## DOI: 文章编号：1671-8860（2014）00-0000-00

## MapVis：一种基于多维度聚合的时空点状大数据可视化方法

谢冲1 关雪峰1 吴华意1

1 武汉大学测绘遥感信息工程国家重点实验室，湖北 武汉，430079

摘 要：对于大数据，可视分析是一项非常重要的研究手段，其有助于快速、直观地理解分析大数据蕴含的价值信息。但因海量、时空、高维等特征，大数据可视化存在内存消耗大、渲染延迟高、可视效果差等问题。针对上述问题，本文以海量时空点数据为例，采用预处理结果可视化方案，设计并实现了一套高可扩展的分布式可视分析框架。首先，本文借鉴瓦片金字塔模型提出了一种多维度聚合金字塔数据组织模型（Multi-dimension Aggregation Pyramid，MAP），该模型将瓦片金字塔的2D空间层级聚合扩展到时/空/属性多维度，同时支持时间、空间、属性的多维层级聚合。进而，本文以Spark集群作为并行预处理工具，以HBase分布式数据库持久化存储MAP模型数据，实现了预处理能力和存储能力的可扩展。最后本文实现了一套开源的分布式可视化框架（MAP-Vis），分布式处理能力支持下系统能实现亚秒级的查询响应，达到良好的交互式可视化效果。

关键词：大数据、多维、层级聚合、分布式数据库 、可视分析

中图分类号：P208 文献标识码：A

随着数据的采集手段成熟与多样化，包括个人智能设备、浮动车GPS、物联网、社交媒体等数据源日益丰富，采集数据量爆炸式增长。这些体量巨大、流式生成、类型众多、价值蕴含的大数据，兼具时空和高维特征[1][2]。时空特征，是指数据都带有空间位置和时间标签或者能体现时空位置的属性字段；高维特征，是指数据常包含多个特征属性，而这些属性所能反映出的信息规律往往更具研究价值，例如广泛应用的POI点数据。

可视化是时空大数据分析挖掘的一种重要步骤和手段，其能够直白地反映出数据中蕴含的模式和规律。随着数据规模的不断增大，现有数据可视化方法和工具越来越不适应。以点状数据可视化为例，如果常规逐点可视化，海量的数据点不但容易造成彼此堆叠压盖，超出用户感知能力，加之硬件设施负载有限导致可视化效率不佳。现有时空大数据可视化解决方案，一是采用高性能硬件环境提升可视化效率，二是基于预处理中间结果可视化方案。前者往往硬件成本代价不菲，内外存空间消耗巨大；后者将结构化的数据进行变换处理，映射为另一种相同或相近易表达的可视化元素，然后通过视图变换将中间可视化结构根据位置、比例、大小等做可视化输出[2]。但这些现有方案对大数据的时空、高维特征支持不够，尤其难以发挥分布式高扩展性的优势。因此，如何高效渲染、表达时空大数据的可视化内容，兼顾大数据的高维度信息，使人们能直观地感受其折射的信息内容，同时能充分利用到集群分布式处理的长处，是时空大数据可视化研究的一大难点。

Godfrey等提出，以可视化查询返回的数据类型划分，大数据交互式可视化类型可以分为：原始数据以及预处理数据[3]。

1）以原始数据作为返回类型

为了提高查询效率，最常用方式是提高处理计算机性能，或采用高性能分布式计算平台。例如，Google's Big Table Query（Dremel系统[4]是其商业版本），它是一个基于云平台的大数据分析网络服务，通过融合分布式计算、列存储、数据压缩等技术，支持亿级记录数据的在线查询分析，并实现交互式可视化。Todd Mostak 等[5]开发的大规模并行数据库MapD利用图形处理单元（GPU）对数据库进行加速处理复杂、实时的时空数据，能以毫秒级来分析处理数十亿行的数据。单节点的情况下，MapD能比集群的CPU数据库快至少2个数量级。

这类方法可以支持精确的单条数据查询，但缺点是所需硬件资源庞大，成本较高，且无法解决可视化出现覆盖、叠加现象，无法清晰的表达分布信息。

2）以预处理作为返回类型

可视分析最常用的概念是数据立方体（Data Cube）[6]，现有许多方法都是扩展此概念来建立。数据立方体，是多维数据结构的层级聚合的形式，包含上卷，下钻，切片，切块，旋转等操作，能够提供数据的多维视图，并允许用户预计算从而快速访问汇总数据。然而对于海量的多维数据集，立方体大部分单元为空，导致存储冗余非常大，内外存消耗过大。

为解决存储冗余问题，许多学者采用多种扩展结构减少数据立方体的内存。Dwarf[7]

通过前缀、后缀冗余来压缩数据立方体，减少内存消耗。imMens[8]将高维数据立方体划分为多个子立方体达到降维的目的，从而减少总内存消耗，并通过GPU并行处理渲染减少查询时间延迟，其缺点是能够支持的最大维度是四维，且无法自由扩展到更高属性维度。Nanocubes[9] 扩展了Dwarf思想，提出了一种基于内存的树结构，其添加了不同层级的空间维度、时间维度、以及属性维度，能够支持多维度以及时空的交互式查询。尽管在数据立方体得到压缩后，Nanocubes效率提升，能够在单机上对一定数据量的大数据进行交互式可视化，但由于其是基于内存的紧耦合结构，难以水平扩展到分布式架构。

