International Journal of Digital Earth

ISSN: 1753-8947 (Print) 1753-8955 (Online) Journal homepage: https://www.tandfonline.com/loi/tjde20

Geoprocessing in Cloud Computing platforms – a comparative analysis

Peng Yue , Hongxiu Zhou , Jianya Gong & Lei Hu

To cite this article: Peng Yue , Hongxiu Zhou , Jianya Gong & Lei Hu (2013) Geoprocessing in Cloud Computing platforms – a comparative analysis, International Journal of Digital Earth, 6:4, 404-425, DOI: 10.1080/17538947.2012.748847

To link to this article: https://doi.org/10.1080/17538947.2012.748847

Accepted author version posted online: 12 Nov 2012.
Published online: 07 Dec 2012.

International Journal of Digital Earth, 2013
Vol. 6, No. 4, 404–425, http://dx.doi.org/10.1080/17538947.2012.748847

Taylor & Francis
Taylor & Francis Group

Geoprocessing in Cloud Computing platforms – a comparative analysis

Peng Yue*, Hongxiu Zhou, Jianya Gong and Lei Hu

State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing (LIESMARS), Wuhan University, Wuhan, China

(*Received 2 December 2011; final version received 8 November 2012*)

The emergence of Cloud Computing technologies brings a new information infrastructure to users. Providing geoprocessing functions in Cloud Computing platforms can bring scalable, on-demand, and cost–effective geoprocessing services to geospatial users. This paper provides a comparative analysis of geoprocessing in Cloud Computing platforms – Microsoft Windows Azure and Google App Engine. The analysis compares differences in the data storage, architecture model, and development environment based on the experience to develop geoprocessing services in the two Cloud Computing platforms; emphasizes the importance of virtualization; recommends applications of hybrid geoprocessing Clouds, and suggests an interoperable solution on geoprocessing Cloud services. The comparison allows one to selectively utilize Cloud Computing platforms or hybrid Cloud pattern, once it is understood that the current development of geoprocessing Cloud services is restricted to specific Cloud Computing platforms with certain kinds of technologies. The performance evaluation is also performed over geoprocessing services deployed in public Cloud platforms. The tested services are developed using geoprocessing algorithms from different vendors, GeoSurf and Java Topology Suite. The evaluation results provide a valuable reference on providing elastic and cost-effective geoprocessing Cloud services.

Keywords: geoprocessing; Cloud computing; geospatial service; GIS; Microsoft Azure; Google App Engine

1. Introduction

Recent advancement of Cloud Computing technologies has shown great promise for building spatial data infrastructures (SDIs) or Cyberinfrastructures (Schäffer, Baranski, and Foerster 2010; Yang et al. 2010). Using Cloud Computing platforms, typical IT resources such as storages, computing utilities, and databases are available as services. Providing geoprocessing functions in Could Computing platforms can bring scalable, on-demand, and cost-effective geoprocessing services to geospatial users (Baranski, Deelmann, and Schäffer 2010; Gong, Yue, and Zhou 2010; Yang et al. 2011).

Some public Cloud Computing platforms, such as Microsoft Azure and Google App Engine (GAE), are already available. Different Cloud platforms have their own development paradigms. For example, GAE provides a development environment using Java and Python. Microsoft Azure fully supports.Net environment. In

*Corresponding author. Email: geopyue@gmail.com

addition, applications in the Windows Azure platform differentiate Web roles and Worker roles, while Google does not. Geoprocessing over different Cloud platforms needs to be compared for recommendations in geospatial Cloud applications.

The paper provides a comparative analysis of geoprocessing in Cloud Computing platforms from the following two perspectives. The first one is to compare features of different Cloud platforms in the development of geoprocessing applications. The analysis compares differences in the data storage, architecture model, and development environment based on the experience to develop geoprocessing services in the two Cloud Computing platforms – Microsoft Windows Azure and GAE; emphasizes the importance of virtualization; recommends applications of hybrid geoprocessing Clouds; and suggests a solution on overcoming heterogeneity in Cloud Computing platforms to bring interoperable geoprocessing Cloud services.

The second perspective is to compare the performance of geoprocessing applications in Cloud platforms. This includes performance analyses on different software packages within the same Cloud environment and one software package within different Cloud environments. The geoprocessing algorithms from different vendors, GeoSurf and Java Topology Suite (JTS), are performed and evaluated in GAE. The goal is to analyze how the same geoprocessing functions developed by different providers can have different performance and costs in the same Cloud platform. In addition, the performance of JTS in both GAE and Azure is also evaluated, which provides a valuable reference for selection of various Cloud platforms.

The remainder of the paper is organized as follows: Section 2 briefly introduces the public Cloud Computing platforms, Azure and App Engine, and presents examples and challenges when migrating legacy geoprocessing functions into Cloud Computing environments. Related work is described in Section 3. Section 4 presents a general architecture on integration of geoprocessing functions in Cloud Computing environments, and Section 5 compares features of the two Cloud platforms, Microsoft Windows Azure and GAE, in the development of geoprocessing applications. The performance evaluation of geoprocessing applications in Cloud platforms is presented in Section 6. Section 7 discusses the results. Conclusions and pointers to future work are given in Section 8.

2. Running examples

This section briefly describes two well-known public Cloud Computing platforms (Section 2.1): Microsoft Azure platform, and GAE. Furthermore, it illustrates challenges when moving legacy geoprocessing applications into Cloud Computing environments by providing some examples (Section 2.2).

2.1. Microsoft Azure platform and GAE

Microsoft Azure was first announced in 2008. After that, a developer version was freely available for public tests. The formal version was published at the beginning of the year 2010. The platform provides a Cloud environment to help developers build, host, and scale applications through Microsoft data centers. The platform consists of the following major components (Chappell 2009; Microsoft 2011):

- Windows Azure: It provides a platform for running Windows applications and storing their data in the Cloud. Windows Azure runs on a large number of servers in Microsoft data centers, which are connected into a unified whole by Windows Azure Fabric. Windows-based compute and storage services for Cloud applications are built on top of this fabric.
- SQL Azure: It provides services for relational data in the Cloud based on SQL Server. Although the eventual goal of SQL Azure is to provide a range of data-oriented capabilities such as data synchronization and reporting, the first SQL Azure component is SQL Azure Database, which provides a Cloud-based database management system.
- Windows Azure platform AppFabric: It provides Cloud services for connecting applications running in the Cloud or on premises. The functions provided by AppFabric are called Cloud-based infrastructure services, including the Service Bus and Access Control Service.

GAE was published in 2008. Currently, it can provide services to customers without charge within the threshold of resources such as central processing unit (CPU) time and storage. The GAE includes following major components (Sanderson 2009; Roche and Douglas 2009; Google 2011):

- App Engine runtime environment: The GAE supports applications written in several programming languages such as Java and Python. The runtime environment is a 'sandbox', which lets the application use only features of the server that can scale without interfering with other applications. The Java runtime environments (JREs) allow the developers to build applications using standard Java technologies, including the Java virtual machine (JVM), Java servlets, and the Java programming language or any other language using a JVM-based interpreter or compiler, such as JavaScript or Ruby.
- App Engine datastore: The GAE provides a distributed data storage service that features a query engine and transactions. The distributed datastore grows with the data. The App Engine datastore is implemented with BigTable and Google File System, which are scalable and fault-tolerant.
- App Engine Services: The GAE provides a variety of services that perform common operations such as URL Fetch, Mail, Memcache, and Image Manipulation. These services are accessible through application programming interfaces (APIs).

