

一种基于sort-last架构的非结构网格并行体可视化算法

范良¹, 张晓蓉¹, 吴亚东², 陈呈^{3,4}, 王昉^{3,4}

1. 西南科技大学计算机科学与技术学院, 四川 绵阳 621010;
2. 四川轻化工大学计算机科学与工程学院, 四川 自贡 643002;
3. 空气动力学国家重点实验室, 四川 绵阳 621000;
4. 中国空气动力研究与发展中心计算空气动力研究所, 四川 绵阳 621000

摘要

体绘制是非结构网格数据体可视化的重要途径之一。然而, 现有的串行非结构网格体绘制算法效率较低, 难以满足大规模数据体可视化需求。因此, 并行体绘制算法研究是高性能体可视化程序设计的趋势。提出一种基于sort-last架构的非结构网格并行体绘制算法。首先, 设计了基于KD树的并行体数据分割算法; 其次, 每个进程使用独立的可视化管线计算体绘制图像, 根据树合成策略合成最终结果; 最后, 采用两层LOD模型技术优化交互体验。实验结果表明, 提出的算法能被很好地应用于大规模非结构网格数据体可视化中, 所有交互时延在毫秒级别, 满足了实时交互需求。

关键词

非结构网格; 体可视化; 并行体绘制; 实时交互

中图分类号: TP333

文献标识码: A

doi: 10.11959/j.issn.2096-0271.2021042

A sort-last architecture based parallel volume visualization algorithm for unstructured grid

FAN Liang¹, ZHANG Xiaorong¹, WU Yadong², CHEN Cheng^{3,4}, WANG Fang^{3,4}

1. School of Computer Science and Technology, Southwest University of Science and Technology, Mianyang 621010, China
2. School of Computer Science and Engineering, Sichuan University of Science and Engineering, Zigong 643002, China
3. State Key Laboratory of Aerodynamics, Mianyang 621000, China
4. Institute of Computational Aerodynamics, China Aerodynamics Research and Development Center, Mianyang 621000, China

Abstract

Volume rendering is one of significant unstructured grid data volume visualization methods. However, existing serial algorithms for unstructured grid volume rendering are inefficient and cannot meet large-scale data volume visualization needs. Therefore, parallel volume rendering algorithm research has become a trend in high-performance volume visualization program design. A sort-last based parallel volume rendering algorithm for unstructured grid was proposed. Firstly, a parallel KD-tree algorithm for volumetric data split was designed. Secondly, each process calculated volume

rendering images with independent visualization pipeline, and the final result was synthesized with tree composite strategy. Finally, a two-level LOD model was leveraged to optimize interactive experience. The experimental results show that the proposed algorithm can be well applied to large-scale unstructured grid data volume visualization, and all interaction delays are in milliseconds, which meets real-time interaction needs.

Key words
unstructured grid, volume visualization, parallel volume rendering, real-time interaction

1 引言

体可视化是科学可视化中的重要研究方向,被广泛应用于医学影像、计算流体力学(computational fluid dynamics, CFD)、气象气候学等领域,体绘制算法是体可视化领域的重要研究内容^[1]。体绘制直接从三维的体数据中生成半透明的二维屏幕图像,空间表现能力强,因此,也被称为直接体绘制。科学家可以通过体绘制结果看穿数据体,探索数据内部结构,获取有用信息。与面绘制相比,体绘制结果多了一个维度的信息,因此,可视化结果不仅更加精确,而且更能体现数据本身的特点^[2]。

体数据是体绘制算法的输入,分为结构网格和非结构网格^[3]。其中,结构网格具有规则的几何结构和拓扑结构,相反,非结构网格的几何结构和拓扑结构不规则,网格单元可以是任意形状。在三维数据场变化剧烈的区域可以用体积较小的网格单元表示,变化比较平滑的区域则使用体积较大的网格单元表示,对于同一个数据场,可以用更少的非结构网格单元进行描述,有效节省了存储空间^[4]。因此,非结构网格数据在科学计算与仿真领域的应用更加广泛。然而,由于非结构网格的几何结构和拓扑结构不规则,相对于结构网格,非结构网格的体绘制算法设计比较困难,涉及复杂的计算与庞大的存储量。目前,非结构网格体绘制算法可以通过基于图像空

间、基于对象空间和两者混合的技术实现^[2]。光线投射算法^[5]是十分具代表性的基于图像空间实现的体绘制技术。然而,对于非结构网格数据而言,无法根据几何结构和拓扑结构计算光线离开当前网格单元后即将进入的下一个网格单元,采用局部或全局搜索下一个网格单元需要极大的开销。Weiler M等人^[6]将沿着光线的采样点限制在单元面上,同时,采用预积分技术,将预积分的颜色和透明度值存入一张纹理图,并且将光线进入和离开单元点的标量值以及光线在网格单元内穿过的长度作为参数查询纹理图。此外,他们还精心设计了纹理数据结构,将体数据以纹理的形式载入显存,通过硬件技术加速算法。投影四面体算法^[7]是十分著名的基于对象空间实现的体绘制技术,首先,计算所有网格单元的可见性顺序;然后,根据可见性顺序将四面体单元在图像平面上的投影拆分为一系列三角形;最后,将三角形的顶点数据载入显存,通过GPU光栅化并合成最终结果。此外,Wylie B等人^[8]利用可编程的顶点着色器,设计了基于硬件加速的投影四面体算法,不需要额外的数据结构与数据预处理。HAVS(hardware-assisted visibility svrting)^[9]算法是最具代表性的基于图像/对象空间混合技术实现的非结构网格体绘制算法,在对象空间,对网格单元执行局部排序,并生成一个近似有序的网格单元片段列表;在图像空间,使用固定深度的排序网络对片段列表执行升序排序。其中,对象空间计算在CPU上进行,图像空间计算在GPU上进