以上实现都是在单机上对数据进行聚合从而实现内存消耗减少，而目前基于分布式平台的聚合方法也日益增多。VisReduce[10]，是通过分布式NoSQL数据库来扩展，在线聚合之后再返回查询结果，其聚合过程采用了MapReduce算法以及数据压缩方法。但该方案是对原始数据在线聚合，随着数据量增大，聚合过程中会消耗更多的时间，使其难以支持交互性可视化。TBVA[11]是通过Spark集群[12]对大数据进行离线预处理，提前生成不同空间层级的瓦片数据，并计算出各个瓦片的属性维度的统计值，达到交互式可视化效果，然而TBVA不支持空间/时间范围查询，不支持时空特征分析。GeoMesa[13]使用Cassandra/HBase等数据库对时空数据进行分布式存储，在处理数据过程中可以使用MapReduce或者Spark处理，当对于大数据进行可视化时，通过时空索引快速获取原始数据并在线进行处理，最终返回聚合结果，也存在随着数据量的增大在线交互式性能显著降低的问题。

综上，面对时空高维大数据，基于数据预处理的可视化仍是主流实现方式。在已有的单机解决方案中，数据量仍有很大的限制，很难具备数据容量的水平扩展性；在已有的分布式解决方案中，在线聚合的实现较难做到流畅的交互式可视化。针对上述问题，本文以时空点类型数据（如出租车点位数据、签到数据等）为可视化对象，提出了一种多维度聚合金字塔数据组织模型，兼顾时空和多维度属性以及地图显示的层级尺度，并设计实现了一套面向此类时空大数据的开源可视化框架，从而解决了大数据可视化时扩展性不佳、查询性能低、时空/高维支撑不足的难题，实现快速查询可视化的效果。

2 　多维聚合金字塔模型（MAP）

2.1 2D金字塔模型

GIS中，瓦片地图金字塔模型常用于2D地图显示，它是一种多分辨率的层次模型，从底层到顶层金字塔的分辨率越来越低。金字塔包含L个图层（Layer/Level），每个图层由M×N个正方形瓦片（Tile）拼接而成，每个瓦片再由2n×2n像素（Pixel）组成（一般默认为256×256），而像素值就是可视化的内容。为了更准确描述2D金字塔模型构建的过程，本文定义如下概念：

1. 像素（**Pixel**）：像素是MAP模型的最小数据单位，即瓦片金字塔层级中瓦片中的单个像素。如式1表示，其中l指代金字塔的层级，x、y指代该像素包含的空间范围，*p*代表在该像素对应的空间范围内的特征采样值，或聚合统计值，如均值，总和等。

|  |  |
| --- | --- |
|  | （1） |

瓦片（Tile）：瓦片金字塔模型中的瓦片，是空间邻近（n×n邻域）的像素组合结果，默认分辨率为256\*256。如式2表示,其中l代表金字塔的层级，X、Y为瓦片的行列号，表示该瓦片空间范围内的像素集合。

|  |  |
| --- | --- |
|  | （2） |

（Pyramid）：即各个层级瓦片共同

组合成一个空间多尺度的模型，可如式3所示：

|  |  |
| --- | --- |
|  | （3） |

以上述概念为基础，金字塔模型构建还

需对应操作，共同完成对空间维度层级聚合。定义的四个基本操作如下：

Extract操作

Extract操作负责将每一条原始记录映射到像素单位上，一个像素可能对应一条或多条原始记录， Extract操作公式如式4。其中需要说明的是，ln指代金字塔的最大层级，[Lon, Lat]代表原始记录数据的地理位置，w为定义的瓦片的分辨率。

|  |  |
| --- | --- |
|  | （4） |

Group操作Group操作主要是建立像素与瓦片之间对应的映射关系，即求得像素应隶属的具体瓦片，而操作公式如式5所示：

|  |  |
| --- | --- |
|  | （5） |

Key-value操作

再经过Group操作得到最大层级的瓦片后，我们可以得到瓦片及其聚合数据，可以将其键值化，令Key为瓦片的四叉树/空间填充曲线编码，Value为该瓦片的像素的集合。键值化操作为后续分布式存储建立基础。

Aggregate操作

不同于Group操作是对单一瓦片内的像素进行组合，Aggregate是层级间瓦片的聚合。以上的3个操作可以获取金字塔最大一层的瓦片集合，接下来需逐级往上聚合，得到各层级的瓦片，最终得到金字塔模型。Aggregate操作公式如式6所示：

|  |  |
| --- | --- |
|  | （6） |

2.2 多维金字塔模型

多维度聚合金字塔（Multi-dimension Aggregation Pyramid， MAP）模型是在传统的瓦片金字塔模型的基础上，提出的时空多维层级聚合模型。在空间维，其以瓦片金字塔的方式进行逐级聚合；在时间维，其通过预先设定的时间粒度对数据进行划分重组；在属性维，其通过扁平化的属性聚合树进行各维度属性值的聚合。其中，时间、空间维属于定位特征的，二者构成的整体标识一个个时空单元，并作为时空单元的Key，离散的属性维聚合树为Value，两者以键值对（Key-Value-Pair）的方式对应。这样实现时空聚合的同时，属性维也随之聚合，得到一个包含空间、时间、属性维度的多维时空聚合金字塔模型，模型如图2所示。相较上节介绍的2D金字塔模型，同样定义类似的概念和操作。

