Computers, Environment and Urban Systems xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Computers, Environment and Urban Systems

journal homepage: www.elsevier.com/locate/compenvurbsys

Parallel map projection of vector-based big spatial data: Coupling cloud computing with graphics processing units

Wenwu Tang *, Wenpeng Feng

Center for Applied Geographic Information Science, Department of Geography and Earth Sciences, University of North Carolina at Charlotte, 9201 University City Blvd., Charlotte, NC 28223, United States

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Map projection
Big spatial data
Cloud computing
Graphics processing units
Parallel computing

ABSTRACT

The objective of this article is to present a framework that couples cloud and high-performance computing for the parallel map projection of vector-based big spatial data. The past few years have witnessed a tremendous growth of a variety of high-volume spatial data—i.e., big spatial data. Map projection is often needed, for example, when we apply these big spatial data into large-scale spatial analysis and modeling approaches that require a common coordinate system. However, due to the size of these data and algorithmic complexity of map projections, the transformation of big spatial data between alternative projections represents a pressing computational challenge. Recent advancement in cloud computing and high-performance computing offers a potential means of addressing this computational challenge. The parallel map projection framework presented in this study is based on a layered architecture that couples capabilities of cloud computing and high-performance computing accelerated by Graphics Processing Units. We use large LiDAR data as an example of vector-based big spatial data to investigate the utility of the parallel map projection framework. As experimental results reveal, the framework provides considerable acceleration for re-projecting vector-based big spatial data. Coupling high-performance and cloud computing, which complement to each other, is a suggested solution for the efficient processing and analysis of big spatial data.

1. Introduction

Advance in computational science and cyberinfrastructure (Atkins et al., 2003) over the past decade has produced digital spatial data at a rapidly increasing rate, pushing us into the era of big data (Manyika et al., 2011; Zikopoulos & Eaton, 2011). These spatial data are generated through various ways, for example, social media platforms, remote sensors, sensor networks, or computer simulation (Goodchild, 2007; Hey, Tansley, & Tolle, 2009). As more and more spatial data are produced, these data are evolving into big data because of their characteristics of large volume (e.g. terabytes or even petabytes), high frequency of update, and diverse types (Zikopoulos & Eaton, 2011). The capacity and capability of advanced computing technologies allow us to collect and store these big spatial data, which drives advance in data-intensive science (as the fourth scientific paradigm; see Hey, Tansley, & Tolle, 2009) in general and geographic information science in particular. However, the efficient manipulation of big spatial data through processing, analysis, and visualization has been rarely investigated because of hurdles from coping with the size and complexity of these data. This manipulation is essentially important for us to garner interesting spatial knowledge that cannot be feasible using aggregated data or conventional desktop computing. Emerging cloud and high-performance computing (Armbrust et al., 2010; Atkins et al., 2003), to which domain scientists draw increasing attention (Agrawal, Das, & El Abbadi, 2011; Shi, 2010; Wang & Liu, 2009; Yang, Wu, Huang, Li, & Li, 2011), have the potential to handle the manipulation of big spatial data. In this study, we aim at investigating the potential of cloud and high-performance computing in the efficient handling of big spatial data. We focus on the map projection of GIS data, a fundamental and representative spatial data handling approach.

Map projection is an important theme in analytical cartography and geographic information systems (see Burrough & McDonnell, 1998; Slocum, McMaster, Kessler, & Howard, 2009). Map projection is referred to as "a systematic transformation of ellipsoidal coordinates of latitude and longitude to a plane coordinate representation" (Usery, Finn, & Mugnier, 2009; p. 91). The presentation of locational information on the Earth (ellipsoid) using 2D flat maps induces the original need of map projection. The transformation between ellipsoidal coordinate system (or geographic

* Corresponding author. Tel.: +1 (704) 687 5988; fax: +1 (704) 687 5973.
E-mail addresses: WenwuTang@uncc.edu (W. Tang), wfeng5@uncc.edu (W. Feng).

http://dx.doi.org/10.1016/j.compenvurbsys.2014.01.001
0198-9715/

coordinate system) and planar coordinate system (or projected coordinate system) is based on mathematic conversion of coordinates using developable surfaces in forms of, for example, cylinder, cone, and plane (see Slocum et al., 2009). Because of transformation between alternative coordinate systems, map projection produces inevitably distortion on such metric properties as area, distance, shape, or scale (see Pearson, 1990; Slocum et al., 2009). The combination of developable surfaces and distortion leads to a set of map projections with different algorithmic complexity (from straightforward to sophisticated). Map projection, typically comprising forward (from ellipsoidal coordinate system to planar) and inverse (from planar to ellipsoidal) transformations (Maling, 1991; Slocum et al., 2009; Usery et al., 2009), preserves some metric properties (e.g., distance) while sacrificing the accuracy of other properties (e.g., area). The choice of which map projection is used depends on the objective of maps, accuracy that users expect to obtain, cartographic scale, study area, and data types (Maling, 1991; Usery et al., 2009).

Map projection or coordinate system constitutes a necessary component of GIS data. The GIS data are often organized in different coordinate systems, but using these data for spatial analysis and modeling requires a common coordinate system. As a consequence, map re-projection is needed to transform GIS data between alternative coordinate systems. In this study, map re-projection is generally included in the category of map projection. The map re-projection of GIS data is composed of two steps: inverse transformation to ellipsoidal coordinate system and then forward transformation to the planar coordinate system to be projected. The inverse and forward transformation can be carried out using an analytical approach or polynomial approximation (Maling, 1991; Usery et al., 2009). Maling (1991) provided an example that details map re-projection via inverse and forward transformation. Because of the algorithmic complexity of mathematic transformation and the volume of GIS data, map projection often presents a notable computational challenge. As an example, for the forward transformation of the Mollweide projection (see Snyder, 1987), a numerical approach required by the projection of coordinates may result in multiple iterations. Researchers have been aware of, and exploiting, the power of high-performance computing for computationally demanding map projection (e.g., Behzad et al., 2012; Finn et al., 2012; Zhao, Cheng, Dong, Fang, & Li, 2011). Zhao et al. (2011) implemented a GPU-based parallel map projection algorithm. Their studies were based on a single GPU and tested using small-size datasets. Finn et al. (2012) presented a parallel map projection solution, pRaserBlaster, which leverages message-passing parallelism and high-performance CPU-based clusters (e.g., supercomputers) for speeding up the re-projection of large raster datasets. These parallel map projection studies demonstrate the tremendous capabilities of high-performance computing in efficient map projection.

The objective of this study is to develop a parallel spatial computing framework for the map projection of vector-based big spatial data. This parallel map projection framework couples cloud computing and high-performance computing accelerated by Graphics Processing Units (GPUs; see Kirk & Hwu, 2010; Owens et al., 2007). This framework is well tailored to best reaping benefits from each component to efficiently handle vector-based big spatial data. Vector-based spatial data that we use for parallel map projection are bare earth LiDAR (Light Detection And Ranging; see Wehr & Lohr, 1999) data for North Carolina, USA. The entire dataset requires hundreds of gigabytes for storage. A simple map projection on this dataset will lead to a tera-scale computing requirement, corresponding to several hours of sequential computing time. Thus, how to use state-of-the-art computing capabilities to (re)project these big spatial data within an acceptable and affordable time limit is our concentration in this study.

The remainder of this article is organized in the following manner. In Section 2, we present the background of this work by focusing on cloud computing and GPU-enabled high-performance computing. In Section 3, we discuss the parallel spatial computing framework for the map projection of big spatial data. In Section 4, we design experiments to evaluate this framework. Section 5 concludes this article with summary and future research directions.

2. Background

2.1. Cloud computing

Cloud computing represents a new computing paradigm that provides on-demand and cost-effective computing support as a form of utility (Armbrust et al., 2010; Buyya, Yeo, Venugopal, Broberg, & Brandic, 2009). Generally, cloud computing is defined as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (Mell & Grance, 2011, p. 2). Cloud computing is built on the basis of distributed computing and characterized by on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured services (Armbrust et al., 2010; Mell & Grance, 2011). Functionality of cloud computing is encapsulated into services that coordinate together to provide on-demand scalable computing capability. Regarding how these services are delivered, cloud services are typically classed into three levels: Infrastructure as a Service (IaaS; e.g., Amazon EC2), Platform as a Service (PaaS; e.g., Google App Engine or Windows Azure), Software as a Service (SaaS; e.g., Google Maps). Depending on the way that clouds are deployed and accessed, four types of clouds (private, community, public, and hybrid cloud infrastructure) can be constructed to meet the computing requirements of different users (Armbrust et al., 2010; Buyya et al., 2009; Mell & Grance, 2011). In particular, virtualization plays an essential role in the provisioning of on-demand and networked computing instances (e.g., virtual machines) over physical computing infrastructures. Typically, capabilities of such computing resources as compute, data storage, and networks are virtualized to collaboratively support the flexible and efficient manipulation (expansion or shrinkage) of virtual machines (i.e., elasticity). As technologies of cloud computing become mature, open-source platforms and tools, represented by OpenStack (OpenStack, 2013), OpenNebula (OpenNebula, 2013), and Eucalyptus (Eucalyptus, 2013), have been available for constructing and deploying cloud computing environments. This further spurs the study and application of cloud computing for scientific discovery.

