Implementing Grid Enabled Web Services for Enhanced Positioning using Low-cost GPS Devices

Yoshida, D.,[1] Realini, E.,[2] Fenoy, G.[3] and Raghavan, V.[4]
[1]Faculty of Liberal Arts, Tezukayama Gakuin University, 2-1823 Imakuma, Osakasayama City Osaka 589-8585, Japan, E-mail: yoshida@la.tezuka-gu.ac.jp
[2]Research Institute for Sustainable Humanosphere, Kyoto University, Gokasho, Uji City Kyoto 611-0011, Japan
[3]GeoLabs SARL, Future Building I 1280, Avenue des Platanes, Lattes 34970, France
[4]Graduate School for Creative Cities, Osaka City University, 3-3-138 Sugimoto, Sumiyoshi-ku Osaka 558-8585, Japan

Abstract
This paper presents web based grid services for enhanced GPS positioning. The adopted positioning engine is goGPS, an open source software package for enhancing the accuracy of low-cost devices. goGPS can be provided as a standardized Web Processing Service (WPS) in order to utilize high-quality location data in a variety of location-related applications. Further, the handling of a large number of users and data volumes is addressed by implementing the services on a grid computing platform for supporting large-scale service developments. Benchmarking tests have been carried out to demonstrate the scalability and interoperability of the system. This research investigates the advantages of utilizing cloud resources for positioning services in order to avoid the direct implementation of a physical grid network and to achieve scalable and fault-tolerant systems. Open source software and standards are extensively utilized in the system development. The outcomes can contribute to widening of market for location-based services (LBS) or location-related businesses by lowering accurate positioning costs and providing standardized and interoperable GPS processing services.

1. Introduction

With the increasing popularity of location-aware services, diffusion of web-enabled mobile computing devices (e.g. smartphones) and miniaturization of GPS hardware, the availability of accurate positioning information is attracting attention in many fields for both personal and business uses. Examples include, but are not limited to, community-based sensor networks and geographic data sharing, sport analysis, work safety, precise monitoring of moving machinery and automations in heavy industry and agriculture sectors. Location-based online collaborative platforms have become remarkably popular and were proven to be useful and widely adopted solutions for geospatial data collection, updating and sharing, especially in emergency situations. This was demonstrated by the crowd sourcing efforts in the aftermath of the Haiti Earthquake in January 2010 and of the Great East Japan Earthquake in March 2011. Volunteer users extensively utilized popular collaborative projects such as OpenStreetMap and other online collaboration services to collect and publish up-to-date geospatial data just after the disasters (Raghavan et al., 2010, and Yoshida and OSGeo Japan, 2011). However, personal mobile devices such as smartphones or Portable Navigation Devices (PND) are in some cases not accurate enough to meet specific requirements of geographic data collection. Built-in GPS receivers in such devices offer positioning with an accuracy ranging from 1 to 5 meters, depending on the survey conditions and hardware quality. In addition, in locations within dense urban areas, often enclosed by tall buildings or skyscrapers, the accuracy degrades further. The use of accurate location information has not been adequately exploited to its full potential yet, since costs for obtaining data with high positioning accuracy (e.g. at sub-meter level) are still prohibitive for various applications. There are various grades of GPS devices and the cost of different receivers depends on their hardware quality (and therefore their accuracy). Lowering the cost of highly accurate data could be a vital element to further widen location-based services or businesses. Additionally, the software embedded in most GPS devices and GPS internal processing units is generally closed (i.e. the source code is not made

available to users). Hence, the positioning algorithms cannot be modified or tuned to improve the accuracy for specific applications (Yoshida, 2011). An open source software library for enhancing GPS positioning, goGPS (Realini, 2009), was adopted in this research. The software has potential to achieve sub-meter accuracy, that usually ranges between 40 cm and 80 cm (Pertusini et al., 2010, Realini et al., 2012, and Yoshida, 2011). Such accuracy, obtained with low-cost devices, is close to that achievable by single frequency professional receivers, the cost of which is generally higher than $1,000 (Figure 1). In order to make goGPS processing available to a wide and diverse user-base and to foster the usage of sub-meter location data in various location-based services (LBS) or location-related businesses, Realini et al. (2012) and Yoshida (2011) developed a standardized web service for the goGPS positioning processes. The web service implementation was done by using the ZOO framework (Fenoy et al., 2012), that enables the implementation of Web Processing Services (WPS) which are compliant to the international standards proposed by the Open Geospatial Consortium (OGC). OGC-compliant WPS are supported by various GIS applications or systems. This enhances service usability and allows creating new services by making a so-called "chain" of different services. A WPS server has the capability to host many different sorts of processes as web services to be provided to the public. Therefore, many concurrent requests from clients could arise, so the server may need to run processes simultaneously or handle very large datasets, especially when dealing with GPS positioning. This could be a possible reason of server crashes or lowering of server response due to overload. This paper introduces a grid implementation of WPS compliant web services using Grid Engine, in order to make the service provider be able to handle a large number of accesses or heavy amount of processes (i.e. computational load) by using distributed grid execution machines. Implementing a grid system requires an appropriate number of computers for handling the processes and services requested by users. Not every grid user can afford to prepare such working environment to implement a grid. Therefore, this can be an issue in implementing a grid computing environment. The research describes utilizations of Amazon public cloud in order to resolve the issue and develop scalable and fault-tolerant systems and explains the advantages and disadvantages. This development is a fundamental research on High Performance Computing (HPC) and large-scale system deployments for providing stable and fast web services to a wide user base.