行。HAVS算法在简化CPU的处理运算的同时,将大量排序计算交给GPU执行,进一步提升了算法性能。

由于各种科学计算与模拟仿真的精度越来越高,非结构网格数据规模呈爆炸式增长。体绘制包含数据重构、传递函数映射和绘制合成,涉及大量的浮点运算,串行的算法难以满足需求。随着技术发展,大量的并行计算软件和硬件出现,这为科学研究提供了更加强劲的计算能力,基于进程并行的体绘制算法研究成为大规模非结构网格数据体可视化设计的趋势^[10]。Fogal T等人^[11]实现了基于集群的并行体可视化算法,使用KD树组织数据,进程之间采用信息传递接口(message passing interface, MPI)通信,可高效绘制大规模标量场数据集。Chen J H等人^[12]设计了分布式的并行体绘制算法,并对大规模燃烧模拟数据的多个变量同时进行体绘制,允许用户实时交互体绘制结果。本文在现有研究成果的基础上,研究并实现了一种基于sort-last^[13]架构的非结构网格并行体绘制算法,并集成到国产自主可控科学可视化软件拓视(TopViz)中,主要贡献如下。

- 实现了基于KD树的并行体数据分割算法和可见性排序算法。首先,使用预处理程序确保每个进程读取的网格数量大致相等;其次,以每个进程上的数据为输入,构造局部KD树;然后,将所有进程上的局部KD树合成为全局KD树;最后,通过相机位置和分割平面计算全局KD树叶节点子节点的可见性顺序。

- 实现了基于二叉树的并行图像合成算法。根据可见性顺序对所有进程上的图片两两配对并合成中间结果,不断重复配对与合成的过程,直到合成最终结果。

- 实现了两层LOD交互模型。首先,同步所有进程的数据边界信息;然后,在主进程上渲染数据边框模型。当处于交互状

态时,渲染边框模型;停止交互时,渲染体绘制模型。

- 设计实验验证,并分析了算法性能。实验结果显示,本文的并行体数据分割算法具有良好的性能和负载均衡能力;并行体绘制算法可以被很好地应用于大规模非结构网格数据体可视化,且交互时延在毫秒级别,满足实时交互的需求。

2 基于进程并行的非结构网格体绘制算法

非结构网格数据的体绘制算法涉及庞大的计算量与存储量,串行算法效率较低,难以满足大规模非结构网格数据体可视化实时交互的需求。因此,基于进程并行的体绘制算法是大规模非结构网格数据体可视化十分有效的解决方案。本节将详细描述基于进程并行的非结构网格体绘制算法流程,算法包含数据分割、绘制和图片合成3个阶段,如图1所示。在数据分割阶段,首先,每个进程读取数量大致相等的网格单元;然后,将每个进程上的体数据作为输入,构造局部KD树;最后,通过MPI将所有进程上的局部KD树合成为全局KD树。在绘制阶段,每个进程使用独立的体绘制管线计算图像。在图片合成阶段,按照可见性顺序配对所有进程上的图片,并合成中间结果,不断重复配对和合成过程,直到合成最终结果。

2.1 数据分割

并行体绘制算法的第一步是将大规模的网格数据拆分为若干个规模较小的网格。由于绘制过程的性能取决于绘制最慢的进程效率,因此,负载均衡是数据分割算法设计首要考虑的问题。本节将详细描述基于KD树的并行体数据分割算法的设计,

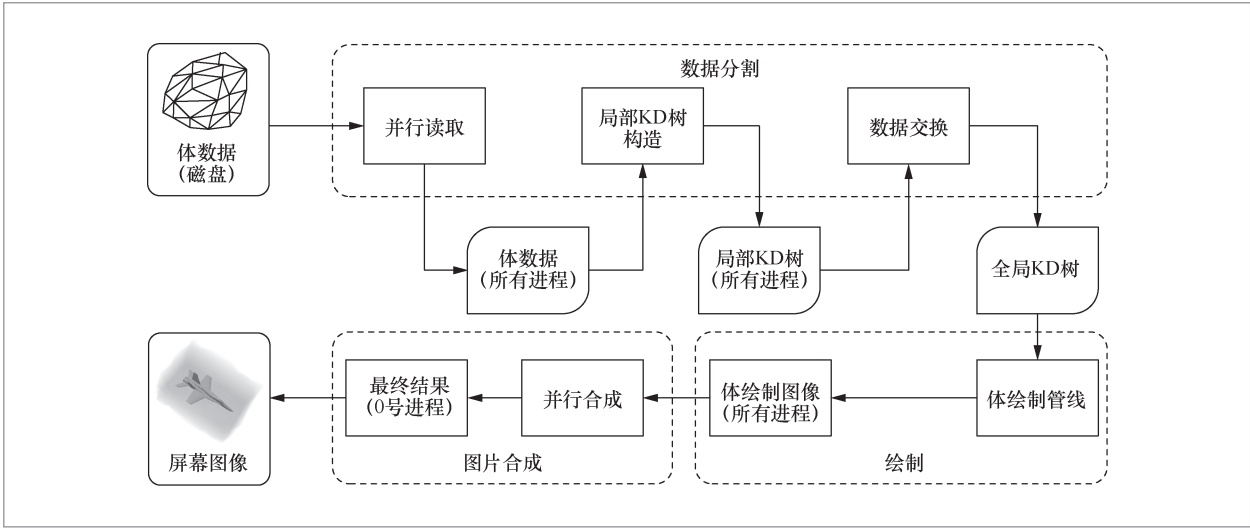


图1 基于进程并行的非结构网格体绘制算法流程

为了确保负载均衡,最终分割结果中KD树叶子节点之间的数据量应基本相同。

2.1.1 局部KD树构造

KD树是一种分割 K 维空间的高效数据结构,属于二叉空间分割树的特殊情况,被广泛应用于各种数据分割和搜索程序中^[14]。其中,KD树的叶子节点表示 K 维空间中的点,非叶子节点表示将 K 维空间一分为二的超平面或超立方体。如果要将KD树算法应用于体数据分割,则需要赋予节点新的意义,其中,叶子节点表示非结构网格单元集合,非叶子节点则表示将空间一分为二的超平面。体数据是三维的,因此,使用与坐标轴平行的分割平面可以最大限度地简化数据分割的运算过程。

