CityGML-based 3D GIS visualization in the cloud environment

Fenghua Huang*

Sunshine College, Fuzhou University, Fuzhou 350015, Fujian, China

Received 1 October 2014, www.cmnt.lv

Abstract

CityGML provides an effective approach to the sharing and inoperability of 3D GIS spatial data. However, due to the large amount of data, CityGML's resolving, transferring, and rendering processes during visualization are inefficient in a standalone system, causing considerable delays in viewing the information of large 3D maps for users. The cloud computing technique is introduced to address these problems in this paper. The Hadoop Distributed File System (HDFS) is employed to store the huge amount of CityGML data. A MapReduce-based parallel CityGML data visualization scheme is proposed. A Hadoop-based public cloud for a 3D city information service is constructed in the cloud, allowing cloud users to interact with the cloud via the service interface and obtain the desired high-quality 3D city scenarios. The visualization effectiveness and interoperability efficiency of the large-scale CityGML 3D virtual city data are improved.

Keywords: cloud computing, virtual city model, CityGML, hadoop open-source platform

1 Introduction

CityGML (City Geography Markup Language) is a general data model for representing the data storage and exchange format of virtual 3D city models. It provides an effective approach to the sharing and inoperability problem of 3D spatial data. However, its model's structural complexity and the huge amount of document data cause inefficiencies in the resolving, transferring, and rendering processes during visualization. As a result, users suffer considerable delays in viewing the 3D maps of large areas. Its visualization and interoperability inefficiencies have caused a bottleneck in the standalone system. Related work [1] shows that efficient CityGML visualization is largely dependent on the rapid loading, resolving, and transferring of CityGML documents. It is feasible to load, resolve, and transfer CityGML more efficiently by parallel processing, because CityGML documents contain many independent and complete elements of the city model and thus can be partitioned [1]. In this paper, HDFS in the Hadoop open-source cloud computing platform is used to implement distributed storage of CityGML documents. A MapReduce-based parallel scheme for visualizing CityGML data is proposed. A Hadoop-based public cloud service for 3D city information is constructed in the cloud, allowing cloud users to interact with the cloud via the service interface and obtain the desired high-quality 3D city scenarios. The visualization effectiveness and interoperability efficiency of the large-scale CityGML 3D virtual city data are improved.

2 CityGML overview

CityGML is a standard format for exchanging 3D city model data, developed in 2002 within the scope of the "Special Interest Group SIG3D" from North-Rhine Westphalia, Germany [2]. It is mainly used to describe the common semantic information of 3D city objects. CityGML is not only a XML-based open coding standard, but also an open, general data model for storing and exchanging virtual 3D city objects. CityGML is an extension of GML 3.0 to 3D city modeling, and its latest version is CityGML 2.0, which has been adopted by the Open Geospatial Consortium (OGC) as the official standard. CityGML defines the classes of most geographical objects in the city and the interrelations between them, and fully considers the regional model's geometric, topological, semantic, and appearance properties, including the subject classification hierarchy, aggregation, relation between objects, and spatial properties [3]. CityGML is modularized, consisting of a core model (CityGML Core) and 11 thematic extension modules [3]. The core techniques of CityGML modeling encompass the LOD (Level of Detail) model, lexeme/geometry integration model, modularization idea, and application domain extension (ADE) [2,3], in which the LOD model is the basis of the CityGML model.

2.1 LOD MODEL

To improve visualization and data analysis efficiency, CityGML adopts different levels of detail to describe the 3D city model. The LOD model can display different details according to different requirements. To visualize and analyze the same object at different granularities, an

* ***Corresponding author's*** e-mail: huangfenghuanfj@163.com

object can have different LOD models that can be integrated with each other. The LOD model has 5 levels (LOD0-LOD4) in an ascending order of the detail's accuracy. An instance of the details at LODs 0-4 is provided in Figure 1 [3], where LOD 0 is the lowest level at an accuracy of 5m, representing a 2.5D digital terrain model (DTM), and can overlay aerial images and rendered maps; LOD 1 usually refers to the city block mode consisting of the 'building model' without the roof structure; LOD 2 mostly represents the appearances of roof structures and buildings or vegetation objects; LOD 3 represents the building model's walls, roof structures, balconies, projections, textures, materials, as well as detailed vegetation and transport objects [3]; LOD 4 is an improvement on LOD 3, and is mostly used to represent the interior of the building (e.g., rooms, doors in the rooms, ladders, and furniture within the buildings), being the most detailed level at an accuracy of 0.2m.