2. goGPS Positioning with Low-cost Devices

goGPS was adopted in this research to remove the systematic positioning errors (i.e. receiver and satellite clock biases, ephemeris errors, atmospheric effects) by applying relative positioning with respect to a GPS base station, and to reduce the low-cost instrumentation noise by Kalman filtering (Kalman, 1960, and Grewal and Andrews, 2001). goGPS applies extended Kalman filtering on double difference observations, specifically designed to address some of the issues of low-cost receivers. goGPS main features include a continuous estimation of float phase ambiguities (managing satellite configuration changes and cycle slips) and receiver-specific observation weighting functions based on satellite elevation and signal-to-noise ratio. goGPS needs raw observations in input either by reading RINEX files or in real-time by receiving raw data streams through a serial port or a TCP/IP connection (using NTRIP - Networked Transport of RTCM via Internet Protocol). Devices using u-blox LEA "T" modules that can provide raw data (i.e. timing, pseudo range, carrier phase and optional auxiliary data), are usually employed for tests. The most efficient way for receiving base station data is to exploit RTK services provided by a network of permanent GPS stations (Brovelli et al., 2008), in particular using Virtual Reference Stations (VRS). This research used the VRS service provided by JENOBA Co., Ltd., Japan. goGPS positioning engine is available both in MATLAB and Java programming languages. goGPS Java was specifically designed to be provided as a web service (Realini et al., 2012), therefore it was chosen for the current research.

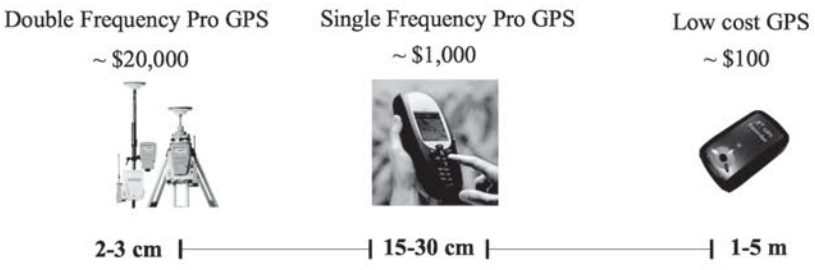

Figure 1: GPS accuracies and costs

3. Web Geo-processing Services for goGPS Positioning

Several online services exist for enhanced positioning, for example, Automatic Precise Positioning Service (APPS) for the Global Differential GPS (GDGPS) System by NASA and the Online Global GPS Processing Service (CSRS-PPP) by Natural Resources Canada. However, these services are not compliant to any international standard, thus they cannot be easily integrated or chained to other existing services and their results cannot be directly used in other applications or systems. Moreover, available online services are not specifically designed for enhancing the accuracy of low-cost receivers. The standardized web positioning services developed by Realini et al. (2012) and Yoshida (2011) communicate with any client software that follows OGC specifications. Figure 2 illustrates an overview of ZOO WPS server and client communication. The ZOO clients can be web browsers (Figure 3) or WPS-enabled GIS applications (e.g. uDig, Quantum GIS) and they must send the WPS request parameters through standard HTTP protocol. The ZOO WPS server receives the request, carries out the processing (e.g. executing goGPS) and finally returns the WPS response as an XML document. ZOO was chosen as a WPS framework because it natively supports various programming languages such as C, Fortran, PHP, Python, Perl and Java, while other currently available WPS engines support only a few specific languages. This multi-language support is convenient for developers and allows using existing code to create new web services. Open source GIS libraries or specific code (spatial based or not) can thus be ported server-side with small modifications and also make them available as open processing services for various applications over the network.