The emergence and advancement of cloud computing have greatly intrigued geographic information scientists and researchers with relevant background into the investigation of its potential in enhancing GIS-related spatial problem-solving (Huang et al., 2013; Yang, Goodchild et al., 2011). Wang, Wang, and Zhou (2009) developed a PaaS-based framework for the efficient retrieval and indexing of spatial data on the platform of Google App Engine. Yang, Wu et al. (2011) stressed that spatial analysis and modeling can greatly benefit from alternative levels of services in spatial cloud computing in terms of resolving data, computing, concurrency, and spatiotemporal intensities. Yang, Wu et al. (2011) highlighted the importance of applying spatiotemporal principles as guidance for efficiently harnessing cloud computing power. Huang et al. (2013) conducted a comparison of three open-source cloud platforms on geospatial applications regarding their features and performance. Huang et al. emphasized that at

Please cite this article in press as: Tang, W., & Feng, W. Parallel map projection of vector-based big spatial data: Coupling cloud computing with graphics processing units. *Computers, Environment and Urban Systems* (2014), http://dx.doi.org/10.1016/j.compenvurbsys.2014.01.001

the present stage parallel computing capabilities provided by virtualized cloud infrastructure may not compete with high-performance clusters for computationally intensive spatial problems because of overhead from virtualization. The findings by Huang et al. (2013) are in agreement with the reported performance analysis of public cloud (e.g., see Jackson et al., 2010). In other words, high-performance clusters built directly on physical computing infrastructure remain as a suggested solution when we are dealing with computationally intensive spatial analysis and modeling. Furthermore, the coupling of cloud computing and high-performance clusters creates potential for resolving the performance issue facing cloud computing.

2.2. GPU-enabled high-performance computing

GPUs, with a foundation on the many-core computing paradigm (Nguyen, 2007; Owens et al., 2007), provide unprecedented massively parallel computing capabilities that are shifting mainstream computing paradigm. Benefiting from numerous cores that are tightly coupled, GPUs allow us to reap computing performance that is often several orders of magnitude of CPUs (Kirk & Hwu, 2010). Therefore, commodity programmable GPUs have been extensively used as co-processors to accelerate general-purpose scientific discovery (Kirk & Hwu, 2010; Owens et al., 2007), far beyond their original purpose on efficient graphics operations. Because of the stream processing mechanism built in GPUs, data parallelism is well suited to harnessing the high-throughput computing power of GPUs (Nickolls, Buck, Garland, & Skadron, 2008). To solve a scientific problem using GPUs is to partition data associated with the problem and map the computation of data partitions to the many-core parallel computing architecture on GPUs. The computation of data partitions ported on GPUs is executed concurrently. Platforms and standards, including CUDA (Compute Unified Device Architecture; see CUDA, 2013) and OpenCL (see OpenCL, 2013) have been implemented on the basis of thread parallelism to enable the programming of GPUs for high-performance acceleration. These platforms and standards are now supported by alternative programming languages (e.g., C/C++, Python, and Fortran).

GPUs have demonstrated their high-performance computing capabilities for analyzing and visualizing big data. However, the computation required by the efficient handling of big data often exceeds the capacity of a single GPU device. In other words, the use of multiple GPUs that are interconnected (i.e., GPU clusters that couple both CPU processors and GPU co-processors) is urgently needed to handle efficiently big data using GPUs. Further, the virtualization of GPUs as a form of high-performance computing utility on cloud infrastructures has been an active research theme (Expósito, Taboada, Ramos, Touriño, & Doallo, 2013; Ravi, Becchi, Agrawal, & Chakradhar, 2011; Shi, Chen, Sun, & Li, 2012). Although GPU virtualization on clouds remains in the initial stage of development, cloud vendors (e.g., Amazon EC2, NIMBIX, Peer 1 Hosting; see http://www.nvidia.com/object/gpu-cloud-computing-services.html) are now providing virtualized GPU resources for public use. Because of the availability, affordability, and scalability of GPU programming environments, scientists are striving to apply GPUs for the acceleration of their domain models and data analytics (Nguyen, 2007; Wang & Shen, 2011). In the field of geographic information science, GPU-enabled massively parallel algorithms for GIS-based spatial analysis and modeling have been reported (Ortega & Rueda, 2010; Xia, Kuang, & Li, 2011; Zhang, You, & Gruenwald, 2010). These parallel spatial analysis and modeling efforts encompass spatial indexing (Zhang et al., 2010), viewshed analysis (Zhao, Padmanabhan, & Wang, 2013), cartogram construction (Tang, 2013), and drainage network analysis (Ortega & Rueda, 2010).

3. Methods

In this section, we present the design of the parallel computing framework for the accelerated map projection of vector-based big spatial data. We choose to use bare earth LiDAR data of North Carolina, USA as an example of vector-based big spatial data (see Fig. 1). This dataset covers the entire North Carolina, resulting in 13,596 sub-datasets (in file format). The LiDAR dataset is collected and released by the North Carolina Floodplain Mapping Program (see http://floodmaps.nc.gov). The original projection of the dataset is the Lambert Conic Conformal projection. About 230 GBs of storage space are needed to maintain the original LiDAR dataset. As an example, it takes about several hours of CPU-based sequential time (2.83 h for projection only; 11.46 h with the consideration of input/output operations; an advanced CPU is used) for re-projecting the entire dataset from Lambert Conic Conformal projection to Mollweide projection. Thus, a parallel framework that allows for efficient map projection is necessary and urgent if we are to transform these vector-based big spatial data. Thus, in this section, we first describe the entire architecture of the framework. Then, we discuss each component of the framework.

3.1. Framework design

The cloud-based parallel map projection framework is based on a layered architecture as shown in Fig. 2. Three layers of functionality are specifically designed to enable the efficient map projection of vector-based big spatial data. These three layers are high-performance computing, cloud-based virtual machine, and web-based GIS portal. The first layer is the collection of GPU-enabled high-performance computing infrastructure. The layer of virtual machines is built on cloud infrastructure to support the manipulation of data, map projection algorithms, and spatial parallel strategies. The third layer of web-based GIS provides user interfaces that allow for specifying parameters for parallel map projection (e.g., input data, and spatial parallel strategies). The design of the layered architecture integrates capabilities of Internet GIS (Fu & Sun, 2011; Peng & Tsou, 2003), cloud computing, and GPU-enabled high-performance computing. The coupling of the three layers provides solid support for leveraging cloud computing and high-performance computing capabilities for the efficient parallel map projection of big spatial data. Fig. 3 illustrates the workflows designed in this framework with respect to interactions between these layers and the specific functionality of each layer. In the rest of this section, we first discuss how to utilize GPU-based high-performance computing for efficient map projection. Then we present the layer of cloud-based virtual machine. Third, we illustrate the use of Web GIS portal for user input and geovisualization of relevant data.