算法1给出了KD树的构造过程。对于一个非叶子节点,首先,计算所有网格单元的质心坐标;其次,计算分割平面的方向,为了确保与分割平面相交的网格单元最少和避免数据分割后被拉长,选择数据边界跨度最长的坐标轴方向作为分割平面方向;然后,计算所有网格单元沿着分割平

面方向的质心坐标中位数,为了确保左右子树的网格单元数量大致相等,将质心坐标的中位数作为分割平面的位置坐标;最后,沿着分割平面将数据一分为二,对于与分割平面相交的网格单元,复制一份并划分到分割平面两侧。KD树构造是一个递归的过程,不断拆分非叶子节点,直到生成指定数目的叶子节点。

算法1 KD树构造

输入:非结构网格数据

输出:KD树根节点指针

步骤1:计算所有网格单元的质心坐标;

步骤2:根据网格数据边界值计算分割平面方向;

步骤3:计算质心坐标的中位数,将中位数质心坐标作为分割面的位置坐标;

步骤4:沿着分割面将网格数据一分为二;

步骤5:递归执行步骤2~4,直到生成指定数目的叶子节点。

通常,一个大规模的网格数据在超级计算机上使用多个并程序模拟计算生成,以多块(multi-block)数据的形式写入磁盘,最好也用并行读取的方法处理。

虽然单块数据的规模不大,但是单块数据之间的网格规模可能差距很大。因此,本文首先使用一个预处理程序统计大规模网格数据中每一块数据的网格单元数量,并保存到一个列表中;然后,根据列表网格数量信息计算每个进程应读取的单块数据编号,确保每个进程读取的网格单元数量大致相等;最后,每个进程根据单块数据的编号并行读取体数据。

在每个进程上使用KD树算法分割体数据,这个过程被称为局部KD树的构造。预处理过程确保了每个进程上读取的网格单元数量大致相等,因此,局部KD树构造算法是负载均衡的。每个进程上的局部KD树具有相同的非叶子节点,即分割平面相同,所有的树节点同步生成。因此,首先构造一个非叶子节点,同步所有进程上该节点对应的体数据边界信息,根据边界范围选择分割平面的方向;然后,并行计算所有进程上该节点对应的体数据网格单元沿着分割平面方向质心坐标的中位数^[15],将中位数的坐标作为分割平面的位置坐标;最后,沿着分割平面将该节点一分为二。递归构造每一个非叶子节点,直到生成指定数目的叶子节点。为了最大化地确保KD树平衡,以及便于后续设计并行图像合成算法,本文构造的KD树均为满二叉树,即叶子节点数量为2的正整数幂。如果叶子节点数量为 N ,则KD树的高度为 $\lg N$ 。

2.1.2 全局KD树合成

全局KD树与局部KD树具有相同的非叶子节点,不同的是,每个进程只存储全局KD树中一个叶子节点的数据,这些叶子节点的数据为最终分割结果。进程之间的局部KD树对应的叶子节点数据交换过程被称为全局KD树的合成。数据分割需要保证每个进程上都有数据,因此,进程数量应

该等于KD树的叶子节点数量,即2的正整数幂。

对于局部KD树,只有其中一个叶子节点的数据属于当前进程,因此,需要进行 N 对 N 的数据交换操作。首先,给所有局部KD树的叶子节点编号(从0开始);然后,通过MPI,根据进程秩将属于其他进程的叶子节点数据发送至对应的进程;最后,将每个进程上的数据合并为完整的网格。

2.1.3 基于KD树的可见性排序算法

对于与分割平面相交的网格单元(也被称为ghost单元^[2]),本文采取的处理方式是复制一份,然后将其划分到分割平面两侧。该处理方式可以确保分割平面两侧的网格数据具有严格的可见性顺序,不存在相互遮挡的情况。其中,可见性顺序^[16]用于描述空间中两个对象的遮挡情况,在给定视线方向的情况下,如果对象 a 遮挡了对象 b ,则可见性顺序 $O_a < O_b$;反之,则可见性顺序 $O_a > O_b$ 。对于KD树的非叶子节点的左右孩子节点,以分割平面为界,定义与相机位置处于同一侧的孩子节点为近点,另一侧为远点。近点的可见性顺序明显小于远点的可见性顺序,因此,通过相机位置和分割平面可以计算KD树中所有叶子节点的可见性顺序。

算法2给出了基于KD树的可见性顺序计算过程。将全局KD树作为输入,从根节点开始,根据相机位置确定近点与远点;按照“先近点,后远点”的顺序计算每个非叶子节点的可见性顺序;如果遇到叶子节点,则将叶子节点对应的进程秩添加到可见性顺序列表。递归处理每个节点,直到所有叶子节点的可见性顺序计算完毕。

算法2 可见性顺序计算

输入:全局KD树

输出:可见性顺序列表list

- 步骤1:如果当前节点为叶子节点,将该节点的进程秩添加到list;
- 步骤2:如果当前节点为非叶子节点,根据相机位置确定近点与远点:
 - 计算近点可见性顺序;
 - 计算远点可见性顺序;
- 步骤3:递归执行步骤1~2,直到所有叶子节点的可见性顺序计算完毕。