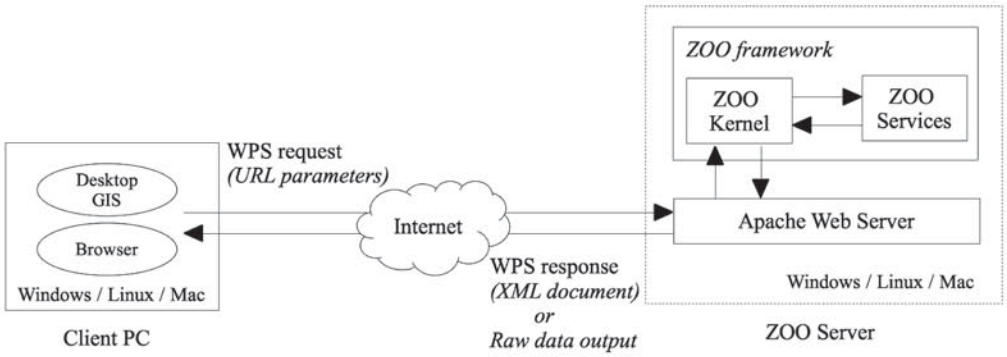

Figure 2: ZOO WPS server and client communication

Figure 3: goGPS web interface

4. GPS Processing Services on Grid Platform

4.1 Introduction

A ZOO server can host many different kinds of processes and provide them as web services through HTTP. Many concurrent requests from clients or handling very large datasets are expected issues when services for processing large volumes of data are made publicly available. Therefore, only scaling up server's performance at a single machine would not be enough for serving possible numerous and concurrent requests from users. Lanig et al. (2008) pointed out the limitation of classical GIS applications to manage the demand of processing and analysing massive raw terrain data. Their research integrated a grid and WPS for processing Digital Elevation Models (DEM) and explained the advantages of utilizing distributed computing power. Padberg and Kiehle (2009) and Coetzee (2011) also developed similar implementations for their Spatial Data Infrastructure (SDI) and showed some possible use cases of grid based SDIs such as flood simulation and emergency routing. Baranski (2009) explained how highly specialized geospatial applications based on large volumes of distributed data, such as live sensor data streams at different scales combined with high-resolution spatial data, which have to be analyzed in real-time, often require functionality for simultaneous execution of multiple processes. In such highly specialized large-scale geospatial applications, not every processing step can potentially be handled by a single processing entity (for example with the resources of a single computer). To improve the computational performance of processing large amounts of dynamic spatial data, grid computing provides appropriate tools. Baranski (2009) also describes WPS advantages for grid computing (WPS-G) and for the various applications using grid resources, concluding that OGC Web Services Phase 6 (OWS-6) and WPS should be able to benefit from their integration with distributed computing resources and technologies to enable applications to scale out. The implementations of a grid system using Sun Grid Engine for goGPS processes and the integration of ZOO with Grid Engine to enable grid requests from web services are described in this chapter.

4.2 Implementation of a Grid System

Grid Engine is one of the most popular open source batch-queuing systems (the latest version of Sun Grid Engine is now called Open Grid Scheduler). The Grid Engine master host efficiently allocates its jobs to the execution hosts according to their workload values. These values are calculated by usage of CPUs and memories, and Grid Engine uses them to make comparisons with a threshold value to assign jobs to appropriate execution hosts. This function was used in this work for the load balancing, especially when handling a large number of processing requests. On the other hand, Apache web server supports the load balancing function using a proxy module since the release of its version 2.2. This function can redirect web requests to cluster servers using specific algorithms to monitor the amount of incoming traffic or the number of requests in the web server. However, this can be handled only at the web request level. Apache server cannot monitor the clusters at the workload level as Grid Engine does. The grid system was implemented using virtual machines. Advantages of using virtual machines are efficient system management (system backup, copy and deployment, etc.) and effective use of physical computer resources. Moreno-Vozmediano et al. (2009) also implemented Grid Engine platform into their virtualized computer resources and described the advantages of virtualization. In this research, Oracle (Sun) VirtualBox was used as virtualization software, on 20 Apple iMac (Mac OSX 10.6) machines and Ubuntu Linux was installed on each one as the virtual OS. Grid Engine execution hosts were set up on these virtual machines. Apple Remote Desktop was used for monitoring and controlling the iMac machines remotely. These machines were used for the benchmark tests, as explained in Section 4.4. The system can work either in a global or private network, or both mixed as far as the name service resolves the host names. However, in case of systems that need fast responses from grid clients, for example the goGPS positioning application in this research, it is recommended to set up the grid system in private networks in order to secure and keep a fast network response. Therefore, this research developed the grid in a single segment of a private network and a Gigabit Ethernet based network was prepared. BIND DNS was set up in the grid master machine for managing host names. Since the grid system uses several distributed machines in a network, Network File System (NFS) was prepared to share input/output data and also execution programs.

4.3 Grid-Enabled Web Services for goGPS Positioning

Generally, grid systems are managed from the host machines (i.e. machines with master or execution hosts installed) by using command lines, for example "qsub" to submit a job to the grid system. The users need to know several commands to operate the grid. Therefore, this restricts the range of users who can manage to setup the system,

especially in the case of beginners not confident with UNIX operations. Distributed Resource Management Application API (DRMAA) is a well-documented open and standardized Application Programming Interface (API) available for controlling grid systems. DRMAA implementation in ZOO makes it possible to control Grid Engine operations as WPS services. In the research, DRMAA Java was adopted as the interface for Grid Engine, since Java was used for goGPS code and it is also supported by ZOO. The integration enables users to submit jobs through web sites or incorporate the grid service in applications on users' machines (Figure 4).

