种方法有很好的效果。

2 基于绘制历史的负载平衡方法

图3显示了本文 sort-first 集群绘制系统的结构。其中,控制节点接受用户的输入,将绘制任务分配到多个绘制节点。绘制任务由多个绘制节点并行完成,图像合成节点将各个绘制节点生成的子图像拼接为最终图像并输出到显示设备。

设有 n 个绘制节点,分别为 $R_1,R_2,\cdots,R_k,\cdots,R_n$,$A_1 \sim A_n$ 为屏幕上的矩形任务块,f 为帧数,A_{kf} 为第 f 帧绘制时节点 k 负责的屏幕矩形区域,第 f 帧的屏幕剖分为 $(A_{1f},A_{2f},\cdots,A_{nf})$,第 f 帧各绘制节点的绘制时间为 $(t_{1f},t_{2f},\cdots,t_{nf})$,系统运行流程如下:

步骤1:$f=1$,以平均划分的原则设定第一帧的屏幕剖分 $(A_{11},A_{21},\cdots,A_{n1})$;

步骤2:控制节点接受用户输入,命令绘制节点 R_k 开始绘制任务块 A_{kf},启动 R_k 的计时器($k=1,2\cdots,n$);

步骤3:R_k 完成 A_{kf} 的绘制,停止计时器,发送 RENDER_OVER(t_{kf}) 消息给控制节点;

步骤4:控制节点等待所有 $R_k(k=1,2\cdots,n)$ 都完成绘制,命令图像合成节点进行图像拼接与同步输出;

步骤5:控制节点调用 Time-Space 算法计算下一帧的屏幕剖分:

$(A_{1f+1},A_{2f+1},\cdots,A_{nf+1}) = Time-Space((A_{1f},A_{2f},\cdots,A_{nf}),(t_{1f},t_{2f},\cdots,t_{nf}))$

步骤6:$f=f+1$;

步骤7:转步骤2。

Time-Space() 算法根据绘制历史信息,即各个节点的绘制时间和任务空间,计算下一帧的屏幕剖分。剖分方式有图4所示垂直剖分和水平剖分二种。假定系统使用图4(a)的垂直剖分,由于所有任务块高度相同,$(A_{1f},A_{2f},\cdots,A_{nf})$ 由宽度集合 $(w_1,w_2\cdots\cdots w_n)$ 决定,故 Time-Space() 的实际输入为:

1) 第 f 帧的宽度集合 $(w_1,w_2\cdots\cdots w_n)$;

2) 第 f 帧的节点绘制时间集合 $(t_1,t_2\cdots\cdots t_n)$。

Time-Space() 的输出为 $f+1$ 帧绘制使用的宽度集合 $(w_1',w_2',\cdots,w_n')$。

垂直剖分的 Time-Space() 的流程如下:

步骤1:计算时间均值 $\bar{t}=\frac{1}{n}\sum_{m=1}^{n}t_m$;

步骤2:计算时间轴上点 $(p_0,p_1,\cdots,p_n)$,其中 $p_u=\sum_{v=1}^{u}t_v$,$p_0=0$;

步骤3:令 $w_0'=0$;

步骤4:通过以下循环计算 $(w_1',w_2',\cdots,w_n')$:

```
FOR ( i = 1 TO n )
  { FOR ( a = 0 TO n - 1 )
      { IF i · t̄ ∈ [p_a, p_{a+1}], BREAK;}
```

$$w_i' = (i \cdot \bar{t} - p_a)\frac{w_{a+1}}{t_{a+1}} + \sum_{j=1}^{a} w_j - \sum_{k=1}^{i-1} w_k'$$