3.2. GPU-accelerated high-performance computing

To efficiently transform big spatial data to alternative map projections, high-performance computing accelerated by GPUs is an imperative option. As GPU technologies advance, high-performance computing clusters are increasingly equipped with GPU accelerators. GPU clusters, including those on cloud infrastructure (e.g., Amazon EC2), create potential for the efficient map projection of big spatial data. The layer of high-performance computing in this framework is targeted on the leverage of GPU clusters for parallel map projection of vector-based big spatial data. GPU clusters in this study include not only physical clusters but also virtualized clusters on cloud infrastructure. Data to be projected need to be transferred from virtual machines and then deployed on the GPU cluster selected for map projection. These

Please cite this article in press as: Tang, W., & Feng, W. Parallel map projection of vector-based big spatial data: Coupling cloud computing with graphics processing units. *Computers, Environment and Urban Systems* (2014), http://dx.doi.org/10.1016/j.compenvurbsys.2014.01.001

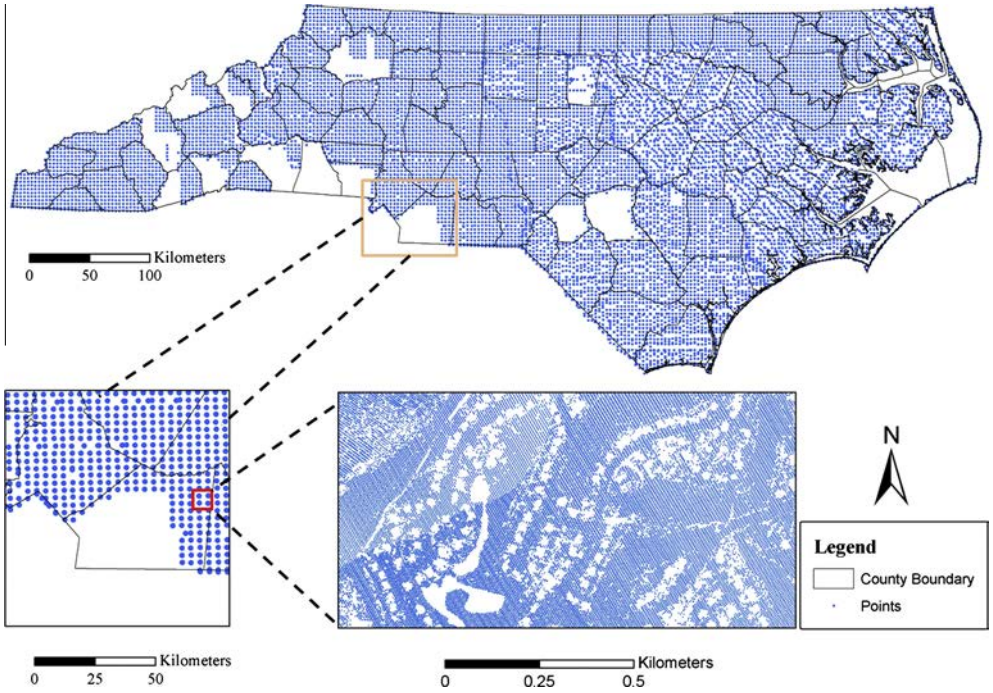

Fig. 1. Map of bare-earth LiDAR data of North Carolina, USA (top and bottom-left figures show the spatial distribution of each sub-dataset; the first point in each sub-dataset was used as the location of the corresponding sub-dataset; bottom-right figure shows the spatial pattern of LiDAR points in selected sub-datasets).

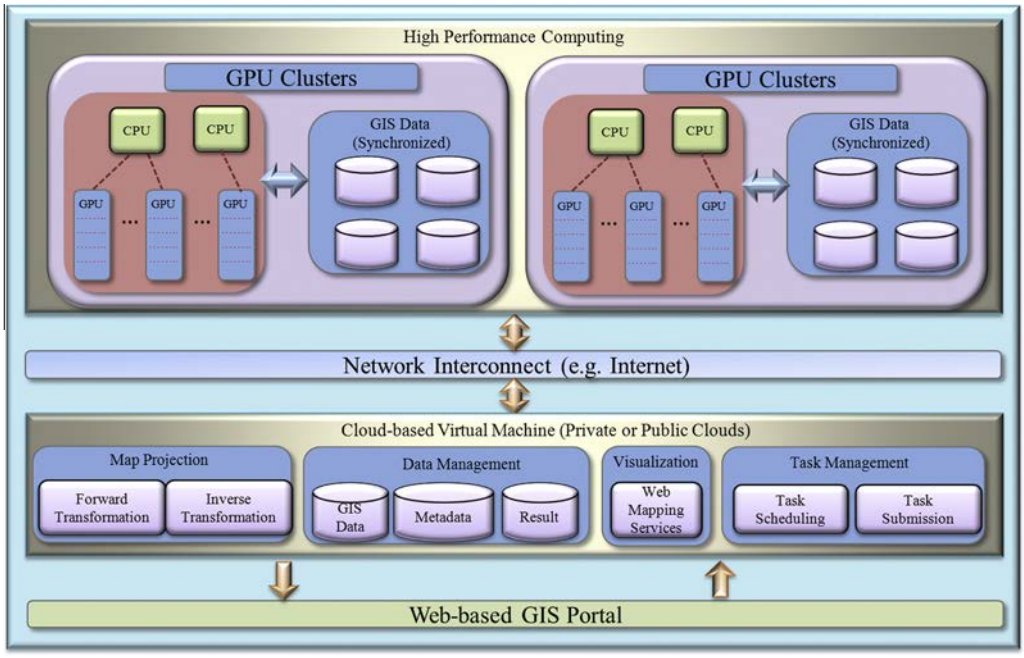

Fig. 2. Cloud-based parallel map projection framework of big spatial data.

data can be transferred once and then updated according to the need of map projection. Once data deployment is completed, models of map projection are updated (for example, when new algorithms are available from the layer of virtual machines) and deployed. If the deployment of data and models is done, tasks retrieved from the layer of virtual machines are encapsulated into computing jobs and ported into the GPU cluster for parallel map projection.

Because this framework allows for the use of multiple GPUs, a load balancing strategy is developed to ensure the workload assigned to GPU devices is close to each other. As a result, multiple GPU resources can be utilized in an efficient manner. For example, the bare-earth LiDAR data used in this study, the number of points can be used as an indicator to guide the load balancing operation. Once jobs are submitted, the information of job status (e.g., queuing, running, error, or completed) is sampled frequently (e.g., every 10 s). This information is sent to the virtual machine in charge of task monitoring and can be further presented to users interacting with Web GIS portal. Those jobs that are not completed due to, for example, a hardware issue can be re-submitted (fault-tolerance).

Please cite this article in press as: Tang, W., & Feng, W. Parallel map projection of vector-based big spatial data: Coupling cloud computing with graphics processing units. *Computers, Environment and Urban Systems* (2014), http://dx.doi.org/10.1016/j.compenvurbsys.2014.01.001

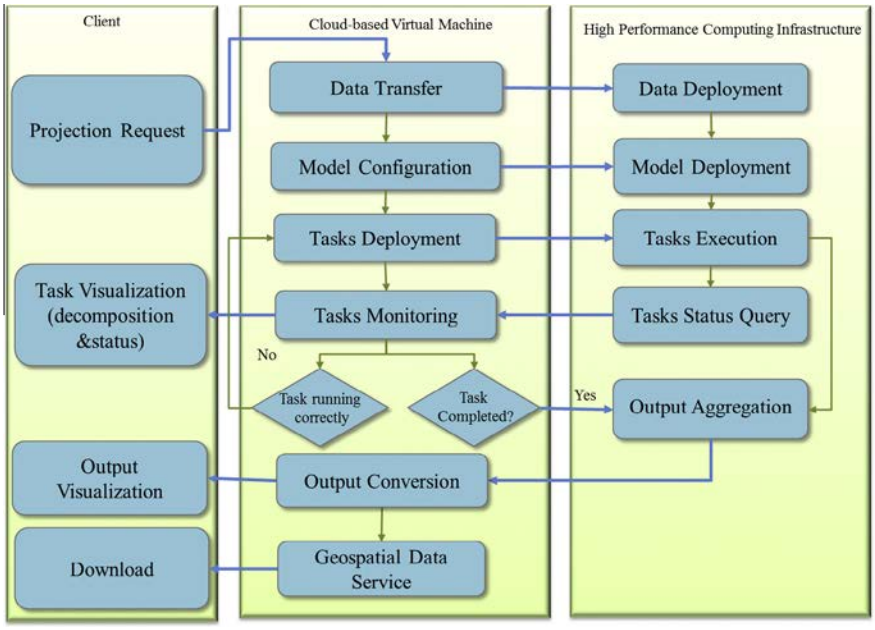

Fig. 3. Illustration of workflows for inter- and intra-layer interactions for cloud-based parallel map projection.

