海量超声数据体可视化研究

潘卫国[1],何 宁[2],薛 健[1],吕 科[1],翟 锐[1],代双凤[1]
(1.中国科学院大学工程管理与信息技术学院,北京 100049;2.北京联合大学信息学院,北京 100101)

摘 要: 近年来,随着科学数据的快速增长,海量数据的可视化分析成了急需解决的难题.越来越多的处理海量数据的方法向着并行、分布式处理的方向发展.本文提出了一种混合的框架来处理海量的超声数据,该框架通过整合多种硬件环境和计算资源来处理海量数据;所有的数据都存放在一个基于高速网络环境的数据共享中心,具有高性能显卡的前端工作站将耗时的处理任务分配到网络中的计算结点,而自身处理显示和交互的操作;同时基于OpenCL和OpenMP实现了可视化算法在GPU和CPU上的并行计算;核外算法应用在本框架中来处理海量的体数据.实验结果表明,本文提出的框架不仅可以处理海量数据,而且具有较高的交互性能.

关键词: 体绘制;图形处理器;核外技术;并行计算;海量数据
中图分类号: TP31 **文献标识码:** A **文章编号:** 0372-2112 (2016)02-0472-07
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2016.02.031

Research of Large Ultrasonic Data Visualization

PAN Wei-guo[1], HE Ning[2], XUE Jian[1], LÜ Ke[1], ZHAI Rui[1], DAI Shuang-feng[1]
(1. *College of Engineering and Information Technology, University of Chinese Academy of Sciences, Beijing* 100049, *China*;
2. *Beijing Union University, College of Information Technology, Beijing* 100101, *China*)

Abstract: In recent years, with the rapid growth of scientific data, large data analysis has become urgent problems. More and more large-data processing methods are modified to perform computation under parallel and distributed computing environment. In this paper, we present a hybrid architecture for large volume data visualization and processing. Various hardware environments and technologies are integrated in this architecture to perform interactive operations on very large volume datasets. All the datasets are stored in a data center with a gigabit network environment. The time-consuming data processing tasks are dispatched to the computing nodes connected to the same network, while the visualization and interaction operations are executed on a high-performance graphics workstation. OpenCL and OpenMP are used to implement volume rendering algorithms for accelerating visualization of a hierarchical volume data structure by both GPU and CPU with multi-cores, and some out-of-core algorithms are also presented to process the large dataset directly. The experimental results and practical application indicate that the hybrid architecture and methods presented in this paper are effective and efficient for the processing and visualization of very large volume datasets.

Key words: volume rendering; graphics processing unit (GPU); out-of-core; parallel computing; large data

1 引言

近年来,随着大型科学计算以及图形图像技术的快速发展,数据可视化领域待处理的数据量已经远远超过了目前的数据处理能力.在体可视化领域,随着体数据的快速增长,迫切期待计算设备的处理能力能够得到进一步提高;尤其是在国内外应用最广泛、使用频率最高的超声检测技术领域,因其具有检测对象范围广,检测深度大;缺陷定位准确,灵敏度高;成本低,使用方便;速度快,对人体无害以及便于现场使用等特点;正向着智能化、自动化、图像化、数字化、信息化和交叉领域的前沿方向发展.当然硬件设备本身的升级换代是最有效的途径,但在绝大部分条件下是不能够无限制地实现的.因此在现有的设备条件下如何能进一步发挥设备使用效率,如何有效提高体绘制速度就成为必然的研究趋势,而可以有效地满足这种对计算能力需求的途径就是充分利用网络分布式环境中的各种计算资源.

收稿日期:2014-06-19;修回日期:2015-03-02;责任编辑:诸叶梅
基金项目:国家自然科学基金(No. U1301251, No. 61271435);北京市自然科学基金(No. 4141003)

作为科学计算可视化技术的一个分支,体绘制是一种三维数据场可视化方法,在近二十年的发展过程中取得了相当大的进步,它利用图形学原理通过计算来显示三维数据场中有意义信息. 对三维数据场内的形体不进行专门的建模,不生成几何图元,二维图像由三维数据场经过计算直接生成,能够显示形体的内部信息. 该方法产生数据场的整体图像,具有图像质量高、对数据的表现力强、便于并行计算的优点,但计算量太大. 尤其是面对这海量的数据,如何快速的处理海量数据来提高绘制速度成为亟待解决的问题[1].