}

图 5 演示了 Time-Space( )的基本原理。假设在第 f 帧,任务块 1、2、3 由绘制节点 R_1、R_2、R_3 分别绘制的时间为 1s、2s、3s,按照以时间作为负载度量的原则,认为 R_1、R_2、R_3 工作负载之比为 1:2:3,以此比例将任务块 1、2、3 剖分为一系列小任务块,然后按照负载均分的原则进行任务块的重组,即得到新的屏幕剖分。图 6 为 Time-Space( )流程中各个变量的含义,该流程将基本原理推广到任意 n 个绘制节点的情况。图 4(b)的水平剖分的 Time-Space( )流程与垂直剖分相同,只需将流程中变量 w 换为 h 即可。

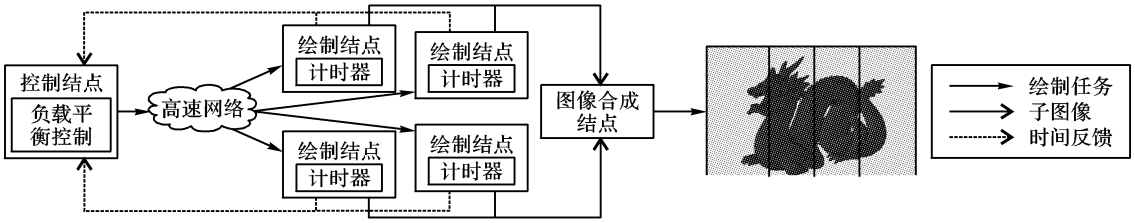
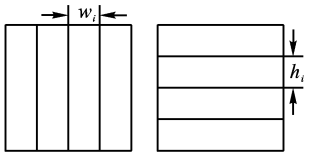


图 3　集群绘制系统体系结构



(a) 垂直分割　(b) 水平分割

图 4　垂直和水平剖分方式

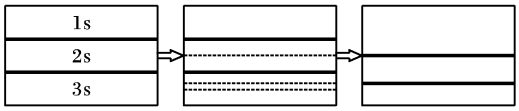


图 5　Time-Space( )算法的基本原理

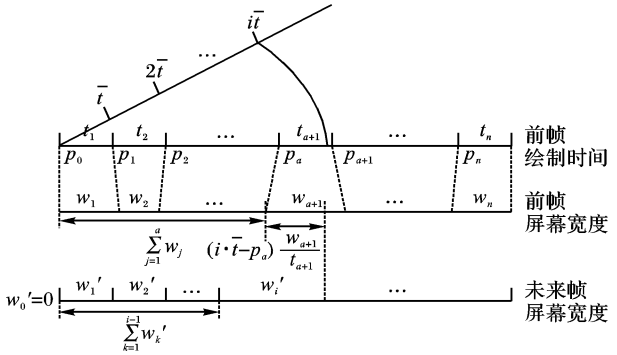


图 6　Time-Space( )中流程变量的含义

与其他的动态负载平衡方法相比,本文方法的特点是:放弃了数量庞大、处理复杂的几何数据,以绘制节点的帧绘制时间作为其负载的度量。Time-Space( )利用历史信息完成下一帧屏幕剖分的计算,是整个方法的关键。由 Time-Space( )流程步骤 4 的循环嵌套知其算法复杂度为 $O(n^2)$。注意其中基数 n 为绘制节点数,取值较小,如 4、8、16、32。而针对几何数据的负载平衡算法的基数的含义为几何元素数量,轻易就可以达到兆、千兆的水平。因此本文负载平衡方法的一个显著特点是开销很小。

3　实验数据分析比较

本文负载平衡方法在以高速网络联结多台配备高端显卡的 PC 而构成的 sort-first 集群绘制系统上进行了大量测试,图 7 为一些应用绘制效果的例子。

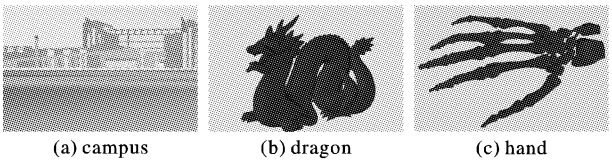
图 8 为 8PC 集群对一些应用进行多帧绘制的时间曲线。为了衡量系统的负载平衡状况,定义变量 $LB = \frac{T_{first}}{T_{render}}$,$T_{first}$ 为首先完成绘制的节点绘制一帧的时间,T_{render} 为整个系统绘制一帧的总时间,LB 是系统负载平衡水平的指标。图 9 为图 8 实验对应的 LB 曲线。表 1 列出了相关的测试数据,可以看出:

1)表 1 中使用负载平衡的帧平均绘制时间是不使用时的 60% ~80%,图 8 中有负载平衡的时间曲线比较低且平缓,说明使用本文负载平衡方法提升了系统的绘制速度。

2)表 1 中使用负载平衡的 LB 是不使用负载平衡时的 1.3 ~ 3.4 倍,图 9 中使用负载平衡的 LB 曲线在大部分区域更接近 1 并且很平缓,而不使用负载平衡的 LB 值相对较小,且曲线波动剧烈,这说明使用本文负载平衡方法提高了系统的负载平衡程度。

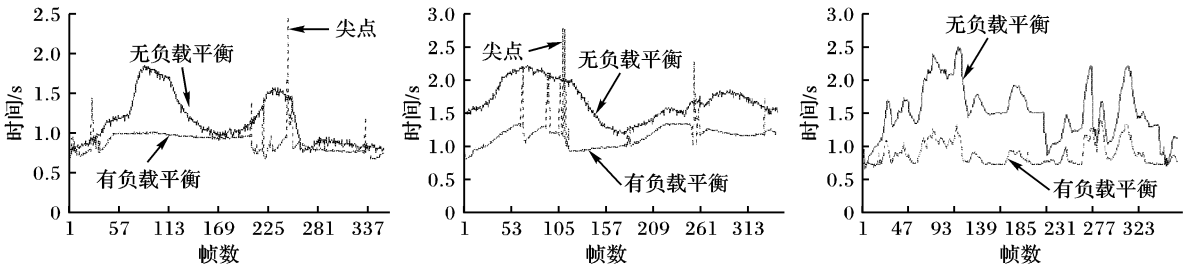
3)本文负载平衡方法本身的开销很小。表 1 负载平衡处理时间只占整个绘制时间的 2.3% ~8.2%,这部分开销包括了网络消息传递和 Time-Space( )模块计算的时间。

总之,以上结果证明了本文方法的有效性。



(a) campus　(b) dragon　(c) hand

图 7　应用绘制效果的例图



(a) hand,包含654 666个面片　(b) dragon,包含871 414个面片　(c) campus,包含133 540个面片,30M纹理

图 8　绘制时间曲线

文献[4]中,对 DisplayWall 系统分别应用 STATIC、GRID、UNION、KD-SPLIT 几种负载平衡方法进行了测试,并采集了以下数据:1)客户端进行负载重分配计算的时间 $ClientTime$;2)非平衡时间,即最先完成绘制的节点等待的时

间 *ImbalTime*;3）帧绘制总时间 *FrameTime*。

可知 $LB = 1 - ImbalTime/FrameTime$，负载平衡开销 $COST = ClientTime/FrameTime$。表 2 中列出了文献[4]和本文 *LB* 和 *COST* 的均值，可以看出：