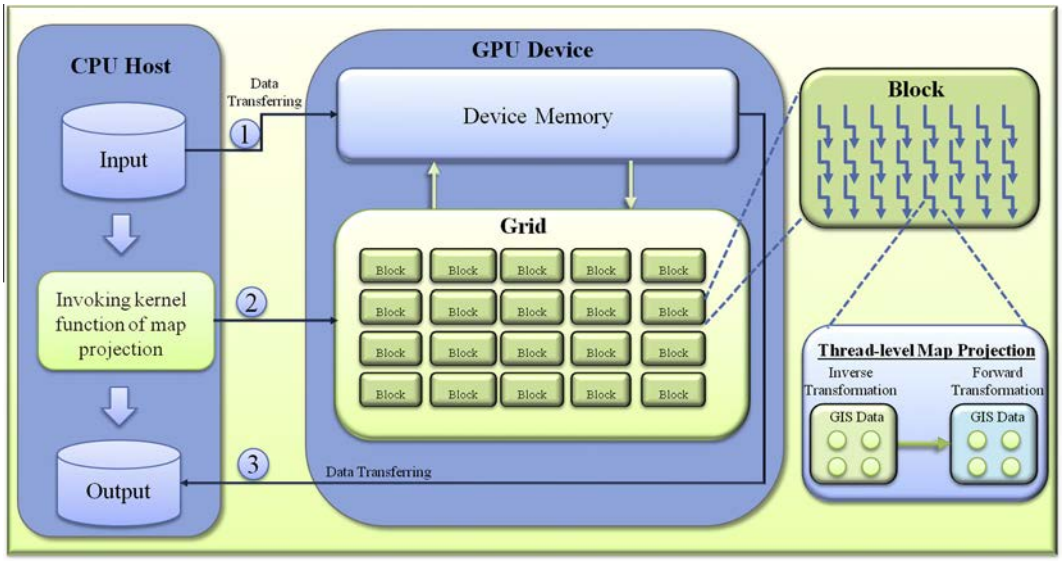

Fig. 4. Workflow of the GPU-enabled parallel map projection based on CUDA.

Based on the GPU programing platform of CUDA, we develop GPU-enabled parallel map projection algorithms. For a specific map projection, the use of GPUs for accelerating the transformation of coordinates is illustrated in the following steps (also see Fig. 4). First, spatial data are read into the CPU host memory and organized into a collection of coordinates. Second, these data are transferred to the global memory of the GPU to be used. Because the capacity of the GPU global memory may be limited, in this study we ensure that memory space required by each dataset assigned to a GPU device does not exceed the memory capacity of the GPU device. Of course, if we want to use a single GPU device to compute a very large dataset that exceeds the global memory capacity, this dataset can be organized into multiple smaller datasets. These small datasets are then transferred to, and computed on, the GPU device.

Once data are reconstructed in the global memory of the GPU device, the kernel function of map projection is triggered. The kernel function comprises the following two steps: inverse and forward transformation. The kernel function recruits a grid of CUDA threads to parallelize the inverse and forward transformation of spatial data. For the LiDAR dataset used in this study, each thread is responsible for re-projecting a sub-set of points. Once all threads complete their map projection tasks, map projection results are transferred back to the CPU memory. As we can see, the use of GPUs for parallel map projection requires cooperation between CPUs and GPUs. This CPU–GPU cooperation

Please cite this article in press as: Tang, W., & Feng, W. Parallel map projection of vector-based big spatial data: Coupling cloud computing with graphics processing units. *Computers, Environment and Urban Systems* (2014), http://dx.doi.org/10.1016/j.compenvurbsys.2014.01.001

makes GPU programming referred to as heterogeneous parallel computing.

3.3. Cloud-based virtual machines

Cloud-based virtual machines are the core layer of the entire framework. This layer is to employ a collection of virtual machines (i.e., IaaS here) that can be virtualized from private or public clouds to manipulate data and algorithms for the parallel map projection. Generally, this layer comprises four modules: map projection, data management, visualization, and task management. The module of map projection is to maintain and update the algorithms of map projection. Each specific map projection includes algorithms for forward and inverse transformations. The data management module supports the storage and transferring of inputs, outputs, and their metadata.

The third module, visualization, provides a set of geospatial web services (i.e., at the level of SaaS) that allows for the access and geovisualization of GIS data, parallel strategies, or the status of computing jobs on the layer of high-performance computing. These geographically referenced information can be delivered through geospatial web services, represented by web map services (WMS; see WMS, 2013) or web feature services (WFS; see WFS, 2013).

The fourth module, task management, implements fundamental capabilities for the utilization of GPU-enabled high-performance computing. The task management module is in charge of the partitioning of computation associated with map projection into collections of tasks, submission and monitoring of these tasks that are deployed and computed on remote high-performance computing resources. Each task consists of the computer program of the chosen map projection and data to be projected. Big spatial data (e.g., the NC LiDAR data used in this study) may be originally organized into a large number of sub-datasets. These sub-datasets may need to be aggregated into larger datasets in terms of estimation on the capacity of computing resources and computational intensity of map projection algorithms. In this study, we use static task scheduling (see Wilkinson & Allen, 2004) to assign tasks to computing nodes—i.e., tasks are assigned to computing nodes before these tasks are executed. Two approaches in terms of considering load balancing or not are designed for the static task scheduling. Without the consideration of load balancing, each computing node is assigned with an equal number of tasks. While load-balancing is applied, it is ensured that the total workload (e.g., the number of points in this study) associated with the tasks assigned to each node is equivalent. But this load-balancing operation may lead to a different number of tasks assigned to each computing node.

The association between the modules in this layer and virtual machines is flexible. For example, modules in this layer can be deployed and handled by a single virtual machine or each module is linked to a virtual machine. Because of the scalability of cloud computing infrastructure, users can recruit an appropriate number of virtual machines to support the functionality of these modules. Furthermore, this layer is able to provision virtual machines with relevant development environments or platforms (i.e., PaaS) so that users can modify and update the functionality of these modules. For example, users can contribute new map projection algorithms to the module of map projection.

3.4. Web GIS portal

The layer of Web GIS portal provides interfaces that allow users to directly interact with the other two layers of the framework (cloud-based virtual machines and high-performance computing). Fig. 5 is the snapshot for configuring parallel map projection of big spatial data. Users can specify map projection and data that they want to apply. Furthermore, two data partitioning strategies are provided for the LiDAR dataset used in this study: partitioning by the number of files or of points. Based on the number of GPU devices that are available on computing clusters of interest, users can determine the number of computing tasks that are needed. A Web GIS interface is needed to assemble geospatial web services (public or customized) or GIS data to guide the configuration of parallel map projection (see Fig. 5).

Once the configuration of parallel map projection is completed, users will be presented with the interface of task monitoring (see Fig. 6). This interface allows for querying and visualizing the status of tasks that are executed on the remote GPU cluster(s). Task status (including queuing, running, error, and completed) is organized in KML format and can be updated accordingly. To geovisualize the status of jobs, the first point in the data assigned to each job is extracted and used as the spatial location to which the job

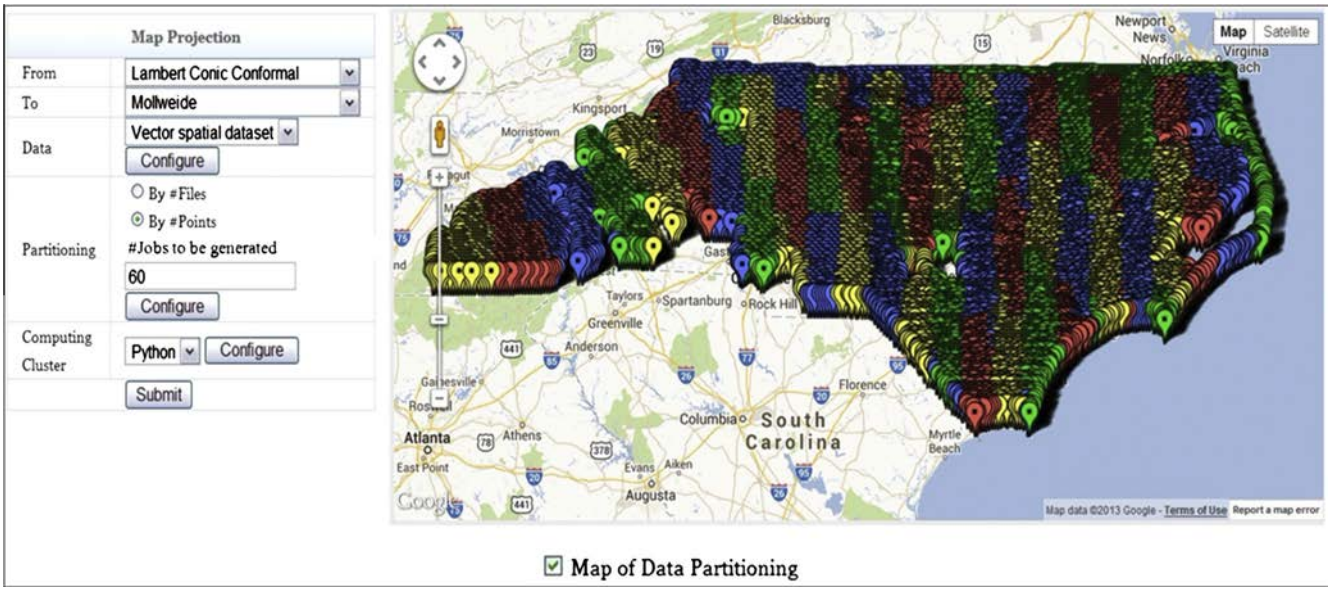

Fig. 5. Web GIS interface for the configuration of parallel map projection of big spatial data.