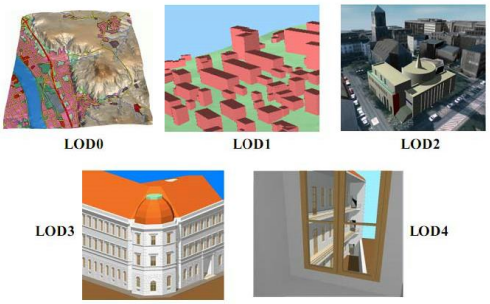


FIGURE 1 An instance of the detail at the five LOD levels (LODs 0-4)

2.2 CITYGML DOCUMENT STRUCTURE

CityGML uses XML as its data storage and exchange format, so its documents conform to the XML specifications [5]. The XML structure is usually represented with the structural file (.xsd) and the instance file (.xml). The former defines the model rules, whereas the latter encompasses the specific data [6]. Each label in the CityGML instance documents has a special meaning, and each document contains a CityModel label as the model's root [7,8]. Each CityObjectMember in CityGML corresponds to a city object (e.g., a building or a road), and has complete model structure. Multiple related CityObjectMembers can form a CityGML city model. Hence, a CityGML document can be divided into many independent CityObjectMember objects [1]. CityGML adopts the boundary representation model, and uses a set of boundary surfaces to describe a 3D entity object (e.g., a building) [9,10]. The most basic element of the CityGML label is gml:pos, representing the 3D coordinate points composing the polygon. CityGML represents the polygon by organizing the points into a linear ring gml:LinearRing or a point list gml:posList. Its feature is that each polygon consists of a sequence of points whose first element meets with the last element. This type of structure is helpful in being transferred into the data structure for image rendering [1].

3 Cloud computing and MapReduce mode

Cloud computing originates from Dell data center solutions, Amazon EC2 products, and Google-IBM distributed computation projects, and is based on distributed computing, parallel computing, grid computing, utility computing, SaaS, SOA, virtualization, and server clusters. Its ancestor is the grid computing that solves large-scale problems via parallel computing and the public computing that provides computing resources as the measurable service. Later on, it flourished in the context of rapid progress in broadband Internet and virtualization [11,12]. Cloud computing uses many techniques, the most important of which include: the programming model, data management, data storage, virtualization, and cloud computing platform management [13]. Other techniques encompass the MapReduce programming model, mass data distributed storage, and management. Cloud computing can be classified into IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service) [14], according to the type of service.

MapReduce is a popular, simplified parallel computing model in the cloud computing platform. It can run parallel applications on the large-scale distributed clusters and features great robustness and availability [15]. MapReduce from Hadoop is open-source and its function is similar to MapReduce from Google. MapReduce is comprised of Map and Reduce, which are responsible for decomposing tasks and aggregating the intermediate results, respectively. Using this computing model, users can easily develop distributed programs to compute mass data [16]. HDFS and MapReduce are the core of Hadoop. The greatest advantage of Hadoop distributed computation lies in efficiently processing local data by combining HDFS with MapReduce.

4 Key techniques to CityGML visualization

4.1 JAXB-BASED CITYGML RESOLVING

Traditionally, XML documents are resolved via the document object model (DOM), data binding, or streaming transformation. The first two models involve reading the entire CityGML document to the memory, thus entailing huge memory consumption and low resolving efficiency [20]. The third method is highly efficient because it can perform rapid streaming of inputs without the need to load the document data into the memory. However, it is poorly intuitive and thus hard to use. There are two common approaches to streaming transformation: the Simple API for XML (SAX) in Java and the XmlReader in .NET [1]. JAXB (Java Architecture for XML Binding) can generate Java classes according to XML Schema [10]. JAXB provides a method for reversing XML instance document into Java object tree, and can also write the contents of the Java class tree into the XML instance document.

In this paper, the CityGML documents are resolved via a combination of SAX and JAXB in Java. The CityGML

documents are fragmented such that each fragment is resolved through data binding. During the resolving process, SAX will first be used to perform a streaming read of the CityGML document and its validity will be verified. Next, the document will be divided into independent fragments according to the CityObjectMember label, and each fragment will be stored in the cache. Finally, JAXB will be employed to transform the fragments in the cache into the object instance. CityGML resolving is considerably sped up, because only a small number of CityObjectMember fragments need to be resolved each time. Furthermore, this fragmenting strategy enables CityGML to be resolved in a parallel manner on the distributed cluster or in the standalone multi-core environment, improving CityGML resolving efficiency.

4.2 CITYGML DATA CONVERSION

To perform CityGML 3D map rendering, the data structure of the resolved CityGML model needs to be converted so that it is suitable for 3D map rendering. It has been proven in [1] that this conversion is usually more time-consuming than resolving, and the time proportion of data conversion in relation to document resolving increases with increased data volume. For large-scale CityGML city models, too much time consumption of conversion will hinder users from viewing in real-time and thus becomes a bottleneck in efficient, real-time visualization.

