remote sensing

MDPI

Article

Auto-Scaling of Geo-Based Image Processing in an OpenStack Cloud Computing Environment

Sanggoo Kang and Kiwon Lee *

Department of Information Systems Engineering, Hansung University, Seoul 02876, Korea; lainer23@naver.com
* Correspondence: kilee@hansung.ac.kr; Tel.: +82-2-760-4254

Academic Editors: Chung-Ru Ho, Guoqing Zhou and Prasad S. Thenkabail
Received: 1 June 2016; Accepted: 15 August 2016; Published: 16 August 2016

Abstract: Cloud computing is a base platform for the distribution of large volumes of data and high-performance image processing on the Web. Despite wide applications in Web-based services and their many benefits, geo-spatial applications based on cloud computing technology are still developing. Auto-scaling realizes automatic scalability, i.e., the scale-out and scale-in processing of virtual servers in a cloud computing environment. This study investigates the applicability of auto-scaling to geo-based image processing algorithms by comparing the performance of a single virtual server and multiple auto-scaled virtual servers under identical experimental conditions. In this study, the cloud computing environment is built with OpenStack, and four algorithms from the Orfeo toolbox are used for practical geo-based image processing experiments. The auto-scaling results from all experimental performance tests demonstrate applicable significance with respect to cloud utilization concerning response time. Auto-scaling contributes to the development of web-based satellite image application services using cloud-based technologies.

Keywords: auto-scaling; cloud computing; OpenStack; satellite image processing

1. Introduction

Currently, cloud computing is considered an important paradigm in Information Technology (IT). The key features for cloud computing capabilities can be summarized as: on-demand self-service, broad network access, resource pooling, measured service and rapid elasticity. The cloud service model uses virtual servers for managing, scheduling, communicating, networking and auto-scaling as the common layer to fulfill a cloud computing scheme [1]. Among the proven benefits of cloud computing and applications, elasticity is the primary feature for both service providers and users. Elasticity, which refers to the extent to which resources provisioned by service providers change in relation to the changing user demand, allows service providers to maintain a high level of performance quality for application services.

Cloud-based application cases have been reviewed and we considered unresolved issues related to cloud platforms [2]. With regard to auto-scaling schemes, related concepts and taxonomy were surveyed [3]. A technical review of auto-scaling for elastic cloud-based applications was provided, and the Gartner group describes auto-scaling as an automatic expansion or contraction of system capacity, and indicated that such a capacity is a commonly desired feature in cloud infrastructure as a service and platform as a service offering [4]. In other words, auto-scaling refers to the significant capability of a cloud computing environment to utilize virtualized computing resources automatically. In this scheme, virtualized resources can be increased or decreased dynamically by adapting resource utilization to satisfy the given requirements. Auto-scaling contributes to cost control. The key features of auto-scaling are the ability to scale-out, i.e., automatic addition of resources during increased demand, and scale-in, i.e., automatic termination of unused resources when demand decreases. Scale-out and scale-in schemes are referred to as horizontal scaling. Unlike horizontal scaling, vertical

scaling increases computation resources in existing nodes. Auto-scaling at the service level is important because services run on a set of connected virtual machines. The optimal model-driven configuration of cloud auto-scaling infrastructure was studied [5]. It was implemented an open-source cloud environment with auto-scaling to access resources for a flexible period with varying requirements in bioinformatics and biomedical workflows [6], and was implemented an auto-scaling model in simulation experiments using the Amazon Elastic Compute Cloud (EC2) to reduce resource costs and test the quality of service in terms of response time and availability [7]. Three features of auto-scaling from a technical perspective were described [8]: covering variable demands, web application architecture, and distributing instances across availability zones. Cloud-based data-processing services in a geospatial web were pointed out as a very important issue in the area of remote sensing [9]. Cloud computing contributes to remote sensing applications for maximizing imagery resources, disseminating on-demand insight to end users globally, and improving efficiency while reducing cost and risk [10]. Advantages of remote sensing image processing within a cloud environment were discussed [11]. The auto-scaling performance test using a concurrent user number and an average response time of a spatial web portal based on a cloud-enabled framework implementation by Amazon EC2, Microsoft Azure, and NASA Nebula was carried out with respect to the dynamic workload [12]. Auto-scaling is a feature of OpenStack [13,14], where Heat is a template-based service that manages the lifecycle of OpenStack applications and defines the relationships among resources. The purpose of this study is to investigate the applicability of auto-scaling schemes to geo-based image processing services through performance tests with respect to some practical remote sensing algorithms, in an OpenStack cloud computing environment.

