

Article

A Performance Evaluation of a Geo-Spatial Image Processing Service Based on Open Source PaaS Cloud Computing Using Cloud Foundry on OpenStack

Kiwon Lee *  and Kwangseob Kim

Department of Electronics and Information Engineering, Hansung University, Seoul 02876, Korea; engintruder@hansung.ac.kr

* Correspondence: kilee@hansung.ac.kr; Tel.: +82-2-760-4254

Received: 13 June 2018; Accepted: 13 August 2018; Published: 13 August 2018



Abstract: Recently, web application services based on cloud computing technologies are being offered. In the web-based application field of geo-spatial data management or processing, data processing services are produced or operated using various information communication technologies. Platform-as-a-Service (PaaS) is a type of cloud computing service model that provides a platform that allows service providers to implement, execute, and manage applications without the complexity of establishing and maintaining the lower-level infrastructure components, typically related to application development and launching. There are advantages, in terms of cost-effectiveness and service development expansion, of applying non-proprietary PaaS cloud computing. Nevertheless, there have not been many studies on the use of PaaS technologies to build geo-spatial application services. This study was based on open source PaaS technologies used in a geo-spatial image processing service, and it aimed to evaluate the performance of that service in relation to the Web Processing Service (WPS) 2.0 specification, based on the Open Geospatial Consortium (OGC) after a test application deployment using the configured service supported by a cloud environment. Using these components, the performance of an edge extraction algorithm on the test system in three cases, of 300, 500, and 700 threads, was assessed through a comparison test with another test system, in the same three cases, using Infrastructure-as-a-Service (IaaS) without Load Balancer-as-a-Service (LBaaS). According to the experiment results, in all the test cases of WPS execution considered in this study, the PaaS-based geo-spatial service had a greater performance and lower error rates than the IaaS-based cloud without LBaaS.

Keywords: cloud computing; Cloud Foundry; data processing; OGC WPS; OpenStack; optical remote sensing; performance test

1. Introduction

Cloud computing, and its component technologies, are receiving a great deal of attention, and a lot of research has been carried out since the moment they were released. There are three main service models in cloud computing: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). The essential elements of cloud computing, defined by the National Institute of Standards and Technology (NIST), are on-demand self-service, broad-networking access, resource pooling, and rapid prototyping infrastructure, as a software model in Mell and Grance [1]. Virtualization software emulates different hardware processes in a virtual machine, and this could result in performance degradation compared to physical servers that are not virtualized. Nevertheless, server virtualization in the cloud environment has an advantage in terms of system management; it can physically consolidate several servers into one, reducing the footprint, or move virtual machines to other servers.

IaaS is a cloud service that provides an infrastructure environment that is highly available without the need to purchase computing resources or equipment. The hardware resources are virtualized and are being offered to potential customers. The user is not paying any hardware infrastructure and maintenance costs, but only an operational cost due to the use of virtualized resources, which are controlled by another party. The user can acquire virtualized resources on demand from the web by exploiting a certain service that is offered by a particular endpoint, but that service is not necessarily centralized. Rapid elasticity, as one of the main advantageous points of a cloud computing environment, enables the user to allocate infrastructure resources to meet the usual incoming traffic and automatically allocate additional infrastructure resources to meet demand when traffic is heavy beyond a specified threshold. On the other hand, if traffic is kept below the lower threshold after a certain period of time, unnecessary resources are cleaned up to keep infrastructure resources to a minimum.

Early on, cloud computing technologies focused on optimizing IaaS, for instance, using the Amazon Web Service (AWS). PaaS is also the core service model for cloud computing, providing services to automate the deployment and management of applications and enabling multiple middleware services. Middleware services are indeed offered in the environment and can be prepared for the user; this would mean that the user can focus more on developing the core functionality of his/her application. The potential for PaaS cloud computing is highly appreciated. The SaaS model virtualizes everything and can be raised to the level of the actual functionality. While, in the case of PaaS, the environment is virtualized, the user still needs to develop the core functionality. In fact, PaaS builds on IaaS, and SaaS might build on PaaS or IaaS. Software with SaaS is provided as a ready-to-use service, such as Adobe Creative Cloud and Microsoft Office 365. Especially, the open source PaaS development and scientific application based on PaaS is the nascent market [2,3]. By comparison of the three types of cloud service models with transportation [4], IaaS is like leasing a car that the client can choose and drive wherever they like, and PaaS is like taking a taxi; the client does not need to drive it himself but rather need only tell the driver the destination. SaaS is like going by bus, which has some routes, and the ride is shared with other clients.