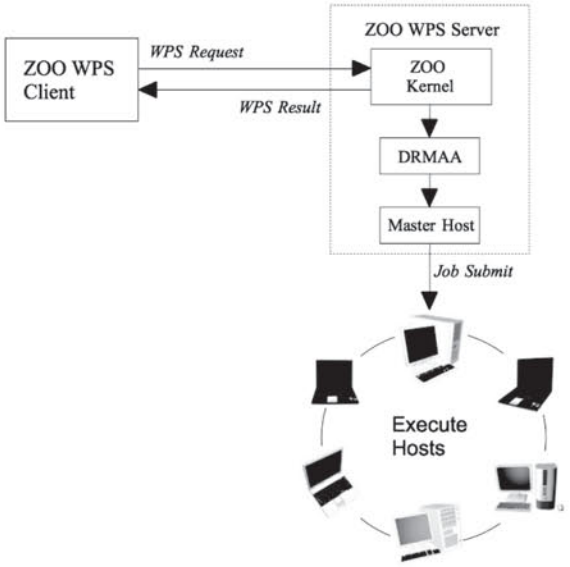

Figure 4: Overview of a grid enabled web system

4.4 Benchmark Tests for Grid, Non-Grid and Local Processing

Benchmark tests were carried out to compare the response performances of the grid, non-grid and local processing. The non-grid system is composed of a standard ZOO installation with Apache system on a single machine, while by local processing we mean that the test observes the execution time of goGPS at a local processing machine, without employing ZOO or Apache systems. The non-grid system was set up on the same machine as the grid master host. In addition, the test separately observed the performance difference between a "single-machine" grid, which is composed of one master and one execution hosts on the same machine, and a normal grid, which is composed of one master and 19 execution hosts on different machines. The specification of the grid system used for the tests is shown in Table 1. In the test, each goGPS positioning process (running Kalman filtering) used a large size dataset (8 hours of GPS observation data, the total size of the dataset is 32MB) as input data. A benchmark tool (httperf) was used for testing the non-grid system. This tool allows sending an HTTP request in various conditions, such as various methods (GET, POST, etc.) and the number of connections or requests. The tool returns detailed performance information such as test duration, connection rate and time, reply rate and time, etc. For testing local processing, a shell script was used to carry out multiple commands simultaneously. Figure 5 compares non-grid and grid systems, and Figure 6 shows the results of the two versions of grids and the local processing. Figure 5 and 6 indicate the response time for a certain numbers of requests. The requests were sent from the same machine in the case of the local processing machine. The benchmark tests clearly indicate the differences in response time for each system. When the server receives 3 or more requests for the non-grid system, the response time significantly increases due to the server's CPU specification (Figure 5), as the test server has a dual core CPU. On the other hand, the response times of the grid system are almost flat because the grid distributes its processes to physically different machines when the machine workloads exceed each threshold values.

Table 1: Specification of the grid system for the benchmark tests

Type	OS	Virtualization Software	Machine	Num. of CPU Cores	Memory	Num. of Machines
Master host	Ubuntu 10.4	VirtualBox	iMac (Intel Core2Duo 3.06GHz, 4GB)	2 Cores	2GB	1
Execute host	Ubuntu 10.10	VirtualBox	iMac (Intel Core2Duo 3.06GHz, 4GB)	2 Cores	2GB	19

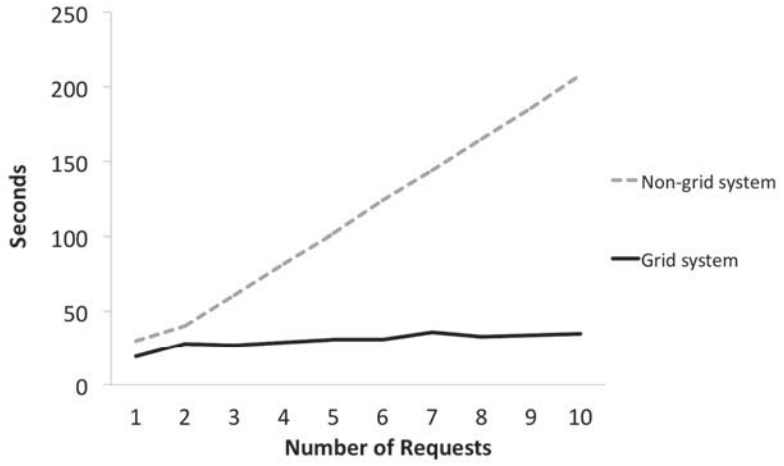

Figure 5: Response time against the number of requests (Non-grid and grid systems)

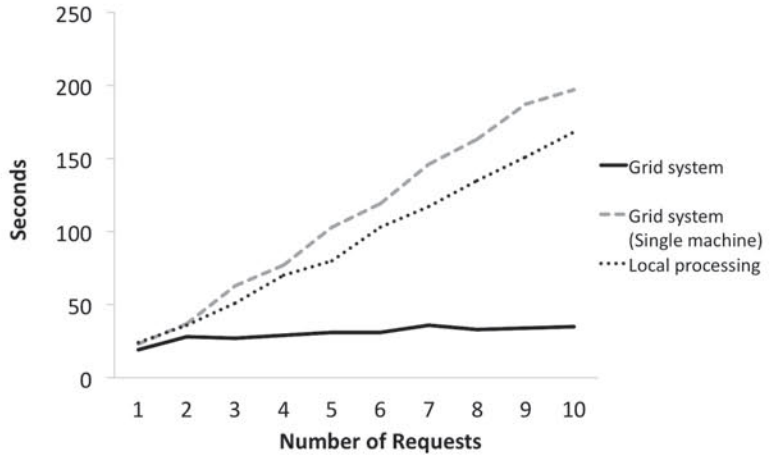

Figure 6: Response time against the number of requests (Grids and local processing)

Figure 7 indicates the response time for a certain numbers of requests on a number of execute host basis. This shows that each grid loses the efficiency when the number of requests exceeds the number of CPU cores. For instance, in the case of a grid consisting of 2 execute hosts, the processing speed is decreased at 5 requests. Therefore, this test system (i.e. consisting of 19 execute hosts) theoretically loses the efficiency when the number of requests exceeds 38 requests. Moreover, the benchmark tests indicated that the response times of the grid system on a single machine and as local processing are not very different from each other, although the grid system on a single machine shows an amount of overhead that tends to increase with the number of requests. One of the advantages of the grid system (on multiple machines) is that it can separate the processes from the ZOO engine and Apache server. Therefore, the processing speed can be kept quite similar to that of the local processing and it is also not affected by a large number of requests.