The components in both Microsoft Azure platform and GAE can be used to address three aspects of Cloud Computing environments: data environment, application environment, and infrastructure services (Figure 1). The data environment provides data storage and access services from Cloud platform vendors, while the application environment emphasizes the computing services. The infrastructure services offer utility services in the Cloud information infrastructure. These three aspects are not addressed by separate components in Cloud platforms. For example, Windows Azure, as a Cloud application environment, also offers blobs, tables, and queues to store data. Both Microsoft Azure platform and GAE are still being developed and tuned. For example, the latest version of Microsoft Azure platform provides Windows Azure Marketplace, which offers ready-to-purchase applications and

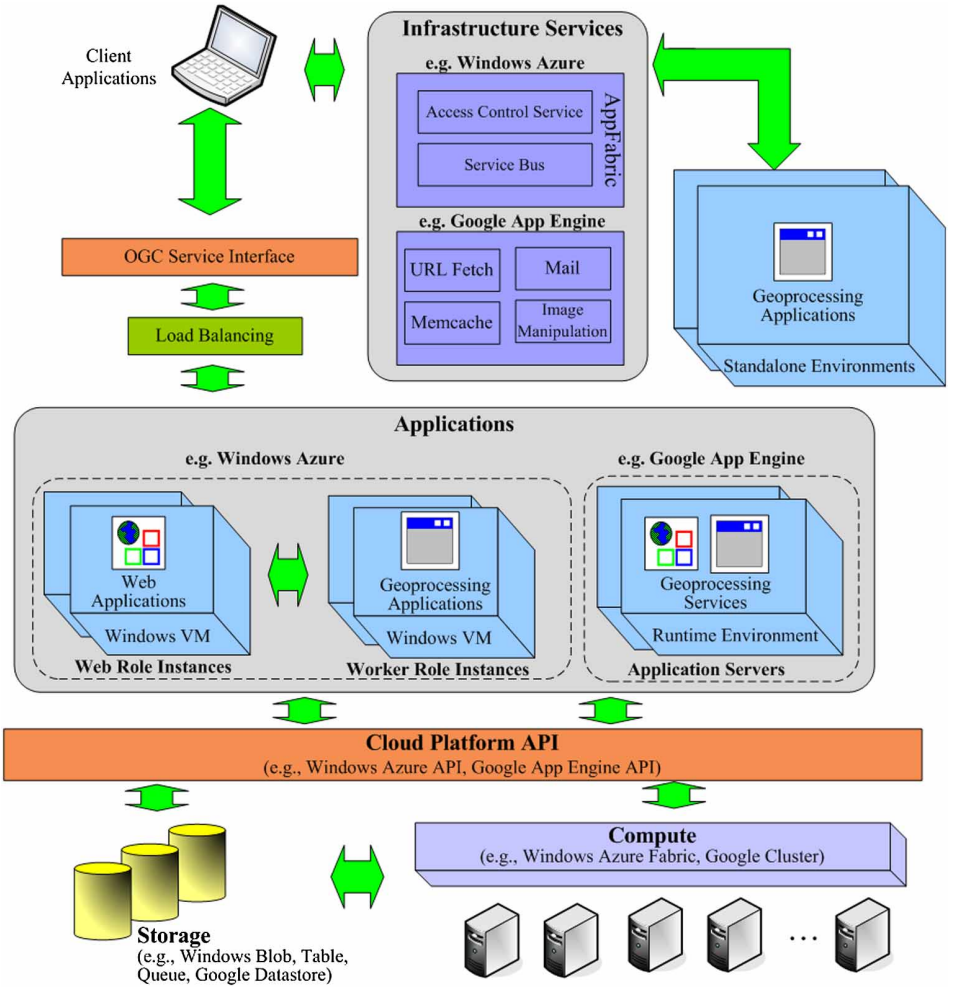

Figure 1. A general architecture for geoprocessing in Cloud platforms.

data. The latest version of GAE also includes the Go runtime environment, which can be used to develop and deploy Web applications written in the Go programming language.

2.2. Examples and challenging issues

The spatial analysis functions in professional GIS software packages have been developed for over decades. However, traditional GIS software systems cannot support an open analysis environment. The analysis functions can only be used in their own proprietary environments. Taking a terrain slope computation as an example, supposing a raster map of the slope can be generated from a raster map for the Digital Elevation Model (DEM) data, using an analysis algorithm written as a Java application. This legacy application works only in a JVM. The task is then how to move this legacy application into different Cloud platforms. The practice can help

identify the difference in the development of Cloud applications in different Cloud platforms, understand the importance of virtualization, and suggest the interoperable solution on geoprocessing Cloud services.

Another example is to move geoprocessing algorithms developed by different providers into Cloud platforms. The geospatial buffer analysis functions from GeoSurf and JTS are tested for this purpose. GeoSurf is a leading Web GIS software system in China and has been used in many projects on digital cities in China. It is developed by the State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing of Wuhan University (Zhu et al. 2003). The spatial analysis component in GeoSurf is implemented using Java. JTS provides a complete and robust implementation of fundamental 2-dimensional spatial algorithms (Vivid Solutions 2011). The implementation code of JTS is written in Java. It is widely used as an open source solution in many Java-based GIS projects. The same functional geoprocessing Cloud services, provided by different Software as a Service (SaaS) vendors in Cloud platforms, offer the possibility to choose cost-effective services.

The primary challenge in dealing with these examples is that conventional spatial analysis applications need to be adapted to running environments in Cloud Computing environments. In particular, how components in Microsoft Azure platform and GAE can be used to support functional requirements in geoprocessing service developments, such as enabling geospatial data management functions (i.e. file inputs/outputs) of legacy applications in Cloud platforms and exposing conventional spatial analysis algorithms as services in a Cloud architecture?

The second challenge is about interoperable service interfaces that geospatial applications interact with Cloud services. Geospatial data are collected by diverse means and are highly complex and heterogeneous. Often, the temporal and spatial coverage, resolution, origin, format, and map projections are incompatible. Because of the multidisciplinary nature of Earth science modeling and applications, geoprocessing functions involved are always diverse. Interoperability issue, therefore, is important in a distributed environment. One possible solution is to promote the use of the Open Geospatial Consortium (OGC) standards, which allow the seamless access to geospatial data and geoprocessing functions in a distributed environment, regardless of heterogeneity of involved geospatial data and geoprocessing platforms.

3. Related work

Geoprocessing functions are important for discovering hidden and useful geospatial information and are widely used in Earth science modeling and applications. The growth of the Web has resulted in the Web-based sharing of large volumes of distributed geospatial data and computational resources. Geoprocessing is a core component in Geospatial Cyberinfrastructure (Yang et al. 2010), which must support geospatial data processing within and across scientific domains. Cloud Computing is associated with a new paradigm to provide a computing infrastructure (Vaquero et al. 2009), which can support geoprocessing over the Web (Yue et al. 2010). Several new aspects of the information infrastructure enabled by Cloud Computing are the illusion of infinite computing resources available on demand, the elimination of an up-front commitment by Cloud users, and the ability to pay for use of computing resources on a short-term basis as needed (Armbrust et al. 2009). A Cloud could be public or private, depending on the deployment approaches. The public Cloud is

available to the public, while the private Cloud is used inside an organization. According to the type of provided capabilities, major categories of Cloud Computing include Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and SaaS (NIST 2009). Microsoft Azure and GAE are examples of PaaS. A known example of IaaS is Amazon's Elastic Compute Cloud (EC2), which allows consumers to manage virtual machines (VMs).

Several efforts on using Cloud Computing in geospatial applications have been reported (Table 1). Baranski, Schaeffer, and Redweik (2009) create a Cloud service for the spatial buffer analysis in GAE and conduct a stress test for scalability evaluation. The interface of the service follows the Web Processing Service (WPS)

Table 1. Typical cases on geospatial Cloud Computing.