Like the resolving process, the CityObjectMember-based parallel processing strategy can be used for CityGML data conversion to enable real-time visualization and reduce time consumption of data conversion.

4.3 JAVA3D-BASED CITYGML 3D GRAPH RENDERING

Java3D is one of the most important technical specifications in Web3D and is widely used in 3D graph rendering and displays. Java3D API is the Sun-defined 3D display interface. Its underlying graphic library, OpenGL, is encapsulated with DirectX [21]. Java3D supports dynamic modeling, efficient rendering, platform independence, and flexible interaction, being particularly suitable for 3D landscape generation and interaction in the Internet environment [21].In Java 3D, two geometric objects (TriangleArray and QuadArray) are used to construct 3D graphs [22].The TriangleArray is always a convex polygon and its rendering efficiency is usually higher than that of QuadArray. Thus, it is easier for the 3D engine to optimize. In addition, the indexed mode consumes less memory than the non-indexed mode does, especially when the data volume is huge, because the indexed mode can reuse the point and color data [22].

The CityGML geometric model adopts the boundary representation model and has a large data volume. So, in this paper, Indexed TriangleArrayis are used to construct the CityGML 3D graph. To improve rendering efficiency, the CityGML 3D graph can be rendered by constructing the grid object.

4.4 CITYGML DATA STORAGE

The data involved in this paper include: the original CityGML data, intermediate data, historical data, and metadata. Due to the considerable size of the CityGML data, the idea behind organizing the data in this paper is that the first three types of data are stored in HDFS, whereas the other metadata is stored into HBase. HBase is a column-based database that differs greatly from the traditional relation database in terms of data table structure [23]. CityGML is based on LOD and thus supports stratified or partitioned storage. The quadtree algorithm is used in this paper to partition and query each LOD. The metadata and quadtree codes can represent the relation between partitions or objects. The metadata information tables in the HBase database encompasses Divide, LODS, and Records.

5 MapReduce-based CityGML parallel visualization

As aforementioned, CityGML document visualization involves CityGML document resolving, transferring, and rendering (Java3D). In the distributed environment, each process can only handle a small number of CityObjectMember segments to process CityGML in a parallel manner and improve CityGML visualization efficiency. The process of MapReduce-based CityGML parallel visualization is shown in Figure 2.

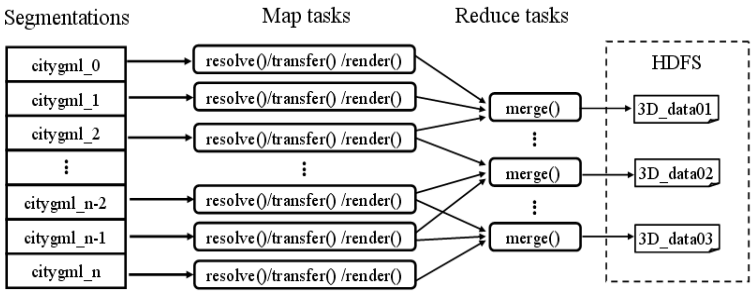


FIGURE 2 MapReduce-based CityGML parallel visualization

First, the input CityGML document is segmented into several file blocks, each of which contains several CityObjectMember segments, and submits them to the master control program at the master node in the cluster [24]. The Master allocates Map or Reduce tasks according to the loads of each node. The Map tasks at each node will perform parallel processing of the input CityGML data blocks via resolve (), transfer (), and Render(), output the intermediate results (i.e., the <key, value>) and store it in the cache [2]. The Reduce node sorts all the intermediate data acquired according to the key values, and puts the data with the same key values into the same group. Finally, the Reduce node calls the user-defined Reduce function (Merge ()) to combine the sorted intermediate data, and temporarily store the output destination files into HDFS. After executing all Map and Reduce tasks, the Master hands control back to the entry to the MapReduce program.

6 Hadoop-based CityGML 3D city information cloud service

The Hadoop-based distributed service clusters and HDFS can be used to perform distributed processing (resolving, transferring, and rendering) and storage of large-scale CityGML data, because of the considerable size of the CityGML data and the limited computing and storing abilities of the traditional servers. Implementation of the Hadoop-based CityGML 3D city information cloud service is shown in Figure 3. In the cloud service clusters, there are two types of nodes: Master and Slave. The Master is responsible for storing CityGML metadata, scheduling tasks, and disseminating map data. The Master uses OGC's W3DS specifications and GeoServer to provide the 3D spatial information services, and includes three service interfaces: the 3D feature data service interface, tile map service (TMS) interface, and the CityGML spatial data access interface. The Slave is responsible for storing and performing parallel processing (resolving, transferring, and rendering) of CityGML data.