5. Implementation of Positioning Services using a Public Cloud System

5.1 Advantages and Disadvantages of Amazon Cloud

Amazon has started its Elastic Cloud Computing (EC2) service in 2006. Developed as Infrastructure as a Service (IaaS), EC2 provides advantages such as low cost of implementation, scalability, rapid deployment, very large and flexible storage infrastructure and computing resources, reliable power supplies and distributed backups. Amazon EC2 is especially useful when the number of accesses for the service increases, as it is easy to augment the performance by increasing the number of computing instances or switching to higher grade of instance. In addition, Amazon EC2 offers various APIs to manipulate its cloud resources.

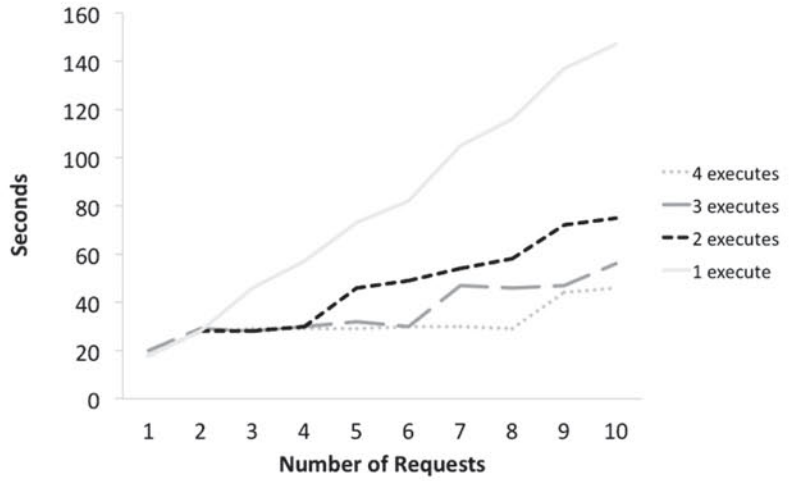

Figure 7: Response time against the number of requests by the number of execute hosts in grid