Geospatial Cloud	Geospatial application	Programming language	Service interface	Types of Cloud	Cloud platform
52 North geoprocessing Cloud (Baranski et al. 2009)	Buffer analysis by JTS	Java	OGC WPS	Public	Google App Engine
Spatial data Cloud (Wang et al. 2009)	Spatial data indexing and management	Java	OGC WKB/ WKT	Public	Google App Engine
Payment model in geoprocessing Cloud (Baranski et al. 2010)	Pay-per-use revenue model for geoprocessing services	Java	OGC WPS/ OASIS XACML	Public	Amazon EC2
Cloud-enabled geoprocessing (Gong et al. 2010)	Terrain slope computation	Java/C#	OGC WPS	Public	Microsoft Azure
FGDC GeoCloud (Huang et al. 2010)	GEOSS Clearinghouse	Java	OGC Catalogue Service for the Web (CSW)	Public	Amazon EC2
Cloud-enabled SDI (Schäffer et al. 2010)	Buffer and intersection analysis by JTS	Java	OGC WPS	Public	Amazon EC2
Cloud-enabled WMS (Blower 2010)	Web Map Service	Java	OGC WMS	Public	Google App Engine
QoS in geoprocessing Cloud (Baranski et al. 2011)	Coordinate transformation	Java	OGC WPS	Hybrid Cloud	Private Cloud/ Amazon EC2

FGDC, US Federal Geographic Data Committee; GEOSS, Global Earth Observation System of Systems; JTS, Java Topology Suite; OASIS, Organization for the Advancement of Structured Information Standards; OGC, Open Geospatial Consortium; QoS, Quality of Service; SDI, Spatial Data Infrastructures; WKB, well-known binary; WKT, well-known text; WMS, Web Map Service; WPS, the Web Processing Service; XACML, eXtensible Access Control Markup Language.

specification, and the implementation is based on the Java-implemented 52 North open source WPS software. A pay-per-use revenue model for geoprocessing services in the Cloud is further proposed and implemented in the Amazon EC2 platform (Baranski et al. 2010). The model is based on a common policy-based eXtensible Access Control Markup Language (XACML) security architecture. The XACML is a standard defined by the Organization for the Advancement of Structured Information Standards. Y. Wang, S. Wang, and Zhou (2009) use OGC well-known binary and well-known text (WKT) for data exchange in the Cloud Computing environment, and demonstrate how spatial indexes can be created in the GAE. In the GeoCloud Sandbox Initiative of the US Federal Geographic Data Committee, Huang et al. (2010) deploy and test the metadata catalog service in the Amazon EC2 platform to support the Global Earth Observation System of Systems Clearinghouse. Ludwig and Coetzee (2010) provide a conceptual comparison among several PaaS Cloud Computing solutions including Microsoft Azure and GAE from the aspect of the Cloud security and discuss its implications for geoprocessing services. Schäffer et al. (2010) discuss Cloud-enabled SDIs. The Amazon Web Services, together with an OGC WPS implementation hosting the buffer and intersection processes, are used in a use case for the public risk management. Blower (2010) provides an implementation of OGC Web Map Service in the GAE. Baranski et al. (2011) propose a hybrid Cloud architecture by composing private Cloud and computational resources of public Cloud. In this architecture, resources of external providers can be requested and integrated on demand into the local infrastructure to match Quality of Service requirements.

Table 1 lists cases of geospatial Cloud applications from a perspective of development. The concept of Cloud Computing has spread widely in the past several years. Some well-known Cloud Computing platforms are already in place. The community, especially the geospatial community, now needs some implementation cases to guide practices and real applications and validate and support the conceptual analysis. Several typical cases with prototype implementations are compared in Table 1. Most implementations are in the public Cloud platforms. The geospatial services in Clouds always try to follow the OGC service standards. In addition, the programming language used for development in these implementations is usually Java due to its platform-independent feature. The advantage of this feature is the ability to move easily from one computer system to another, which is crucial in distributed environments.

Gong et al. (2010) present an architecture and implementation for a terrain slope computation service in the Microsoft Azure platform. It uses the blob to store the binary image data, implements the slope computation algorithm in a Worker role application, develops a WPS interface for a Web role application, and communicates messages between Web and Worker roles using a queue service. However, as shown by most cases in Table 1, there is no implementation comparison between different Cloud platforms. The previous work is extended by choosing another Cloud platform for development, i.e. implementing the same slope computation service using the GAE. Furthermore, to the best of our knowledge, there is no comparison on services with the same function (e.g. the buffer services by GeoSurf and JTS in this work) in one Cloud platform or different Cloud platforms. The practice on moving the legacy software to the Cloud platform can provide an informed

understanding of the work involved in the migration. The performance evaluation can also help us select cost-effective geoprocessing services.

4. A general architecture for integration of geoprocessing functions in Clouds

By analyzing existing components in Microsoft Azure platform and GAE, we propose that the integration of geoprocessing functions and Cloud platforms takes place at three points, in order to address challenges in Section 2.2. Figure 1 shows a proposed architecture of the integrated system.

The first point is to move computational applications into the application environment of Cloud platforms. Conventional analysis applications work in their own standalone environments. The applications in Cloud platforms can scale to distributed servers. The Windows Azure platform differentiates Web roles and Worker roles. An application with a Web role is a Web application supported by Internet Information Services (IIS). An application with a Worker role is used for the generic development of applications in the Cloud and can perform background processing for a Web role. Both Web and Worker role applications can have multiple instances on Windows Azure, each running in its own Windows VM. The VMs are maintained by Windows Azure and allow role applications to access resources in data servers managed by Windows Azure Fabric via Windows Azure API. The GAE provides application servers, which are responsible for distributing requests and starting up instances. The application code executes in a runtime environment, which provides access to system resources such as CPU and memory. For example, in the JRE, the App Engine Java Software Development Kit (SDK) supports the application development using Java standard APIs. One advantage of using Cloud Computing platforms for providing geoprocessing services is that instances of applications are running as needed to maximize the throughput. And another advantage is that existing Cloud platforms provide built-in load balancing to Web applications when multiple instances of the same application are running.

The integration of geospatial analysis functions with Windows Azure can use both Worker role and Web role applications. For example, Worker role applications can be developed for running geoprocessing algorithms, and Web role applications can implement Web service interfaces for geoprocessing services. The development of Worker and Web role applications uses the Windows Azure API. The development of geoprocessing functions in the GAE is to make traditional application code, including the Web interface and geoprocessing algorithms, executable in the Cloud runtime environment. The service interface for geoprocessing services in different Cloud platforms can adopt the OGC WPS standard. OGC WPS specifies a standard interface and protocol for discovering and executing distributed geoprocessing processes (Schut 2007). The use of the standard interface makes these Cloud services interoperable.

The second point is to manage the application data using storage services from the data environment of Cloud platforms. The storage service in Microsoft Azure offers several ways including blobs, tables, and queues to store data. The blobs store binary or text data. The queues store messages for communication between components of Windows Azure applications such as the communication between Web role instances and Worker role instances. The tables provide a structured way to store non-relational data. For the relational data, the Microsoft SQL Azure extends

SQL Server capabilities to the Cloud and offers a relational database service. The App Engine datastore provides distribution, replication, and load-balancing services for storage. Geospatial data that are required for geoprocessing, therefore, can be stored in both Cloud platforms using robust storage services.

The third point is to connect Cloud-based services and legacy geoprocessing applications using infrastructure services. For example, the Service Bus component in the AppFabric provides a facility for identifying service endpoints, publishing them, routing messages among them, and connecting them despite common Internet connectivity challenges such as translations between external and internal network addresses. The URL Fetch service in the GAE allows applications to access resources on the Internet, such as Web services or other data. The service retrieves Web resources using the same high-speed Google infrastructure that retrieves Web pages for many other Google products. These Cloud-based infrastructure services can be connected in distributed applications, whether in a standalone environment or in the Cloud, thus generating a hybrid Cloud. Here the hybrid means that Cloud services can be connected with legacy services to support Cloud applications. Legacy geoprocessing applications, therefore, can be reused by exposing itself via the Service Bus.