If the client requests a 3D feature (or image layer) service, then the Master in the cloud receives the request first, loads and segments the CityGML data of the corresponding area according to the requested spatial area and the CityGML metadata. Data blocks in HDFS are entrusted to different Slave nodes for parallel resolving, transferring, and rendering in order to generate their respective 3D feature data blocks. Next, the feature data blocks are combined. The client can obtain the 3D feature data address using the dedicated interface from Master, download the feature data from the server via the address, and render and display the data through Java3D. If the client requests the CityGML spatial data, then the Master can directly provide the client with the CityGML spatial data access interface. Using this interface, the client can obtain the address to directly download the CityGML spatial data from the server. After being resolved and processed at the client, the downloaded 3D data can display the 3D scenario via the web browser. In addition, if the size of the CityGML data requested by the client is huge, then the client can perform parallel resolving, transferring, and rendering of the CityGML data. The generated 3D graphs can be delivered to users in the TMS manner, i.e., segmenting the graph into block-shaped maps using the image pyramid's multi-resolution model. This strategy can desirably improve the downloading speed and 3D rendering effectiveness [25,26].

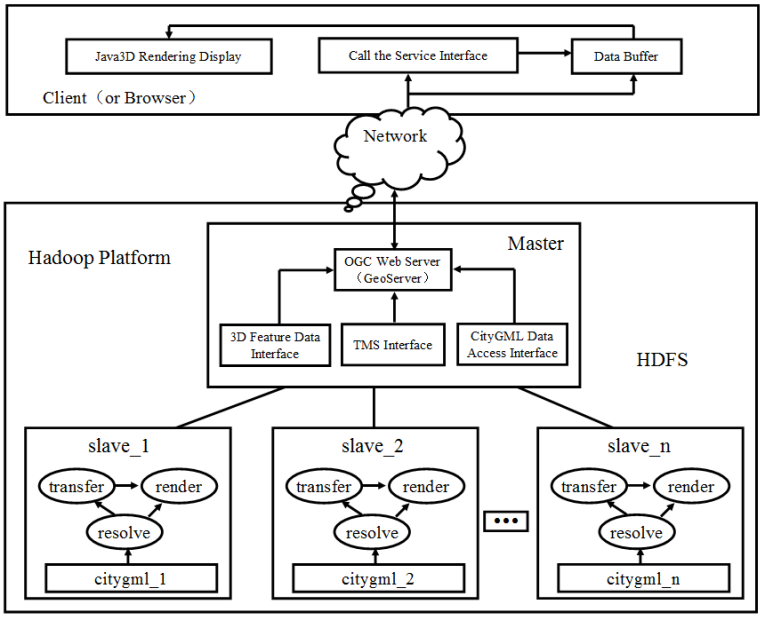


FIGURE 3 Implementation of a Hadoop-based CityGML 3D city information cloud service

The CityGML data volume is large, so when users request to view the same areas, it suffices for the cloud to offer the historical results stored without repeating the procedures. The information requested by the users and the cloud processing results should be outputted and stored into HDFS according to the stratification and partitioning principles. This method amounts to the construction of a cache for data acquisition in the cloud, thus greatly reducing the cloud's implementation complexity and improving the response speed [27,28]. During the MapReduce-based parallel processing of the CityGML data, users can view the scenario in real-time without the need to wait for the data to be loaded completely, performing various operations (e.g., roaming and scaling) on the scenario smoothly during data loading.

7 Experimental results and analysis

7.1 EXPERIMENTAL PLATFORM SETUP

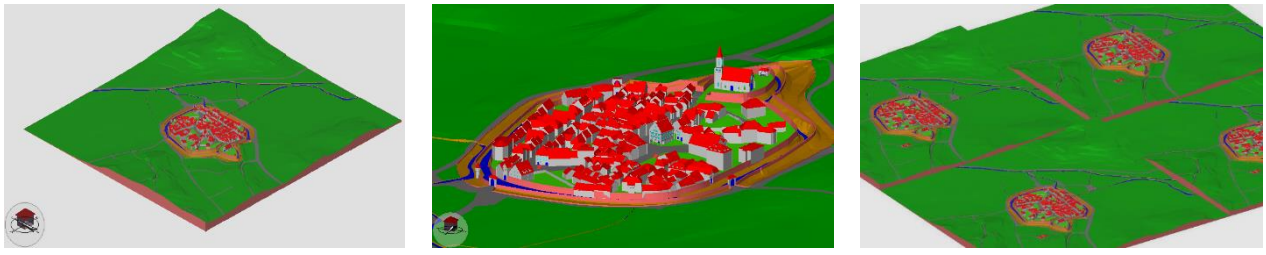
To test the response speed of the Hadoop-based CityGML 3D information service, a Hadoop cloud computing platform is developed in LAN in this paper. This experimental platform adopts the bus topology, including a Hadoop server (Master, NameNode), a backup Hadoop server (Secondary NameNode), 10 Hadoop data nodes (Slave, DataNode), and a virtual server. The platform's software and hardware configurations are shown in Table 1.