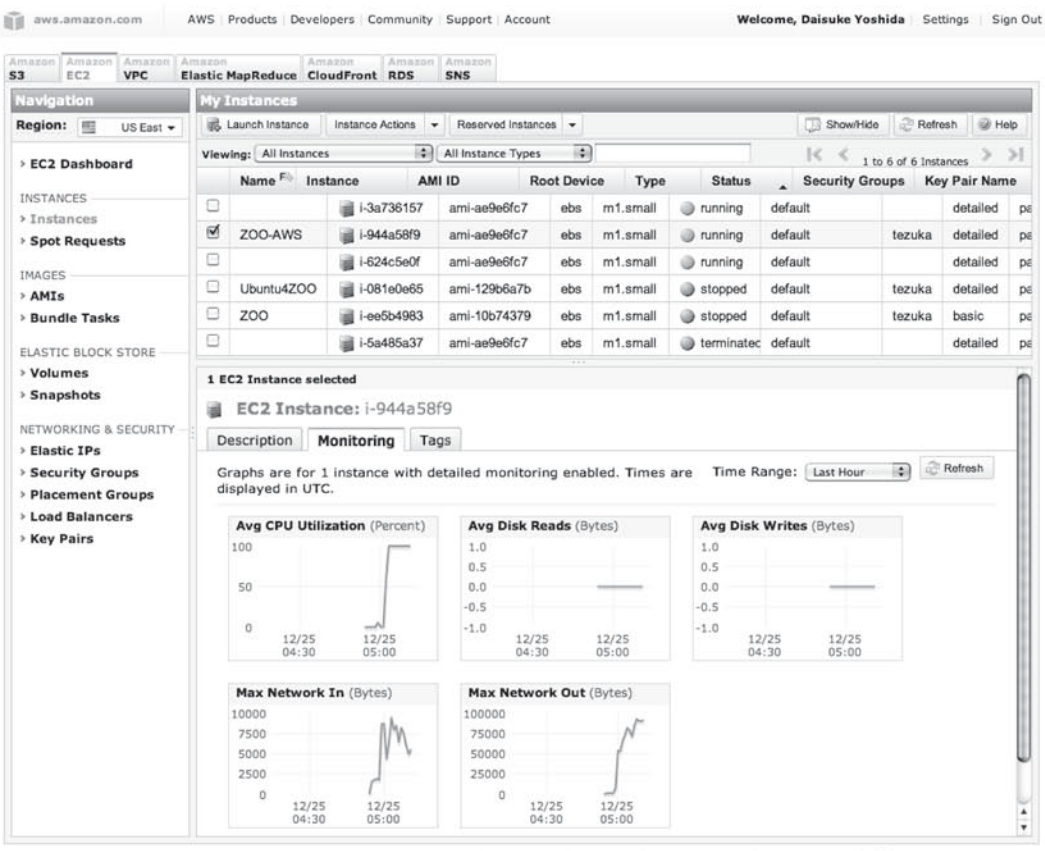

Figure 8: Web interface of Amazon Management Console

Users can apply these programmable APIs in their own applications (e.g. automatically adjusting the number of instances in accordance with the access hits of the web sites or schedules). To achieve scalable and fault-tolerant systems, this research investigated several scalable functions of Amazon EC2, especially Elastic Load Balancing and Auto Scaling to support a large number of requests. Auto Scaling allows flexibility in the system for scaling out and scaling down, which means that virtual machines (what Amazon refers to as "instance") can be started up or shut down automatically according to be distributed to the registered instances, as Apache proxy module does. Elastic Load Balancing

and Auto Scaling functions are generally used together with Amazon CloudWatch, which can monitor instance work performances. When workloads of instances exceed a threshold such that the average of CPU utilization remains over 90% for some time, a new instance is started up and also automatically registered in Elastic Load Balancing. In this research, goGPS and ZOO were implemented in the Amazon EC2 using the US-East data center (Virginia) and setting up the load balancing and Auto Scaling functions. Figure 8 shows Amazon Management Console interface and two instances, i-3a736157 and i-624c5e0f, which were started up because CPU usage was exceeded for a certain time, which is managed by CloudWatch and shown in the lower section of Figure 8. Amazon cloud also has some disadvantages. One is that the data centers are currently available only in the US (North California and Virginia), Ireland, Singapore and Japan. The connection speed for those countries that do not have the data center is relatively slow. Another disadvantage in Amazon load balancing functions is that they do not monitor workloads of the instances like grid master does: they only monitor the health (online or offline) of the registered instances and distribute one or more instances according to the incoming traffic. Moreover, in the Auto Scaling function some of the threshold values can be set, but the lowest intervals of monitoring periods are relatively high. For example, the breach time is at least 120 seconds, which indicates that a new instance can only be started after the CPU usage workload threshold is exceeded for 120 seconds. In addition, the monitoring period is at least 60 seconds and booting of new instances takes some time. Therefore, the functions cannot handle fast dynamic changes. Nonetheless, the functions are adequate for systems that have stable incoming traffic.

5.2 Cloud Based Grid Platform for Positioning Service

Implementing a grid system requires an appropriate number of computers for handling the processes and services requested from users. This can be an issue for implementing an effective grid computing network. On the other hand, cloud computing services such as Amazon EC2 can offer abundant computer resources at low cost and it can be utilized for developing scalable and fault-tolerant grid platforms that can handle a large number and heavy volume of requests simultaneously.