5. Implementation comparison

The Windows Azure Tools for Microsoft Visual Studio 1.2 (June 2010 version) are used to develop geoprocessing services in Windows Azure. The tools are combined with Visual Studio 2010, SQL Server 2005 Express, and IIS 7.0 to provide a Windows Azure development environment including development fabric and storage utilities, which can simulate Windows Azure compute and storage services in desktop machines. Thus the tools allow the creation, building, debugging, and running of Cloud applications in a simulated Windows Azure environment. Once these services pass tests in the development environment and are ready to be used by public, they can be packaged and deployed on Windows Azure by using utilities in the tools.

Geoprocessing Cloud services including terrain slope computation and buffer analysis in the GAE are developed using the App Engine Java SDK). The SDK includes software for a Web server, which simulates the App Engine environment for running and testing Java applications. The Java 6 VM is supported in the App Engine and used for compiling applications. The applications can be published freely in the GAE under certain quotas and limits, such as up to 1 GB of storage and up to five million page views a month (Google 2011). Three geoprocessing services are published in the GAE and available to the public.

The comparison is provided as follows, using the examples in Section 2.2 as illustrations.

5.1. Creating the application framework

The Windows Azure Tools for Microsoft Visual Studio provides a project template for developing a Cloud service. Using this template, a Cloud service solution for the terrain slope computation can be generated. The solution includes three projects: a Cloud service project, a Work role project, and a Web role project. The Work and Web role projects are intended for development of the background geoprocessing

and Web service interface. The Cloud service project in the solution includes files for the service definition and service configuration. The service definition file specifies roles in a service and optional local storage resources. The service configuration file specifies the number of role instances to be deployed for each role in the service and other values of configuration settings.

The Google Plugin for Eclipse provides an easy way to develop, test, and upload App Engine applications. A Web application project for Cloud services can be created by checking the option of using GAE. The project uses the Web Application Archive standard layout and the Java Servlet for interacting with the Web server environment. The environment is similar to the conventional development of Java Web applications. Additional configurations specific to Google Cloud services are required. For example, the configuration file, named appengine-web.xml, specifies the app's application ID registered in GAE.

5.2. Developing WPS interface

According to the WPS specification, the WPS interface includes three operations: GetCapabilities, DescribeProcess, and Execute. The GetCapabilities operation allows a client to request and receive a service capabilities document that describes operations and processes of a specific WPS implementation. The DescribeProcess operation allows a client to get detailed information, such as input and output parameter types, about specific processes. The Execute operation allows the client to run a specific process in a WPS server (Schut 2007). These operations can be invoked via the HTTP (Hypertext Transfer Protocol) GET method with a key value pair (KVP) encoded request or the HTTP POST method with an Extensible Markup Language (XML) based encoding. The following examples show the implementation of the KVP encoded DescribeProcess requests for retrieving the process description of the terrain slope computation in the Windows Azure and GAE respectively:

Window Azure:
http://127.0.0.1:81/wps.aspx?Request=DescribeProcess&Service=WPS&Version=0.4.0&Identifier=DEM2SlopeProcess

GAE:
http://dem2slope.appspot.com/SlopeCloud?Request=DescribeProcess&Service=WPS&Version=0.4.0&Identifier=DEM2SlopeProcess

The two requests are same except the server address. The interoperability can be achieved when using the standard interface. One example on execution using HTTP GET in a Web browser is shown in Figure 2. The data on sides of the Web browser are input DEM and output terrain slope.

5.3. Providing the storage of application data

In Microsoft Azure, the Blob service is used for the robust data storage. Figure 3 shows the sequence diagram of processing a WPS Execution request in Azure. In the blob service, blobs are organized into a container beneath a storage account. Geospatial data can be available as either a Web URL to a data product or an URI to the address of a blob. In the former case, the data need to be downloaded and stored into blobs. In the latter one, the blob can be accessed directly in the same

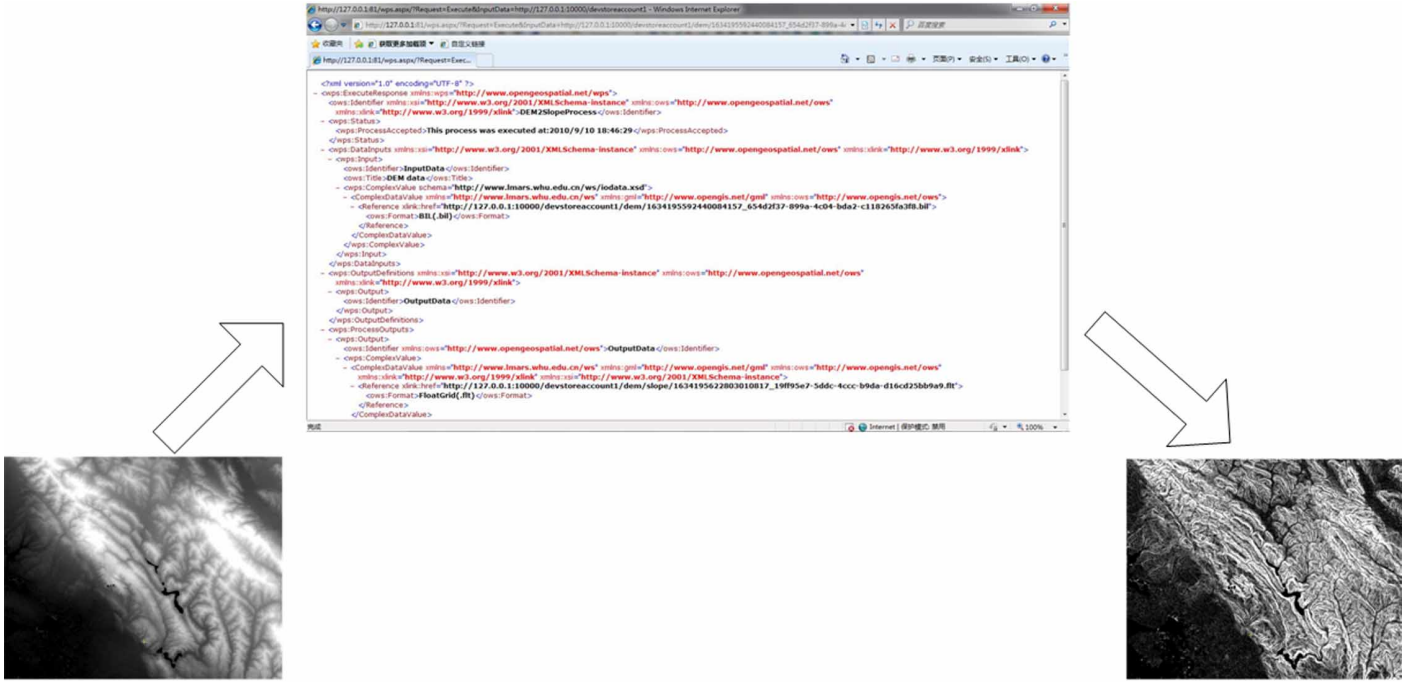

Figure 2. WPS Execution request using the HTTP GET method.

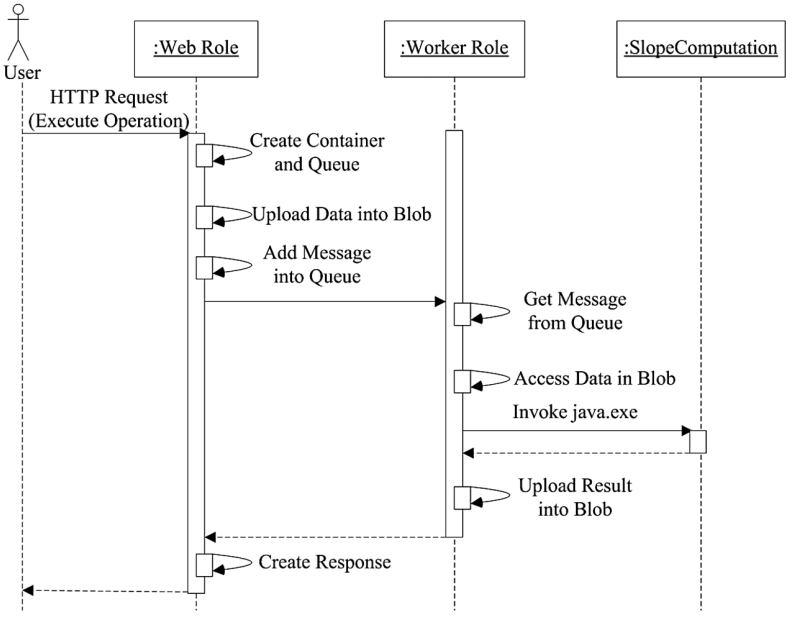

Figure 3. Workflow on using storage services in Web and Worker roles.

storage account. The Queue service is used to store communication messages between Web and Worker roles. When an Execution request comes, the container and queue are created if they do not already exist. The data in the WPS request is loaded into blobs of the container. The path to the blob is recorded in a message and added into the queue.