2. Auto-Scaling Concept

In a cloud computing environment, a practical methodology that elastically modifies and automatically adjusts the amount of used resources is called an auto-scaling scheme. However, the concept of auto-scaling has been defined from many perspectives.

Figure 1 shows a conceptual view of auto-scaling in a cloud computing environment. State 1 is a type of ready state implemented with two types of virtual servers, i.e., servers A and B, which are subject to an auto-scaling scheme. This represents the basic concept; however, cloud computing environments with two or more multiple servers are also possible. Increasing or decreasing computing resources for a particular application can be achieved by horizontal and vertical scaling. Horizontal scaling is a resource distribution scheme that adds or removes expansible server nodes to an application system. Vertical scaling is a scale-up and/or scale-down process that adds capacity to a single server node in an application system or removes capacity from that system. This study does not consider scale-up and scale-down schemes because mobile operating systems applied in this study are in a persistent state without intermediate interruptions for system upgrading. The client system in this application was designed with user interface components running on an iPad mini, a mobile tablet. Cases 1 and 2 are examples of scale-out and scale-in at the horizontal level depending on an application rule for auto-scaling. Auto-scaling represents one of the core services of a cloud computing environment. The number of virtual servers is monitored and controlled automatically according to the rules or conditions established by the cloud system manager. For example, if more than 50% of a virtual server's CPU resources are used for more than 10 min, additional servers are applied. Similarly, if less than 10% of the virtual servers are accessed for more than 30 min, the number of servers is reduced automatically.

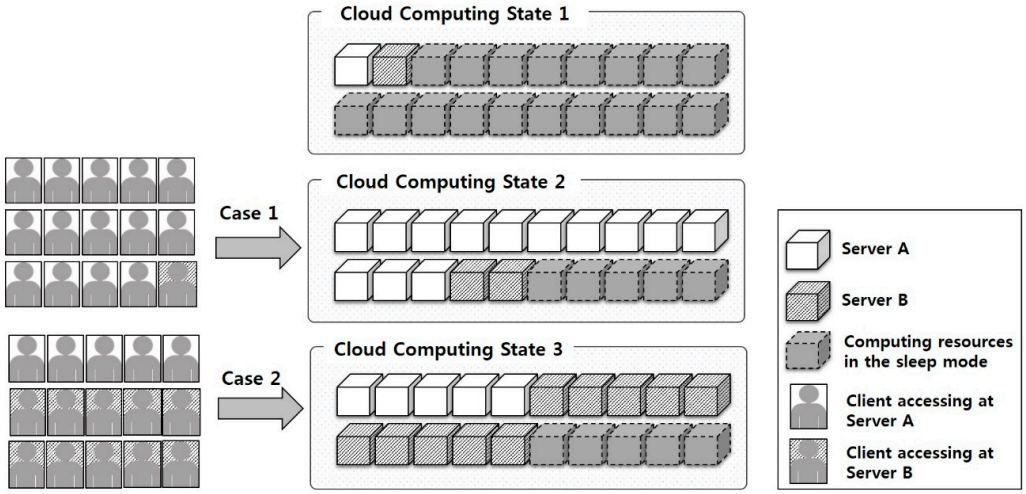

Figure 1. Basic concept of horizontal auto-scaling, edited from Reference [4].

3. Auto-Scaling Application Model for Geo-Based Image Processing Services

3.1. Example Case for Auto-Scaling Application for Geo-Based Image Processing Service

The test system implementation in this study was based on fully open sources and comprised the separate parts of the object storage component, the management instance server, and the processing instance server, which was designed in a satellite image processing system running in a cloud computing environment as shown in Figure 2. The object storage component is a computing environment based on Ceph [15] that stores geo-based satellite image sets in a scene unit for further analysis processing. In object storage processing using Ceph, which supports management functions of a large volume of data sets in a distributed computing environment, image sets formatted as geotiff were used. Stored objects, as data sets for image processing, in Ceph were imported into MariaDB in a cloud environment.