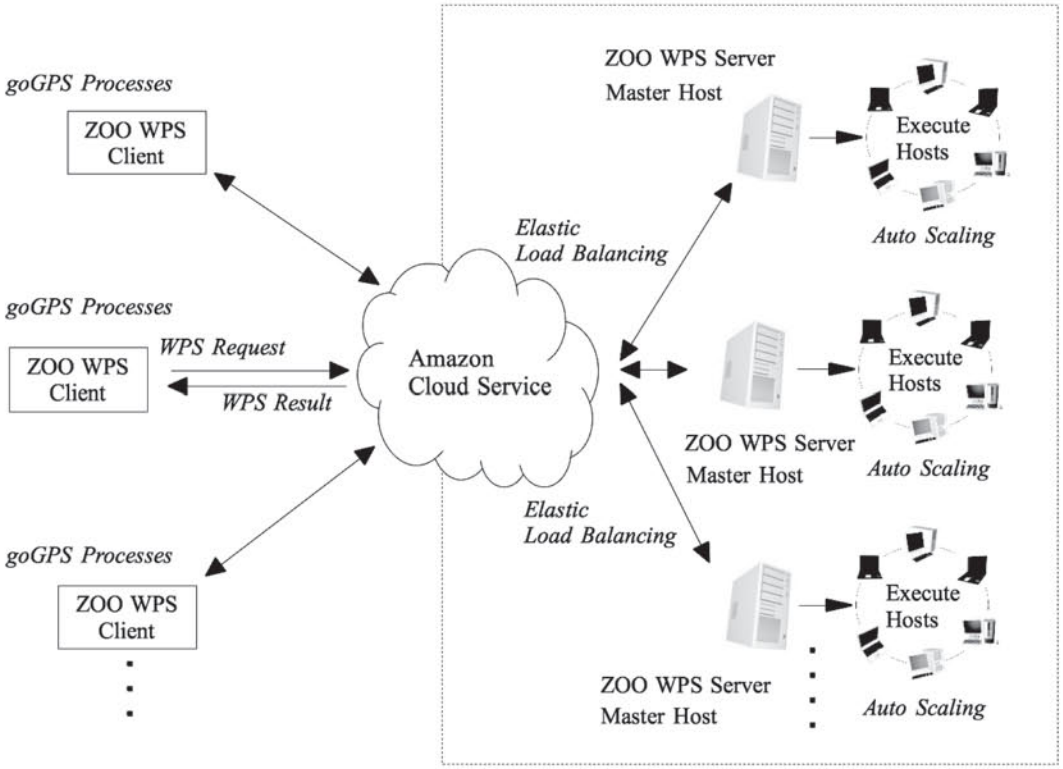

Figure 9: Overview of a redundant grid platform using Amazon cloud services

In addition, a grid is basically implemented in a fixed computer resource and this would be an issue when the master host in the grid is down, or when network failure or fire accident happen. Moreover, as shown in section 4.4, there is a limit after which the processing efficiency, according to the composition of physical computer specifications. Therefore, it would be impediment to handle the resources for keeping the efficiency in grid when the workload in the grid exceeds the limit. On the other hand, a cloud system has the capability to change the computer resources flexibly and automatically as scaling up the resources according the workloads. Further, the cloud-based implementation is able to achieve higher level of fault-tolerant system using the abundant and backup resources. Such capabilities fill in these disadvantages of the grid. Figure 9 illustrates an overview of a grid platform implemented in Amazon EC2 and its functions. The system takes advantage of both grid and cloud computing features. Multiple grid systems (a master and sufficient number of execution hosts) can be prepared in different Availability Zones (i.e. physically independent data centers not affected by problems occurring in other zones) in order to host servers that are located at physically different locations in the same Region (i.e. Regions consist of several Availability Zones and they are located in several countries). This can prevent system downtime from network or power failures. Elastic Load Balancing function can be set up on top of ZOO servers to distribute the requests. Auto Scaling function can be set up at the execution hosts in order to adjust the processing workloads. The function can also be useful for grid master hosts as it provides failsafe functionality like Shadow (i.e. backup) master of Grid Engine. Llorente et al. (2009) conducted a similar approach for implementing Globus Toolkit in Amazon EC2. They describe the use of virtualization, along with an efficient virtual machine management, to create a new virtualization layer that isolates the service workload from the resource management. The integration of the cloud within the virtualization layer can be used to support on-demand resource provisioning, providing elasticity in modern Internet-based services and applications, and allowing adapting dynamically the service capacity to variable user demands. Cluster and grid computing environments are two examples of services which can obtain a great benefit from these technologies.

6. Conclusion

This paper describes the development of open and standardized web grid services for enhanced positioning using goGPS software and investigates cloud advantages for achieving scalable and fault-tolerant systems. A grid functionality using Grid Engine was integrated with the ZOO WPS framework for controlling Grid Engine through web services. This research indicated that the functionality can be applied for large-scale or HPC system deployments and that cloud resources such as Amazon EC2 can also be utilized for developing large-scale systems. Further, open source software and standards have been extensively utilized in the system development of this research. Therefore, the developed services can be used in various applications in a cost-effective manner. As Sriram and Khajeh-Hosseini (2010) mentioned, cloud computing is the latest effort in delivering computing resources as a service. It represents a shift away from software as a product that is purchased, to Software as a Service (SaaS) that is delivered to consumers over the Internet from large-scale data centers. An increasing number of companies and computer users tend to shift their storage and computing power to non-local systems, mainly by using cloud services. Therefore, providing cloud services for various geospatial processing and data, as described in this work, could be useful for the public. Moreover, ZOO WPS can host different kind of processes in various programming languages and it can enhance the re-usability of legacy applications. For instance, Bozon et al. (2010) implemented Fortran programs for pesticide atmospheric dispersion modelling in ZOO WPS. Song et al. (2010) and Yoshida et al. (2010) developed various filters such as map matching and line simplification algorithms for processed GPS data, which were implemented in PHP and Python. These can be offered as WPS services and chained services for making better results after goGPS positioning. Some users consider using computing and storage in public clouds risky for data privacy, security reasons or cost matter. In addition, data centers of public clouds are not available in most countries. Therefore, good server responses are not always available in those countries. Instead of using public clouds, an option is to develop a private cloud environment in own closed and secure network. Moreno-Vozmediano et al. (2009) implemented OpenNebula to develop their private cloud environment and described on-demand provisioning of virtualized resources as a service. The cloud platform offers Amazon EC2 compatible API that allows controlling Amazon EC2 standardized clouds and integrating own private cloud resource and the Amazon cloud. Implementing a grid platform in a private cloud and investigating its fault-tolerant functions such as load balancing and auto scaling, etc. are among the future

works of this research. To increase the processing speed further in goGPS positioning, developing a local multi-GPU cluster using Compute Unified Device Architecture (CUDA) is also one of the future studies. This can be expected to increase a large amount of computing speed in highly n-dimensional calculations in the Kalman filtering in General Purpose computing on Graphics Processing Units (GPGPU) distributed processing. To enhance the accuracy of positioning, another future development is to support Japan's Quasi-Zenith Satellite System (QZSS) to improve the accuracy of the goGPS positioning service.