The Worker role detects the message in the queue and processes it using domain business logics, including extracting the path to the blob from the message, accessing data in the blob, invoking the geoprocessing process, uploading the processing result into a blob belonging to a new container, and deleting the old message. The Windows Azure Tools provide the Windows Azure Storage Explorer to view read-only blob and table data.

Applications in GAE can still use the structure in conventional Java Web applications. As shown in Figure 4, the message processing still follows the standard Servlet process. The data storage, however, needs to use the Cloud storage services of Google. Conventional file creation and I/O stream operations require revisiting since they might not work in the Cloud storage environment. The implementation here adopts the GAE Virtual File System (GaeVFS 2011). GaeVFS provides a portability layer that allows users to write application code to access the file system. The functions include creation of folder and files and related read and write functions. GaeVFS is implemented using the GAE datastore and Memcache APIs. The advantage of using GaeVFS is that it simplifies the use of Google datastore. In addition, the code can run unmodified in either GAE or non-GAE servlet environments.

Although the application data for cases in the paper can be managed successfully by GAE and Azure, the management of huge volumes of geospatial data in the Cloud environments needs more investigations. For example, how can traditional

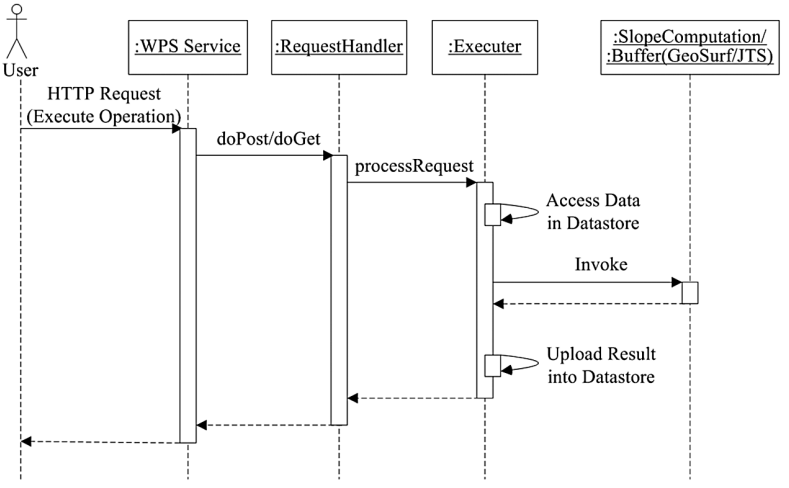

Figure 4. Workflow on using storage services in Google App Engine.

spatial indexes and queries be supported in Cloud environments? Can the MapReduce programming model or BigTable be leveraged with the spatial data management? These are outside the scope of this paper. For some existing approaches on how to manage massive amounts of geospatial data in Cloud environments, please refer to Gonzalez et al. (2010) and Cary et al. (2010).

5.4. Implementing geoprocessing algorithms

It usually requires the rewriting, compiling, or building computer source code, when migrating legacy geoprocessing programs into Cloud Computing platforms. Marx (2009), a developer of Microsoft Windows Azure, suggests an approach to run Java applications in Windows Azure by taking advantage of VMs in Windows Azure. A JRE is packed with a Java application and deployed into a VM in the Windows Azure. The latest version of Microsoft Azure platform now starts to provide a Windows Azure SDK for Java, which also packs together the JRE and Java project for each application. Thus Java code for the slope computation does not need to be modified and can work in Windows Azure.

In GAE, the JRE already includes the Java Standard Edition Runtime Environment (JRE) 6 platform and libraries. However, it provides limited access to JRE classes (the JRE Class White List) due to the restrictions of the sandbox. The source code of conventional geoprocessing algorithms have to be tested and modified to follow the restrictions. For a heavy-weight geoprocessing package such as that in GeoSurf, it requires considerable efforts, since GeoSurf is the Web GIS software including many Java libraries that are not related to geoprocessing. It is better to peel off these unrelated libraries before the migration. On the other hand, the source code for the slope computation case and JTS focus on geoprocessing algorithms and can be considered as light weight. The slope and buffer (GeoSurf and JTS) services are available at http://geopw.whu.edu.cn/ws/quickstart.html. They use the same servlet framework for supporting the WPS interface. As shown in Figure 5, the HTML page provides a data upload function. The test data is the Yangtze River in China, which

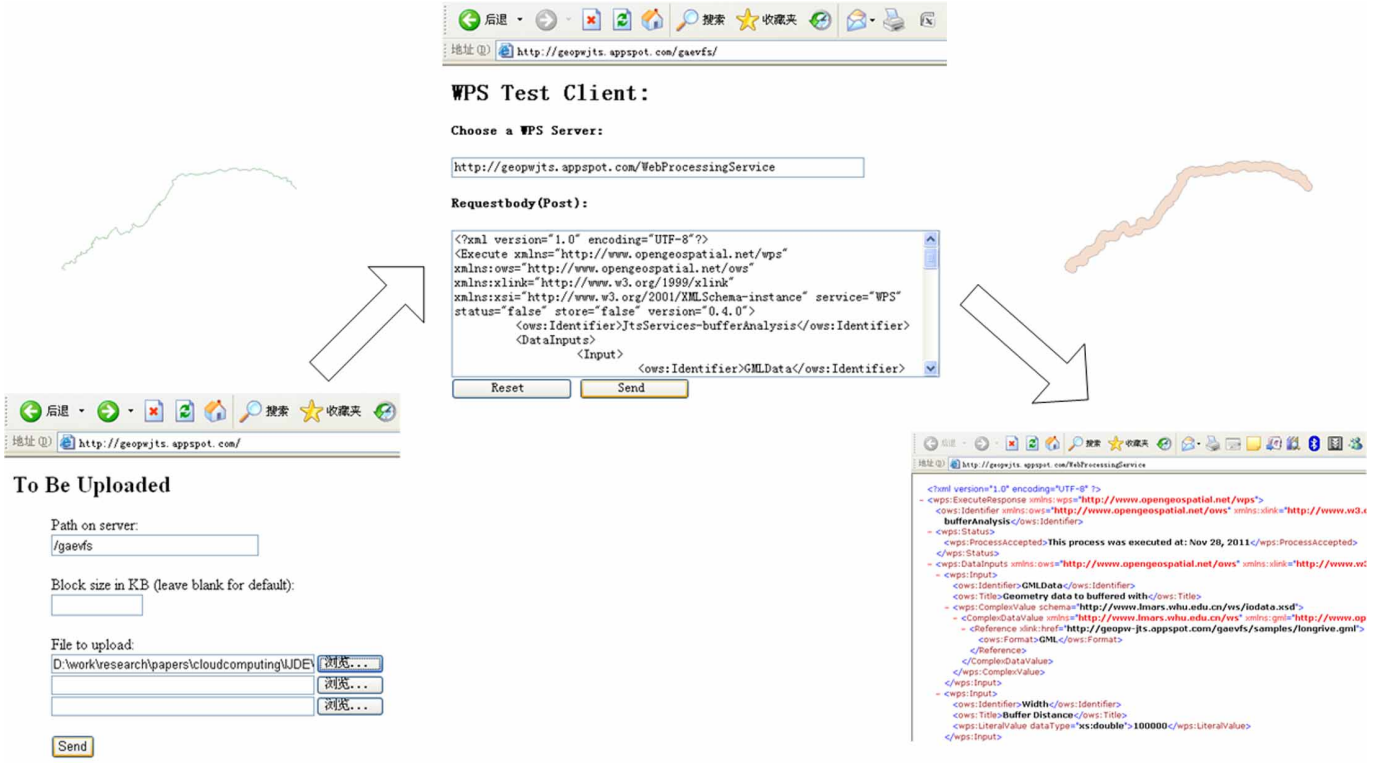

Figure 5. User interface for geoprocessing Cloud services in Google.