2.2 绘制

并行体绘制算法的第二个阶段(绘制)以第2.1节中KD树算法的分割结果为输入,在每个进程上执行独立的可视化管线,从而计算体绘制图像。本文采用体绘制管线技术,如图2所示,整个体绘制管线分为数据处理、体绘制算法、体绘制参数和渲染4个部分,其中,管线中每个阶段的输出作为下一个阶段的输入。在数据处理部分,使用Reader加载KD树算法的分割结果,如果需要对体数据进行裁剪和四面体化等操作,可以通过Filter完成;在体绘制算法部分,使用高效的线程并行投影四面体算法^[17]计算图元信息;在体绘制参数部分,建立体数据标量值与颜色/透明度之间的映射关系;在最后的渲染部分,使用OpenGL API函数将体绘制图元数据与传

递函数参数载入显存,通过GPU光栅化并合成最终的体绘制图像。体绘制管线技术具有高度的可扩展性与灵活性,管线每个部分只提供通用的接口,实现体绘制管线则需要提供所有通用接口的具体实现。如果要研究不同的体绘制算法性能,只需要提供新算法的Mapper即可。

2.3 图片合成

如图3所示,基于sort-last架构的并行体绘制算法最后需要将所有进程上的体绘制图片合成最终结果。由于体绘制结果是半透明的二维图片,因此,需要根据可见性顺序按照从前向后的顺序进行合成。可以使用over运算将两张体绘制图片合成一张图片,其中,over运算不满足交换律,但是满足结合律^[18]。因此,在不改变合成顺序的前提下,可以对所有进程上的体绘制图片任意分组,采取分组合成策略。over运算的性质为并行图像合成算法设计提供了条件^[15]。

本文采用基于二叉树的图像合成算法,首先,根据第2.1.3节中的方法计算所有进程体绘制图片的可见性顺序列表;然后,根据可见性顺序列表对体绘制图片进行两两配对;最后,通过MPI将图片发送至其配对进程,并使用over运算合成中间

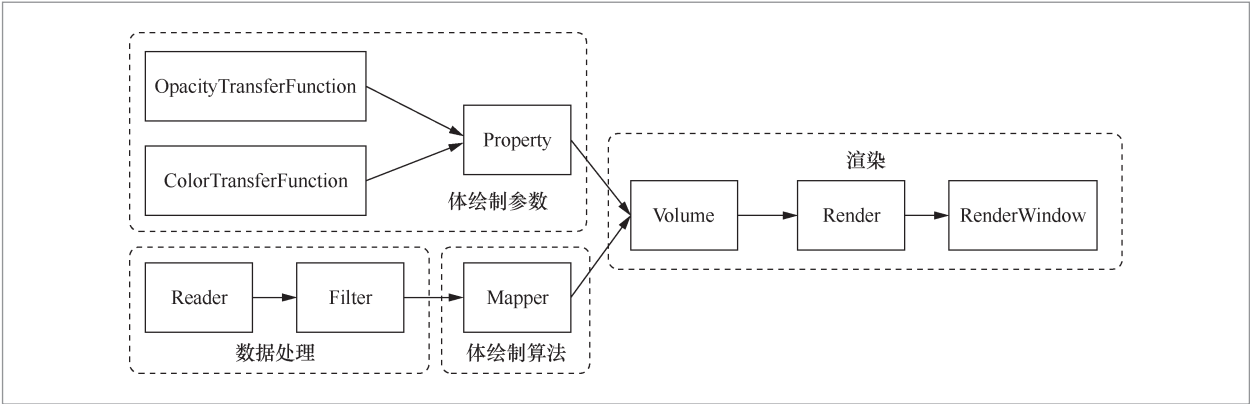


图2 体绘制管线

结果。不断重复配对与合成的过程,直到所有中间结果都被合成为最终图片。与串行的图像合成算法相比,虽然基于树的合成算法增加了通信次数和合成次数,但是它将时间重叠利用,减少了资源闲置,算法并行粒度更高,效率优势更明显。

以4个进程为例,二叉树图像合成算法流程如图4所示。假设4个进程上的体绘制图片可见性顺序为 $O_0 < O_2 < O_1 < O_3$,第一次合成,2号进程和3号进程分别将图片发送至0号进程和1号进程,在0号进程和1号进程上合成中间结果;第二次合成,1号进程将中间结果发送至0号进程,在0号进程上合成最终结果。

2.4 两层LOD模型

体绘制算法涉及大量的距离运算、求交运算和插值运算,同时,图片合成与进程之间的通信开销不可忽略。对于大规模非结构网格数据体可视化,难以满足实时交互的需求,因此,本文采用层次细节(level of detail, LOD)技术优化用户体验。如图5所示,一共设计了两层LOD模型,首先,所有从进程将数据边界信息发送至主进程(0号进程),在主进程上同