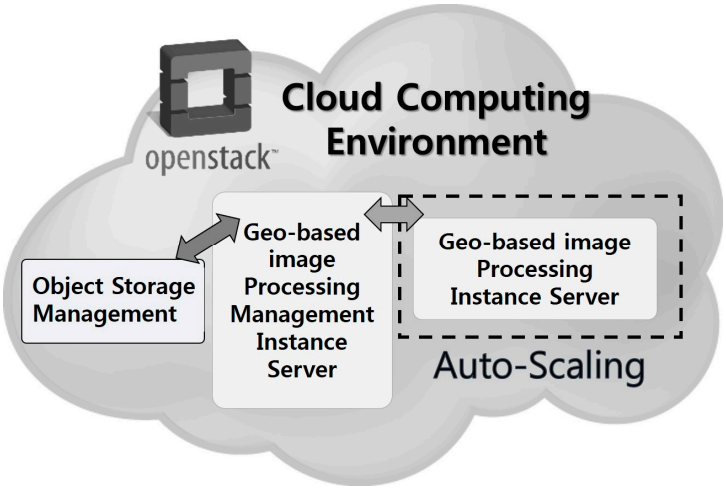

Figure 2. Auto-scaling components in satellite image processing system in a cloud computing environment.

The management instance server is a Web-based system that is accessible to users and provides user interfaces for image analysis processing functions. This is for general management processing regarding user requests. The processing instance server is for practical geo-based image analysis

processing with scale-out or scale-in horizontal auto-scaling. Virtual servers in these parts are built from fully open source stacks, and message processing is applied for communication between servers.

OpenStack consists of several separate and closely interconnected projects, such as Cinder, Neutron, Keystone, Glance, Ceilometer, Horizon, Nova, and Heat [16].

The cloud environment in this study was composed of three types of servers: system controller, network, and compute. To build a cloud computing environment for geo-based image processing services, a server with an Intel® Core™ i5-3470 CPU, 8 GB memory, and 500 GB of storage was used for the system controller and network server. The controller server was utilized with Keystone, Glance, Cinder, Nova, Ceilometer, Heat, Horizon, and Neutron services in OpenStack. The network server used Neutron. The controller server was utilized with Keystone, Glance, Cinder, Nova, Ceilometer, Heat, Horizon, and Neutron services in OpenStack. The network server used Neutron. These two virtual servers were located between the internal network accessing the compute servers and the external network to Internet for the client system of cloud services. The compute server, which comprised Nova, Neutron, and Ceilometer, was implemented on a server with an Intel® Core™ 2 Q9550 CPU, 4 GB memory, and 500 GB of storage. By exploiting the virtualization functions provided by OpenStack, 10 virtual servers were implemented to build the cloud computing environment. As for a virtual machine for two types of instance servers, the virtual machine specifications for the geo-based image processing management instance server were four virtual CPU(VCPU)s, 4 GB RAM, and 120 GB storage, and those of the geo-based image processing instance server were four VCPUs, 4 GB RAM, and 10 GB storage. The latency time to start virtual machines influences the cloud computing environment including the network status and the size of virtual image. The latency time applied in this study was established as conditions that the virtual machines were generated and image processing preparation was finalized within 2 min.

3.2. Experimental Condition for Auto-Scaling

Table 1 shows a summary of the specifications for the auto-scaling experiment for the open-source cloud computing environment. Table 2 shows a summary of the auto-scaling experiment for the cloud-based geo-based image processing. Table 2 also shows the number of users and processing requests and the types of processing algorithms. It is assumed that a single user makes 60 processing requests in 30 min. Thus, the total number of processing requests delivered to the virtual server in the cloud computing environment is 6000 by 100 users. The number of users can be considered the concurrent user access in the same period or the total number of users who want to use a certain algorithm. Here, a spatial coverage of 1 km $\times$ 1 km is an example for the experiment; however, this can be changed. The algorithms applied in the geo-based image processing instance server have four functions or filters provided by the open-source Orfeo Toolbox (OTB) [17].

The binary threshold image processing is aimed at transforming an input image into a binary image by substituting new pixel values according to the user definition of two thresholds and two intensity values. The binary thresholding is based on the Image Adaptor filter to perform pixel-wise computations on a geo-rectified image dataset. The cloud detection processing using a single image is based on a Gaussian factor and a spectral angle that is newly computed, to obtain a thresholding binary image. The cloud detection function in the OTB performs chained processing using the Spectral Angle Functor, the Cloud Estimator-Functor, and the Cloud Detection Filter. The maximum autocorrelation factor (MAF) is a kind of spatially extensive factor of the principal component analysis to maximize auto-correlation between neighboring pixels. For this factor, the Maximum Autocorrelation Factor Image Filter in the OTB computes a set of orthogonal linear transforms to maximize the spatial auto-correlation between a component and a unitary shifted version of that component. The normalized difference vegetation index (NDVI) is a well-known index to extract areas containing a dense vegetation canopy using multispectral image data. The OTB provides the NDVI function the Rand NIR Index Image Filter to compute the difference between the near-infrared (NIR)

channel and the red channel. On a mobile device in cloud computing, the system implementation model for satellite image processing based on OTB algorithms has been studied [18,19].