2　研究现状

体绘制最早由 Levoy[2] 于 1988 年提出,用于辅助科学家理解三维数据场,如 CT,MRI 类型的医学数据或有限元分析(FEA)的计算结果等. 该方法采用光线投射(ray casting)算法对数据场进行采样和积分,直接将三维数据转化为图像. Kniss 等[3] 引入了高维传递函数并设计了编辑界面,使得用户能够通过调节传递函数观察数据场中不同特征的分布情况.

海量数据的单机绘制方法可以分为硬件加速、数据压缩和核外计算;硬件加速利用图形硬件自带的三线性插值等功能将复杂的光线投射过程转移到图形处理器(Graphics Processing Unit,GPU)上进行,通过 GPU 的流式并行计算模式实现加速. 储骏[4] 提出了一种改进的算法,使用 Cg 语言编写顶点和片段着色程序,只需要绘制一个代理面就能获取光线的起始点,但其使用预先计算梯度并保存的方法获取体素法向量,内存占用量大. 杨金柱[7] 等基于 CPU 的代理几何生成算法和 GPU 共同完成体绘制,很好的解决了多组织标定与重建速度优化问题. 张慧滔[5] 等人利用 GPU 来加速单层螺旋 CT 数据的重建. 袁斌[6] 针对均匀数据场可视化的问题,提出了一种改进的 GPU 光线投射算法,算法采用按需实时计算梯度的方法,省略无效体素的梯度计算过程,效率较高,但算法使用汇编语言实现,编程比较复杂. 基于 GPU 的体绘制虽然计算速度得到了提升,但是也存在着瓶颈;体数据将通过总线从系统内存传递到显存中,这是 GPU 流水线中最慢的一个部分,因此传递的数据量应该尽可能地小;对大纹理进行采样是一个相当耗时的操作,如果纹理的大小大于 GPU 内的纹理高速缓存,那么纹理采样耗时将会非常耗时. 复杂的像素渲染[7] 也会造成明显的瓶颈.

在数据压缩方面,Wang 等[8] 引入图像质量度量的方法,使得压缩后的数据更易于用户理解. Lindstrom 等[9] 则提出了一种对浮点格式数据进行快速压缩的方法,它能够与应用程序的 I/O 环节无缝连接且适用于可变精度的浮点或整型数据,这些方法与绘制过程无关,具有普遍的适用性. 但是这些方法由于对原始数据进行了压缩,降低了成像的质量.

在核外计算(out-of-core)方面,Farias 等[10] 最早将核外计算用于大规模非结构化网格的体绘制,利用外部存储器对每个网格单元执行求交、排序和积分操作,从而得到大数据的绘制结果. Gobbetti 等[11] 提出了一种只需遍历数据一次的核外计算方法. 这类方法利用有限的计算资源处理大规模数据,能够在一定程度上解决大数据的绘制难题,但效率远不如并行的方法. 汪萌等在海量多媒体数据可视化[12] 和辅助标记[13] 方面的研究也取得了卓有成效的效果.

近年来,利用 GPU 构建分布式的计算与可视化平台成为了可视化领域的研究热点. Fan 等[14] 利用 GPU 集群进行流体的仿真计算和可视化,提出了一种新的架构,通过设计双层的体系结构(粗粒度下操作全局纹理,细粒度下进行单节点的运算)将 MPI 与分布式共享内存(distributed shared memory DSM)有机地结合起来,从而大幅提升计算和绘制的效率. Fogal 等[15] 基于 MPI 在分布式的多 GPU 上进行了大规模数据的并行体绘制,通过使用 k-D 树对数据进行划分和负载平衡的优化,数秒内便能绘制出千亿级体素的数据集. 曹轶等[16] 基于本地并行机和分布式图形工作站,给出了一种混合并行绘制模型. 该模型的工作原理是先将源数据存留在并行机,然后通过并行机的多处理器发布远程绘制命令流,进而通过操控工作站的图形硬件完成绘制;后期图像合成在并行机上执行,以发挥共享存储通信优势. 但是这些处理方法需要后期执行图像合成的操作,所以系统的内存的大小将会成为处理海量数据能力的瓶颈[17].

在海量标量场数据处理和可视化领域,虽然国内外研究者已经取得了大量基础性研究成果[18],但仍然缺乏一个技术成熟,目标明确,方便可用的海量数据处理和可视化系统,本文提出了一种基于分布式混合架构的海量数据处理及可视化系统及与其相适应的数据处理和可视化方法,由分布式混合架构的海量数据处理及可视化硬件系统和运行此基础上实现相应数据处理和可视化方法的软件系统.