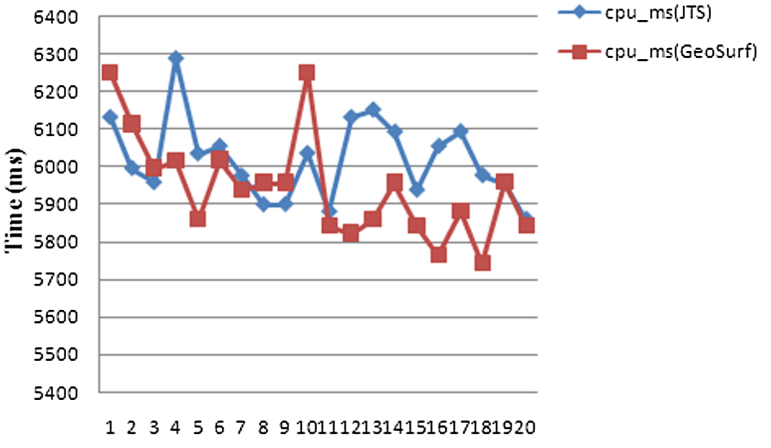

Figure 6. Performance tests of GeoSurf and Java Topology Suite services (20 executions).

consists of 223 vector points. After a user uploads the data, the page is directed to an execution page, where a HTML form is used to post the execution request.

6. Performance evaluation

To evaluate the performance of the Cloud services, we have conducted two experiments. One is to execute buffer services from both GeoSurf and JTS 20 times using the same input data and the other one is to invoke these services using input data with increased size. The resource usage for each execution can be collected from the monitoring dashboard of GAE.

Figure 6 shows the performance tests of GeoSurf and JTS services in 20 executions. The following indexes are used when monitoring the executions:

cpu_ms: It reports the usage of CPU to fulfill the request by using the adjusted time in milliseconds as a reference measurement. It includes the time spent by the Google API usage.

***cpm_usd*:** An estimate of the cost (in USD) of 1000 similar requests.

The values of these indexes are monitored in the dashboard of GAE and can be recorded for each execution. Since the input data for both GeoSurf and JTS services are same, the usage of storage resources for both services in the dashboard is same and therefore is not compared here.

In Figure 6, the execution time is fluctuating for each execution of the same service using the same input data. This is due to the changing status of the network and hosting servers. It can be observed that the ***cpu_ms*** of JTS service is usually higher than GeoSurf. This means that the performance of buffer analysis in GeoSurf is better than that in JTS. The higher the ***cpu_ms***, the more users will pay. Figure 7 shows that the value of ***cpm_usd*** in using the JTS service is higher than that in GeoSurf.

To analyze the effect of data size on the performance of Cloud services, the input vector data are divided into 10 groups, where most of them include 22 vector points,

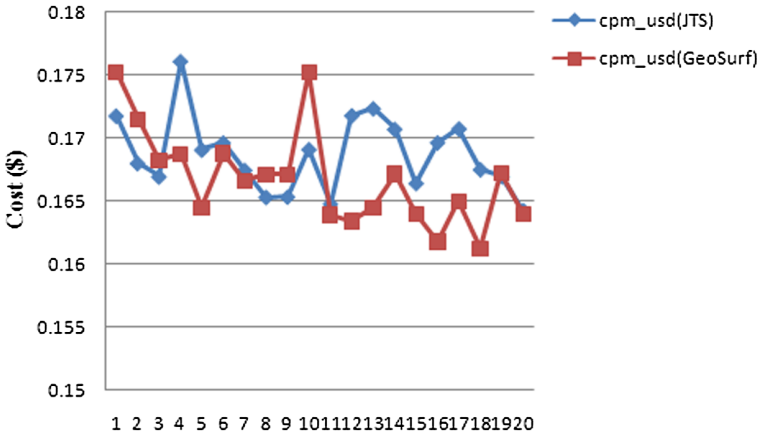

Figure 7. Costs of GeoSurf and Java Topology Suite services (20 executions).

with the final group including 25 vector points. The input data are increased from 1 to 10 groups. To decrease the network effect on the index values, 10 executions are conducted at each data size, and the average value is recorded. A new index – ***api_cpu_ms*** – is added, which records the CPU time spent by the Google API usage.

Since both GeoSurf and JTS services use the same GAE API, the values of ***api_cpu_ms*** are the same. As shown in Figure 8, the performance of the GeoSurf service is still better than JTS as the size of data increases, since the ***cpu_ms*** for the JTS service is higher than that of the GeoSurf service. In addition, when the size of data increases from six groups (the actual size is 6224 bytes) to seven groups (the actual size is 7192 bytes), the ***api_cpu_ms*** increases from 4037 to 4357 ms, while it is stable before and after. It means that the CPU time spent for processing is stable in a certain range, yet can be scaled on demand. Figure 9 demonstrates that the cost of the JTS service is still higher than the GeoSurf service as the size of data increases. The cost increase sharply when the data size is changed from six to eight. This is due to the increase of ***api_cpu_ms***. The experimental results show that Cloud Computing can provide on-demand allocation of computing resources to support geoprocessing

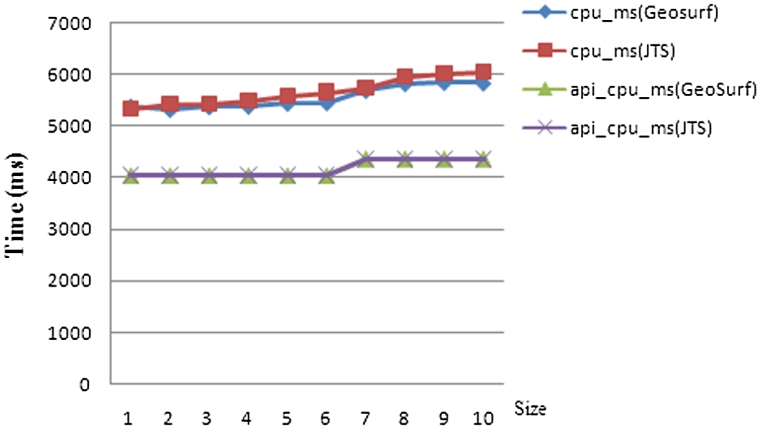

Figure 8. Performance tests of GeoSurf and Java Topology Suite services (different sizes).

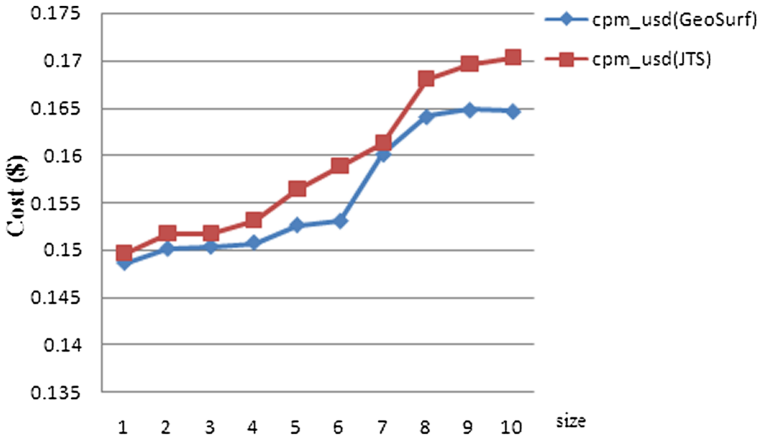

Figure 9. Costs of GeoSurf and Java Topology Suite services (different sizes).

services. The measurability of the Cloud Computing also allows the economic selection of geoprocessing services.