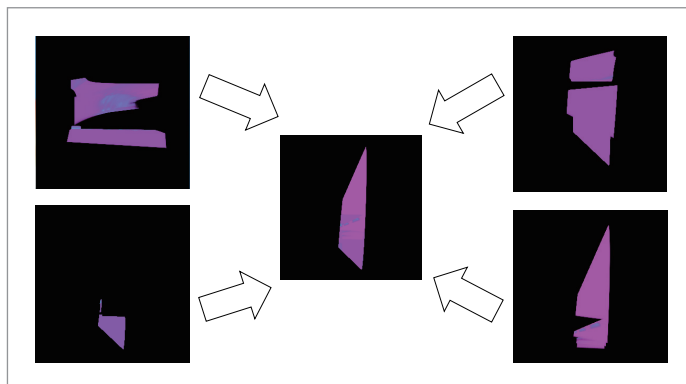


图3 图片合成

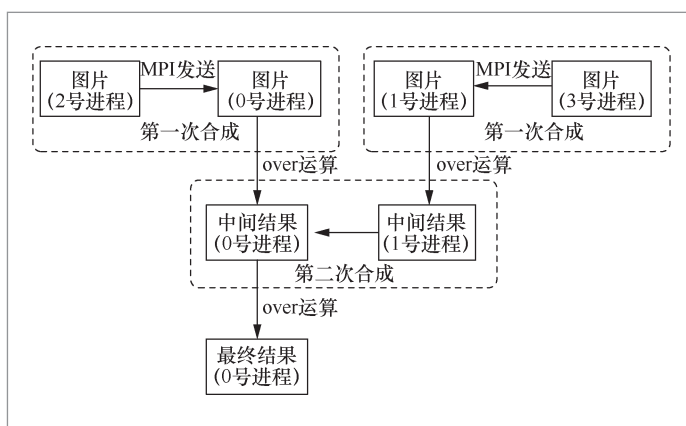


图4 二叉树图像合成算法过程(4个进程)

步边界信息,并渲染一个刚好能包围住体数据的边框模型。当处于交互状态(按下鼠标)时,渲染边框模型;当交互状态结束

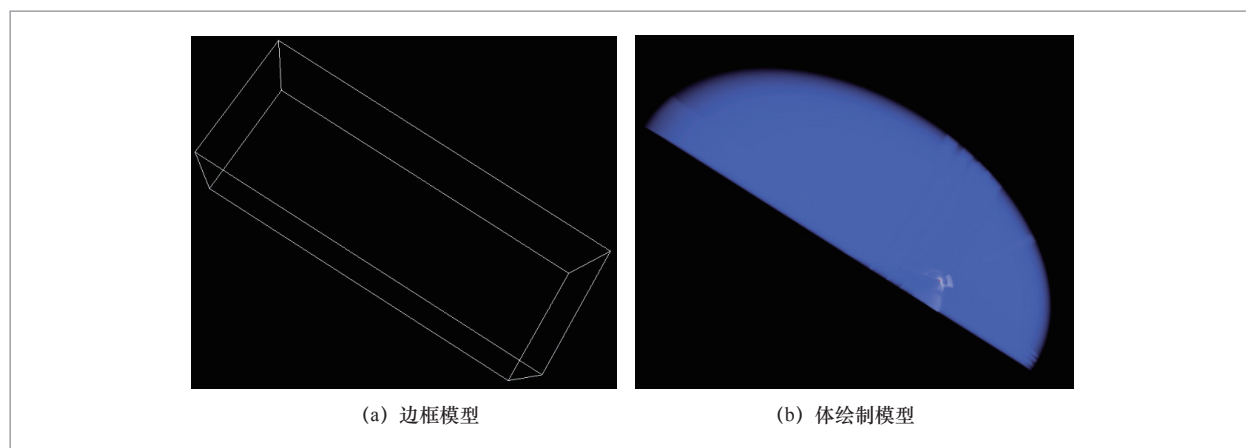


图5 两层LOD模型

(释放鼠标)时,开始渲染体绘制模型;当体绘制模型渲染完成后,替换边框模型。使用两层LOD模型,用户在交互时可以根据边框模型判断当前体数据的位置和角度,同时,边框模型的渲染在主进程上进行,只需要极小的开销,并且不需要图片合成,因此,可以满足实时交互的需求。

3 实验分析

为了研究算法的性能,本文设计了节点内并行实验和跨节点并行实验。节点内并行实验配置为高性能Intel服务器,其中,CPU为4个20核心的Intel(R) Xeon(R) E5-2698 v4 @ 2.20 GHz,内存大小为500 GB。跨节点并行实验配置为国产高性能服务器,其中,CPU为飞腾处理器,内存大小为120 GB。所有的算法均使用C/C++语言实现,并行框架采用MPI,编译器的版本为gcc/g++ 8.3,操作系统为64位Ubuntu 16.04。所有的算法执行时间通过C语言API函数ftime()获取。同时,为了避免偶然性,所有的结果均通过多次重复实验计算平均值得出。

3.1 数据分割算法性能

为了研究数据分割算法性能,采用节点内并行实验配置,从数据分割开销、负载均衡测试和VTK(visualization toolkit)算法对比3个方面设计实验。首先,在不同进程数量下研究数据分割后的

网格数量增加情况;然后,对数据分割算法的负载均衡能力进行测试;最后,对比了本文算法与VTK算法的性能。

3.1.1 数据分割开销

在构造局部KD树时,与分割平面相交的单元会被复制一份,并被划分到分割平面的两侧。因此,数据分割后,网格单元总数会增加。**表1**给出了使用不同数量的进程分割数据后的网格规模。与预期相符,数据分割后,网格规模增大;并且,随着进程数量增加,更多的网格单元与分割平面相交,产生了更多的ghost单元,因此,网格规模随之增大。ghost单元是数据分割开销的重要部分,直接影响后续并行体绘制算法的效率。

3.1.2 负载均衡测试

一个好的数据分割算法应该具有良好的负载均衡能力。本节将最多网格进程与最少网格进程的网格单元数量差值作为衡量标准,该值越小,说明算法负载均衡能力越好。**表2**给出了不同进程数量下的网格单元数量差值,结果显示,所有的差值均在4 000以内,全局KD树叶子节点之间的网格单元数量大致相等,本文数据分割算法的负载均衡能力较好。