3　系统框架

本框架的主要的思想是充分利用现有的计算资源和硬件加速设备. 同时为了满足实时交互的需要,使用预览体数据的结构来提升交互性能. 本文所提出的可视化硬件系统包括前端工作站和计算节点;前端工作站要求性能较高,主要处理实时性和交互性要求较高的操作;计算结点则可以是各种类型的计算机或硬件,可以运行各种类型的主流操作系统,用于耗时操作的

分布式处理. 前端工作站上运行的是图形界面,负责渲染显示,交互操作以及各种计算任务的协调,同时负责将耗时的计算任务分配到各计算结点执行;计算结点上运行的是一个后台计算服务程序,监听从前端工作站发送过来的计算指令并调用相应的计算模块执行计算任务. 系统框架通过前端工作站和计算结点的协同工作可以处理海量数据. 系统的总体框架结构如图 1 所示:

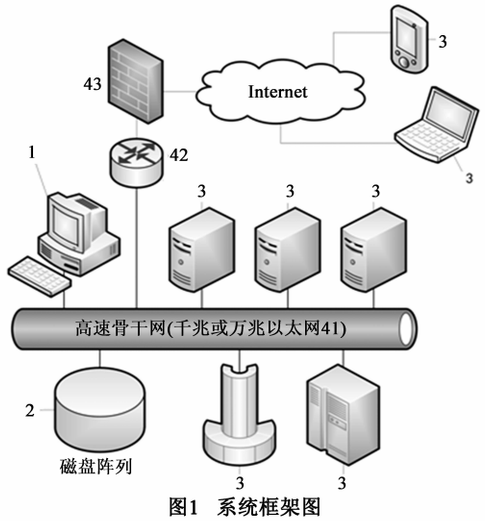

图1 系统框架图

从上述图 1 中可以看出,多种硬件环境和技术整合在系统框架中来完成对于海量数据的三维交互操作;所有的体数据都存储在数据存储中心,通过高速网络环境进行共享. 需要实时显示的交互操作由硬件加速和并行计算算法运行在前端工作站. 同时前端工作站也可以自适应地选择合适的数据尺寸进行处理. 耗时的计算任务通过高速网络分配到各个计算结点进行计算. 使用 OpenCL 和 OpenMP 实现了体绘制算法在 GPU 和 CPU 上的并行处理来加速绘制速度,同时使用了层次的体数据结构使交互级别达到实时的效果;核外算法应用在本框架中来处理海量的数据. 体数据首先通过前端工作站从数据共享中心加载,在加载的过程中将会根据初始体数据的大小来产生预览体数据和分级体数据,这样可以提升交互的响应性能. 如果初始体数据大于预设的阈值,该计算任务将会分配到相应的计算结点进行计算,当计算任务完成后,结果将会存放到共享数据中心并通知前端工作站进行显示.

3.1 分级体数据

当处理海量数据时,耗时的加载过程将会影响系统的总体性能. 本文提出的框架采用分级的体数据结构解决这个问题,如图 2 所示.

当开始加载数据时,系统首先产生一个预览体数据,这样使用者就可以在第一时间观测到所加载的体

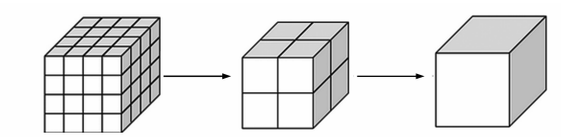

图2 分级体数据示意图

数据;第二步是根据加载数据的大小产生分级的体数据,如果产生的下一级体数据大于预设的阈值,将会按照图 2 继续分割,最后产生的体数据将会存放在一个列表中. 在这一部分,我们设计并实现了一个类来缩放相关的分级体数据,这个类的输入和输出都是一个体数据,可以自适应的来寻找要显示的体数据. 本系统中最小的预览体数据是 32 MB,这一阈值的确定与前端工作站的硬件性能有关,其基本要求是数据量等于这一阈值的数据做体绘制渲染时渲染速度至少要达到 15 帧/秒以上,以满足交互操作的需求. 这一预处理过程可以作为数据加载的一部分,由计算节点完成,处理完成后多级标量场数据存储在数据存储中心备用.

3.2 GPU 加速

目前,图形处理器的计算能力越来越强大,速度也越来越快,极大地提高了计算机图形处理的速度和图像质量,为可视化工作提供了很好的基础.