TABLE 1 Hadoop cloud computing experimental platform's software and hardware configurations

Hardware	Processor	Memory	Operation System	Hard disk space	Number of hosts
Hadoop server (Master, NameNode)	Intel dual-coreE5200 (CPU frequency 2.50GHZ)	3.2GB	Ubuntu Server 10.104	500GB	1
Backup Hadoop server (Secondary NameNode)	Intel dual-core E5200 (CPU frequency 2.50GHZ)	3.2GB	Ubuntu Server 10.104	500GB	1
Hadoop data node (Slave, DataNode)	Intel dual-core E5200 (CPU frequency 2.50GHZ)	3.2GB	Red Hat Linux(5 hosts) Windows XP SP3(5 hosts)	500GB	10
Virtual server	Intel dual-core E5200 (CPU frequency 2.50GHZ)	3.2GB	Windows 2003 Server	500GB	1

7.2 SELECTION OF EXPERIMENTAL DATA

The CityGML dataset Ettenheim-LoD3) of the virtual 3D model of a community in Ettenheim, Germany, was selected as the experimental data the data are available at http://www.citygml.org). This dataset is based on CityGML 1.0.0 standard, 41.0 M in size, and the corresponding experimental scenarios are shown in Figures 4a and 4b. The CityGML model has over 300 objects at LODs 0-3, including the digital terrain models, rivers, roads, vegetation, buildings, materials, and textures of the interior and exterior of the buildings. Multiple identical communities can be put together to construct larger CityGML 3D scenario maps and provide more experimental data. Combinations of multiple Ettenheim-LoD3 datasets can be represented as Ettenheim-LoD3-$m\times n$, where m denotes the number of communities horizontally, and n denotes the number of communities vertically, i.e., the combined dataset is geometrically $m\times n$ times the original size of the dataset. Figure 4c shows the CityGML scenario that combines four communities Ettenheim-LoD3-2×2).



a) b) c)

FIGURE 4 CityGML virtual 3D model scenario of the experimental area: a) Panorama of the community, b) Buildings in the community, c) Combined CityGML 3D scenario Ettenheim-LoD3-2×2)

7.3 COMPARATIVE EXPERIMENTS AND ANALYSIS

Recall that clients suffer considerable delay when viewing large 3D maps, because the resolving, transferring, and rendering processes are inefficient due to the huge data volume in the CityGML document to be visualized. The response time refers to the time interval from the moment that the user issues the request to view the entire scenario at the client to the moment that the server returns the processing results and the returned results are rendered and displayed, amounting to the sum of the time needed to load, resolve, transfer, transmit, and render the CityGML data. Obviously, the response time is an intuitive measure

of the delay that the user has to suffer in viewing large 3D maps. The impacts of different factors on the response time are analyzed as follows.

7.3.1 Impact of the service mode on the response time

As shown in Figure 3, the cloud can deliver the CityGML 3D information service by offering 3D feature data, TMS tile map or directly the CityGML data. These three methods were used, respectively, for comparative experimentation at a default Hadoop segmenting granularity), as shown in Table 2.

TABLE 2 Comparison of response time at clients adopting different service modes

Service mode	Dataset	File size(MB)	Number of nodes	Response time(s)
3D feature data	Ettenheim-LoD3	41.0	12	55.21
TMS	Ettenheim-LoD3	41.0	12	32.65
CityGML data	Ettenheim-LoD3	41.0	12	85.33
3D feature data	Ettenheim-LoD3-2×2	164.0	12	161.06
TMS	Ettenheim-LoD3-2×2	164.0	12	85.96
CityGML data	Ettenheim-LoD3-2×2	164.0	12	316.39
3D feature data	Ettenheim-LoD3-4×4	656.0	12	425.21
TMS	Ettenheim-LoD3-4×4	656.0	12	152.65
CityGML data	Ettenheim-LoD3-4×4	656.0	12	1132.33

It shows that compared with other two methods, TMS has a shorter response time, allowing the user to view the large 3D map with slight delay. The reason for this is that the CityGML data in this method is all processed in a parallel manner by the cloud, and the processing results are delivered to the client in the form of tile maps, minimizing the sum of the requesting, processing and response time.