1）STATIC 算法 $LB = 0.302$，$COST = 2.6\%$，平衡效果最差，开销最小，这正是静态方法的特点。

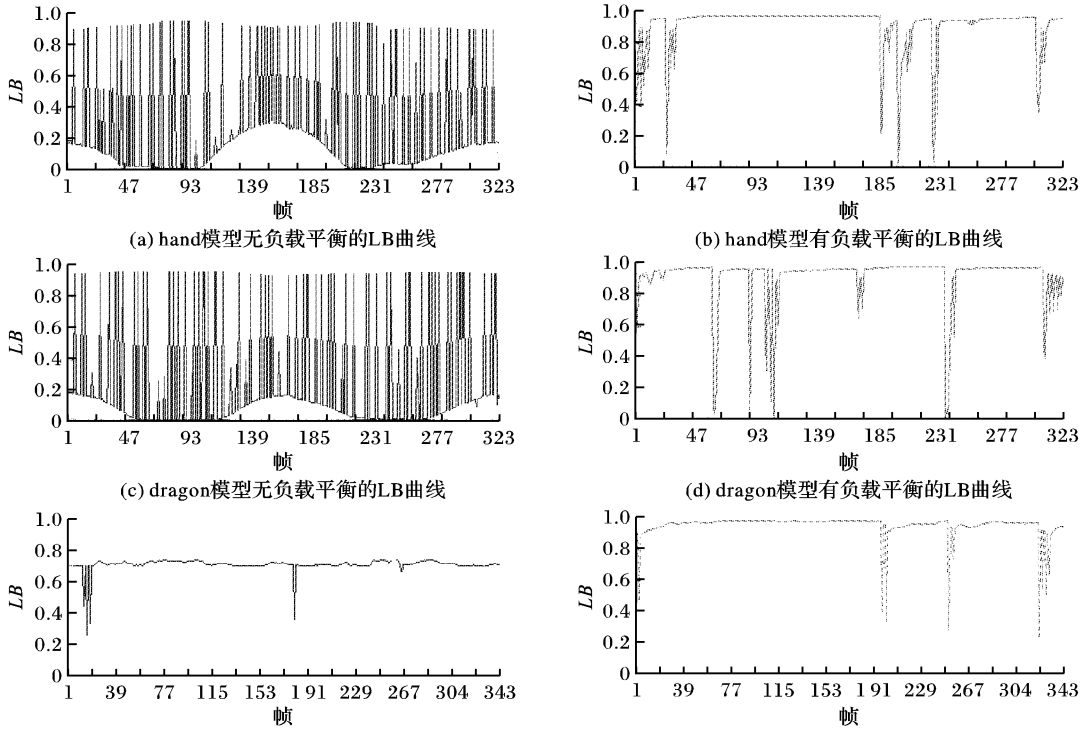
2）本文方法与 GRID 算法相比，平衡效果略好，开销是其 50%。

3）本文方法与 UNION 算法相比，平衡效果略好，但开销是其 11%。

4）本文方法与 KD-SPLIT 算法相比，平衡程度略不如，但开销是其 16%。

从比较结果可以看出，本文方法能实现较优的负载平衡效果，同时开销很小，与其他方法相比有相当的优越性。

另外，在时间曲线和 LB 曲线上都存在一些穿透基本（不使用负载平衡）曲线的尖点，形成这种现象的原因是：用户视角的运动趋势存在一些转折点，在这些点上，绘制历史对未来的预测失效了。如在浏览一个大建筑物若干帧后，它离开了视见区，同时在相反方向出现了新的建筑物。在这一帧里，如果原来的屏幕剖分与新的负载分布相差过大，会使绘制时间突升，*LB* 值突降，之后负载平衡模块即获得这一帧的绘制时间并立即作出反应，重新剖分屏幕使之适应新的运动趋势，绘制时间和 *LB* 又回到正常水平。这种现象是基于绘制历史的负载平衡方法所特有的。



(a) hand模型无负载平衡的LB曲线　(b) hand模型有负载平衡的LB曲线

(c) dragon模型无负载平衡的LB曲线　(d) dragon模型有负载平衡的LB曲线

(e) campus漫游无负载平衡的LB曲线　(f) campus漫游有负载平衡的LB曲线

图 9　LB 曲线

表 1　与无负载平衡控制的比较

应用名称	bunny	horse	hand	dragon	campus
无负载平衡帧平均绘制时间 *t*1 (s/frame)	0.298	0.344	1.158	1.700	1.796
使用负载平衡的帧平均绘制时间 *t*2 (s/frame)	0.252	0.282	0.909	1.192	1.061
*t*2/*t*1 (%)	84.6	82.0	78.5	70.1	59.1
负载平衡计算使用时间 *t*3 (s/frame)	0.017	0.023	0.024	0.028	0.046
COST(*t*3/*t*2) (%)	6.7	8.2	2.6	2.3	4.3
无负载平衡 *LB*1	0.304	0.284	0.299	0.265	0.721
负载平衡 *LB*2	0.650	0.679	0.911	0.911	0.950
*LB*2/*LB*1	2.1	2.4	3.0	3.4	1.3