Table 1. System specifications for the auto-scaling experiment.

System Component	Specification and Version
Server operating system	Ubuntu server 64 bit/14.04 LTS
Cloud computing	OpenStack Juno/2014.2.3
Database management system	MariaDB/5.5.44
Storage	Ceph Hammer/0.94
Storage access library	librados(python)/0.94.2
Web server	Apache/2.4.7
Web framework	Django/1.6.1

Table 2. Summary of the auto-scaling experiment.

	Experimental Condition
Number of users	100
Number of processing requests	60
Ramp-up duration	30 min
Applied algorithms	Binary thresholding, NDVI, Cloud Detection, MAF
Spatial coverage	1 km × 1 km

Figure 3 shows tiled pieces of auto-scaling experiment data and the results obtained by OTB-based algorithms in a cloud computing environment. Figure 3a shows a tiled part of the data image set, and Figure 3b–d show the results of the binary thresholding, MAF, and NDVI, respectively. Figure 3e shows a tiled part of the data image set for a cloudy scene and Figure 3f shows the cloud detection result. Two high-resolution satellite image sets provided by the Korean Aerospace Research Institute were used as the processing datasets, i.e., Korea Multi-Purpose Satellite-2 (KOMPSAT-2; ground sample distance of 1 m panchromatic data and 4 m multispectral data) and KOMPSAT-3 (0.70 m panchromatic data and 2.8 m multispectral data). A total of 20 full-sized scenes per experiment were used for the test algorithm.

Table 3 summarizes the virtual server rules for the auto-scaling experiment and gives information about average CPU utilization, the scale-out and scale-in conditions, and the maximum or minimum number of instances. Apache JMeter was used for this experiment. The measurement results were obtained as the total time from the start point of the user's request for the management server to the task termination time stamp of the processing server. Many types of performance metrics are available in the OpenStack compute environment.

In telemetry measurement [20], the telemetry metric named cpu_util, which possesses the type of gauge and the unit of %, is used for average CPU utilization in this experiment. In the performance test experiment, response time is used as the metric, which comprises the total time for the first user request for a certain algorithm, parsing that request, pre-processing for image partitioning relative to a pre-set ortho-rectified full scene, delivering the message and its parsing, actual image processing using the algorithm on the geo-based image processing instance server, sending the results to the geo-based image processing management instance server, and finalizing the result message. These experimental conditions are an example to check the practical auto-scaling applicability under the same conditions and computing environment. Note that the scale-out and scale-in conditions with cpu_util and the processing time are changeable to unlimited cases.