3.1.3 VTK算法对比

为了研究本文中的数据分割算法性能,将开源可视化框架VTK中的D3

表1 网格规模

数据	进程数量/个						
	1	2	4	8	16	32	64
DATASET1	18 986 688	19 070 167	19 174 907	19 338 967	19 544 837	19 819 439	20 148 062
DATASET2	7 695 032	7 735 498	7 808 708	7 899 696	7 996 802	8 135 826	8 339 999
DATASET3	4 330 768	4 375 315	4 462 776	4 552 178	4 648 963	4 756 936	4 879 697

(vtkDistributedDataFilter)类作为对比,在不同进程数量下进行实验,结果如图6所示。对于VTK算法,当进程数量在2个和8个之间时,算法执行时间有所下降;但是,当进程数量超过8个时,算法执行时间反而随着进程数量的增加而增加。本文算法则是随着进程数量的增加,执行时间降低。出现该结果的原因是,VTK算法以块数据为单位并行读取数据,为了确保每个进程上的网格数量大致相等,需要在数据读取完毕后进行一次同步操作;而本文算法使用预处理程序统计每个数据块的网格数量,确保了每个进程上读取的网格数量大致相等,不需要同步。随着进程数量增加,VTK算法的同步操作需要更大的通信开销,算法执行时间因此增加;与VTK算法相比,本文算法少一次数据同步开销,随着进程数量增加,局部KD树构造时间下降,算法执行时间随之下降,性能优势更加明显。

3.2 并行体绘制算法性能

为了研究本文的并行体绘制算法性能,首先,使用节点内并行实验配置,测试算法在不同进程数量下的算法执行时间和加速比;然后,使用跨节点并行实验配置,测试算法在大规模网格体绘制下的性能,并结合实验数据分析了算法的效率和瓶颈。

表 2 负载均衡测试结果

数据	进程数量/个					
	2	4	8	16	32	64
DATASET1	0	17	3 837	3 076	3 170	2 133
DATASET2	0	2 584	1 239	3 012	2 119	448
DATASET3	0	6	3	41	57	117

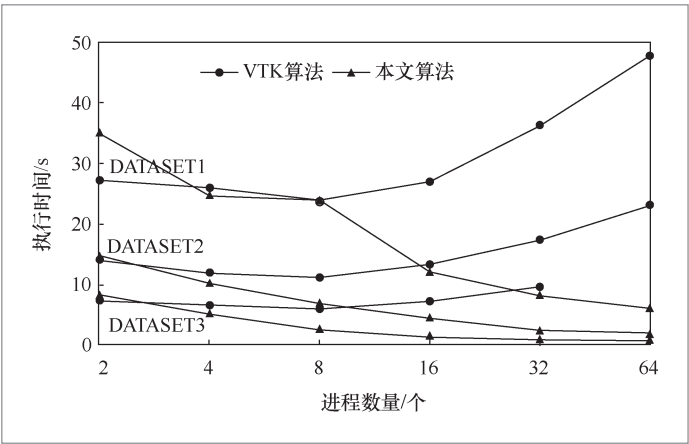


图 6 数据分割算法性能对比

3.2.1 算法性能

由于本文使用的实验数据均是曲面网格,经过四面体化处理后,网格规模将会大幅度增加,分别为DATASET1(113 920 117)、DATASET2(46 412 858)、DATASET3(25 984 608)。所有的并行体绘制图片分辨率均为1 000×800。表3给出了本文算法在不同进程数量下的交互时间和绘制时

表 3 并行体绘制算法性能

数据	算法性能	进程数量/个						
		1	2	4	8	16	32	64
DATASET1	交互时间/s	0.06	0.06	0.05	0.06	0.05	0.06	0.07
	绘制时间/s	150.68	67.61	41.29	22.89	13.91	6.34	4.24
DATASET2	交互时间/s	0.05	0.04	0.04	0.04	0.05	0.04	0.06
	绘制时间/s	70.09	28.77	15.42	11.48	5.43	2.94	2.16
DATASET3	交互时间/s	0.02	0.03	0.03	0.02	0.02	0.02	0.04
	绘制时间/s	57.71	15.87	9.98	5.29	3.31	1.71	1.28

间,结果显示,随着进程数量增加,算法执行时间缩短,使用64个进程并行绘制时获得最佳性能。由于使用了两层LOD交互模型,交互开销非常低,均为毫秒级别,并且基本不受进程数量影响,可以满足实时交互需求。其中,在64个进程下,飞行器数据DATASET3周围的流场体绘制结果如图7所示。

3.2.2 加速比

为了进一步研究算法性能和瓶颈,统计了本文算法相对于串行算法的加速比。

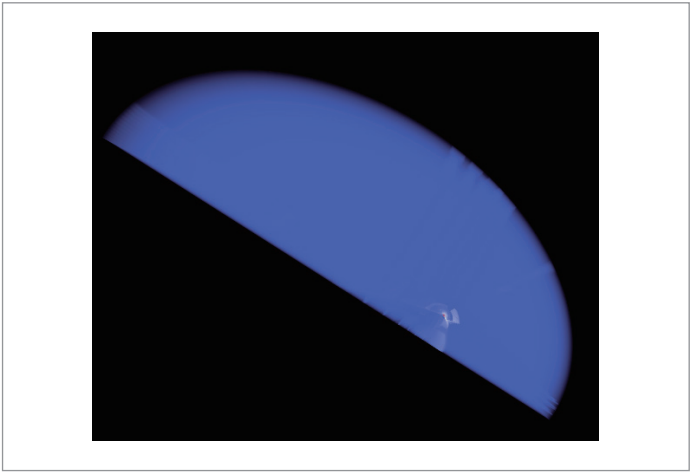


图7 飞行器 DATASET3 周围的流场体绘制结果 (64 个进程)