Please cite this article in press as: Tang, W., & Feng, W. Parallel map projection of vector-based big spatial data: Coupling cloud computing with graphics processing units. *Computers, Environment and Urban Systems* (2014), http://dx.doi.org/10.1016/j.compenvurbsys.2014.01.001

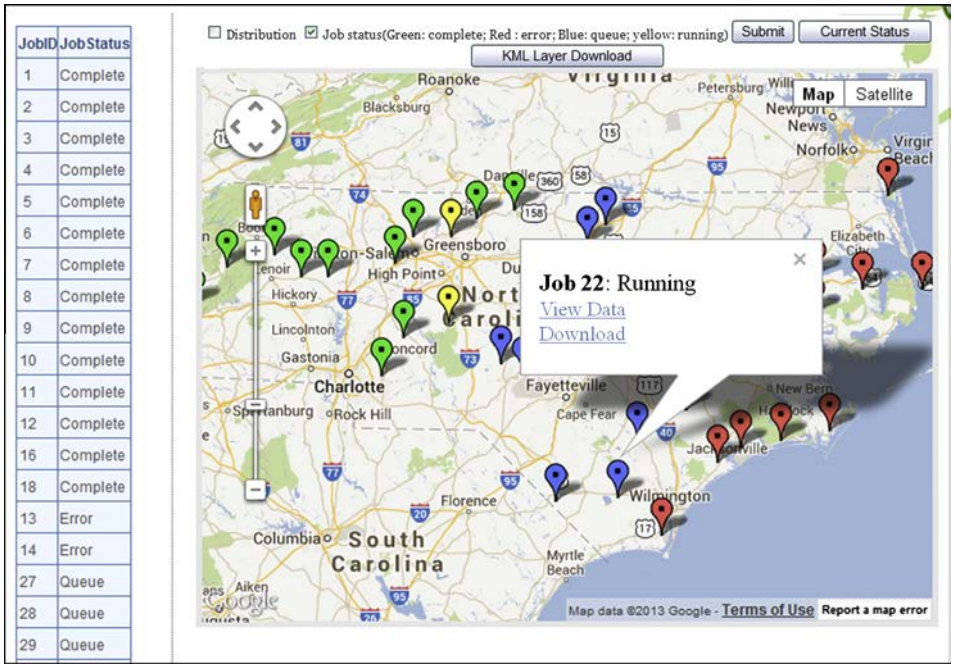

Fig. 6. Web GIS interface for the monitoring of job status for parallel map projection.

corresponds to. Users have the flexibility to visualize task status and download GIS data of interest.

3.5. Implementation and computing resources

Cloud-based virtual machines play a pivot role in coordinating interactions with the front-end Web GIS portal and back-end high-performance computing clusters. We construct a private cloud infrastructure to support the on-demand request of virtual machines. An open-source cloud computing platform, OpenStack (OpenStack, 2013), is used to build this private cloud at the IaaS level. OpenStack is composed of three basic modules: Nova (compute), Swift (data storage), and Glance (image), to empower the on-demand delivery of virtual machine instances. These modules together provide cloud-required capabilities of compute, network, and storage. OpenStack provides a collection of services that coordinate together to install these modules either on a single server or across multiple machines. In this study, we install and configure the OpenStack platform using a single server. The OpenStack private cloud that we build allows us to request a set of virtual machine instances for cloud-enabled geocomputation (physical configurations of host and computing machines are the same: 2.93 GHz of clock rate and 4 GB of memory). The functionality in the second layer of the parallel map projection framework is implemented within Ubuntu Linux environment, encapsulated into an image for OpenStack that allows for re-deployment and extension.

High-performance computing resources used in this study are a GPU cluster that comprises 32 nodes each consisting of 12 CPUs (2.67 GHz of clock rate), 3 GPU devices and 12 GBs computer memory. The cluster has about 26 terabytes of storage and uses infiniband network connections. GPU devices are Nvidia Fermi GPU cards each having 448 cores (14 streaming multiprocessors by 32 CUDA cores), 3 GBs of global memory, and 1.15 GHz of clock rate. The maximum number of CUDA threads per block is 1,024, and the maximum number of block of a grid is 65,535. The code of the map projection algorithms is adapted from GEOTRANS (GEOTRANS, 2013), and CUDA version 4.2 is used for GPU-enabled parallel programming. We use TORQUE (TORQUE, 2013) to manage computing resources on the GPU cluster for parallel map projection. The Web GIS portal is developed using a server-side scripting language, PHP (see PHP, 2013), and Google Maps Javascript API (version 3; see GoogleMap, 2013).

4. Experiments

In this section, we focus on evaluating the parallel map projection framework for transforming vector-based big spatial data. We design two experiments: the first experiment is to evaluate the acceleration performance of the GPU-enabled parallel map projection algorithm; in the second experiment, we target on comparing the performance of the framework using multi-GPUs in response to load balancing. In both experiments, we re-project datasets from Lambert Conic Conformal projection to Mollweide projection.

We take advantage of two types of metrics, computing time and acceleration factor (see Preis, Virnau, Paul, & Schneider, 2009), to help assess the computing performance of the parallel map projection framework. In this study we ignore I/O time during which GIS data are pre- and post-processed. In other words, we only consider computing time spent on map projection so as to obtain a clear understanding of how the parallel algorithm accelerates the map projection. Acceleration factor is obtained by the following equation:

$$AF = T_{cpu}/T_{gpu} \quad (1)$$

where AF is acceleration factor. T_{cpu} denotes CPU-based sequential computing time, and T_{gpu} represents the computing time of the GPU-enabled parallel map projection algorithm. Acceleration factor reflects the ratio of CPU-based sequential computing time over computing time spent on a GPU device. A large acceleration factor represents that a high speed up is obtained from GPU acceleration. There are two types of computing time regarding whether taking into account the data transferring time between GPUs and CPUs. Correspondingly, we have two acceleration factors: with and without the consideration of time for data transferring between CPUs and GPUs.

Please cite this article in press as: Tang, W., & Feng, W. Parallel map projection of vector-based big spatial data: Coupling cloud computing with graphics processing units. *Computers, Environment and Urban Systems* (2014), http://dx.doi.org/10.1016/j.compenvurbsys.2014.01.001

4.1. Experiment 1: GPU performance in response to problem size

The purpose of this experiment is to investigate how problem size influences the computing performance of the GPU-enabled parallel map projection. Because map projection of vector-based GIS data per se operates at the point level, altering the number of points in the data used in this study is a way to examine the relationship between problem size and computing performance. To generate data for this experiment, we randomly pick 10 sub-datasets from the original data set. We aggregate these 10 datasets together to generate base data (2,362,074 points) for different treatments. We have fourteen treatments in which points are randomly drawn from the base dataset. The number of points varies from 2,362,074 to 37,793,184 at an interval of 2,362,074. Each CUDA thread handles ten points and each block has 512 threads for the GPU-enabled parallel map projection algorithm.

Computing performance results of the GPU-enabled map projection algorithm for the fourteen treatments are shown in Table 1 and Fig. 7. From these results, it can be observed that the GPU-enabled parallel map projection reduces the computing time of transformation (without data transfer time) to the level of milliseconds, while the CPU-based sequential algorithm needs multiple seconds. The acceleration factor without considering data transferring remains relatively stable (around 139–141). Taking into account data transfer time between host and device, the computing time of transformation is from 0.5 to 1.68 s for the fourteen treatments. This difference between these two types of GPU computing time indicates that the data transfer time dominates the total computing time spent on GPUs due to the low bandwidth of transferring data between CPUs and GPUs. This explains that the acceleration factors with data transfer time get down to the range of 5.51–39.57. Acceleration factors with data transfer time exhibit an interesting variation pattern. As the number of points increases, acceleration factors quickly jump from 5.51 to 32 for the first three treatments, then remain relatively stable between the range of 22 and 39 for the rest of the treatments. The low acceleration factors for the first and second treatments can be attributed to the high proportion of data transferring time in the total GPU computing time for map projection.

