## 基于分布式文件系统的海量车载点云管理

摘 要：激光雷达，以其高精度和抗干扰能力强等特点而受广泛应用。然而，超大范围的地形激光雷达扫描造成的海量三维点云数据已超过当前单台电脑的存储管理能力，同时，如何进行海量点云数据的实时调度同样成为当下研究的难题。针对该问题，结合现在应用较为广泛的Hadoop分布式文件系统，本文开展了面向车载海量点云数据的分布式存储与调度技术研究。在研究车载海量点云的索引结构基础之上，采用了一种结合多级文件目录的点云分段八叉树索引存储与调度方法，为进一步实现分布式的海量点云管理奠定了基础。进一步对Hadoop分布式文件系统开展了深入学习与研究，并为了克服原始 HDFS 结构无法有效支撑大量点云索引小文件存储和调度的问题，引入HBase分布式数据库来支持高效率的海量点云索引文件分布式存储与调度。通过实验表明，本文所采用的方法能够有效的支撑车载海量点云的分布式存储与可视化调度。

关键词：车载激光扫描系统；海量点云数据；空间索引；分布式文件系统； Hadoop；HBase

对于车载激光雷达采集的海量三维点云数据，一般而言，平均每公里的点云数据量可达千万级，针对城区或高速道路的单次作业数据量可达GB级甚至TB级。而大范围的车载激光雷达海量点云数据的处理过程包括了精度控制、 点云滤波、特征提取等多个需要多人协同或流程化作业的关键环节， 传统的作业方式往往涉及到点云数据的多次存储、 索引构建及备份等过程，耗费了大量的时间和存储空间。因此，研究并实现车载海量点云的分布式存储和调度方法，以支撑海量点云数据的协同分布式作业模式对提高大范围车载点云数据处理效率具有重要的研究意义，也是目前亟需解决的关键问题。

目前，针对海量车载激光点云空间管理的研究，基本都是对点云建立索引组织及内外存调度策略为主。王彦民和郭明提出的大规模点云数据的二维与三维的混合索引方法，先按照点云扫描线的横向和纵向两个方向对点图像数据进行二分，生成二维矩形，再根据生成的矩形在三维上生成立方体， 从而构建 3D-MBB（minimal bounding box）树。它可以兼顾二维和三维的特征， 而且分割后的点图像数据没有重叠点[1]。陈驰提出的通过双层四叉树建立空间索引的方法， 先将空间分割成一个个矩阵，矩阵内的点云数据用四叉树索引，矩阵之间也通过四叉树索引，如此双层四叉树相较于传统四叉树建立索引速度更快，而且占用内存更小[2]。梁钰立提出的基于八叉树的金字塔结构模型点云数据组织方式，对点云数据建立了金字塔模型，更符合可视化过程的分层显示[3]。

针对分布式文件管理的方法，目前比较流行的有Storm，Apache Spark，MongoDB以及本文所采用的Hadoop。Storm 是一个分布式流计算平台，目前已经相当成熟，它擅长做复杂事件处理（complex event processing, CEP）和流处理（stream processing），具有适用场景广泛、良好的伸缩性、保证数据无丢失、异常健壮、良好的容错性以及支持多语言编程等优点[4,5]。Apache Spark 是一个与 Hadoop 相似的开源集群计算环境， 在性能和迭代计算上很有优势，可用来构建大型的、低延迟的数据分析应用程序。Spark 启用了内存分布式数据集， 它在交互式查询和优化迭代工作负载方面有很大帮助。 Spark 最大的优点在于可以以毫秒级的速度达到对一定数据量的查询， 并且在某些迭代计算的场景（如机器学习算法）中可以领先 Hadoop 数十倍甚至到一百倍[6]。MongoDB 本身是一个非关系型数据库，通过 key-value 的形式存储数据形成文档结构。而且它也支持分布式存储， 支持横向扩展和系统可扩展性， 对数据读写的速度相当快。不过 MongoDB 目前还不够成熟， 没有相对应的维护工具， 而且不支持数据的事务性工作[7,8]。涉及到云计算技术，应用最广泛的还是 Hadoop 基础架构。Hadoop 发展到现在，已经形成了自己的一套生态圈。从功能上说， Hadoop 可以称作一个“大数据管理和分析平台”。其中，Hadoop 框架中最核心的部分就是 Hadoop 分布式文件系统（HDFS）[9]和分布式计算模型 MapReduce[10]。Hadoop 适合做大文件处理，是目前搜索网页以及大数据挖掘最主流且应用最广泛的软件框架。本文所采用的的分布式文件系统就是Hadoop的底层部分——Hadoop分布式文件系统。