本框架利用图形硬件自带的三线性插值功能来完成光线投射算法中耗时的采样、插值过程,并将繁琐的合成运算转移到 GPU 上进行. 实现的基本流程图如图 3 所示:

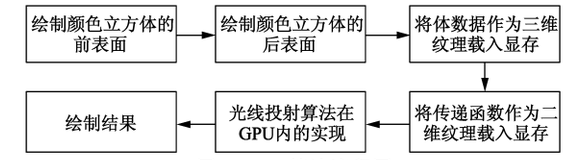

图3 GPU算法流程图

计算光线进入数据场的点:将颜色立方体的前表面绘制到一幅 2D 纹理上,此时由纹理像素的 RGB 值所组成的三元数组即为当前视线与由三维数据场映射成的 3D 纹理的第一个交点的坐标,即光线的进入数据场的点的坐标.

计算光线离开数据场的点:将颜色立方体的后表面绘制到一幅 2D 纹理上. 将这幅纹理和前表面纹理逐像素相减,就可以得到光线在数据场中行进的距离,设定合适的采样步长对数据场进行遍历.

将体数据作为 3D 纹理载入显存:使用基于 GPU 的光线投射算法需要在进行绘制之前将数据载入到显存中,所以在这一步进行时首先要确定对体数据进行插值前分类还是插值后分类,因为只有确定分类方法,才能确定将体数据以何种形式映射到显存中. 采用插值前分类方法,不仅需要 4 倍的存储空间(R,G,B,A 四个

通道,即体数据大小×4),而且由于采样时的插值对象不再是原始数据,会降低图像的质量,而采用插值后分类方法,直接将体数据作为 3D 纹理载入到显存中,就不会出现上述情况.

将传递函数表作为 2D 纹理载入显存:将传递函数存储为 2D 纹理是为了快速地将从数据场中获得的体素值转化为相应的灰度和不透明度值.

顶点着色器主要用来对图形的顶点数据信息进行坐标变换,即通过世界变换、取景变换以及投影变换等一系列变换将一个顶点由局部坐标系变换到齐次裁剪坐标系中. 在本程序顶点变换的过程中,为了保持着色器输出的纹理坐标与先前生成的颜色立方体的前后表面的纹理坐标相一致,需对输出的纹理坐标做如下操作:

$$\begin{aligned} Out.\ TexCoor.\ x &= (Out.\ Pos.\ x + 1)/2;\\ Out.\ TexCoor.\ y &= (Out.\ Pos.\ y + 1)/2;\\ Out.\ TexCoor.\ z &= 1\text{-}Out.\ TexCoor.\ y; \end{aligned} \tag{1}$$

式中 *Out. TexCoor. x* 和 *Out. TexCoor. y* 是由顶点着色器输出的纹理坐标,*Out. Pos. x* 和 *Out. Pos. y* 为由顶点着色器输出的顶点坐标.

像素着色程序的编写:这一部分是该程序的核心部分,光线投射算法中采样、插值、分类、合成运算以及对体元的光照渲染全在这部分程序中实现. 首先通过对写入前表面信息的纹理的访问来求出光线在三维数据场中的入射点位置,然后通过对存有后表面信息的纹理的访问来计算光线离开数据场时的点的坐标,接着给定一个采样步长来对数据场进行采样,通过依赖纹理提取获取采样点的 RGBA 值,最后采用由前往后的图像合成方法对得到的采样点的 RGB 值进行混合. 为了在重建出的三维物体中更加突出地显示不同物质之间的边界面,可以对其进行明暗计算,本文采用 Phong[19] 光照模型.

3.3　核外算法

一般将海量的数据以致于无法完全加载到内存的数据称为 Out-of-Core 数据,将处理这类数据的算法称为 Out-of-Core 算法. 在具体的实现中,计算框架的设计采用了如图 4 所示的数据流模型[20].

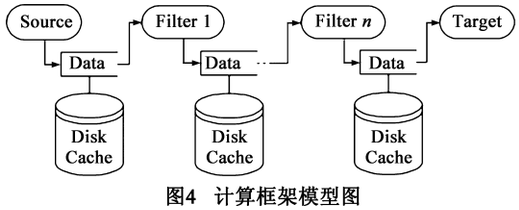

图4　计算框架模型图

在该数据流模型中,数据处理是核心,数据和算法分立于不同的抽象模块中. 数据和算法都由对象来表示,数据被抽象为 Data 类,处理数据的算法被抽象为 Filter 类,每一个 Filter 接受一组输入数据,经过处理产生一组输出数据,这样的一组 Filter 连接成一个流水线,构成统一的计算框架. 为了能够以统一的方式处理海量数据,每个 Data 对象都可以带一个磁盘缓冲,而 Data 类则封装了内外存数据交换的操作细节,对外提供了统一的访问接口.