In general, our GPU-enabled parallel computing approach accelerates the transformation of map projection in a considerable manner. From experimental results, we see that the larger the size of input data (i.e., problem size), the higher speed up that we gain in terms of pure computing time for map projection. Further, it needs to be ensured that the size of input data for a GPU device does not exceed the capacity of the GPU device with respect to, for example, its memory. The transformation operation of coordinates for map projection is independent with each other—in other words, the transformation of a spatial entity (point in this study) does not need information from other spatial entities. This leads to a good linear relationship between acceleration factors without considering data transfer and problem size. However, because of the dominant influence of data transfer between CPUs and GPUs, this relationship is modified considerably after data transfer is taken into account. It is necessary to port a large number of points into GPUs if we want to fully take advantage of the many-core architecture of GPUs for massively parallel computing. Further, this suggests that small GIS-based spatial datasets may need to be pre-processed and aggregated into datasets that are sufficiently large to best exploit parallel computing power on GPUs.

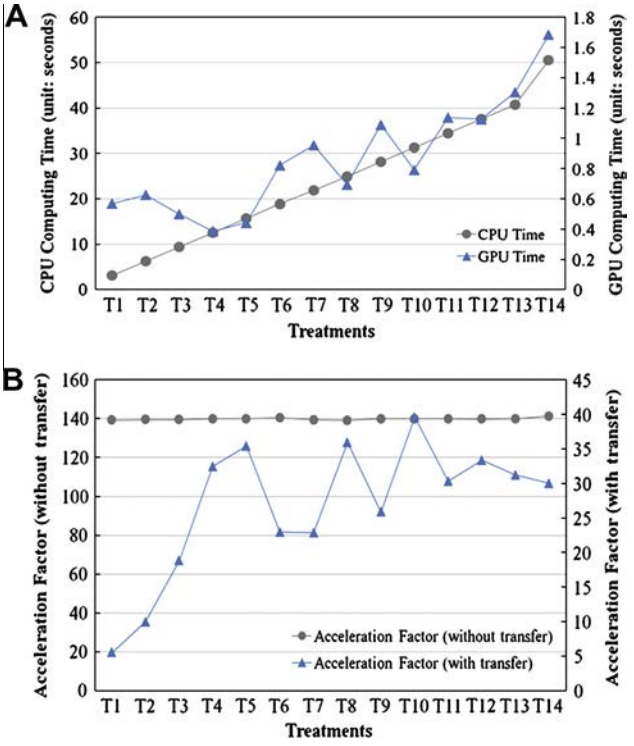

Fig. 7. Comparison of the computing performance of GPU-enabled parallel map projection in response to problem size (A: computing time; B: acceleration factors; time unit: seconds).

Table 1
Results of computing performance for experiment 1 (time unit: seconds; AF: Acceleration Factor; datasets were resampled from the entire North Carolina LiDAR dataset).

Treatments	CPU time	With data transfer		Without data transfer	
		GPU time	AF	GPU time	AF
T1	3.13	0.5672	5.52	0.0225	139.29
T2	6.26	0.6266	9.99	0.0449	139.56
T3	9.39	0.4992	18.81	0.0673	139.56
T4	12.54	0.3864	32.45	0.0896	139.92
T5	15.66	0.4419	35.44	0.112	139.86
T6	18.89	0.8219	22.98	0.1344	140.50
T7	21.85	0.9538	22.91	0.1567	139.40
T8	24.95	0.6942	35.94	0.1791	139.27
T9	28.20	1.0896	25.88	0.2015	139.94
T10	31.33	0.7916	39.58	0.2238	140.01
T11	34.45	1.1366	30.31	0.2461	139.97
T12	37.56	1.1261	33.35	0.2686	139.85
T13	40.72	1.3037	31.24	0.291	139.95
T14	50.57	1.6841	30.03	0.358	141.24

4.2. Experiment 2: Load balancing effect

In the second experiment, we compare the effect of load balancing on the computing performance of the parallel map projection framework. We use the entire NC LiDAR dataset consisting of 13,596 sub-datasets. From results in experiment 1, we see that spatial data ported into GPUs should be sufficiently large. Thus, in experiment 2, we first aggregate these 13,596 sub-datasets by randomly picking a fixed number of original sub-datasets (14 in this experiment). As a result, 990 sub-datasets are generated. We design two treatment groups for the comparison of load balancing effect. The first treatment group is a control group in which 990 sub-datasets are randomly allocated to the GPU devices to be used and the number of sub-datasets for each GPU is equivalent. For the second treatment group, we apply a load balancing strategy that ensures the total number of points assigned to each GPU is close to each other. Each treatment group includes five treatments that use alternative number of GPUs for map projection. The numbers

Please cite this article in press as: Tang, W., & Feng, W. Parallel map projection of vector-based big spatial data: Coupling cloud computing with graphics processing units. *Computers, Environment and Urban Systems* (2014), http://dx.doi.org/10.1016/j.compenvurbsys.2014.01.001

Table 2
Results of computing time for parallel map projection using multiple GPUs (time unit: seconds; the entire North Carolina LiDAR dataset was used).

#GPUs	Computing time (with transfer time)			Computing time (without transfer time)		
	Load balancing		Time difference	Load balancing		Time difference
	Before	After		Before	After	
20	30.72	27.11	3.61	4.56	3.78	0.79
30	19.64	19.03	0.61	2.97	2.58	0.39
40	15.32	15.28	0.03	2.27	2.02	0.25
50	12.51	12.26	0.25	1.96	1.64	0.33
60	10.91	10.78	0.13	1.70	1.42	0.29

Table 3
Acceleration factors of the parallel map projection algorithm using multiple GPUs (acceleration factor was derived with respect to a single CPU).

#GPUs	Acceleration factor (with transfer time)		Acceleration factor (without transfer time)	
	Before load balancing	After load balancing	Before load balancing	After load balancing
20	331.65	375.78	2232.96	2699.08
30	518.84	535.43	3435.28	3955.19
40	665.22	666.62	4481.78	5034.20
50	814.44	831.31	5177.09	6209.15
60	933.64	944.89	5983.73	7199.43

of GPUs for the five treatments in each treatment group are 20, 30, 40, 50, and 60. For CUDA configuration, each block consists of 512 threads, and ten points are assigned to a single thread for map projection.

The sequential time using a single CPU for transforming the entire LiDAR dataset is 10,189.28 s (about 2.83 h). Table 2 and 3 report the results of computing time and acceleration factors for the GPU-enabled map projection algorithm before and after applying load balancing. As we can see, it only takes several seconds to re-project the entire LiDAR dataset when we use multiple GPUs to accelerate map projection. This is the case when we consider data transferring between CPUs and GPUs. For the situation of considering data transfer time, computing time for re-projection drops from about 31 s down to 11 s when the number of GPUs used increases from 20 to 60. Correspondingly, acceleration factors exhibit an increasing pattern: from about 300 (331.7 without load balancing and 375 with load balancing) to 900 (933.64 without load balancing and 944.9 with load balancing). These results show that acceleration from the use of multiple GPUs is considerable. Further, with respect to the effect of load balancing, it can be observed that computing time after applying load balancing is generally lower than that without load balancing. As the number of GPUs increases, reduction in computing time after applying load balancing tends to decline. As reflected from acceleration factors, the use of load balancing leads to increase in acceleration factors (both for considering data transferring or not).

To further study the effect of CPU–GPU data transferring, we run the treatment of 60 jobs (before applying load balancing) using 60 CPUs (i.e., we conduct parallel map projection on multiple CPUs). Fig. 8 illustrates the scatterplots of computing time of each job on GPUs and CPUs. CPU computing time for the 60 individual jobs varies from about 110 to 240 s. GPU computing time for each of these jobs ranges from 0.8–1.7 s (without data transferring time) and 5–11 s when data transferring time is considered. GPU time without considering data transferring exhibits a fairly good linear relationship with CPU time. When data transferring time is taken into account, GPU computing time is perturbed but it is positively related to CPU computing time.