7.3.2 Impact of the segmenting granularity on the response time

Table 2 shows that TMS provides the best response speed. In the following image, for a given amount of data Ettenheim-LoD3-6×6, about 3.8GB), TMS performance is evaluated at different granularity, as shown in Figure 5.

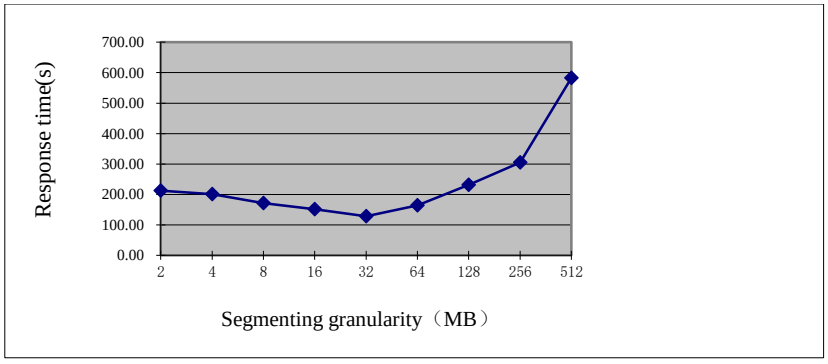


FIGURE 5 Impact of the segmenting granularity on the response time

Figure 5 shows that the delay that the client suffers when viewing large 3D maps varies according to the segmenting granularity. When the granularity is in the range from 2-32 MB, the response time decreases with increased granularity. When the granularity is over 32 MB, the response time increases, that is, the response time is minimized when the granularity is 32 MB. The reason for this is that for a given amount of CityGML data to be processed, a smaller granularity means the data is segmented into a larger number of data blocks. Although each data block is processed more quickly, the large number of data blocks will increase the consumption of time for processing, loading, and managing data blocks at each node. On the other hand, if the granularity is large, then the number of data blocks will decrease, causing some nodes in the cluster to stand idle, whereas other nodes slow down while processing large data blocks. Some nodes may even wait for each other. Thus, in this case, the larger the granularity, the longer the response time at the client.

7.3.3 Impact of data volume on the response time

Ettenheim-LoD3 is used in our work as the baseline data block 41.0MB), and multiple baseline data blocks are combined to form the composite CityGML datasets of varying sizes. In the following image, the number of data blocks is used to represent the data volume the fragmenting granularity is 32MB). TMS is evaluated using different numbers of data volume blocks) at a fixed data size Ettenheim-LoD3-6×6, about 3.8GB).

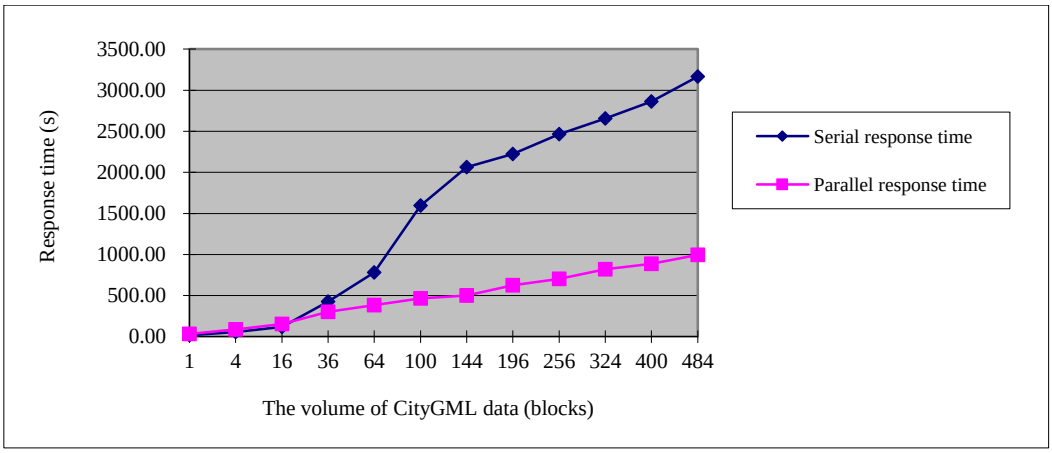


FIGURE 6 Impact of data volume on the response time

Figure 6 shows that in the case of a small CityGML data volume, the MapReduce parallel mode has no obvious advantages, but in the case of the CityGML data volume with 1-16 data blocks, the serial response time is even shorter than the MapReduce parallel response time. The MapReduce parallel mode's advantages, however, become conspicuous when the data volume is huge. When CityGML has 16+ data blocks, the serial response time increases almost exponentially, whereas the MapReduce parallel mode's response time rises slowly. Hence, the MapReduce parallel mode is more suited for parallel processing of large-scale CityGML data. In the case of 144 data blocks, the MapReduce parallel mode's response time is less than one quarter of that of the serial mode