我们先前的工作为 Out-of-core 数据设计了一套统一的访问接口[21],可以进行快速的数据交换,以块为单位存储当前正在处理的数据,同时维护了 4 个列表来管理整个缓冲区如图 5 所示:

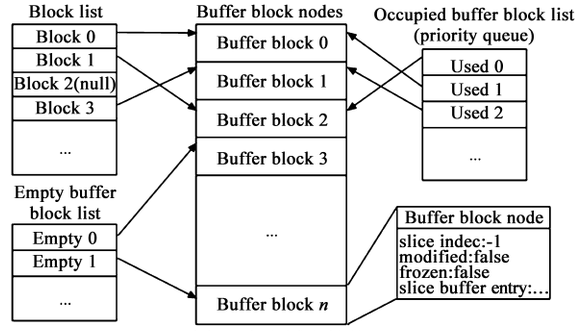

图5　内存缓存结构

Block List:一个线性列表,存储分块后原始数据的每一子数据块所用内存缓冲区的首地址,用于对子数据块的快速访问(可能为空,表示该子数据块尚未导入内存缓冲区);Empty Buffer Block List:一个环形队列,保存当前空闲缓冲区块的指针,这些空闲块可用于缓冲新的子数据块;Occupied Buffer Block List:一个优先队列,维护当前被占用的缓冲区块,同时决定当缓冲区满的时候哪一个缓冲区块的数据最优先被兑换回磁盘;Buffer Block Nodes List:一个线性列表,存储每个缓冲区块的相关参数和状态信息.

获取一个子数据块的基本操作按算法 1 步骤进行:

算法 1

步骤 1:检查 Block List,如果该 Block 内存首地址非空,则转步骤 5;

步骤 2:从 Empty Buffer Block List 取得一个空闲缓冲区块,若成功则转步骤 4;

步骤 3:取得 Occupied Buffer Block List 的头元素,其所指向缓冲区块中的数据若有改动的话写回磁盘,将该缓冲区块归还入 Empty Buffer Block List,然后转步骤 2;

步骤 4:将所需 Block 数据从磁盘导入所找到的空闲缓冲区块;

步骤 5:更新 Occupied Buffer Block List 和相关缓冲区块的状态信息;

步骤 6:返回存储 Block 数据的内存首地址或将 Block 数据拷贝入指定地址的内存区.

3.4　前端工作站和计算结点的协作

本文提出的框架在加载海量数据的过程中,包括

数据加载和处理环节. 可视化的软件系统运行在前端工作站上,包含图形用户界面,在计算节点必要的配合下,完成数据加载、数据处理、可视化和交互操作等工作. 数据加载的步骤包含以下算法 2 流程:

算法 2

步骤 1:计算待加载数据的数据量;

步骤 2:若待加载数据的数据量超过预先给定的阈值(是海量数据)则转步骤 3,否则直接将数据加载到前端工作站内存并转步骤 6;

步骤 3:遍历计算节点列表,查找可用计算节点,与可用计算节点建立 TCP 连接,将数据加载任务(包括待加载数据的名称、存储位置、加载参数等信息)传送到可用计算节点;

步骤 4:监听已建立连接的计算节点发回的状态信息;

步骤 5:若计算节点返回错误信息,则直接转步骤 6;若计算节点发回任务完成信息,则根据计算节点发回的加载(预处理)后的数据存储位置到数据存储中心读取相应数据并转步骤 6,否则转步骤 4;

步骤 6:结束数据加载过程并显示加载结果.

本文提出的框架针对标量场数据处理(如降噪、平滑、锐化等)分别实现其对应的内存算法和外存算法,内存算法运行在前端工作站上,用于处理普通规模的数据,外存算法运行在各计算节点上,用于处理海量数据. 基于这些基本算法,数据处理方法包括算法 3 步骤:

算法 3

步骤 1:计算待加载数据的数据量;

步骤 2:若待加载数据的数据量超过预先给定的阈值(是海量数据)则转步骤 3,否则直接在前端工作站执行相应内存算法对数据进行处理并转步骤 6;

步骤 3:遍历计算节点列表,查找可用计算节点,与可用计算节点建立 TCP 连接,将数据处理任务(包括待处理数据的名称、存储位置、数据处理命令及参数等信息)传送到可用计算节点;

步骤 4:监听已建立连接的计算节点发回的状态信息;

步骤 5:若计算节点返回错误信息,则直接转步骤 6;若计算节点发回任务完成信息,则根据计算节点发回的处理后的数据存储位置到数据存储中心读取相应数据并转步骤 6,否则转步骤 4;

步骤 6:结束数据处理过程并显示处理结果.