Another experiment is to test the performance of JTS in both GAE and Azure. The JTS is deployed into Azure, and its performance is compared with the JTS in GAE. Figure 10 shows the performance tests of buffer services in GAE and Azure in 20 executions using the same data. Azure has geolocation-specific support, which for example allows selection of geographical regions of Microsoft data centers for hosting JTS services. Tests on JTS services in Azure in East United States and Azure in East Asia are performed and shown in Figure 10. Execution times are collected by services themselves instead of clients and recorded in response messages of WPS. Thus communication costs between services and clients are not included in execution times.

Figure 10 shows that the performance of JTS services in either GAE or Azure is stable. However, performance difference exists among GAE, Azure in East United States, and Azure in East Asia. The JTS service in Azure (East Asia) has the best performance, followed by GAE and Azure (East United States). Figure 11 shows the

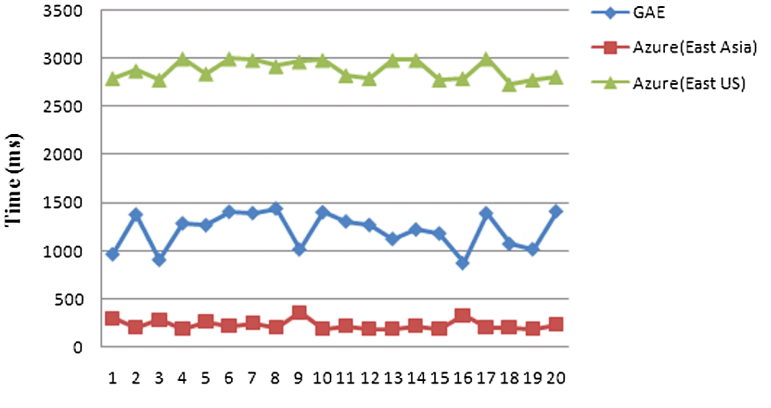

Figure 10. Performance tests of Java Topology Suite services in Google App Engine and Azure (20 executions).

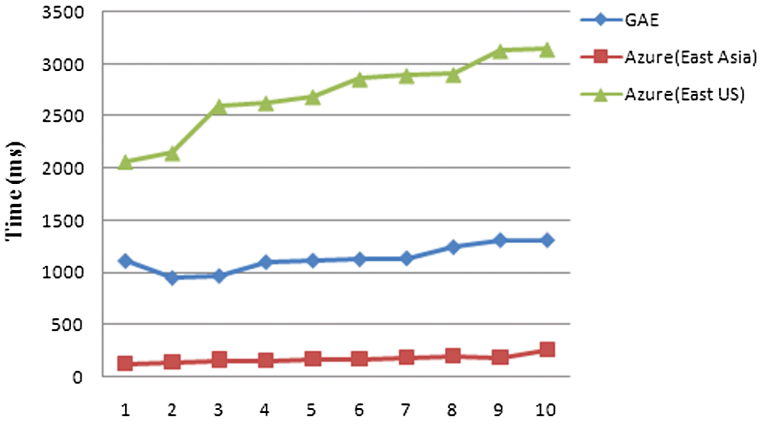

Figure 11. Performance tests of Java Topology Suite services in Google App Engine and Azure (different sizes).

performance tests of JTS services in GAE and Azure using input data with increased size. Ten executions are conducted at each data size, and the average value is recorded. The JTS service in Azure (East Asia) still has the best performance, followed by GAE and Azure (East United States). The execution time increases when the size of input data increases. The execution time of the JTS service in Azure (United States) increases more quickly than others. The better the performance, the slower the increase will be. Since the performance results could be affected by the working load of Cloud centers, it cannot be concluded which platform or data center has the best performance from this single case. However, the experiment shows that the selection of a platform or a specific data center provides potential for improving performance.

7. Discussion

The key feature of the Cloud Computing technology is that it hides the underlying complexity of using IT resources, while at the same time bring the scalable, reliable, sustainable, ready-to-use, on-demand, and cost-effective IT services to the general public. There are already various public Cloud platforms in the general information domain. They provide a firm technical base for on-demand provision of sufficient resources for both data-intensive and computing-intensive geoprocessing applications. For example, when the data size increased in the buffer analysis, the computational power can be allocated and scaled automatically (NIST 2009; Armbrust et al. 2009). Users no longer need to manage their own IT infrastructure. Applications and data run on platforms operated by third parties. Such provision has its business values by providing flexible costs. The resources usage is monitored and cost could be measurable and economic. As demonstrated in the buffer analysis, the cost for using the JTS service is higher than GeoSurf. When large volumes of data are involved in the geoprocessing or if the geoprocessing algorithm is complex and time-consuming, the usage cost is an important concern for practical geoprocessing applications.

The APIs of Cloud platforms are designed to hide as much as possible the complexity of managing resources in the infrastructure. The comparison between Microsoft Azure and GAE in developing and implementing geoprocessing Cloud services suggests that virtualization is the key issue for migrating legacy geoprocessing applications into the Cloud Computing environment. The migration requires the investigation of the Cloud component architecture, running environment, programming language, application framework, storage service, and platform APIs. The virtualization could happen at all these aspects. In the infrastructure level, the Amazon EC2 provides the virtualization at the system kernel level. For example, the Amazon EC2 provides VMs such as Linux Amazon Machine Images, which allow users to control nearly the entire software stack. Users can work in a shell environment. They will install all related software including the servlet container like the Tomcat when developing services (Huang et al. 2010). Although this gives users more control on the application environment, the cost will also increase for using dependent software systems such as Tomcat. In the platform level, Microsoft Azure and GAE try to provide the higher virtualization at the language (common language runtime) and Web application level, respectively (Armbrust et al. 2009). There is no consensus yet on the virtualization level. As a result, the comparison shows considerable work such as rewriting the file I/O functions has to be conducted on adapting conventional geoprocessing applications to Cloud environments. The implementation of geoprocessing Cloud services is locked into a specific vendor platform. The virtualization of file systems such as the virtual file system provided by the GaeVFS shows promise to ensure applications portable to other hosting vendors.

The vendor-specific APIs complicate the development of geoprocessing services in different Cloud Computing platform. Although the Cloud Computing interoperability is an open issue (Ortiz Jr 2011), the OGC service standards provide an interoperable solution in the geospatial domain for finding and accessing data and geoprocessing services. In fact, as shown in Table 1, most cases advocate the use of OGC standards for accessing geospatial Cloud services. The standard interfaces enable the interoperation of services, so that services developed by different vendors can be combined to fulfill users' requests. The interoperable services can be published, discovered, and executed. They can also be chained together for collaborative scientific problem solving. The standard-compliant geospatial Cloud services, legacy geoprocessing services, and infrastructure services can be connected together to support hybrid Cloud applications. This appears to be practical when new geospatial applications try to provide solutions on combining legacy systems and Cloud Computing resources, since the entire migration might require intensive work and it is only worth the effort when the advantages of Cloud Computing really count in applications.

8. Conclusions and future work

This paper provides a comparative analysis of the design and implementation of geoprocessing services in two Cloud Computing platforms – Microsoft Azure and GAE. A general architecture on how various components in Microsoft Azure platform and GAE can be used to address storage and computing demands of geoprocessing services are presented. The work compares the difference in the running environment, programming language, application framework, storage

service, and platform APIs for both Cloud platforms based on the practice in supporting geoprocessing functions. Geoprocessing functions from legacy GIS software including GeoSurf and JTS are migrated into Cloud platforms and compared in performance.