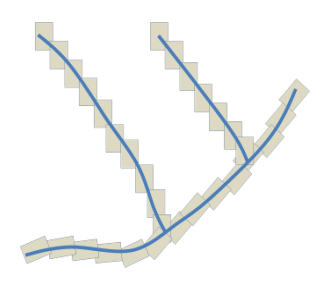


图2 沿POS轨迹的点云测段划分

本文针对点云索引采用一种分段及八叉树结构的海量点云数据管理方法，并将八叉树层级结构与文件系统的树形结构相结合，建立基于多层级目录的八叉树存储方法以及自适应 LOD 调度与更新机制[11,12]，并将其存储在Hadoop + HBase的分布式架构上，以有效支撑车载海量点云的实时可视化和后续处理过程。

### 多级文件目录的车载海量点云组织方法

传统的点云空间索引组织方法，无论是八叉树还是KD树的索引方法，都是将整个点云数据和模型建立在内存中，对内存占用较大，而且存在预处理时间长，数据更新困难，不支持并行访问的问题。因此，为了更好的支持车载移动测量点云数据高效快速处理，本文研究并采用一种适用于海量车载点云数据的管理调度和实时可视化的方法。 该方法将原始数据按采集里程或数量进行分段，对分段的每段数据建立八叉树模型， 并且设计了分段式多分辨率存储结构将点云数据以文件的方式存在磁盘里，有效减少内存占用，同时解决了快速更新和并行访问的问题。

#### 车载海量点云数据分段

作为车载数据处理的主体数据，由于其平面沿车载轨迹线性分布，因而尽管对于城市范围的 GB 级数据，在其解算或后期信息提取过程中，一般都是根据轨迹进行分段处理。为此，在车载点云索引组织过程中，提出并采用一种沿 POS 轨迹进行测段划分并针对每个沿轨测段进行八叉树 LOD 结构构建的方法，以实现海量车载点云数据的二级调度。沿POS轨迹的测段划分示意图如图2所示。

#### 多级文件目录的索引存储结构

在按 POS 轨迹对点云数据进行分段之后，针对每段数据构建八叉树 LOD 结构。并采用多级文件目录的方式进行存储，即按照文件目录表示八叉树层级的方法进行八叉树节点文件的存储。文件夹系统中的文件夹表示八叉树的节点，文件夹目录的层级表示八叉树结构的深度。节点文件夹中存放该节点包含的点云数据文件和属性文件。 索引结构如图3所示，文件名为 0-7 的文件夹表示该八叉树节点的子节点,后缀名为.node 的文件记录了该节点的属性信息（如当前深度、点云数目、外包盒大小等），后缀名为.pointcloud 格式文件存储了属于该节点内的点云数据。采用这种方式进行数据存储， 只需将原始数据读取一遍， 计算节点位置后写入到对应节点文件夹中即可。同时各个节点中的数据存储相互独立，任意节点的数据更新都不会影响其它节点的数据， 有利于并行访问。 并且这种分段式的结构也适用于 C/S 模式的软件架构， 很好的满足了高效管理调度车载点云数据的需求。

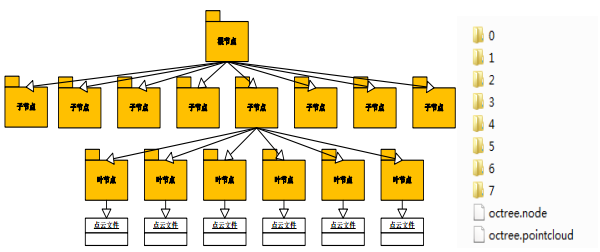


图3 文件夹式点云存储结构

八叉树的具体创建过程如下：

（1）计算八叉树深度，不同的八叉树层级存储不同分辨率点云的数据。为了计算八叉树的深度，可以如下列伪代码表示：

while(length>10){

length=length/2;