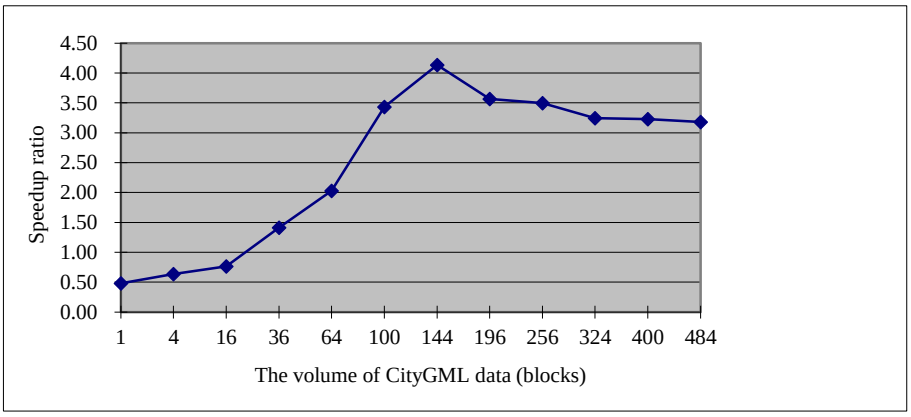


FIGURE 7 Impact of data volume on cloud parallel speedup ratio

Speedup is an important measure of parallel performance. Let r denote the number of processes or nodes, T_c and T_p denote the serial computation time and the parallel computation time, respectively, so the speedup ration, S, can be defined as $S=T_c/T_p$. Ideally, $S=r$, that is, the parallel programs with r processes or nodes can run r times as far as the serial programs. In this case, the speedup ratio is also called the linear speedup ratio. The ideal speedup ratio is 12 for our work. Figure 7 shows, however, that the speedup ratio of the MapReduce-based parallel processing of the CityGML data is clearly less than 12. The reason is that the frequent communication and resource scheduling between nodes during actual parallel computation cause severe time overheads, making it only possible to approximate, rather than reach, the linear speedup ratio [30]. Figure 7 shows that the speedup ratios of the cloud clusters change with increased volume of CityGML data. When the CityGML data volume is less than 144 data blocks, the speedup ratio of the MapReduce parallel mode increases obviously. But when the CityGML data volume is 144+ data blocks, the speedup ratio of the MapReduce parallel mode decreases. The reason for this is that in the case of a huge CityGML data volume, each node has to process a much larger number of data blocks, resulting in much larger time overheads caused by frequent communication and resource scheduling between nodes. It is even possible that some data blocks may be in the waiting state and thus cannot be processed promptly.

8 Conclusions

Clients suffer enormous delays when viewing large CityGML 3D maps. To address this problem, a MapReduce-based CityGML data parallel visualization scheme was proposed. The CityGML data is resolved, transferred, and rendered in the cloud. The Hadoop-based

3D information service public cloud is constructed in the cloud, enabling users to smoothly view, zoom in on, and roam large CityGML scenarios. Experimental results showed that the proposed scheme enables the large-scale CityGML 3D virtual city data to be visualized and inter-operated more efficiently.

Acknowledgment

The work was supported by the Science and Technology Project of Fujian Province Education Department of China NO. JA13364).