表 2　与 DisplayWall 系统的比较

方法	*LB*	*COST* (%)
STATIC	0.302	2.6
GRID	0.780	9.6
UNION	0.754	44.6
KD-SPLIT	0.942	30.8
本文	0.820	4.8

4　结语

集群绘制、乃至并行计算领域的负载平衡问题，其难点在于算法的设计，以及如何控制负载平衡的计算开销。算法越强则计算开销越大，实际的工作环境可能相当苛刻，如果负载平衡方法开销超过了使用它获得的好处，就失去了实用意义。并行绘制处理的几何图元数量庞大，因此开销控制是 sort-first 集群绘制负载平衡的关键问题。DisplayWall 采取的策略是粗粒度地对图元进行组织，由于要进行预处理，对系统的兼容性有影响。本文采取另一种思路：放弃了几何数据，根据帧间相似的原理，用绘制历史信息作为负载平衡计算的输入。测试结果证明了本文方法的有效性和优越性。

在本文工作的基础上，可有两种方向的发展：一是增强对

绘制历史信息的应用,如利用多帧(本文使用了一帧)的历史来进行下一帧的预测,以提高算法有效性;甚至保留所有绘制历史,以某种智能型的决策达到负载平衡。二是将本文算法与文献[4]的方法进行混合,有选择地使用几何数据,以达到有效性和开销两利的效果。这些工作将在未来进一步展开。

参考文献:

[1] DAVIS T, CHALMERS A, JENSEN HW. Practical Parallel Processing for Realistic Rendering[A]. ACM SIGGRAPH[C]. 2000.

[2] MOLNAR S, COX M, ELLSWORTH D, *et al.* A Sorting Classification of Parallel Rendering[A]. IEEE computer graphics and applications[C]. 1994. 23 - 31.

[3] HUMPHREYS G, ELDRIDGE M. WireGL: A Scalable Graphics System for Clusters[A]. Proceedings of ACM SIGGRAPH[C]. 2001.

[4] SAMANTA R, ZHENG J, FUNKHOUSER T, *et al.* Load Balancing for Multi-Projector Rendering Systems[A]. Proceedings of the SIGGRAPH/Eurographics Workshop on Graphics Hardware[C]. 1999.

[5] MONTRYM JS, BAUM DR, DIGNAM DL, *et al.* InfiniteReality: A Real-Time Graphics System[A]. Proceedings of SIGGRAPH '97[C]. 1997. 293 - 302.

[6] CROCKETT TW. Design Considerations for Parallel Graphics Libraries[R]. ICASE Report No. 94 - 49, 1994.

[7] MOLNAR S, EYLES J, POULTON J. PixelFlow: High-Speed Rendering Using Image Composition[A]. Proceedings of SIGGRAPH '92[C]. Chicago, Illinois, 1992. 231 - 240.

[8] MITRA T, CHIUEH T. Implementation and Evaluation of Parallel Mesa Library[A]. IEEE International Conference on Parallel and Distributed Systems[C]. 1998.

[9] MOLL L, SHAND M, HEIRICH A. Sepia: Scalable 3D Compositing Using PCI Pamette[A]. Laurent Proceedings of the Seventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines[C]. 1998.

[10] SHI JY, JIN ZF. A Survey On Parallel Polygon Rendering[J]. Computer Aided Design and Computer Graphics, 2003, 15(6): 637 - 642.