计算节点软件运行在各计算节点上,以守护进程的方式运行,监听特定端口,检测到有计算任务发来,则根据任务信息启动相应的外存算法对数据进行处理,并将计算状态信息(如进度信息、出错信息、结束信息等)返回给前端工作站. 由于计算节点软件只负责进行比较耗时的数据处理任务,没有显示和交互的需求,因此其软件结构和数据处理流程较为简单,可以部署在各种不同的硬件平台和操作系统上,也可以很方便地将已有的数据处理外存算法集成到计算节点软件中,不断丰富整个系统的数据处理功能. 此外,由于计算节点和前端工作站之间只存在比较简单的任务发布和状态回传的通信,计算节点故障不会影响前端工作站软件的运行,更不会引起整个系统的崩溃,因此整个系统的稳定性和可靠性得到了有效的保障.

4 实验结果

本系统采用 C ++ 语言实现前端工作站的 GUI 和计算结点上的后台服务程序,同时结合 OpenCL 和 OpenMP 实现了可视化算法在 GPU 和 CPU 的并行处理. 实验中前端工作站的具体配置:CPU:Core2 2.5GHz,内存:4GB,64 位操作系统,显卡:NVIDIA NVS5400,显存:1GB;计算节点使用的是浪潮服务器,具体配置:CPU:Xeon E5-2407 2.2GHz,内存:8G DDR3,硬盘容量:1TB. 实验中所使用的超声数据的从 10MB 到 25GB.

图 6 所示是不同大小的体数据在本系统处理后的效果,本系统在上述的实验环境下能够处理的超声数据可以达到 25GB.

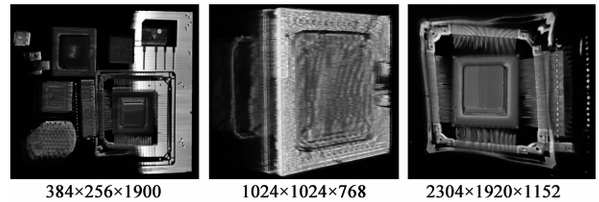

图6 不同超声体数据大小绘制效果

上图展示了本系统处理海量超声数据的能力,该框架不仅可以应用在无损检测领域,也可以扩展到医学数据三维可视化方面和气象数据三维可视化方面.

数据加载过程所耗费的时间是影响一个系统实时交互处理能力的主要因素. 图 7 展示了在加载不同大小超声体数据过程中所消耗的时间.

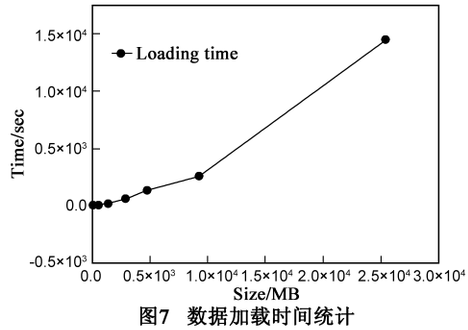

图7 数据加载时间统计

从图 7 可以看出当加载的体数据小于 500MB 时,加载所耗的时间可以看成是瞬时的. 当加载的体数据大于 1GB 小于 10GB 时,加载体数据所耗的时间也是可以接受的. 当体数据的大小增长到 25GB 时,所耗的时间较长;考虑到前端工作站可以同时分配多个计算任务到计算结点,这样可以减缓等待的时间. 为了提高系

统加载体数据所耗的时间,本系统采用在原体数据第一次加载过程中产生的 out-of-core 缓存数据,这样以后每次加载时所需的时间将大大减少. 如图 8 所示,从图中可以看出,数据加载所需的时间明显得到了提升. 即使 25GB 大小的体数据加载时间也在可以接受的范围,考虑到系统可以同时处理多个任务,这样可以抵消等待时间,使得加载的时间可以达到实时的效果.

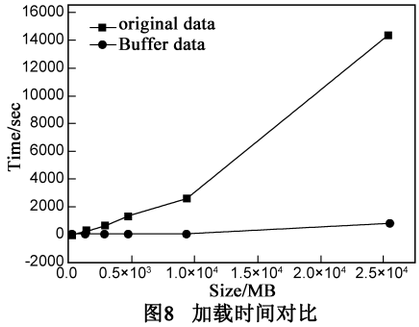

图8　加载时间对比

从上述可以得出,本文提出的处理框架具有在处理海量数据的能力,同时前端工作站具有高配置的显卡和内存,使得系统的交互性能也能够达到实时的效果.