Results in experiment 2 indicate that the use of multiple GPUs has a compounded influence on the acceleration of map projection of spatial data. Using a single GPU allows us to gain about 20–40 times of acceleration (as shown in experiment 1). Once multiple

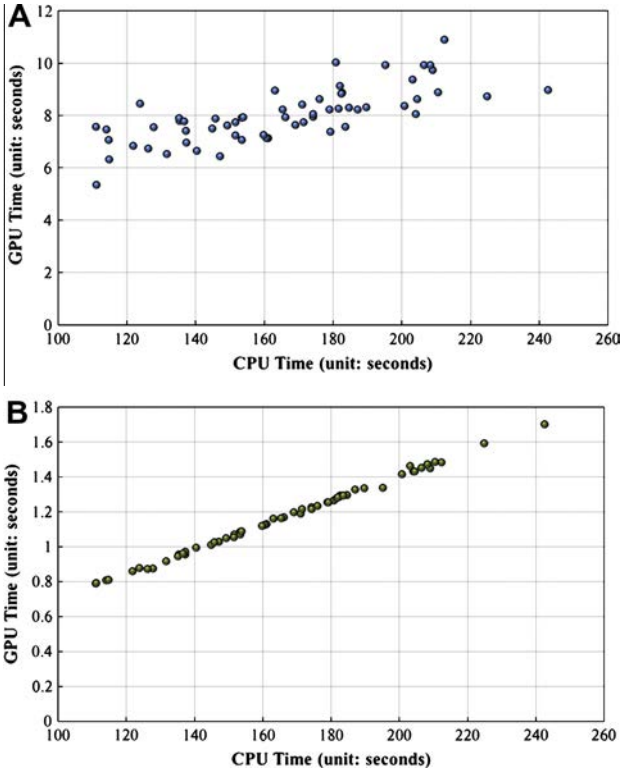

Fig. 8. Effect of CPU–GPU data transferring in parallel map projection accelerated by multi-GPUs (A: scatterplot between CPU computing time and GPU time including data transferring for 60 individual jobs; B: scatterplot between CPU computing time and GPU time excluding data transferring for 60 individual jobs; numbers of GPUs and CPUs: 60).

GPU devices are employed, the acceleration factors can reach a hundred level (from an approximate range of 300–900). Because map projection tasks are independent with each other, there is no communication overhead among GPUs. This explains such substantial acceleration for map projection using multi-GPUs. Moreover, because data transferring between CPUs and GPUs dominates the entire map projection procedure using GPUs, it is imperative to seek a way that reduces data transferring (either at

Please cite this article in press as: Tang, W., & Feng, W. Parallel map projection of vector-based big spatial data: Coupling cloud computing with graphics processing units. *Computers, Environment and Urban Systems* (2014), http://dx.doi.org/10.1016/j.compenvurbsys.2014.01.001

hardware or software levels). This may lead to further enhancement in GPU acceleration of map projection. More importantly, the design and use of appropriate load balancing strategies are necessary to best exploit the massively parallel computing power from multiple GPUs.

5. Conclusion

In this article, we develop a parallel map projection framework for the transformation of vector-based big spatial data among alternative map projections. The parallel map projection framework is based on the integration of capabilities of three layers: GPU-enabled high-performance computing, cloud computing, and Web GIS. This integrative framework provides substantial support for best leveraging each component that complements to each other with respect to the map projection of big spatial data. The layer of cloud computing provisions a collection of virtual machines that serves a key role in terms of interacting with front-end users and back-end high-performance computing clusters. GPU clusters provide many-core massively parallel computing power that holds great promise for handling efficiently big spatial data. Experimental results reported in this study demonstrate substantial acceleration obtained through GPU-enabled parallel map projection. Moreover, it is imperative to apply pre-processing (e.g., aggregation) on big spatial data of interest and design parallel strategies tailored to the spatial data to leverage the massively parallel computing power on many-core GPUs. Parallel strategies, represented by domain decomposition and load balancing, are of particular importance for achieving the best exploitation of high-performance computing resources on the processing and analysis of GIS-relevant data (see Wang, 2010; Xia, Liu, Ye, Wu, & Zhu, 2012; Xia et al., 2011; Yang, Wu et al., 2011).

The emergence of cloud computing extended from service-oriented computing (Foster, 2005; Foster, Zhao, Raicu, & Lu, 2008) exposes us with scalable and on-demand computing resources. The cloud-enabled capabilities of the parallel computing framework can be deployed and further extended on alternative cloud computing infrastructures according to the need of managing the size and complexity of big spatial data. However, cloud computing is on the basis of the virtualization of compute, network, and storage capabilities. The current virtualization technologies lead to the fact that the computing performance of clouds may not outperform computing clusters that are directly built on physical computing hardware (see Huang et al., 2013; Jackson et al., 2010). This leads to the integration of cloud computing and high-performance computing for the efficient map projection of big spatial data in this study.

The transformation of vector-based data among alternative map projections is a fundamental spatial data handling step when we conduct spatial analysis and modeling using different GIS data. This transformation operation often leads to a computational bottleneck in face of big spatial data. The parallel map projection framework based on the coupling of cloud and high-performance computing provides a potential solution that relieves the computational bottleneck. Furthermore, this framework offers insights into the efficient GIS-based handling of big spatial data (vector or raster) using state-of-the-art cloud computing. Before the performance issue induced by virtualization is solved, the integration of both cloud computing and high-performance computing clusters remains as a suggested solution for the manipulation of big spatial data.

Future studies will focus on the following directions. First, more spatial domain decomposition strategies will be developed to further improve the acceleration performance of the GPU-enabled parallel map projection algorithms. Second, the use of this framework for the map projection of raster-based big spatial data is a promising direction. Many raster-based spatial data are becoming big data as spatiotemporal resolution is finer and the extent that they cover becomes larger (e.g., continental or global). Third, GIS-based spatial analysis approaches built on these big spatial data will be parallelized and incorporated into this framework to facilitate big spatial data analytics.

Acknowledgements

The authors would like to thank support from US NSF XSEDE Supercomputing Resource Award (TG-SES090019): "Extending and Sustaining CyberGIS Discovery Environment" and Faculty Research Grant from the University of North Carolina at Charlotte. GPU computing resources were provided by University Research Computing (URC) at the University of North Carolina at Charlotte. The authors thank Ran Tao for assistance on the development of private cloud computing infrastructure in this study, and Huifang Zuo for proofreading the early version of this manuscript.

References

Agrawal, D., Das, S., & El Abbadi, A. (2011). Big data and cloud computing: Current state and future opportunities. In *Proceedings of the 14th international conference on extending database technology* (pp. 530–533). ACM.

Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., et al. (2010). A view of cloud computing. *Communications of the ACM, 53*, 50–58.

Atkins, D. E., Droegemeie, K. K., Feldman, S. I., Garcia-Molina, H., Klein, M. L., Messerschmitt, D. G., et al. (2003). *Revolutionizing science and engineering through cyberinfrastructure: Report of the national science foundation blue-ribbon advisory panel on cyberinfrastructure*.

Behzad, B., Liu, Y., Shook, E., Finn, M. P., Mattli, D. M., & Wang, S. (2012). A performance profiling strategy for high-performance map re-projection of coarse-scale spatial raster data. In *Auto-Carto 2012, a cartography and geographic information society research symposium*, Columbus, OH.

Burrough, P. A., & McDonnell, R. A. (1998). *Principles of geographical information systems*. New York, NY: Oxford University Press.

Buyya, R., Yeo, C., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems, 25*, 599–616.

CUDA (2013). *CUDA*. <http://www.nvidia.com/object/cuda_home_new.html> (Retrieved 07.11.13).

Eucalyptus (2013). *Eucalyptus*. <http://www.eucalyptus.com/> (Retrieved 07.11.13).

Expósito, R. R., Taboada, G. L., Ramos, S., Touriño, J., & Doallo, R. (2013). General-purpose computation on GPUs for high performance cloud computing. *Concurrency and Computation: Practice and Experience, 25*, 1628–1642.

Finn, M. P., Liu, Y., Mattli, D. M., Guan, Q., Yamamoto, K. H., Shook, E., et al. (2012). pRasterBlaster: High-performance small-scale raster map projection transformation using the Extreme Science and Engineering Discovery Environment. In *The XXII international society for photogrammetry & remote sensing congress*, Melbourne, Australia.

Foster, I. (2005). Service-oriented science. *Science, 308*, 814–817.

Foster, I., Zhao, Y., Raicu, I., & Lu, S. (2008). Cloud computing and grid computing 360-degree compared. In *2008 Grid computing environments workshop* (pp. 1–10). Austin, TX, USA: IEEE.

Fu, P., & Sun, J. (2011). *Web GIS: Principles and applications*. Redlands, CA: ESRI Press.