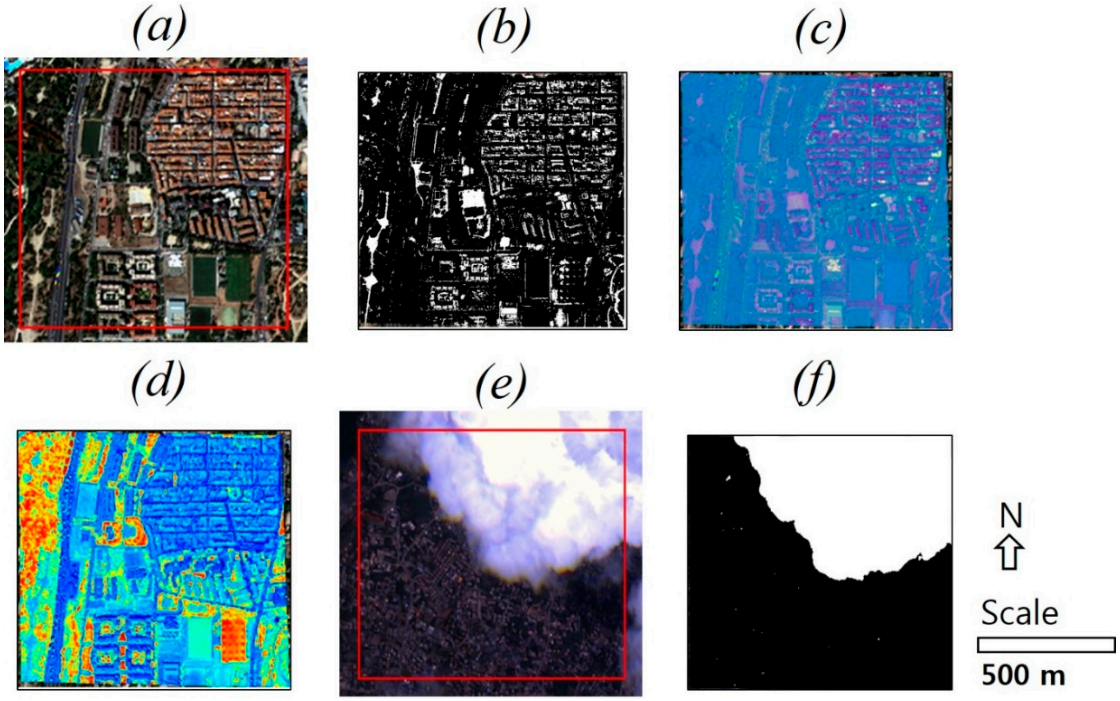

Figure 3. Example data for auto-scaling experiments and one case from OTB-based algorithms in a cloud computing environment: (**a**) tiled part of KOMPSAT 3 data image set (10 September 2014; 40°25°52°N, 3°42°20°W); (**b**–**d**) show the results of binary thresholding, MAF, and NDVI, respectively; (**e**) shows a tiled part of the KOMPSAT 3 image set for a cloudy scene (2 May 2015; 27°41°54°N, 85°19°27°E); and (**f**) shows the cloud detection result.

Table 3. Virtual server rules for auto-scaling experiment.

	Applied Condition
Measurement	Average CPU load check (name: cpu_util, type: gauge, unit: %)
Interval	60 s
Scale-out conditions	cpu_util > 40% and 180 s
Scale-in conditions	cpu_util < 5% and 3600 s
Maximum number of instances	5
Minimum number of instances	1

4. Experimental Results and Discussion

The experimental results for auto-scaling are as follows. Figure 4 shows the results for the performance measurement with the binary thresholding algorithm in a cloud computing environment. Figure 4a,b show a single virtual server without auto-scaling and auto-scaled virtual servers, respectively.

Auto-scaling in the cloud environment controls the optimized number of virtual servers to exploit within those that are in the available state. Auto-scaled virtual servers mean those that are elastically evoked as the compute servers among pre-set virtual servers. The number of virtual servers in the processing state varies depending on the type of user requests, the number of concurrent users or instances and other workloads depending on the auto-scaling scheme applied. It contributes to load balance for the stabilization of Web-based application services.

The response time with auto-scaling is faster than that of the single server in a cloud computing environment. Scale-out processing is applied automatically according to the experimental conditions and the rules in Tables 1 and 2. Figures 5 and 6 show the results obtained with the cloud detection algorithm and the MAF algorithm in the cloud computing environment, respectively. Figure 5a shows

an application on a single server and Figure 5b shows the auto-scaling result. Figure 6a shows a pattern similar to that shown in Figures 5a and 6b can be compared to Figure 5b.

Figure 7 shows the performance measurement results obtained with the NDVI algorithm in a cloud computing environment. Compared to the previous case using a binary thresholding algorithm, the total processing time is longer and the response time is slower. With practical auto-scaling processing for NDVI, more virtual server resources are required; thus, the auto-scaling rules and conditions can be adjusted by a cloud computing environment system manager to take advantage of high-performance features.

An elapsed time of over 30 min under the auto-scaling conditions is caused by different starting and finishing processing times with a virtual server of 100 users. Some are finished just in 30 min, while others may be delayed at the starting and intermediate processing.

Figure 8 summarizes the average response time and total processing time by the virtual servers rule applied in this study. As shown in Figure 8a, binary thresholding, cloud detection, and MAF demonstrate high performance with the auto-scaling scheme. However, NDVI shows a different average response time (24 s) despite using the same auto-scaling conditions and rules as the other algorithms. For the total processing time from initiating the first user request to responding to the final processed results (Figure 8b), the binary thresholding, cloud detection, and MAF algorithms terminate in approximately 30 min. This result is on the satisfactory level for 60 processing requests, within 30 min for 100 users. Due to relatively long average response times, the NDVI algorithm requires more than 49 min compared to the other three algorithms because the NDVI algorithm requires infrared and red bands, unlike the other algorithms, which use only a single image set. If two or more bands are necessary for a certain algorithm in an online environment, more processing time is required.