时空像素（Spatio-Temporal Pixel

, ST-Pixel）

有别于2D地图瓦片的像素零维特征，时空像素是1维的结构，是容纳属性维信息的容器，也是MAP模型的最小数据单位。其延伸了瓦片金字塔像素概念的内涵，除了具有空间坐标范围，还添加了时间刻度坐标。其中t代表预设的时间粒度刻度坐标，faa\_tree代表属性聚合树，是可视化内容的真正来源。

|  |  |
| --- | --- |
|  | （7） |

2） 属性聚合树(Flattened Attribute Aggregation Tree, FAA\_Tree）

属性聚合树是对时空对象的属性各维度单元进行逐维度聚合，从而所形成的树形结构，以广度优先遍历的结果简化实际的存储空间。定义公式如下, 以图2下方表格所示为例，4条出租车的记录，每条记录有三个属性，分别为载客与否、车辆颜色以及司

机性别。图中的4条记录，通过统计聚合得到右侧的属性聚合树结构，再对其做序列化，得到可结构化存储的定长一维数组。相较于树形结构，定长数组结构为后续的储存以及处理提供了极大的便利。

|  |  |
| --- | --- |
|  | （8） |

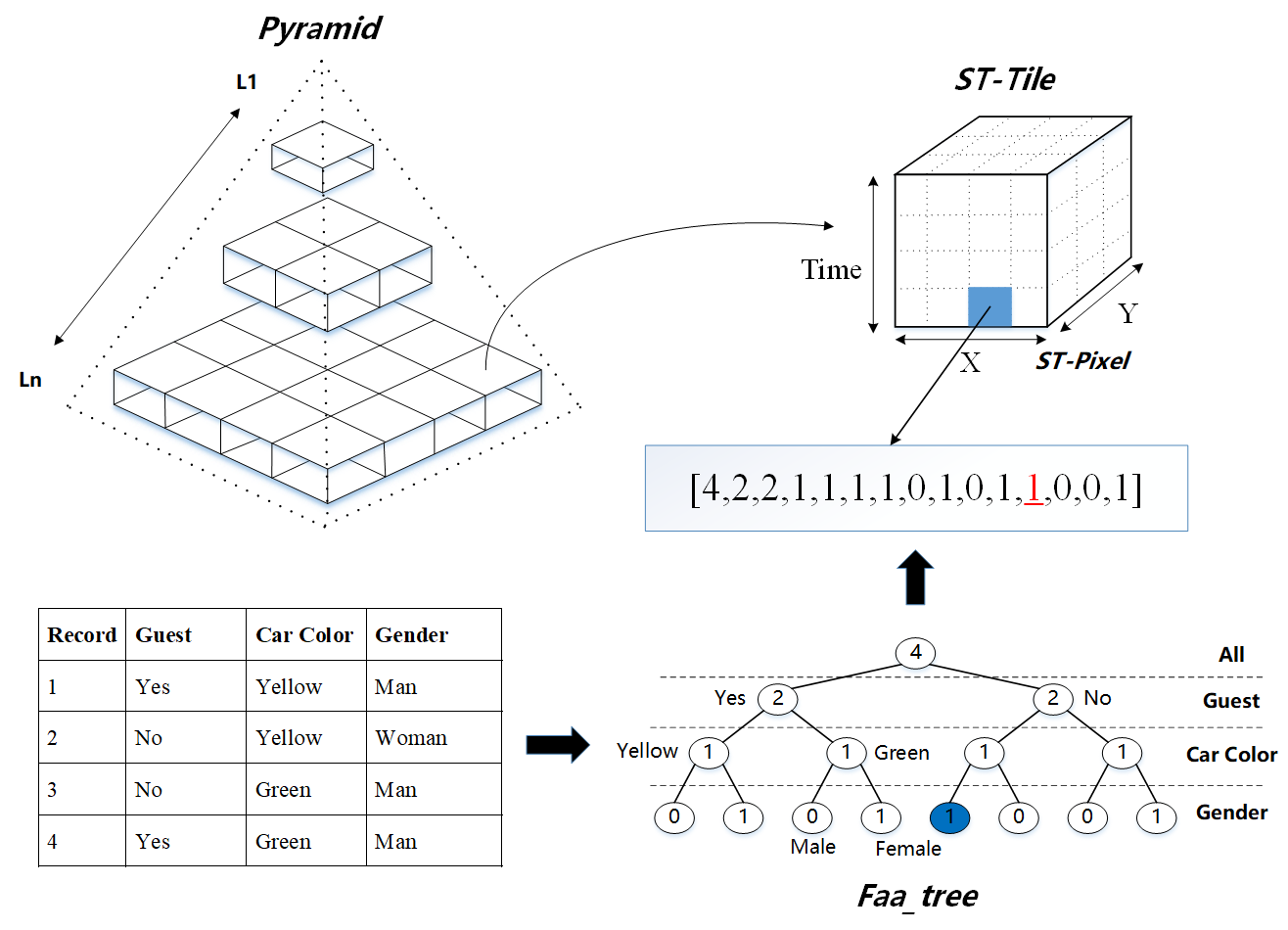


图1 多维金字塔模型

Fig.1 Multi-dimension Aggregation Pyramid

* 1. 时空瓦片（Spatio-Temporal Tile, ST-Tile）

时空瓦片的概念类比于2D模型的瓦片，由时空像素组合而成。时空瓦片包含时空像素，时空像素挂载了属性维聚合树，就形成了一个可实现层级聚合的时空立方体。时空瓦片是MAP模型显示基础单位，它的定义为式9的五元组，其中*l*同样指代金字塔的层级，*X、Y*指代瓦片的行列号，T代表预设了时间粒度的时间单位，Faa\_tree是瓦片层面的属性聚合树。