目前大多数的体可视化系统,主要集中于跨平台,可扩展和交互性等方面开发,如 Volvis,3Ddoctor 和 Microview. 当数据大于内存容量时,这些系统将无法进行加载. ParaView 是与本文所提框架设计目标很相似的软件;与这些系统在加载数据方面的比较如表 1 所示.

表 1　系统加载时间对比

	Oursystem	Paraview	3Ddoctor	VolVis	Microview
10MB	0. 2s	0. 7s	0. 8s	0. 98s	1. 34s
11. 4MB	0. 3s	0. 9s	1. 8s	2. 3s	3. 53s
43. 5MB	0. 46s	1. 34s	2. 21s	2. 68s	4. 54s
87MB	0. 71s	4. 71s	5. 94s	7. 71s	9. 83s
260MB	2. 4s	5. 3s	6. 83s	8. 75s	12. 62s
416MB	17. 95s	25. 1s	39. 3s	45. 81s	55. 85s
1. 3GB	217. 7s	289. 2s	297. 52s	375. 2s	412. 6s
2. 8GB	637. 5s	683. 8s	958. 6s	1206. 1s	1390. 2s
4. 7GMB	1374. 3s	1569. 7s	N/A	N/A	N/A
9. 3GB	3565. 1s	2557. 5s	N/A	N/A	N/A
25. 6GB	14400. 3s	16820. 8s	N/A	N/A	N/A

从表 1 可以看出,当数据大于系统内存时,3Ddoctor,VolVis 和 Microview 都不能加载数据;而 ParaView 虽然在数据处理能力上和我们的系统较为接近,但由于其采用分布式存储和计算的模式,基于 MPI 实现计算节点之间的协同工作,导致计算节点配置复杂,对计算节点本身以及网络传输的可靠性要求较高,不利于构建稳定可靠的分布式海量数据处理和可视化系统.

5　结论与展望

本文提出了一种混合了 GPU 加速、核外算法、和分布式协同等技术处理海量体数据的框架,具有可扩展性强,节点配置简易和数据传输量小等优点. 从实验结果上可以看出,本架构在处理海量数据时可以达到实时的交互效果. 同时本框架还有很多可以改进的地方,如可以采用一种更有效的数据结构来组织体数据,将核外算法应用到 GPU 加速阶段,使并行处理的能力扩展到计算结点间等等.

参考文献

[1] 王纲,季振洲,张泽旭. 大规模真实感雪景实时渲染[J]. 电子学报,2012,40(9):1746 – 1751.
Wang Gang,Ji Zhen-zhou,Zhang Ze-xu. Large scale realistic snow scene real-time rendering[J]. Acta Electronica Sinica,2012,40(9):1746 – 1751. (in Chinese)

[2] Levoy M. Display of surfaces from volume data[J]. IEEE Computer Graphics and Applications,1988,8(3):29 – 37

[3] Kniss J,Kindlmann G,Hansen C. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets[A]. Proceedings of the Conference on Visualization[C]. Washington C:IEEE Computer Society,2001. 255 – 262

[4] 储骏,杨新,高艳. 使用 GPU 编程的光线投射体绘制算法[J]. 计算机辅助设计与图形学学报,2007,19(2):257 – 262
Chu Jingjun,Yang Xin,Gao Yan. Ray-casting-based volume rendering algorithm using GPU programming[J], Journal of Computer-aid Design & Computer Graphics, 2007,19(2):257 – 262. (in Chinese)

[5] 张慧滔,于平,胡修炎,张朋. 利用 GPU 实现单层螺旋 CT 的三维图像重建[J]. 电子学报,2011,39(1):76 – 81.
Zhang Hui-tao,Yu Ping,Hu Xiu-yan,Zhang Peng. Single-sclice helical CT three-dimensional image reconstruction using GPU[J],Acta Electronica Sinica,2011,39(1):76 – 81. (in Chinese)

[6] 袁斌. 改进的均匀数据场 GPU 光线投射[J]. 中国图象图形学报,2011,16(7):1269 – 1275.
Yuan Bin. Improved GPU ray-casting for uniform grid[J]. Journal of Image and Graphics,2011,16(7):1269 – 1275. (in Chinese)

[7] 杨金柱,赵大哲,栗伟,耿欢,王艳飞. 基于 GPU 的体绘制算法研究[J]. 电子学报,2010,38(2):202 – 206.
Yang Jin-zhu,Zhao Da-zhe,The research volume rendering

algorithm based on GPU[J]. Acta Electronica Sinica, 2010,38(2):202 - 206. (in Chinese)

[8] Wang C, Garcia A, Shen H W. Interactive level-of-detail selection using image-based quality metric for large volume visualization[J]. IEEE Transactions on Visualization and Computer Graphics,2007,13(1):122 - 134.