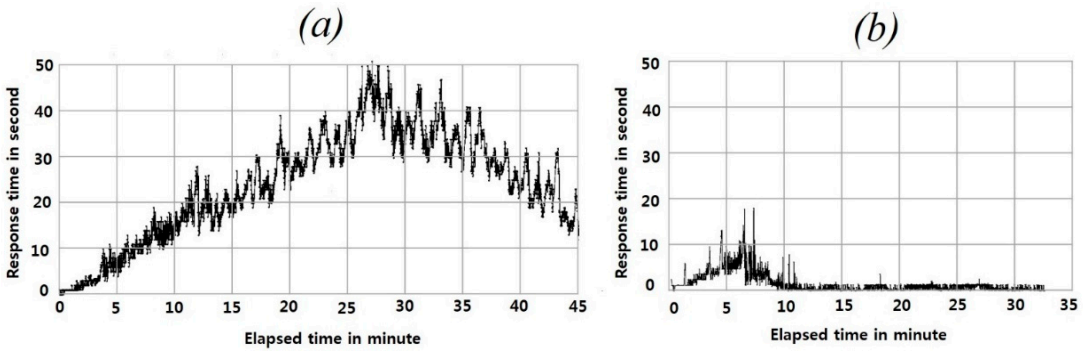

Figure 4. Performance measurement with the binary thresholding algorithm in a cloud computing environment: (**a**) single virtual server without auto-scaling; (**b**) auto-scaled virtual servers.

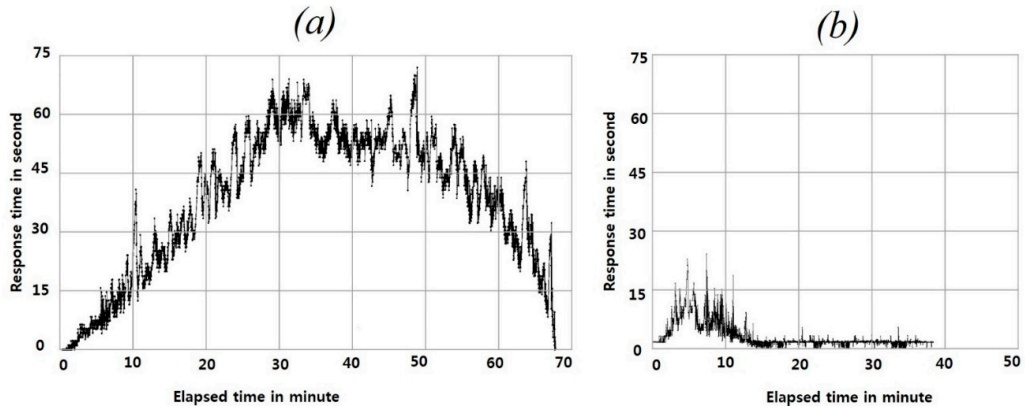

Figure 5. Performance measurement with the cloud detection algorithm in a cloud computing environment: (**a**) single virtual server without auto-scaling; (**b**) auto-scaled virtual servers.

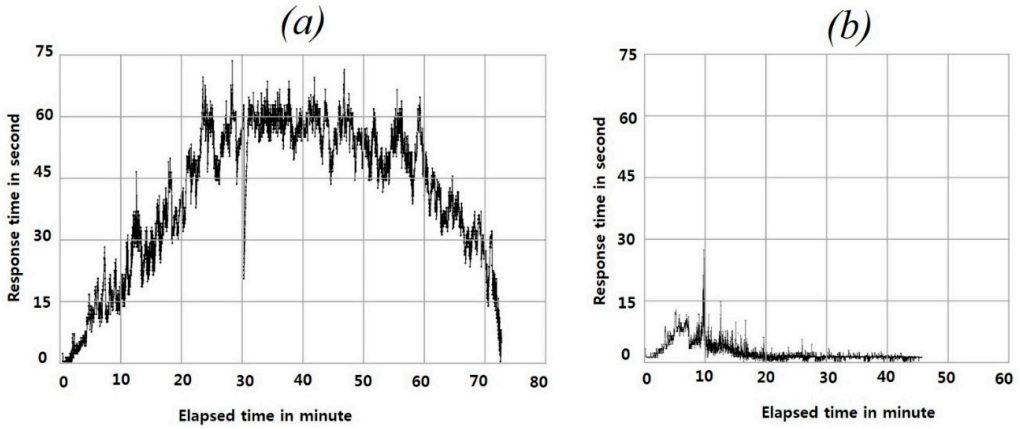

Figure 6. Performance measurement with the MAF algorithm in a cloud computing environment: (**a**) single virtual server without auto-scaling; (**b**) auto-scaled virtual servers.