|  |  |
| --- | --- |
|  | （9） |
|  | (10) |

从公式9以及图2可以看出，定义的时空瓦片从概念上已经不再是二维的平面，构建所需的基本操作对应进行了扩展。

**1） Extract**操作

除了提取地理位置信息，还需提取时间信息，以及属性维度，生成属性聚合树并序列化。

|  |  |
| --- | --- |
|  | （11） |

1. **Group**操作

分为2步，首先主要是建立时空像素与时空瓦片之间对应的映关系，即求得像素应隶属的具体瓦片，而不止在空间上定位其属于某一瓦片，在时间上该像素的归属也是确定的。操作公式如下：

|  |  |
| --- | --- |
|  | （12） |
|  | |
|  | （13） |
|  | （14） |

其中，下标*T*表示每一个对应关系都是有时间限定，因此通过该公式可找出每一个像素所属的时空单元。公式12的作用是将时空像素映射到具体的时空瓦片上。Group操作的第二步即是对所有映射相同时空瓦片的时空像素进行归约，操作如公式13所示，Group操作是以时空瓦片作为Key来reduce所有像素的过程，即时空瓦片***ST-Tile***的聚合树*Faa\_tree*是所有时空像素***ST-Pixel***的一维定长数组*faa\_tree*对应位置累加。

3） **Key-value**操作拆分*Faa\_tree*里的每一个节点，并分别将节点与瓦片的四叉树编码、时间*t*联合编码，生成Key，每个节点对应得像素集合作为Value，生成键值对。其真实意义即在该瓦片的空间范围内，该时间段*t*内，发生某一属性节点的事件统计结果。

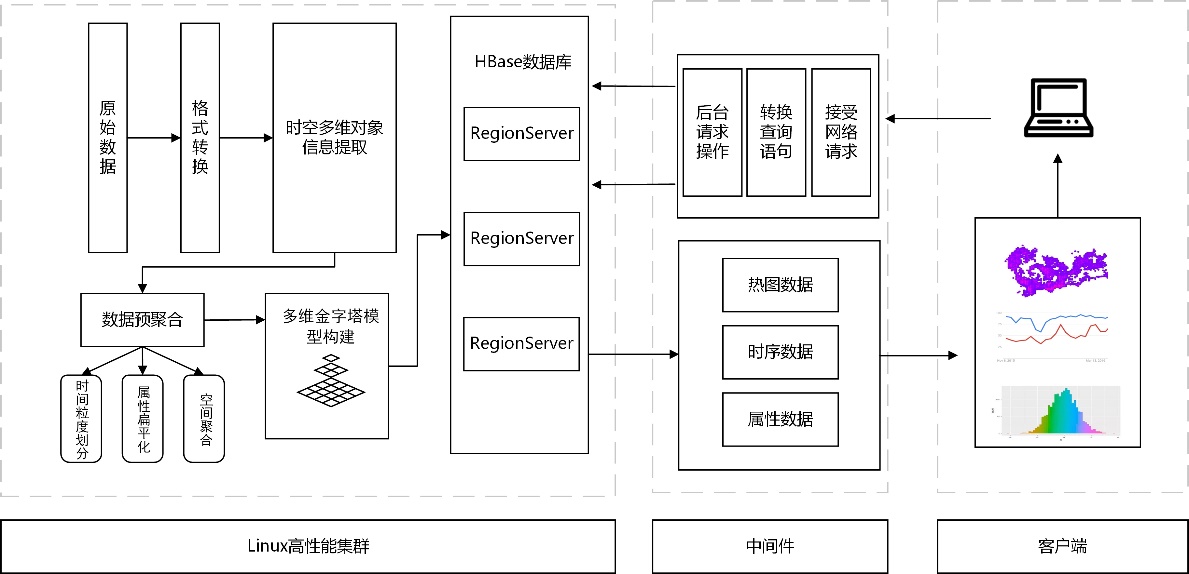


图 2 Map-Vis系统架构图

Fig.2 Multi-dimension Aggregation Pyramid

4）**Aggregate**操作，即在对于key-value化之后的瓦片进行聚合，即需限定在同一时间，同一属性节点attribute的瓦片。公式15表示**Ψ**代表一个可选的算子，可选择属性值以何种计算方法（求和、平均、最大最小值等）向上聚合。



3 基于MAP模型的时空大数据可视化框架(MAP-Vis)

**3.1** 框架总体架构

为验证本文提出的MAP模型和可视化

框架的有效性及性能，本文实现了一套时空数据可视化原型系统MAP-Vis。如图3所示，MAP-Vis系统采用B/S架构，系统架构分为客户端，中间件，以及后台集群三部分组成。其中，预处理及存储模块建构于Linux高性能集群之上，其主要负责原始数据的多维金字塔模型建立以及预处理数据的分布式存储；中间件包含数据访问接口，其主要解析用户的查询请求，并从数据库端抽取相应数据作简单的统计操作；客户端主要负责对从后台获取的热图数据、时序数据和属性数据进行可视化。

4.2 时空数据预处理

在MAP数据模型构建中，数据是由自下而上不断聚合抽象，首先需完成最精细一层，即最大层级的时空瓦片，再不断往上层聚合迭代生成上面层级。迭代聚合过程的伪代码如表1所示，且预处理步骤如下：