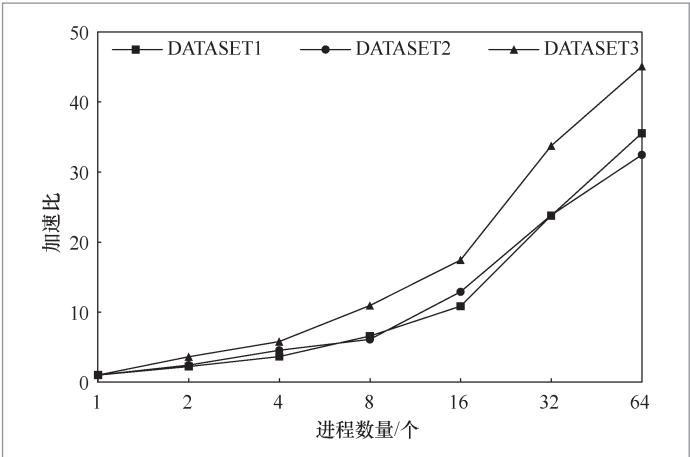


图8 并行体绘制算法加速比

如图8所示,当进程数量在8个以内时,增加进程数量,算法性能提升不明显;当进程数量大于8个时,算法加速比开始明显提升;当进程数量为64个时,获得最大平均加速比,为37.6。这是因为,进程数量在8个以内时,单个进程需要处理的网格单元数量较大,绘制阶段占据整个算法执行时间的绝大部分;进程数量大于8个后,当进程数量翻番时,单个进程需要处理的网格数量减半,绘制阶段占据整个算法执行时间的比例明显下降,加速比随之明显提升。

3.2.3 跨节点性能

由于单个计算节点性能有限,为了研究大规模网格体绘制性能,本节实验采用了32个计算节点。其中,使用的3个非结构网格数据规模为10亿级别,分别为yA31(12.1亿)、DATASET4(29.5亿)、DATASET5(46.5亿),所有的并行体绘制图片分辨率均为1 000×800。表4给出了实验结果,其中,“-”表示进程数量太少,不足以计算体绘制结果。根据实验结果可知,在不同进程数量下,交互开销基本一致,并且都在毫秒级别,满足了实时交互需求。其中,行星yA31撞击地球海面模拟数据的并行体绘制结果如图9所示。

4 结束语

本文提出了一种基于sort-last架构的非结构网格并行体绘制算法。整个算法分为数据分割、绘制和图片合成3个阶段。在数据分割阶段,设计了基于KD树的并行体数据分割算法,首先,使用预处理程序确保每个进程读取的网格数量大致相等;然后,在每个进程上同步构造局部KD树;最后,使用MPI API函数进行数据交换并合成全局KD树。此外,还根据相机位置和分

割平面计算了全局KD树叶节点的可见性顺序。在绘制阶段，在每个进程上建立可视化管线，使用高效的线程并行投影四面体算法计算体绘制图片。在图片合成阶段，设计了基于二叉树的图像合成算法，根据可见性顺序对体绘制图片分组合成。为了提升用户的交互体验，本文还设计了两层LOD模型，处于交互状态时，渲染开销较低的边框模型；停止交互后，再渲染开销较大的体绘制模型。

在Intel服务器和国产服务器上测试了本文的数据分割算法和并行体绘制算法。对于数据分割算法，首先，研究了不同进程数量下的网格规模增加情况；然后，测试了数据分割算法的负载均衡能力；最后，对比了本文算法与VTK算法的性能。实验结果显示，本文算法在不同进程数量下可保持良好的负载均衡能力；同时，与VTK算法相比，本文算法更具性能优势。对于并行体绘制算法，设计了节点内并行实验和跨节点并行实验，首先，测试了在节点内并行实验配置下算法的执行时间，并计算了本文算法相对于串行体绘制算法的加速比；然后，测试了在32个计算节点并行实验配置下算法的执行时间。实验结果显示，本文算法在64个进程下可以获得的最大平均加速比为37.6，由于采用了两层LOD模型，交互时间均控制在毫秒级别，本文算法可以很好地应用于大规模非结构网格体可视化，满足实时交互的需求。

本文实现了基于sort-last架构的并行体可视化原型系统，但是，还存在一些需要继续优化的地方。在数据分割算法中，并行计算分割平面的方向和位置是一个耗时的过程，下一阶段的研究工作可以考虑保存这些KD树分割平面信息，避免同一个数据被再次分割时重复计算分割平面。在图片合成算法中，使用over运算逐像素串行地合成两张图片，下一阶段的研究工作

表 4 并行体绘制算法性能（32 个节点）

数据	算法性能	进程数量/个				
		64	128	256	512	1 024
yA31	交互时间/s	0.09	0.10	0.06	0.06	0.07
	绘制时间/s	48.69	31.45	16.93	11.05	6.45
DATASET4	交互时间/s	—	—	—	0.08	0.06
	绘制时间/s	—	—	—	20.74	12.53
DATASET5	交互时间/s	—	—	—	0.08	0.08
	绘制时间/s	—	—	—	52.73	25.26