depth++;}

其中 length 表示外包盒边长， depth 表示八叉树深度。

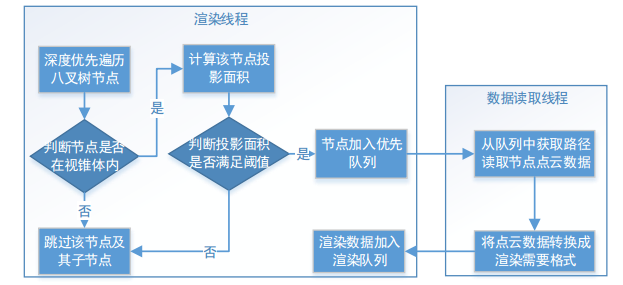


图4 点云渲染调度过程

（2） 计算八叉树节点位置。遍历点云文件， 逐点读入， 根据点坐标信息计算节点位置。计算公式如下，

其中*bx,by,bz*表示点在子节点的位置， *Xmid，Ymid，Zmid* 为节点中心坐标。以节点中心坐标为原点，1表示正半轴， 0表示负半轴， *num*为子节点的编号。例如，树深度为 5，计算得到节点路径为0-3-2-5-3，则在计算机中寻找路径 root\0\3\2\5\3，若该路径不存在，则新建路径，并创建点云文件， 写入该点；若该节点路径已经存在，则直接写入。

（3）更新子节点数据，反复第二个步骤，将所有点云数据分配到叶节点文件中，之后回溯自下向上更新中间节点的数据，生成整个八叉树数据文件存储结构。

#### 1.3. 点云快速渲染以及多线程动态调度

完成八叉树结构的点云多分辨率存储结构后，便可以实现点云数据的快速渲染。本文采用了视锥裁剪技术（View Frustum Culling）和多线程动态调度技术。视锥裁剪技术，即假设一个视点、 视线方向以及视角，判断点云八叉树节点是否在视锥体内。如果在则计算投影面积，并将节点信息加入到事先建立的优先队列之中。之后实现依次在优先队列中读取节点点云数据完成渲染（优先读取和渲染投影面积较大的节点，即离视点较近的节点）。 多线程动态调度技术， 即采用多线程的方式在队列中读取数据并转换为渲染所需格式的数据，加到渲染队列等待线程进行渲染。具体渲染流程如图3。

### Hadoop + HBase文件架构

前面介绍的几种传统的点云数据空间索引方法，都是基于本机内外存调度的单机模式。然而现在所进行的作业模式大部分都是团队协作，具有拥有大量的超过单台电脑存储能力的点云数据以及需要反复拷贝的问题，因此单机模式的点云管理方式对这种情况不是很适应。因此本文采用基于分布式文件系统的点云管理，将分段后的点云文件存储到分布式文件系统之中，有效解决数据存储问题以及团队协作下数据共享的问题。同时为了解决单纯的Hadoop只适应大文件读取的问题，引入HBase[13,14]作为分布式数据库对点云文件进行存储管理，从而很大程度上增加了小文件读写的效率问题。

#### **HBase** 访问接口

HBase访问接口有多种方式，不同方式的适用场景不同， HBase 的访问接口有Native Java API、HBase Shell、Thrift Gateway、 REST Gateway、 Pig和Hive等几种。由于目前针对点云的可视化平台一般采用C++语言实现，因此需要用C++语言实现HBase 的访问。所以采用Thrift Gateway 的方法。它利用Thrift 序列化技术，支持 C++、 PHP、 Python 等多种语言。

#### 基于HBase的车载海量点云组织方法实现

由于在HBase分布式数据库中无法支持类似于上一章节中的多级文件目录存储结构，只能采用“Key-Value”的方式来存储各分段点云的八叉树不同层级的索引节点。在车载海量调度过程中，首先根据视点获取当前需要渲染的索引节点唯一编码（见上一章相关内容），即“Key”值，然后通过访问接口从 HBase 中读取相应的节点数据，最后在节点数据解析基础上利用可视化平台进行节点中存储的点云数据的可视化。主要包括以下几个关键步骤：