1. 输入参数，包括预处理的原始记录、空间的最大层级ln，以及时间粒度。提取原始记录的地理信息，时间信息，以其

他属性维度信息，得到每条记录对应的属性聚合树，并利用3.2节介绍的Extract操作，得到相应的ST-Pixel；

表1 预处理伪代码

|  |
| --- |
| **Algorithm 1** Preprocessing for the system |
| **Input:** 原始数据, Rm(*lng, lat, time, categories*), m=1, 2, …, N;  **Input:** The maximum map level, *l*n;  **Input:** The time granularity;  **Output:** *l*n(*i*=0, 1, 2, …, n), hierarchical aggregation result;  1: **for** each record R*i* in Origin data **do**  2: cate = FaaTree(categories)  3: *t*k = tBin(time)  4: Pixel*ik* = Extract(*lng, lat, l*n*, cate, t*k)  5: **end for**  6: **for** each Pixel*ik* in the same Tile with  Tile*jk* **do**  7: Map Pixel*ik* to Tile*jk*  8: **end for**  9: **for** each Tile*jk* with cate = cat*all*, cat*1*, …, cat*s* **do**  10: Key-Value Tile*ik* to Tile*jks*  11: **end for**  12: Tileln ← All(Tilejks)  13: **for** *i* = *l*n-1, *l*n-2, …, 1 **do**  14: Tile*li* = Aggregate(Tile*li*+1)  15: **end for**  16: **return** Tile*li*, *i* = 0, 1, 2, …, n; |

2）利用Spark的Reduce操作，将属于同一

时间t内同一瓦片内的像素聚合起来，生成聚合数据。由于每个像素的属性聚合树是序列化的之后的数组结构，属性聚合树中每一个下标索引即代表一种属性字段，可以进一步对属性聚合树下的各个属性字段进行聚合，得到每个瓦片在不同时间下不同属性维度的聚合数据。对应3.2节介绍的的Group操作；

3）对瓦片的四叉树编码，时间序列t，以及属性聚合树中的各个节点联合生成Key，聚合之后的像素集合作为Value，生成键值对。最终得到最大层级ln中各属性节点下各个时间段的聚合数据，对应3.2节介绍的Key-Value操作。

4）生成最大层级ln所有瓦片序列之后，可以根据金字塔逐层聚合关系不断聚合便可以逐层得到ln-2，ln-3，…1层级的瓦片数据，构建多维度的聚合金字塔。对应3.2节介绍的Aggregate操作。

4.3 MAP数据组织与存储

为增强MAP-Vis的存储扩展性，本文选择分布式的HBase作为存储数据库。HBase作为Google BigTable的开源实现，是一款分布式、多版本、面向列存储的非关系型数据库 [14-17]。HBase主要用于存储海量的结构化数据，底层以HDFS作为存储系统[16,17]，具有可伸缩、随机实时读写、高可靠性的特点。

如图4为MAP模型的存储映射结构，其中图4可以看出，一个带有属性节点的时空瓦片可以由四叉树编码Quadkey，属性节点An，以及时间坐标t三个坐标唯一确定。再结合HBase的数据存储模型特征，我们对MAP模型数据设计了如下的表结构。横向上以Quadkey+An联合编码的方式作为每一行的Rowkey，设置2个列族分别为Heatmap列族和Sum列族。Heatmap列族按照时间刻度坐标设置各列，负责存储相应时间坐标t下的热图数据。Sum列族同样按照时间刻度坐标设置各列，负责存储相应时间坐标t下热图数据中所有像素的特征采样值总值。

本框架的表结构设计，没有把时间维度加入Rowkey，而是把既定的属性维度加入到Rowkey中，让时间做横向的扩展，属性做纵向的扩展。该设计出发点是，若时间维度信息加入Rowkey，则会因数据时间序列过长导致Rowkey的数目急剧增加，从而影响到查询效率。而对于属性维度而言，一般可视化关注的属性维度数量有限，属性聚合树层级不会过大。

4.4 面向可视化的时空查询

通常HBase数据访问方式是使用全盘扫描（Scan）或单点查询（Get）两种模式获取数据，用户在获取数据后在客户端进行业务运算。当返回数据量非常大的情况下，上述两种模式获取数据就会在网络I/O层面遇到瓶颈。为此HBase提出了协处理器（CoProcessor）的概念，通过协处理器可以在服务器端对查询范围内的结果进一步聚合，得到中间聚合结果返回客户端，从而提高查询效率。图4为MapVis系统基于HBase协处理器概念实现的时空查询示意图。Map-Vis系统可视化所需的查询是空间、时间、属性查询多维度的联动查询。客户端可视化时发送数据获取操作，中间件先分析请求参数，将请求内容分解为时、空、属具体查询类别对HBase数据库进行查询，各个Region的数据通过协处理器再聚合返回给客户端。