GEOTRANS (2013). *GEOTRANS*. <http://earth-info.nga.mil/GandG/geotrans/> (Retrieved 07.11.13).

Goodchild, M. F. (2007). Citizens as sensors: The world of volunteered geography. *GeoJournal, 69*, 211–221.

GoogleMap (2013). *Google Maps API*. <https://developers.google.com/maps/documentation/javascript/> (Retrieved 07.11.13).

Hey, T., Tansley, S., & Tolle, K. (2009). *The fourth paradigm: Data intensive scientific discovery*. Redmond, Washington: Microsoft Research.

Huang, Q., Yang, C., Benedict, K., Rezgui, A., Xie, J., Xia, J., et al. (2013). Using adaptively coupled models and high-performance computing for enabling the computability of dust storm forecasting. *International Journal of Geographical Information Science, 27*, 765–784.

Jackson, K. R., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., et al. (2010). Performance analysis of high performance computing applications on the Amazon Web services cloud. Cloud Computing Technology and Science (CloudCom). In *2010 IEEE second international conference* (pp. 159–168).

Kirk, D. B., & Hwu, W. W. (2010). *Programming massively parallel processors: A hands-on approach*. Burlington, MA: Morgan Kaufmann.

Maling, D. (1991). Coordinate systems and map projections for GIS. In D. J. Maguire, M. F. Goodchild, & D. W. Rhind (Eds.), *Geographical information systems: Principles and applications*. New York, NY: John Wiley & Sons.

Please cite this article in press as: Tang, W., & Feng, W. Parallel map projection of vector-based big spatial data: Coupling cloud computing with graphics processing units. *Computers, Environment and Urban Systems* (2014), http://dx.doi.org/10.1016/j.compenvurbsys.2014.01.001

Manyika, J., Institute, M. G., Chui, M., Brown, B., Bughin, J., Dobbs, R., et al. (2011). *Big data: The next frontier for innovation, competition, and productivity*. McKinsey Global Institute.

Mell, P., & Grance, T. (2011). The NIST definition of cloud computing (draft). *NIST Special Publication, 800*, 7.

Nguyen, H. (2007). *GPU Gems 3: Programming techniques for high-performance graphics and general-purpose computation*. Upper Saddle River, NJ: Addison-Wesley Professional.

Nickolls, J., Buck, I., Garland, M., & Skadron, K. (2008). Scalable parallel programming with CUDA. *Queue, 6*, 40–53.

OpenCL (2013). *OpenCL*. <http://www.khronos.org/opencl/> (Retrieved 07.11.13).

OpenNebula (2013). *OpenNebula*. <http://opennebula.org/> (Retrieved 07.11.13).

OpenStack (2013). *OpenStack*. <http://www.openstack.org> (Retrieved 07.11.13).

Ortega, L., & Rueda, A. (2010). Parallel drainage network computation on CUDA. *Computers & Geosciences, 36*, 171–178.

Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A. E., et al. (2007). A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum, 26*, 80–113.

Pearson, F. (1990). *Map projection: Theory and applications*. Boca Raton, FL: CRC Press.

Peng, Z., & Tsou, M. (2003). *Internet GIS: Distributed geographic information services for the internet and wireless networks*. Hoboken, NJ: John Wiley & Sons.

PHP (2013). *PHP*. <http://us.php.net/> (Retrieved 07.11.13).

Preis, T., Virnau, P., Paul, W., & Schneider, J. J. (2009). GPU accelerated Monte Carlo simulation of the 2D and 3D Ising model. *Journal of Computational Physics, 228*, 4468–4477.

Ravi, V. T., Becchi, M., Agrawal, G., & Chakradhar, S. (2011). Supporting GPU sharing in cloud environments with a transparent runtime consolidation framework. In *Proceedings of the 20th international symposium on high performance distributed computing* (pp. 217–228). ACM.

Shi, X. (2010). High performance computing: Fundamental research challenges in service oriented GIS. In *Proceedings of the ACM SIGSPATIAL international workshop on high performance and distributed geographic information systems* (pp. 31–34). San Jose, California: ACM.

Shi, L., Chen, H., Sun, J., & Li, K. (2012). Vcuda: Gpu-accelerated high-performance computing in virtual machines. *Computers, IEEE Transactions on, 61*, 804–816.

Slocum, T. A., McMaster, R. B., Kessler, F. C., & Howard, H. H. (2009). *Thematic cartography and geovisualization*. Upper Saddle River, NJ: Pearson Prentice Hall.

Snyder, J. P. (1987). *Map projections – A working manual*. Washington, DC: USGPO.

Tang, W. (2013). Parallel construction of large circular cartograms using graphics processing units. *International Journal of Geographical Information Science*, 1–25.

TORQUE (2013). *TORQUE*. <http://www.adaptivecomputing.com/products/open-source/torque/> (Retrieved 10.15.13).

Usery, E. L., Finn, M. P., & Mugnier, C. J. (2009). Coordinate systems and map projections. In M. Madden (Ed.), *Manual of geographic information systems* (pp. 87–112). Bethesda, MD: American Society for Photogrammetry and Remote Sensing.

Wang, S. (2010). A CyberGIS framework for the synthesis of cyberinfrastructure, GIS, and spatial analysis. *Annals of the Association of American Geographers, 100*, 535–557.

Wang, Y., Wang, S., & Zhou, D. (2009). Retrieving and indexing spatial data in the cloud computing environment. In M. Jaatun, G. Zhao, & C. Rong (Eds.), *Cloud computing* (pp. 322–331). Springer, Berlin Heidelberg.

Wang, S., & Liu, Y. (2009). TeraGrid GIScience Gateway: Bridging cyberinfrastructure and GIScience. *International Journal of Geographical Information Science, 23*, 631–656.

Wang, K., & Shen, Z. (2011). Artificial societies and GPU-based cloud computing for intelligent transportation management. *Intelligent Systems, IEEE, 26*, 22–28.

Wehr, A., & Lohr, U. (1999). Airborne laser scanning—An introduction and overview. *ISPRS Journal of Photogrammetry and Remote Sensing, 54*, 68–82.

WFS (2013). *WFS*. <http://www.opengeospatial.org/standards/wfs> (Retrieved 11.07.13).

Wilkinson, B., & Allen, M. (2004). *Parallel programming: Techniques and applications using networked workstations and parallel computers* (second ed.). Upper Saddle River, NJ, USA: Pearson Prentice Hall.

WMS (2013). *WMS*. <http://www.opengeospatial.org/standards/wms> (Retrieved 11.07.13).

Xia, Y., Liu, Y., Ye, Z., Wu, W., & Zhu, M. (2012). Quadtree-based domain decomposition for parallel map-matching on GPS data. In *Proceeding of 15th IEEE Intelligent Transportation Systems Conference (ITSC 2012)*, Anchorage, AK, USA.

Xia, Y., Kuang, L., & Li, X. (2011). Accelerating geospatial analysis on GPUs using CUDA. *Journal of Zhejiang University-Science C, 12*, 990–999.

Yang, C., Goodchild, M., Huang, Q., Nebert, D., Raskin, R., Xu, Y., et al. (2011). Spatial cloud computing: How can the geospatial sciences use and help shape cloud computing? *International Journal of Digital Earth, 4*, 305–329.

Yang, C., Wu, H., Huang, Q., Li, Z., & Li, J. (2011). Using spatial principles to optimize distributed computing for enabling the physical science discoveries. *Proceedings of the National Academy of Sciences, 108*, 5498–5503.

Zhang, J., You, S., & Gruenwald, L. (2010). Indexing large-scale raster geospatial data using massively parallel GPGPU computing. In *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic, information systems* (pp. 450–453). ACM.

Zhao, Y., Cheng, Z., Dong, H., Fang, J., & Li, L. (2011). Fast map projection on CUDA. In *2011 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)* (pp. 4066–4069), Vancouver, Canada.

Zhao, Y., Padmanabhan, A., & Wang, S. (2013). A parallel computing approach to viewshed analysis of large terrain data using graphics processing units. *International Journal of Geographical Information Science, 27*, 363–384.

Zikopoulos, P., & Eaton, C. (2011). *Understanding big data: Analytics for enterprise class hadoop and streaming data*. New York, NY: McGraw-Hill Osborne Media.

Please cite this article in press as: Tang, W., & Feng, W. Parallel map projection of vector-based big spatial data: Coupling cloud computing with graphics processing units. *Computers, Environment and Urban Systems* (2014), http://dx.doi.org/10.1016/j.compenvurbsys.2014.01.001