References

[1] Xu J L 2010 CityGML-based Key 3D GIS Techniques *National University of Defense Technology* Changsha *(in Chinese*)
[2] Chen Y H 2009 On City Geography Markup Language *Science of Surveying and Mapping* **34**(5) 145-6+135 *(in Chinese*)
[3] Huang F H, Yan L M 2013 CityGML-based Virtual 3D Digital City Modelling *Computer Applications and Software* **30**(5) 104-7 *(in Chinese*)
[4] Ge G H 2011 CityGML-based 3D Modelling of Interiors of Large Buildings *Nanjing Normal University* Nanjing *(in Chinese*)
[5] Ou Y Qun D, Wu Z C, Hu Z W 2011 3D Modelling of CityGML Applications *Science of Surveying and Mapping* **36**(3) 166-8 *(in Chinese*)
[6] Wu Z C, Ou Y Qun D, Li F F 2010 CityGML-based Remote Sensing Information Sharing *Wuhan University Journal of Natural Sciences* **35**(4) 424-6 *(in Chinese*)
[7] Zhou N, Zhang J 2010 CityGML-based Description of 3D City Models *Surveying Engineering* **19**(4) 50-5 *(in Chinese*)
[8] Sun X T 2011 CityGML-based 3D City Modelling and Sharing, *Chongqing Normal University* Chongqing *(in Chinese*)
[9] Chen Y C,Wang Q S 2011 CityGML-based 3D Modeling of City Buildings *Beijing Surveying and Mapping* (3) 8-10 *(in Chinese*)
[10] Yi H Q 2010 CityGML-based Spatial Data Storage *Central South University* Changsha *(in Chinese*)
[11] Tom W 2008 Running Hadoop MapReduce on Amazon EC2 and Amazon S3 *http://highscalability.com/running-hadoop-mapreduce-amazon-ec2-and-amazon-s3*
[12] Yang H, Dasdan A, Hsiao R, Parker D S 2007 Map-reduce-merge:Simple relational data processing on large clusters *SIGMOD'07* (5) 1029-40
[13] McKusick M K, Quinlan S 2009 GFS: Evolution on Fast-forward *ACM Queue* **7**(7) 1-11
[14] Wu X C 2009 Data center integration development technology: The next generation GIS architecture and development model *Earth Science-Journal of China University of Geosciences* **34**(3) 540-6 *(in Chinese*)
[15] Hao S K 2012 Analysis of Hadoop HDFS and MapReduce Strucutres *Post Design Technology* (7) 37-42 *(in Chinese*)
[16] Fan Y S 2012 Comparison of Two Prevailing Databases in Cloud Computing *Journal of Mianyang Normal University* **31**(8) 68-71 *(in Chinese*)
[17] Wan Z Z 2008 MapReduce-based Parallel Platform Design and Implementation *Zhejiang University* Hangzhou *(in Chinese*)
[18] Dean J, Ghemawat S 2008 MapReduce: Simplified Data Processing on Large Clusters *Communication of the ACM – 50th anniversary issue* **51**(1) 107-13
[19] Zhang L J, Huan F, Wang Y D 2012. Implementation of Parallel Image processing in Hadoop Cloud *Communications Technology* 10) 59-62 *(in Chinese*)
[20] Qiu G, Li Z H, Zhang Y 2003 Object Constructing and Storage Patterns of XML Data Binding *Computer Engineering* **29**(19) 75-82 *in (Chinese*)
[21] Wang G Y 2007 Design and Practice of JAVA3-based Network 3D Scenario Visualization Techniques *Northwest University* Xi'an *(in Chinese*)
[22] Lu P, Li L, Xie S D 2012 Java 3D-based Terrain Visualization and Key Techniques *Geomatics & Spatial Information* **35**(10) 74-6 *(in Chinese*)
[23] Zhu Z 2008 Hadoop-based Mass Data Processing Models and Application *Beijing University of Posts and Telecommunications* Beijing *(in Chinese)*
[24] Wan Z Z 2008 MapReduce-based Parallel Computing Platform Design and Implementation *Zhejiang University* Hangzhou *(in Chinese*)
[25] Zhao L L, Zhu J J, Liu S, et al. 2009 Rapid 3D Feature Extraction and Visualization of City GML Documents *Computer Engineering and Applications* **45**(26) 226-9 *(in Chinese*)
[26] Ji Y S, Shi S H,Zhang C 2009 CityGML and W3DS-based Distributed 3D Spatial Data Sharing Cases *Coal technology* **28**(11) 109-11 *(in Chinese*)
[27] Jin B X, Bian F L 2004 3D Spatial Data Interoperability Approaches in the Grid Environment *Wuhan University Journal of Natural Sciences* **29**(12) 1075-9 *(in Chinese*)
[28] Chen L, Xiang N P, Jiao Y P 2010 WebServices-based 3D Spatial Data Representation and Transmission *Science of Surveying and Mapping* **35**(3) 162-4 *(in Chinese*)
[29] Liu L 2013 Design of a Platform for Real-time Processing of MPI-based Satellite Remote Sensing Data *Spacecraft Engineering* 22(3) 130-4 *(in Chinese*)

Author

Fenghua Huang, September 1982, Fujian, P.R. China.

Current position, grades: PhD, lecturer, Sunshine college, Fuzhou university, China.
University studies: Fujian Normal university, College of Geographical Sciences 2011-2014), Fuzhou university, College of mathematics and computer 2006-2009.
Scientific interests: cloud computing, geographic information system, pattern recognition and image processing.
Publications: 10 papers.