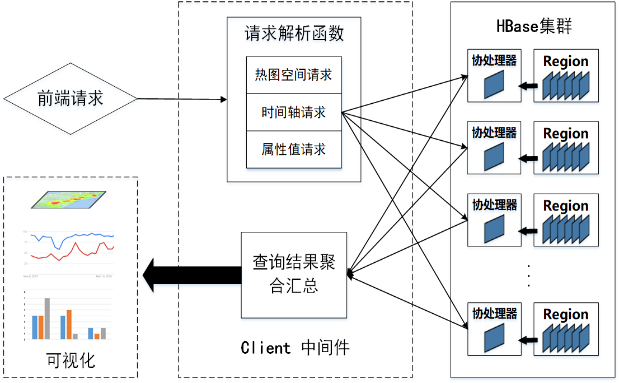


图3 MAP-Vis系统时空查询示意图

Fig.3 Map-Vis。。。。。。。。。。。。。。

空间查询，采用的是异步请求，即每一

张瓦片单独向服务端发送查询请求。当服务端接收到查询请求操作时，中间件将请求参数解析为HBase数据库的空间过滤条件，行键Rowkey为瓦片四叉树编码Quadkey与属性索引位置An组合，以Heatmap为列，列限定符为查询时间范围内的各个时间刻度坐标t，经初次查询后可以得到各个时间坐标的聚合数据。再调用HBase数据库端的协处理器，将查询时间范围内各时间节点的数据做聚合计算，便得到最终的聚合结果返回前端渲染。

时间查询，行键Rowkey为瓦片四叉树编码Quadkey与属性索引位置An组合，Sum

为列，查询时间范围内的各个时间刻度坐标

t为列限定符，在得到各个瓦片相应时间范围的数据后，在中间件上进一步统计可视范围内的瓦片序列的每一个时间坐标节点的总值，并最后将结果提交至服务端进行前端渲染绘制。

属性查询，查询条件与时间查询相类似，以行键Rowkey为瓦片四叉树编码Quadkey与各个属性索引组合，以Sum为列，查询时间范围内的各个时间刻度坐标t为列限定符，在得到各个瓦片在查询时间范围的总值之后，在中间件上进一步统计可视范围内的瓦片序列的每一个属性节点的总值，并最后将结果提交至服务端进行前端渲染。

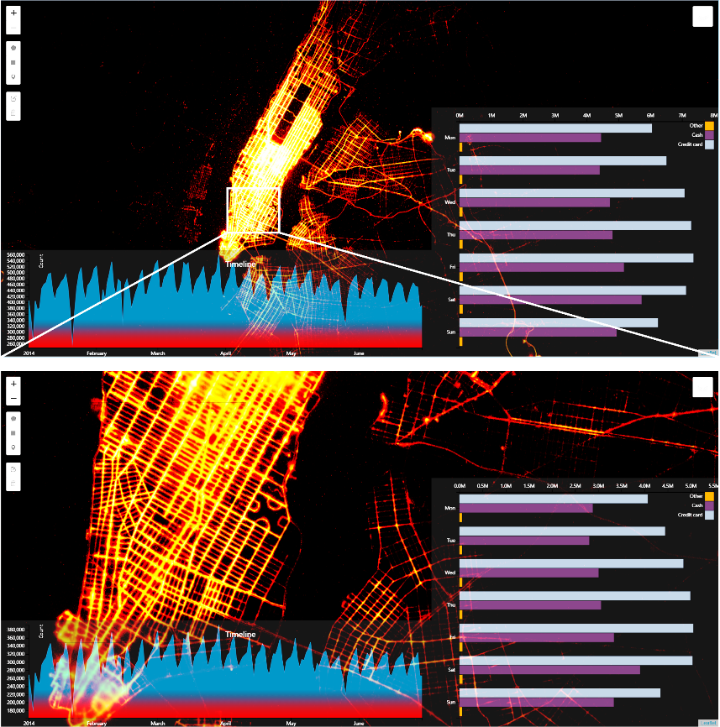


图4 MAP-Vis系统时空查询示意图

Fig.4 xxxxxxxxxxxxxxxxx

# 实验与分析

实验数据以纽约曼哈顿区的出租车数据为可视化对象，该数据集记录了2014年至2016年6月份共30个月的出租车上下客点位等信息，大小共计约54.5G，总记录数约3亿条。单条记录包含了出租车乘客的上下车空间位置，乘客支付方式等其他信息。所示，是MAP-Vis框架实现的出租车数据可视化显示界面，基于HTML，CSS，Javascript实现，地图显示使用Leaflet库[19]

进行交互、时间轴以及属性柱状图部分使用d3[20]库来进行交互。

实验分别从MAP模型的有效性、存储扩展性、预处理能力扩展性3个角度测试了MAP-Vis的数据可视化性能。集群测试的基本环境共有7个节点，集群的节点配置如表4所示：

表 2 集群节点配置情况

|  |  |
| --- | --- |
| 属性 | 配置信息 |
| CPU | 双路六核Intel(R)  Xeon(R) E5-2620 2GHz |
| Memory | 32G |
| Network | 1Gbps |
| OS | CentOS 6.2 64bit |
| JVM | JVM 1.8.0 |
| HBase | HBase 1.2.0 |
| Spark | Spark1.6.0 |
| Zookeeper | Zookeeper 3.4.5 |

实验一：模型的有效性



图5 MAP-Vis系统时空查询示意图

Fig.5 Map-Vis。。。。。。。。。。。。。。

该实验以查询的聚合数据所对应得原始数据大小作为自变量，其中通过改变查询的时间跨度进而控制所对应的原始数据的大小。由于原始数据共30个月，以3个月为间隔，在实验过程中，以时间跨度所对应原始数据为自变量，对地图进行完全相同的操作如放大缩小平移等，并统计各自的平均响应时间，其变化趋势如图6所示，其中平均查响应时间是从请求发起到最后渲染完毕进行计时，中间包括网络传输等因素，总体可以看出其响应延迟稳定在700ms-1000ms之间，能够保证亚秒级的响应，达到流畅的交互可视化体验。图6同时显示，随着原始数据规模的扩大，响应时间曲线基本保持平稳，未呈明显的线性增长，说明MAP模型应用于可视化时能够很好地适应大规模的时空数据集。