[9] Lindstrom P, Isenburg M. Fast and efficient compression of floating point data[J]. IEEE Transactions on Visualization and Computer Graphics,2006,12(5):1245 - 1250

[10] Farias R, Silva C. Out-of-Core rendering of large, unstructured grids[J]. IEEE Computer Graphics and Applications,2001,21(4):42 - 50.

[11] Gobbetti E, Marton F, Iglesias Guitian J A. A single-pass GPU ray casting framework for interactive out-of-core rendering of massive volumetric datasets[J]. Visual Computing,2008,24(7):797 - 806.

[12] Meng Wang, Guangda Li, Zheng Lu, Yue Gao, Tat-Seng Chua. When amazon meets google: product visualization by exploring multiple web sources[J]. ACM Transactions on Internet Technology,2013,12(12):1 - 17.

[13] Meng Wang, Bingbing Ni, Xian-Sheng Hua, Tat-Seng Chua. Assistive tagging: a survey of multimedia tagging with human-computer joint exploration[J]. Journal of ACM Computing Surveys,2012,44(25):1 - 24.

[14] Fan Z, Qiu F, Kaufman A E. Zippy: A framework for computation and visualization on a GPU cluster[J]. Computer Graphics Forum,2008,27(2):341 - 350.

[15] Fogal T, Childs H, Shankar S, et al. Large data visualization on distributed memory multi-GPU clusters[A]. Proceedings of the Conference on High Performance Graphics[C]. Aire-laVille: Euro graphics Association, 2010: 57 - 66.

[16] 曹轶,莫则尧,王弘堃,袁斌. 协同分布式图形硬件的混合并行体绘制[J]. 中国图象图形学报,2008,13(7):1379 - 1384.
Cao Yi, Mo Ze-Yao, Wang Hong-Kun, Yuan Bin. Hybrid parallel volume rendering with distributed graphics hardware[J]. Journal of Image and Graphics, 2008, 13(7): 1379 - 1384. (in Chinese)

[17] 赵云松,张慧滔,赵星,张朋. 双能谱 CT 的迭代重建模型及重建方法[J]. 电子学报,2014,42(4):666 - 671.
Zhao Yun-song, Zhang Hui-tao, Zhao Xing, Zhang Peng. Iterative reconstruction model and reconstruction method for dual energy computed tomography[J]. Acta Electronica Sinica,2014,42(4):666 - 671. (in Chinese)

[18] 钱正莲,杨亦春,滕鹏晓,韩宝坤,王昌田. 阵列可视化噪声源检测中的声 - 光偏离校准方法研究[J]. 电子学报,2014,42(10):2092 - 2097.
Qian Zheng-lian, Yang Yi-chun, Teng Peng-xiao, Han Bao-kun, Wang Chang-tian. Study of calibration method of acoustic & video image deviation in microphone array's visualized noise identification[J]. Acta Electronica Sinica,2014,42(10):2092 - 2097. (in Chinese)

[19] Phong B T. Illumination for computer generated pictures[J]. Communications of the ACM, 1975, 18(6): 311 - 317.

[20] Zhao MC, Tian J, Zhu X, Xue J, Cheng ZL, Zhao H. The design and implementation of a C++ toolkit for integrated medical image processing and analyzing[A]. Proc of SPIE[C]. San Diego: Int'l Society for Optical Engineering,2004. 39 - 47

[21] 薛健,田捷,戴亚康,陈健. 海量医学数据处理框架及快速体绘制算法研究[J]. 软件学报,2008,19(12):3237 - 3248.
Xue Jian, Tian Jie, Dai Ya-Kang, Chen Jian. Processing framework and the fast volume rendering algorithms for out-of-core medical data[J]. Journal of Software,2008,19(12):3237 - 3248. (in Chinese)

作者简介

潘卫国 男,1984 年生于河北邯郸,中国科学院大学在读博士生. 研究方向为三维可视化,计算机图形学.
E-mail: asherbuu@163.com

吕 科(通信作者) 男,1971 出生于宁夏西吉,教授,博士生导师,主要研究方向为数字图像处理、计算机图形学、智能信息处理技术.
E-mail: luk@ucas.ac.cn