[11] YANG J, SHI JY, JIN ZF, *et al.* Design and Implementation of A Large-scale Hybrid Distributed Graphics System[A]. Eurographics Workshop on Parallel Graphics and Visualization[C]. Saarbrueck-en, Germany, 2002.

[12] ELDRIDGE M, IGEHY H, HANRAHAN P. Pomegranate: A Fully Scalable Graphics Architecture[A]. Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH[C]. New Orleans, Louisiana, 2000. 443 - 454.

[13] MUELLER C. The Sort-First Rendering Architecture for High-Performance Graphics[A]. Proceedings of the 1995 Symposium on Interactive 3D Graphics[C]. 1995. 75 - 82.

[14] DOUGLAS R. A Load Balanced Parallel Scanline Z Buffer Algorithm for the iPSC Hypercube[A]. Proceedings of Pixim 88[C]. Paris, France, 1988. 177 - 192.

[15] DANIEL W. Animac: A Multiprocessor Architecture for Real-Time Computer Animation[D]. Ph. D. dissertation, California Institute of Technology, 1985.

[16] WHITMAN S. Dynamic Load Balancing for Parallel Polygon Rendering[J]. IEEE Computer Graphics andApplications, 1994, 14(4).

(上接第 2842 页)

给出了经过区域合并处理后的分段图像,图 2(c)给出了最终分析得到的视觉重要性评价,该图像中的区域的亮度值反映其在视觉上感兴趣的程度,亮度值越高则感兴趣程度越高。它基本上能够反映出人眼视觉的主观特性,例如播音员的脸部和衣襟部分亮度值较大,说明其视觉敏感度较高,这与人类的视觉特性基本吻合。同时,表 1 列出了 News 场景部分区域的视觉感兴趣值及相关特性如区域尺寸和平均亮度。对于较复杂的场景如视频序列 Carphone,该算法也能给出较为理想的结果,如图 3 所示。

表 1　实验结果的部分数据统计

区域编号	1	2	3	4	5	6	7	8	9	10	11	12	…
面积	26	29	31	23	136	1099	35	66	49	32	41	57	…
亮度均值	122	109	93	111	105	135	108	125	92	85	95	102	…
视觉评价值	1	1	0.95	0.93	0.91	0.90	0.87	0.87	0.85	0.84	0.82	0.81	…

5　结语

通过对人眼在观察视频场景的一些低级视觉特性和高级视觉特性的综合分析,提出了一种基于高级视觉特性的感兴趣区域自动判定算法,它结合了不同的视觉因素,能够自动地计算每个区域的视觉重要性估计值,这个估计值反映了每个区域的视觉感兴趣程度。最后的实验结果表明:该算法所获得的结果基本上符合人眼主观视觉特性,且该算法实现的计算复杂度不高,因此具有较强的实用价值。同时,该算法也存在一些不足,如对细节较多的图像判别效果不是很理想,如何结合更好的图像分割算法,同时加入人脸检测的方法,使得感兴趣区域判定更为准确,是值得进一步研究的问题。

参考文献:

[1] 胡栋,郑宝玉. 数字视频关键技术的若干新进展[J]. 通信学报, 2003, 24(7).

[2] 一种新的基于率失真优化的感兴趣区域编码[J]. 计算机应用, 2005, 25(1).

[3] TRAN TD, SAFRANEK R. A locally adaptive perceptual masking threshold model for image coding[A]. ICASSP[C]. Atlanta, 1996.

[4] WOLF S, WEBSTER A. Subjective and objective measures of scene criticality[A]. ITU meeting on subjective and objective audiovisual quality assessment methods[C]. Turin, 1997.

[5] VILLEGAS P, MARICHAL X, SALCEDO A. Objective evaluation of segmentation masks in video sequences[A]. WIAMIS'99[C]. Germany, 1999. 85 - 88.

[6] HOROWITZ SL, PAVLIDIS T. Picture segmentation by a directed split-and-merge procedure[R]. Technical report, Department of Electrical Engineering, Princeton University, N. J. 08540, 1975.