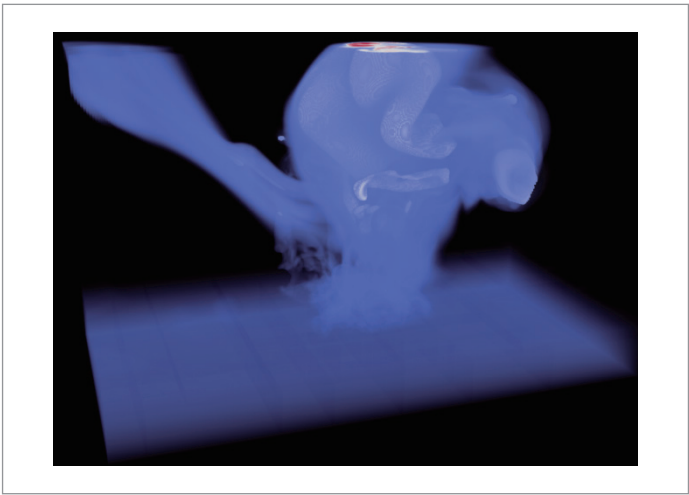


图 9 行星 yA31 撞击地球海面模拟数据的并行体绘制结果

可以考虑使用线程并行的方法优化over运算。在用户交互体验优化方面，由于体绘制模型渲染需要极大的开销，下一阶段的研究工作可以考虑过滤无效的鼠标事件，避免造成无效的体绘制更新。

参考文献:

[1] 袁晓如, 张昕, 肖何, 等. 可视化研究前沿及展望[J]. 科研信息化技术与应用, 2011, 2(4): 3-13.
YUAN X R, ZHANG X, XIAO H, et al. Visualization research frontier and prospect[J]. E-Science Technology & Application, 2011, 2(4): 3-13.

- [2] 李思昆, 蔡勋, 王文珂, 等. 大规模流场科学计算可视化[M]. 北京: 国防工业出版社, 2013.
LI S K, CAI X, WANG W K, et al. Large-scale flow field scientific visualization[M]. Beijing: National Defense Industry Press, 2013.
- [3] HANSEN C D, JOHNSON C R. The visualization handbook[M]. Amsterdam: Elsevier Press, 2005.
- [4] 陈为, 沈则潜, 陶煜波, 等. 数据可视化[M]. 北京: 电子工业出版社, 2013.
CHEN W, SHEN Z Q, TAO Y B, et al. Data visualization[M]. Beijing: Publishing House of Electronics Industry, 2013.
- [5] LEVOY M. Display of surfaces from volume data[J]. IEEE Computer Graphics and Applications, 1988, 8(3): 29–37.
- [6] WEILER M, KRAUS M, MERZ M, et al. Hardware-based ray casting for tetrahedral meshes[C]//Proceedings of the 14th IEEE Visualization. Piscataway: IEEE Press, 2003: 333–340.
- [7] SHIRLEY P, TUCHMAN A. A polygonal approximation to direct scalar volume rendering[J]. ACM SIGGRAPH Computer Graphics, 1990, 24(5): 63–70.
- [8] WYLIE B, MORELAND K, FISK L A, et al. Tetrahedral projection using vertex shaders[C]//Proceedings of IEEE Symposium on Volume Visualization and Graphics. Piscataway: IEEE Press, 2002: 7–12.
- [9] CALLAHAN S P, IKITS M, COMBA J L D, et al. Hardware-assisted visibility sorting for unstructured volume rendering[J]. IEEE Transactions on Visualization and Computer Graphics, 2005, 11(3): 285–295.
- [10] ELVINS T T. A survey of algorithms for volume visualization[J]. ACM SIGGRAPH Computer Graphics, 1992, 26(3).
- [11] FOGAL T, CHILDS H, SHANKAR S, et al. Large data visualization on distributed memory multi-GPU clusters[C]//Proceedings of the Conference on High Performance Graphics. Goslar: Eurographics Association, 2010: 57–66.
- [12] CHEN J H, CHOUDHARY A, SUPINSKI B D, et al. Terascale direct numerical simulations of turbulent combustion using S3D[J]. Computational Science & Discovery, 2009, 2(1): 015001.
- [13] EILEMANN S. Parallel rendering and large data visualization[D]. Zurich: University of Zurich, 2019.
- [14] CHOI B, KOMURAVELLI R, LU V, et al. Parallel SAH k-D tree construction[C]//Proceedings of the Conference on High Performance Graphics. Goslar: Eurographics Association, 2010: 77–86.
- [15] MORELAND K, AVILA L, FISK L A, et al. Parallel unstructured volume rendering in ParaView[C]//Proceedings of SPIE. [S.l.:s.n.], 2007: 1–12.
- [16] WILLIAMS P L. Visibility-ordering meshed polyhedral[J]. ACM Transactions on Graphics, 1992, 11(2): 103–126.
- [17] FAN L, CHEN C, ZHAO S, et al. Multi-threaded parallel projection tetrahedral algorithm for unstructured volume rendering[J]. Journal of Visualization, 2020, 32(3): 1–14.
- [18] PORTER T K, DUFF T D. Compositing digital images[J]. ACM SIGGRAPH Computer Graphics, 1984, 18(3): 253–259.

作者简介



范良 (1995–), 男, 西南科技大学计算机科学与技术学院硕士生, 主要研究方向为科学可视化与并行计算。



张晓蓉 (1976-), 女, 西南科技大学计算机科学与技术学院讲师, 主要研究方向为数据可视化、可视分析与人机交互。



吴亚东 (1979-), 男, 博士, 四川轻化工大学计算机科学与工程学院教授、博士生导师, 主要研究方向为科学可视化、信息可视化和可视分析。



陈呈 (1988-), 男, 博士, 中国空气动力研究与发展中心计算空气动力研究所高级工程师, 主要研究方向为并行计算。



王昉 (1976-), 男, 博士, 中国空气动力研究与发展中心计算空气动力研究所高级工程师, 主要研究方向为科学可视化。

收稿日期: 2020-12-31

通信作者: 张晓蓉, zhangxiaorong@swust.edu.cn

基金项目: 国家自然科学基金资助项目 (No.61872304)

Foundation Item: The National Natural Science Foundation of China (No.61872304)