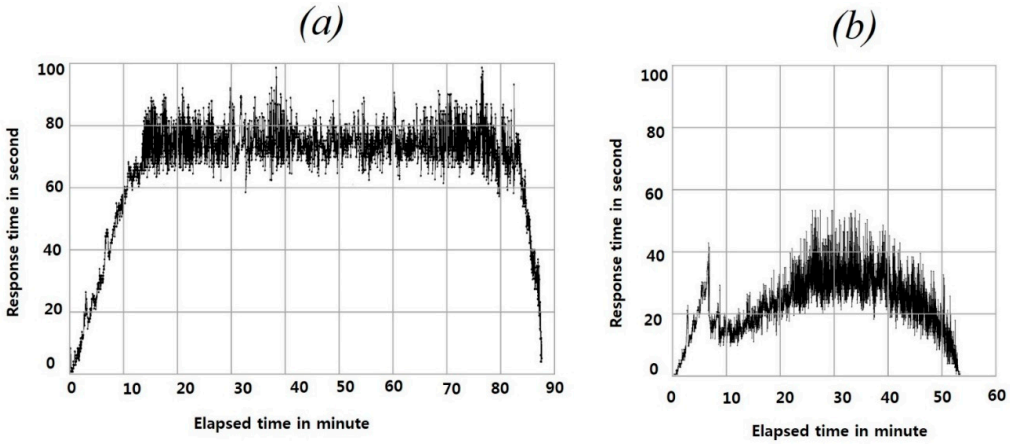

Figure 7. Performance measurement with the NDVI algorithm in a cloud computing environment: (**a**) single virtual server without auto-scaling; (**b**) auto-scaled virtual servers.

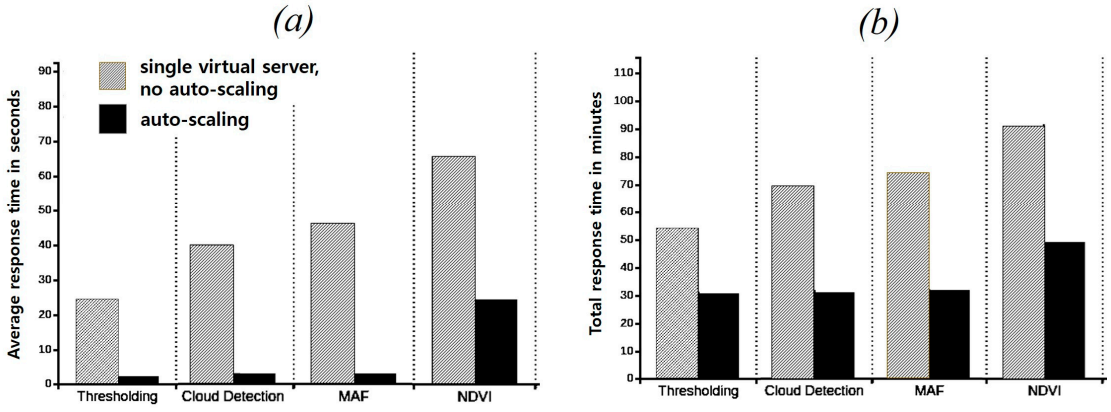

Figure 8. Experimental results for auto-scaling: (**a**) average response time; (**b**) total processing time.

5. Conclusions

Auto-scaling schemes realize automatic scalability in cloud computing environments. By comparing the performance of a single virtual server to that of auto-scaled virtual servers, this study focuses on the applicability of auto-scaling to geo-based image processing algorithms. In both cases, other conditions during data request and result generation are the same in this experiment. OpenStack,

an influential open-source cloud computing environment, was used in this study. In addition, four algorithms—binary thresholding, cloud detection, MAF, and NDVI in the OTB, which is an open-source remote sensing library—were used for satellite image processing experiments.

The auto-scaling results demonstrate prominent cloud utilization features relative to response time for all experiments. In the future, criteria and optimal conditions for applying auto-scaling to remote sensing and practical case studies are required. They include consideration points such as setting of the number of virtual servers, the size of the spatial coverage, and scale-out or scale-in conditions for separate processing algorithms or schemes. Conclusively, the proposed scheme and experimental results contribute to the development of web-based satellite image application services and their load balancing tasks using cloud-based technologies.