1）对分段点云构建八叉树索引文件，并利用分段文件名称以及八叉树索引节点所在的层级进行索引节点的全局唯一编码，即“Key”值；

2）利用 HBase 存储接口，将每个分段文件的八叉树索引节点以二进制的方式存储到 HBase 中；

3）在数据的动态调度过程中，根据视点获取动态获取当前需要渲染的节点以及对应的“Key”值；

4）利用 HBase 读取接口，获取第 3）步骤中“Key”值所对应的二进制文件；

5）解析该二进制文件，获取文件中存储的点云信息进行可视化绘制；

### 3. 实验与分析

本次实验数据采用的是在陕西某城市采集的车载点云数据中经过分段后的连续八段点云数据，点云平均采样间隔为 5 cm，平均每段点云数据约为 800 万（500（米/段））。在实验过程中，分别针对八段点云数据依序构建八叉树索引并写入到HBase 分布式数据库中，并统计了不同段的写入时间。同时，为了测试 HBase 的调度效率，采用 OSG[15-17]渲染引擎实现了基于HBase的索引文件动态调度及可视化过程，统计了当加载不同分段数量点云的点云索引文件调度效率和渲染帧率，并进一步进行了点云分段数和节点数目与 HBase 的写入和读取时间的相关性回归分析。

本次实验平台中，服务器端 Hadoop采用伪分布式模式布置在一台 Linux 机上，其配置为intel(R) Core(TM) i7-3770 3.40GHz 8 核的 CPU，运行内存为8G，系统是Ubuntu 16.04 LTS 64位的版本。客户端采用家用笔记本电脑，配置为intel(R)Core(TM) i7-6700HQ 3.50GHz 8核的CPU，运行内存为16G，系统是 Windows10旗舰版64位。 连接网络为局域百M网络。本次实验的 Hadoop采用2.7.3版本， HBase采用 1.3.0 版本，thrift采用 0.10.0版本。

#### 3.1. 车载海量点云的可视化动态调度

对文件读取之后通过 LOD 调度和 OSG 渲染进行可视化。以下是实验图表。

表.1 多段多视角点云渲染帧率

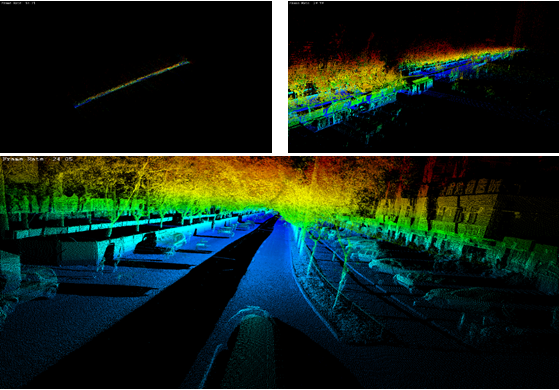


图5 车载激光点云远中近渲染图

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **段数** | **远** | **中** | **近** | |
| 1 | 91.68 | 41.22 | | 35.04 |
| 2 | 62.54 | 57.59 | | 29.98 |
| 4 | 40.08 | 29.99 | | 23.44 |
| 6 | 59.71 | 29.99 | | 24.05 |

如表1可以看出，远距离的点云图像由于点数较少，渲染帧率相较于近距离的要高很多。不过无论是远距离还是近距离，而且无论多少段的渲染，其帧率都在 20 以上，已经被认为是连贯的，证明论文方法能够有效的支持车载海量数据的动态调度。

#### 3.2. 不同段点云写入和读取速度对比

统计每一次写入和读取的平均速度（即图表中趋势线的斜率）， 单位是S/KB，计算读取或者写入 1KB 所花时间。 用直方图的形式表示如下：

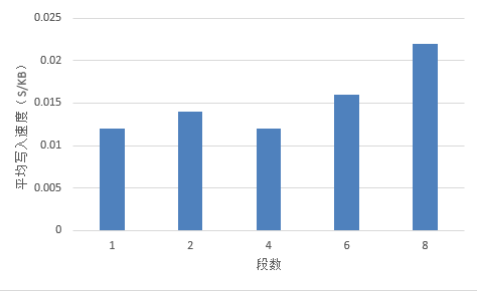


图6 平均写入速度对比直方图

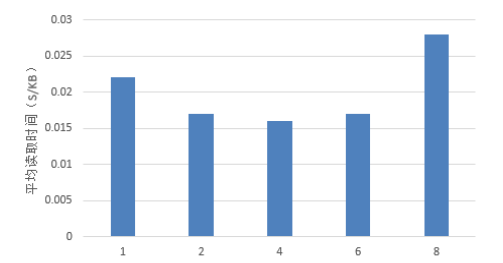


图7 平均读取速度对比直方图

在记录每次读写的平均速度并作成直方图后， 发现直方图高度相差不大，并且都在30ms/KB的速度一下，也就是 HBase 读取速度基本不受HBase 库中已有文件多少的影响，而且速度基本符合实时可视化要求。另外，实验时是客户端是通过无线网外网连接服务器的，还是在伪分布式模式下进行的测试，数据全都是从一台服务器上读取的。 在实际工作环境中如果有很多台服务器存储数据， 并且可以通过网线实现内网连接，读取的时候从不同的节点上获取数据，速度应该可以更快。