Acknowledgements

The authors acknowledge the positioning services provided by JENOBA (Japan) for their support. This research was supported by JSPS Grant-in-Aid for Scientific Research (Issue No. 2109737 and No. 24700105).

References

Baranski, B., 2009, OGC OWS-6 WPS Grid Processing Profile Engineering Report, *OGC Public Engineering Report.*<http://portal.opengeospatial.org/files/?artifact_id=34977>

Bozon, N., Fenoy, G., Raghavan, V. and Mohammadi, B., 2010, Drift-X WPS: Pesticide Atmospheric Dispersion Web GIS, *Proceedings of FOSS4G2010 Conference*, Barcelona, Spain. http://2010.foss4g.org/papers/3631.pdf>

Brovelli, M. A., Realini, E., Reguzzoni, M. and Visconti, M. G., 2008, Comparison of the Performance of Medium and Low Level GNSS Apparatus, with and without Reference Networks. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36, part 5/C55, 54-61.

Coetzee, S., 2011, Reference Model for a Data Grid Approach to Address Data in a Dynamic SDI, *Geoinformatica*, 16(1), 111-129, DOI:10.1007-/s10707-011-0129-4.

Fenoy, G., Bozon, N. and Raghavan, V., 2012, ZOO Project: The open WPS platform. *Applied Geomatics*, DOI:10.1007/s12518-011-0070-0.

Grewal, M. S. and Andrews, A. P., 2008, *Kalman filtering: Theory and Practice using Matlab*, 592. (Wiley-IEEE Press)

Kalman, R. E., 1960, A New Approach to Linear Filtering and Prediction Problems. *Transaction of the ASME, Journal of Basic Engineering*, 82(Series D), 35-45.

Lanig, S., Schilling, A., Stollberg, B. and Zipf, A., 2008, Towards Standards-Based Processing of Digital Elevation Models for Grid Computing through Web Processing Service (WPS), *Computational Science and Its Applications – ICCSA 2008, Lecture Notes in Computer Science*, 5073, 191-203, DOI:10.1007/978-3-540-69848-7_17.

Llorente, I. M., Moreno-Vozmediano, R. and Montero, R. S., 2009, Cloud Computing for on-Demand Grid Resource Provisioning, *High Speed and Large Scale Scientific Computing*, 18, 177-191, (IOS Press), DOI:10.3233/978-1-60750-073-5-177.

Moreno-Vozmediano, R., Montero, R. S. and Llorente, I. M., 2009, Elastic Management of Cluster-Based Services in the Cloud, *Proceedings of ACDC '09 Proceedings of the 1st Workshop on Automated Control for Data Enters and Clouds*, 19-24, DOI:10.1145/1555-271.155277.

Padberg, A. and Kiehle, C., 2009, Towards a Grid-Enabled SDI: Matching the Paradigms of OGC Web Services and Grid Computing, *The International Journal of Spatial Data Infrastructures Research*,4.<http://journals.vaggi.org/index.php/record/view/57938 >

Pertusini, L., Realini, E. and Reguzzoni, M., 2010, goGPS: Accurate Road Mapping using Low-cost GPS Receivers. *Proceedings of GIS-IDEAS 2010*, Hanoi, Vietnam, 5, 67-72.

Raghavan, V., Erle, S. and Realini, E., 2010, Open Software and Data for Emergency Response - How Crowdsourcing Changed Disaster Relief Forever. *Proceedings of Mizunami International Symposium on Earthquake Casualties and Health Consequences*, Mizunami, Gifu, Japan, 15-16, November 2010.

Realini, E., 2009, goGPS - Free and Constrained Relative Kinematic Positioning with Low Cost Receivers. *PhD. Dissertation*, Politecnico Di Milano. <http://geomatica.como.polimi.it/corsi/tesi/E_Realini.tar>

Realini, E., Yoshida, D., Reguzzoni, M. and Raghavan, V., 2012, Enhanced Satellite Positioning as a Web Service with goGPS Open Source Software, *Applied Geomatics*, DOI:10.1007/s12518-012-0081-5.

Song, X., Raghavan, V. and Yoshida, D., 2010, Matching of Vehicle GPS Traces with Road Networks for Precise Vehicle Tracking and Navigation in Urban Environment, *Current Science*, 98(12), 1592-1598.

Sriram, I. and Khajeh-Hosseini, A., 2010, Research Agenda in Cloud Technologies, *Proceedings of 1st ACM Symposium on Cloud Computing (SOCC 2010)*. <http://arxiv.org/pdf/1001.3259>

Yoshida, D., 2011, Development of Processing Services for GPS Accuracy Enhancement by Data Filtering and Kinematic Relative Positioning. *PhD. Dissertation*, Graduate School for Creative Cities, Osaka City University.

Yoshida, D. and OSGeo Japan, 2011, Response of OSGeo Japan with other Communities to the Great East Japan Earthquake, *Proceedings of FOSS4G2011 Conference*, Denver, the USA. <http://2011.foss4g.org/sessions/response-osgeo-japan-other-comunities-great-east-japan-earthquake-0>

Yoshida, D., Song, X. and Raghavan, V., 2010, Development of Track Log POI Management System using Free and Open Source Software, *Applied Geomatics*, 2(3), 123-135.