Acknowledgments: This research was supported by the National Land Space Information Research Program from the Ministry of Land, Infrastructure and Transport, Korea (No. 14NSIP-B080144-01).

Author Contributions: Kiwon Lee conceptualized the research objectives, drafted the manuscript and provided revisions. Sanggoo Kang performed data processing.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mell, P.; Grance, T. The NIST Definition of Cloud Computing, NIST Special Publication 800-145. Available online: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf (accessed on 24 July 2016).
2. Diaz, M.; Martin, C.; Rubio, B. State-of-the-art, Challenges, and Open Issues in the Integration of Internet of Things and Cloud Computing. *J. Netw. Comput. Appl.* **2016**, *67*, 99–117. [CrossRef]
3. Alipour, H.; Liu, Y.; Hamou-Ihadj, A. Analyzing Auto-Scaling Issues in Cloud Environments, 2014. Available online: https://users.encs.concordia.ca/~abdelw/sba/papers/CASCON14-Autoscaling.pdf (accessed on 20 November 2015).
4. Lorido-Botran, T.; Miguel-Alonso, J.J.; Lozano, J.A. A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments. *J. Grid Comput.* **2014**, *12*, 559–592. [CrossRef]
5. Dougherty, B.; White, J.; Schmidt, D.C. Model-driven Auto-scaling of Green Cloud Computing Infrastructure. *Future Gener. Comput. Syst.* **2012**, *28*, 371–378. [CrossRef]
6. Kreiger, M.T.; Torreno, O.; Trelles, O.; Kranzlmuller, D. Building an Open Source Cloud Environment with Auto-scaling Resources for Executing Bioinformatics and Biomedical Workflows. *Future Gener. Comput. Syst.* **2016**. [CrossRef]
7. Qu, C.; Calheiros, R.N.; Buyya, R. A Reliable and Cost-efficient Auto-Scaling System for Web Applications using Heterogeneous Spot Instances. *J. Netw. Comput. Appl.* **2016**, *65*, 167–180. [CrossRef]
8. Auto Scaling Developer Guide. Available online: http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/as-dg.pdf (accessed on 20 January 2016).
9. Evangelidis, K.; Ntouros, K.; Makridis, S.; Papatheodorou, C. Geospatial services in the Cloud. *Comput. Geosci.* **2014**, *63*, 116–122. [CrossRef]
10. Navulur, K. Demystifying Cloud Computing for Remote Sensing Applications. Available online: http://eijournal.com/wp-content/uploads/2013/06/cloudcomputing.pdf (accessed on 1 June 2016).
11. Wang, P.; Wang, J.; Chen, Y.; Ni, G. Rapid Processing of Remote Sensing Images based on Cloud Computing. *Future Gener. Comput. Syst.* **2013**, *29*, 1963–1968. [CrossRef]
12. Xia, J.; Yang, C.; Liu, K.; Gui, Z.; Li, Z.; Huang, Q.; Li, R. Adopting Cloud Computing to Optimize Spatial Web Portals for Better Performance to Support Digital Earth and Other Global Geospatial Initiatives. *Int. J. Digit. Earth* **2015**, *8*, 451–475. [CrossRef]
13. Siddiqui, A.A. *OpenStack Orchestration*; Packt Publication: Birmingham, UK; Mumbai, India, 2015.
14. Auto-Scaling in OpenStack. Available online: http://keithtenzer.com/2015/09/02/auto-scaling-instances-with-openstack/ (accessed on 17 October 2015).
15. Singh, K. *Learning Ceph*; Packt Publication: Birmingham, UK; Mumbai, India, 2015.
16. Radez, D. *OpenStack Essentials*; Packt Publication: Birmingham, UK; Mumbai, India, 2015.

17. The ORFEO Tool Box Software Guide, Updated for OTB-5.0. Available online: http://www.orfeo-toolbox.org (accessed on 7 June 2015).
18. Kang, S.; Lee, K. Mobile App Approach by Open Source Stack for Satellite Images Utilization. *Remote Sens. Lett.* **2013**, *4*, 648–656. [CrossRef]
19. Lee, K.; Kang, S. Mobile Cloud Service of Geo-based Image Processing Functions: A Test iPad Implementation. *Remote Sens. Lett.* **2013**, *4*, 910–919. [CrossRef]
20. Openstack. Available online: http://docs.openstack.org/admin-guide-cloud/telemetry-measurements.html (accessed on 15 November 2015).