实验二：数据分布式存储扩展性

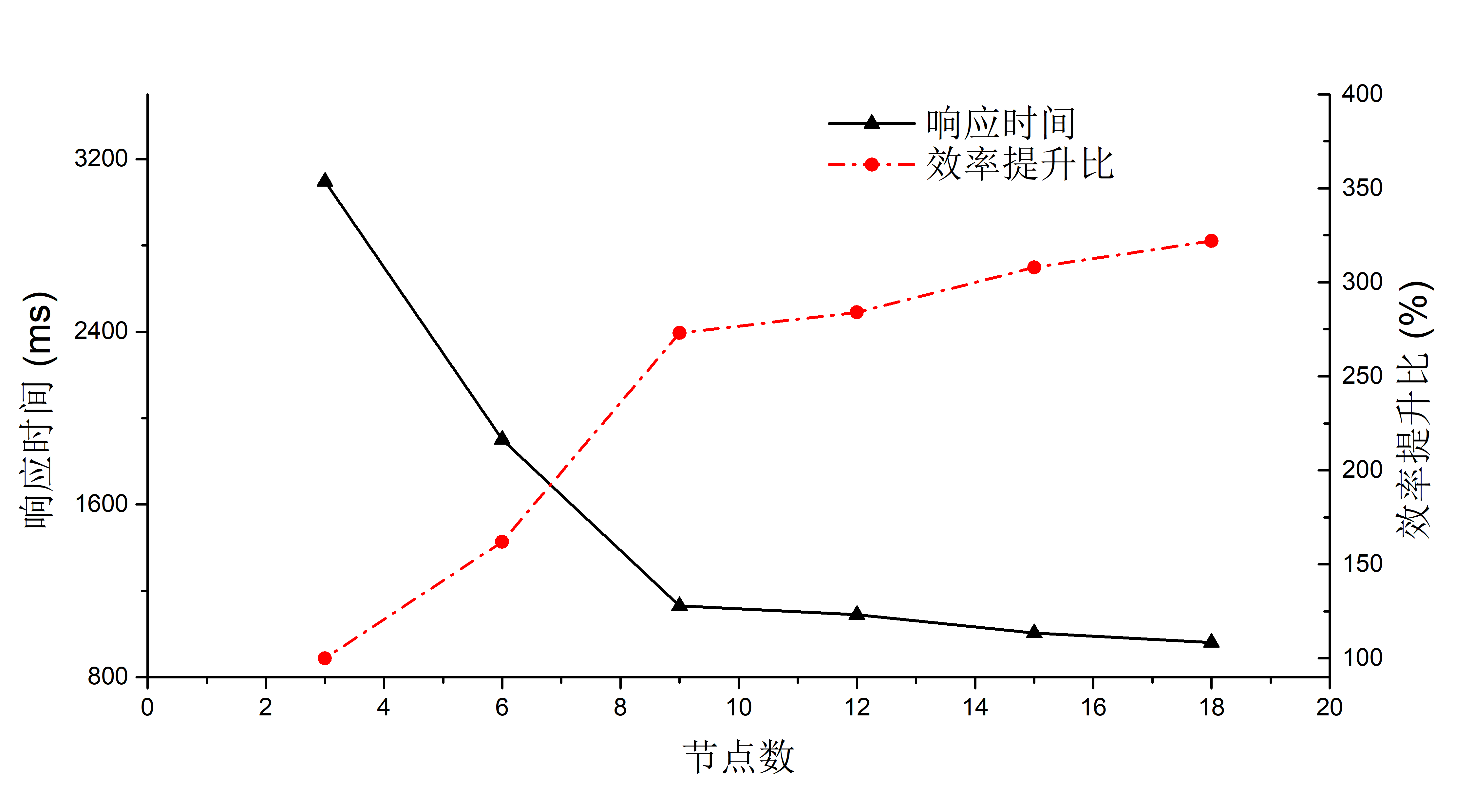


图6 MAP-Vis系统时空查询示意图

Fig.6 Map-Vis。。。。。。。。。。。。。。

该实验以HBase集群使用的Region数目作为自变量，观察查询响应时间的变化。实验数据是2014-2015两年24个月共47G数据。实验结果如图7所示，实线是响应时间的变化趋势，虚线是查询效率提升比例。从图6可以看出，随着集群Region数目的增加，整体的响应时间不断下降，从3.2s减少到0.9s，效率提升约2.4倍，这是因为数据存储得更加分散，单个Region访问频率降低，访问排队时间也随之减少。因此，本框架通过增加Region的数量提升了查询效率，充分体现了集群分布式优势，显著改善可视化交互体验。

1. 实验三：预处理能力扩展性

该实验以Spark预处理集群为实验对象，以集群的计算节点数作为自变量，统计预处理时间变化（不包括导入HBase数据库的时间）。预处理的原始数据是30个月数据。预处理时间的变化趋势如图7所示。由图8可知，节点的增加使得预处理时间有明显的下降趋势，从380min下降到180min，效率提升约1.3倍。因此，通过引入更多的计算节点，使得Spark集群有更多的Worker节点和Executor进程分摊预处理任务，从而显著提高预处理效率。