### 结语

本文首先从激光雷达扫描技术入手，研究分析了针对激光雷达扫描获取的海量点云的空间索引方法。然后根据车载激光雷达的特点，采用基于 POS 轨迹的分段方法，并针对分段点云数据采用多级文件目录的八叉树索引存储和调度方法，将得到的八叉树结构存储到磁盘里，有效减少程序内存的占用。

同时，针对点云节点文件大数据的存储问题，提出基于 Hadoop 分布式文件系统的存储方法， 通过 C/S 模式解决海量点云存储和数据共享的问题。进一步在分析 HDFS 特点以及实验过程中发现的原始HDFS结构对大量小文件存储不兼容的问题，引入 HBase 分布式数据库对小文件进行数据库式管理，有效提高了海量小文件的读写效率。

最后，本文采用 thrift 接口实现C++编程，结合OSG 渲染引擎以及多线程动态调度实现了从客户端到服务器读取节点文件并进行实时可视化调度的过程。

然而， 由于时间和精力有限，本文针对的分布式文件系统只采用了目前较为流行的 Hadoop 分布式文件系统。目前流行的分布式文件系统还有 Storm，Apache Spark， MongoDB等软件框架，进一步的研究方向可以通过实现不同分布式文件系统的点云节点文件的存储来比较它们的优缺点以及相应的改善点。

参 考 文 献（References）

[1] 王晏民, 郭明. 大规模点云数据的二维与三维混合索引方法[J]. 测绘学报,2012, 41(4):605-612.

[2]. 陈驰, 王珂, 徐文学等. 海量车载激光扫描点云数据的快速可视化方法[J].武汉大学学报(信息科学版), 2015, 40(9):1163-1168.

[3]. 梁钰立, 海量车载点云数据组织与快速可视化技术研究[D]. 首都师范大学,2012, 第 53 页.

[4]. 彭明喜. 基于 Storm 的实时数据平台研究[J]. 电信快报, 2015(7):24-28.

[5]. 马可. 基于 Storm 的流数据聚类挖掘算法的研究[D]. 南京邮电大学, 2016.

[6]. Shyam R, Bharathi G H B, Sachin K S, et al. Apache Spark a Big Data  
Analytics Platform for Smart Grid ☆[J]. Procedia Technology, 2015, 21:171-178.  
[7]. 丁洁, 基于 MongoDB 的机载 LiDAR 点云数据的组织与管理[D], 2016, 华南理工大学. 第 77 页.

[8]. 许伟平, 吴晨, 朱庆,等. 基于 MongoDB 海量地形小文件入库优化方法[J].地理信息世界, 2014(6):72-76.

[9]. Franklin M, Halevy A, Maier D. From databases to dataspaces: a new abstraction for information management[M]. ACM, 2005.

[10]. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[C], Conference on Symposium on Operating Systems Design &Implementation. USENIX Association, 2004:10-10.

[11]. 闫利, 胡晓斌, 谢洪. 车载LiDAR海量点云数据管理与可视化研究[J]. 武汉大学学报(信息科学版), 2017, 42(8):1131-1136.

[12]. 谢洪, 胡晓斌, 龚珣. 一种联合瓦片索引的车载海量点云数据管理方法[J]. 测绘通报, 2017(3):17-21.

[13]. 李振举等, 基于 HBase 的海量地形数据存储. 计算机应用, 2015(07): 第1849-1853 页.

[14]. 焦冬冬, 徐新国. 一种基于 HBase 的海量微博数据高效存储方案[J]. 微型机与应用, 2014(11):75-78.

[15]. Chang F, Dean J, Ghemawat S, et al. Bigtable: a distributed storage system for structured data[C], Symposium on Operating Systems Design and Implementation.USENIX Association, 2006:205-218.

[16]. 张元艺, 基于 OSG的道路交通事故三维演示平台设计[D], 2012, 长安大学.  
第70页.

[17]. 张丽娟, 基于 OSG 的矿井突水应急虚拟仿真系统关键技术研究, 2014, 中国矿业大学:北京. 第167页.