The prototype implementation illustrates how conventional geoprocessing functions can be migrated into the Cloud Computing environment. The comparison provides the reference on selectively utilizing Cloud Computing platforms or hybrid Cloud pattern. The Cloud services help geoprocessing applications enjoy benefits of the new IT infrastructure including the load balancing, data replication, and resource management. The practices show that virtualization is the key problem for portable geoprocessing Cloud services. The standard interfaces for Cloud-based geoprocessing services ensure wide applications of these services in geospatial communities. The performance tests demonstrate how the Cloud Computing can support the on-demand geoprocessing and economic choice of geoprocessing services.

Future work includes the practice on migration of more types of geoprocessing services into the Cloud platforms. Such practices can help further validate and support the analysis and provide more insight on the working mechanism of geoprocessing Cloud services. For example, over 100 geoprocessing services have been developed in GeoPW (Yue et al. 2010). Porting all these services into the Cloud environment can help provide a reliable and scalable environment for scientific problem solving. Furthermore, we will consider how different geoprocessing services in the Cloud environment can work together for better performance of geoprocessing service chaining.

Acknowledgments

We are grateful to the anonymous four reviewers for their valuable comments. This work was funded jointly by National Basic Research Program of China (2011CB707105), Project 41271397 and 41023001 supported by NSFC, and LIESMARS (Wuhan University) Special Research Funding.

References

Armbrust, M., A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, et al. 2009. *Above the Clouds: A Berkeley View of Cloud Computing*. Technical Report. Berkeley: EECS Department, University of California, 23 pp.

Baranski, B., T. Deelmann, and B. Schäffer. 2010. "Pay-Per-Use Revenue Models for Geoprocessing Services in the Cloud." Proceedings of the First International Workshop on Pervasive Web Mapping, Geoprocessing and Services (WebMGS 2010), Como, Italy, August 10–12, 6 pp.

Baranski, B., T. Foerster, B. Schäffer, and K. Lange. 2011. "Matching INSPIRE Quality of Service Requirements with Hybrid Clouds." *Transactions in GIS* 15 (s1): 125–142.

Baranski, B., B. Schaeffer, and R. Redweik. 2009. "Geoprocessing in the Clouds." In *GeoInformatics*, edited by E. van Rees, Vol. 12, 386–395. Emmeloord, Netherlands: CMedia B.V.

Blower, J. D. 2010. "GIS in the Cloud: Implementing a Web Map Service on Google App Engine." Proceedings of the First International Conference on Computing for Geospatial Research and Application, Washington, DC, June 21–23, 4 pp.

Cary, A., Y. Yesha, M. Adjouadi, and N. Rishe. 2010. "Leveraging Cloud Computing in Geodatabase Management." Proceedings of the 2010 IEEE Conference on Granular Computing GrC-2010, Silicon Valley, August 14–16, 73–78.

Chappell, D. 2009. *Introducing the Windows Azure Platform*. Chappell & Associates, 21. Whitepapers. Accessed 23 March, 2010. http://www.microsoft.com/windowsazure/

GaeVFS. 2011. *Google App Engine Virtual File System*. Accessed June 16, 2011. http://code.google.com/p/gaevfs/

Gong, J., P. Yue, and H. Zhou. 2010. "Geoprocessing in the Microsoft Cloud Computing Platform – Azure." Proceedings of the joint symposium of ISPRS Technical Commission IV & AutoCarto 2010, Orlando, FL, November 15–19, 2010, 6 pp.

Gonzalez, H., A. Halevy, C. S. Jensen, A. Langen, J. Madhavan, R. Shapley, and W. Shen. 2010. "Google Fusion Tables: Data Management, Integration and Collaboration in the Cloud." Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC'10, Indianapolis, IN, June 6–11, 2010, 175–180.

Google. 2011. *Google App Engine*. Accessed September 6, 2011. http://code.google.com/appengine/docs/whatisgoogleappengine.html

Huang, Q., C. Yang, D. Nebert, K. Liu, and H. Wu. 2010. "Cloud Computing for Geosciences: Deployment of GEOSS Clearinghouse on Amazon's EC2." Proceedings of the ACM SIGSPATIAL International Workshop on High Performance and Distributed Geographic Information Systems, HPDGIS'10, San Jose, CA, November 3–5, 2010, 35–38.

Ludwig, B., and S. Coetzee. 2010. "A Comparison of Platforms as a Service (PaaS): Clouds with a Detailed Reference to Security and Geoprocessing Services." Proceedings of the First International Workshop on Pervasive Web Mapping, Geoprocessing and Services (WebMGS 2010), Como, Italy, August 10–12, 6 pp.

Microsoft'. 2011. *Microsoft Windows Azure*. Accessed September 12, 2011. http://www.microsoft.com/windowsazure/Whitepapers/introducingwindowsazureplatform/

Marx, S. 2009. *Does Windows Azure Support Java?* Accessed July 10, 2010. http://blog.smarx.com/posts/does-windows-azure-support-java

NIST. 2009. *National Institute of Standards and Technology (NIST) Definition of Cloud Computing*. Accessed August 16, 2010. http://csrc.nist.gov/groups/SNS/cloud-computing/index.html

Ortiz, S., Jr 2011. "The Problem with Cloud-Computing Standardization." *Computer* 44 (7): s13–s16.

Roche, K., and J. Douglas. 2009. *Beginning Java Google App Engine*. New York: Apress, 264 pp.

Sanderson, D. 2009. *Programming Google App Engine* (p. 400). Sebastopol, CA: O'Reilly Media Inc.

Schäffer, B., B. Baranski, and T. Foerster. 2010. "Towards Spatial Data Infrastructures in the Clouds." In *Geospatial Thinking: Proceedings of the Thirteenth AGILE International Conference on Geographic Information Science*, edited by M. Painho, M. Santos, and H. Pundt, 399–418. Guimarães, Portugal, Berlin: Springer Verlag Lecture Notes in Geoinformation and Cartography.

Schut, P. 2007. *OpengGIS® Web processing service*. Version 1.0.0, OGC 05-007r7. Open Geospatial Consortium, Inc., 87 pp.

Vaquero, L. M., L. Rodero-Merino, J. Caceres, and M. Lindner. 2009. "A Break in the Clouds: Towards a Cloud Definition." *SIGCOMM Computer Communication Review* 39 (1): 50–55.

Vivid Solutions. 2011. *Java Topology Suite (JTS)*. British Columbia: Vivid Solutions, Inc. Accessed 16 June, 2011. http://www.vividsolutions.com/jts/JTSHome.htm

Wang, Y., S. Wang, and D. Zhou. 2009. "Retrieving and Indexing Spatial Data in the Cloud Computing Environment." In *Proceedings of the 1st International Conference on Cloud Computing (CloudCom 2009)*, edited by M. G. Jaatun, G. Zhao, and C. Rong, Vol. 5931, 322–331. Berlin: Springer Verlag, Lecture Notes in Computer Science (LNCS).

Yang, C., M. Goodchild, Q. Huang, D. Nebert, R. Raskin, Y. Xu, M. Bambacus, and D. Fay. 2011. "Spatial Cloud Computing: How Can Geospatial Sciences Use and Help Shape Cloud Computing." *International Journal of Digital Earth* 4 (4): 305–329.

Yang, C., R. Raskin, M. Goodchild, and M. Gahegan. 2010. "Geospatial Cyberinfrastructure: Past, Present and Future." *Computers, Environment and Urban Systems* 34 (4): 264–277.

Yue, P., J. Gong, L. Di, J. Yuan, L. Sun, Z. Sun, and Q. Wang. 2010. "GeoPW: Laying Blocks for the Geospatial Processing Web." *Transactions in GIS* 14 (6): 755–772.

Zhu, X., J. Gong, N. Chen, and L. Liu. 2003. "Web GIS – GeoSurf Based on J2EE Systematical Structure and its Realization Technology." *Geomatics World* 1 (6), 18–25 (in Chinese).