图6 MAP-Vis系统时空查询示意图

Fig.6 Map-Vis。。。。。。。。。。。。。。

# 结语

时空大数据因其海量、时空、高维等特征，可视化存在内存消耗大、渲染延迟高、可视效果差等问题。为解决该问题，本文提出了一种融合瓦片金字塔模型和键值对化方法的时空大数据组织模型—多维度聚合金字塔（MAP），并设计实现了一套基于MAP模型的开源可视化框架（MAP-Vis）。MAP组织模型能够同时考虑到时间空间维度，兼顾属性信息，三者逐级聚合汇聚，适应时空大数据快速高效可视化。设计的开源可视化框架（MAP-Vis）以Spark作为预处理工具，以分布式HBase数据库作为存储平台。其中，HBase持久化存储MAP模型数据具备高可扩展性，时空查询高效，充分体现了分布式存储与计算的优势，实现亚秒级数据查询与读取，达到良好的交互式可视化效果。

后续MAP-Vis将进一步完善，例如：1）推广至线/面类型数据；2）支持对矩形、多边形查询；3）添加分析功能模块，扩展MAP-Vis成完整的可视分析系统。

# 参考文献

[1] 李德仁,马军,邵振峰.论时空大数据及其应用[J].卫星应用,2015(9):7-11.

[2] 任磊, 杜一, 马帅, 等. 大数据可视分析综述[ J]. 软 件学报, 2014, 25(9): 1909-1936.

[3] P. Godfrey, J. Gryz, and P. Lasek. Interactive visualization of large data sets. Technical Report EECS-2015-03, York University, March 2015.

[4] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar,M. Tolton, and T. Vassilakis, “Dremel: Interactive analysis of web-scale datasets,” Proc. Very Large Data Bases Endowment,vol. 3, no. 1-2, pp. 330–339, 2010.

[5] T. Mostak, “An overview of mapd (massively parallel database),” in White paper. Massachusetts Institute of Technology, 2013.

[6] Gray, J., Bosworth, A., Layman, A., and Pirahesh H,“Data Cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals”,In Proceedings of the 12th International Conference on Data Engineering, pages 152-159, 1996.

[7] Y. Sismanis, A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Hierarchical dwarfs for the rollup cube. In Proceedings of the 6th ACM International Workshop on Data Warehousing and OLAP, pages 17–24.ACM, 2003

[8] Z. L. Liu, B. Jiang, and J. Heer. imMens: Real-time visual querying of big data. Computer Graphics Forum (Proc. EuroVis), to appear, 2013.

[9] L. Lins, J. T. Klosowski, and C. Scheidegger, “Nanocubes for realtime exploration of spatiotemporal datasets,” IEEE Trans. Vis.Comput. Graph., vol. 19, no. 12, pp. 2456–2465, Dec. 2013.

[10] J. Im, F. G. Villegas, and M. J. McGuffin, “Visreduce: Fast and responsive incremental information visualization of large datasets,” in Proc. IEEE Int. Conf. Big Data, 2013, pp. 25–32.

[11] D. Cheng, P. Schretlen, N. Kronenfeld, N. Bozowsky, and W. Wright, “Tile based visual analytics for twitter big data exploratory analysis”, IEEE International Conference on Big Data, 2013.

[12] <http://spark.apache.org/>

[13] Fox, A., Eichelberger, C., Hughes, J., and Lyon, S. 2013. Spatio-temporal Indexing in Non-relational Distributed Databases, In Proceedings of the IEEE International Conference on Big Data, October 2013.

[14] 张延辉,孟鑫,李立松.HBase企业应用开发实战[M]. 北京:机械工业出版社,2014.

[15] Nick Dimiduk, Amandeep Khurana. HBase实战[M]. 北京:人民邮电出版社,2013.

[16] Lars George. HBase权威指南[M]. 北京:人民邮电出版社,2013.

[17] HBase[EB/OL].http://en.wikipedia.org/wiki/NoSQL.

[18] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. Pervasive Computing, IEEE, 7(4):12–18, 2008

[19] V. Agafonkin. Leaflet - a Javascript library for mobile-friendly interactive maps, 2014.

<http://leafletjs.com/>

[20] https://d3js.org/

MapVis：一种基于多维度聚合的时空大数据可视化方法

XIE Chong, GUAN Xuefeng, WU Huayi

1 State Key Laboratory of Information Engineering in Surveying Mapping and Remote Sensing, Wuhan University, Wuhan 430079, China

**Abstract：**Visual analysis is a very important method for big data mining, which can quickly and intuitively help researchers analyze and understand intrinsic values. However, because of its massive volume, spatio-temporal correlation and high dimensions, big data visualization poses much challenges to current implementations, including large memory consumption, high rendering delay, and poor visual effects.

To address these problems, this paper takes massive spatio-temporal point datasets as an example, adopts a pre-processing visualization scheme, and designs a highly scalable distributed visual analysis framework. Firstly, the paper proposes a multi-dimension aggregate pyramid (MAP) model based on the well-known tile pyramid model. It can support multi-dimensional aggregation of time, space, and attributes. Furthermore, this paper uses the Spark cluster as parallel preprocessing platform and distributed HBase as final storage to store the generated MAP data. This greatly increases the scalability of preprocessing capability and the storage capacity. Finally, this paper designs and implements an open-source distributed visualization framework (MAP-Vis). With the distributed processing capabilities, it can achieve sub-second query response and achieve good interactive visualization efficiency.

**Keywords:** big data, multi-dimensional, hierarchical aggregation, distributed database, visual analysis