Cloud computing technologies are also being utilized for geo-spatial image services with data distribution and processing purposes in distributed environments containing data servers and processing programs. Yang et al. [5] analyzed cloud technologies, currently in commercial and open source services, from the IaaS perspective, and they reviewed them for practical application to obtain geo-spatial information. Kang [6] performed a performance evaluation regarding the auto-scaling of geo-based image processing in an OpenStack environment to reveal advantageous aspects of the IaaS cloud, such as cost savings, scalability, and flexibility, and high availability [7]. Recent applications of service systems have to increasingly cope with both larger, diverse issues and workload volumes, but they are requested to operate at minimum costs and maintain desired performance guarantees in Costache et al. [8]. Wu et al. [9] evaluated the performance of finite element analysis on several public clouds by solving a large-scale engineering problem using performance metrics, including elapsed time, speed-up, scalability, and stability. Duan [10] emphasized that the performance of the cloud-based service has a significant impact on future information infrastructure. IaaS-based geo-spatial data processing services with actual image processing functions, such as change detection, segmentation, and normalized difference vegetation index (NDVI) under an OpenStack environment, were presented in Lee et al. [11]. According to the meta-analysis of Senyo et al. [12], the majority of cloud computing researchers carried out an experiment or simulation, as methods of inquiry, rather than theoretical frameworks and models. Yue et al. [13] provided the evaluation results that geoprocessing cloud services could be elastic and cost-effective by a comparative analysis of those of two cloud computing platforms, such as Microsoft Windows Azure and the Google App Engine. Noor [14] suggested future directions for mobile cloud computing. If web services or software applications, deployed or developed via the use of PaaS services, can be provided, the system experts can easily deploy and develop such components. However, cloud application services using PaaS in geospatial data

application fields are just in the early stage of development. There is a need for investigating the benefits of geospatial application deployment or development via PaaS, as well as the closure of some gaps or issues related to these two tasks, either in general or with respect to the use of IaaS services. In this study, the PaaS-based service linked with OGC WPS 2.0 was built and tested on the basis of the performance results of an edge extraction algorithm, in three cases of 300, 500, and 700 threads, against IaaS-based service with the same function.

The remainder of this article is structured as follows. Section 2 briefly reviews the IaaS- and PaaS-based service models. Section 3 describes how the open-source geospatial image service has been realized via the PaaS-based technology. Section 4 presents the results of performance tests, in terms of response time and error rate, on both IaaS and PaaS cloud computing environments with OGC WPS 2.0. Section 5 discusses the results and Section 6 concludes the paper.

2. IaaS- and PaaS-Based Application Services

To deliver services at any level or service model in the cloud, the first step is to select the optimal solutions for a cloud computing service if a client wants to use either commercial ones or free open source ones. In general, the cloud services provided by commercial vendors may be more convenient than open source solutions because commercial ones make it easy to access, maintain, and implement a cloud-based service as well as security checking. Rodero-Merino et al. [15] surveyed the security of multitenant software platforms to check PaaS security issues, such as the access control mechanism, block by synchronized static components, thread termination, and resource accounting of different layers. Kritikos et al. [16] studied a security-enhanced PaaS platform for multi-cloud applications and concluded that pragmatic results should be considered to select the technologies applied to build and manage IaaS and PaaS systems for performance demanding cases. There is the benefit of a direct deployment of cloud-based services, defined by the client, which focus on specific areas because they can implement specialized functions and features in cloud technologies. However, this requires a great deal of skill and experienced knowledge, as they are built directly from the hardware configuration for service deployment. Several open source solutions to building IaaS- and PaaS-based cloud computing service models are already available [17–19]. An application service is built through services offered by each service model. Kim and Lee [20] studied factors, from hardware components to middleware, in relation to designing an application service for geo-based data processing. Kozhircbayev and Sinnott [21] presented a comparison relating to the memory, network bandwidth, and storage overhead performances of container-based technologies in the platforms, such as native, Docker, and Flockport for the PaaS cloud. According to their experiments, the input–output latency is exacerbated by these overheads. The CPU cycles, required for utility, can cause performance degradation. Containers are one of the building blocks for PaaS, which allow clients to create applications on any virtual or physical infrastructure, combining all its dependent elements, such as codes, executables, and system libraries.

There are nine layers of technology for IaaS and PaaS service models, showing computing layers of hardware and software needed to implement IT services, which are illustrated in Figure 1. Layers, such as storage, server, networking, virtualization, runtime, middleware, application, and data, are all necessary to build web services in a cloud environment, and there are many ways to implement them, depending on the purpose or user of the service. IaaS usually provides hardware, such as servers, networks, and virtualization. This allows developers to easily configure the physical portion of hardware through virtual servers (VSs). Virtual servers logically divide one physical server and allocate resources, such as CPU and memory. Each virtual server can operate a separate OS or application. Developers need to install the required runtime components and middleware software, databases, and web servers inside one or more of the provided instances and upload the application processor, which represents the processing function and application view of user interfaces, as well as the required data for operation into a PaaS-based service. Since software services differ in different applications, functional analysis is necessary before providing any functionality in a cloud-based system. A stack of provided software components is the runtime environment in which the application developed by

the developer runs, and the middleware software is required when the application works. In PaaS environments, it is converted to packs and services. This enables cloud clients or operators to operate the application view as soon as they upload the application processor and required data to PaaS.

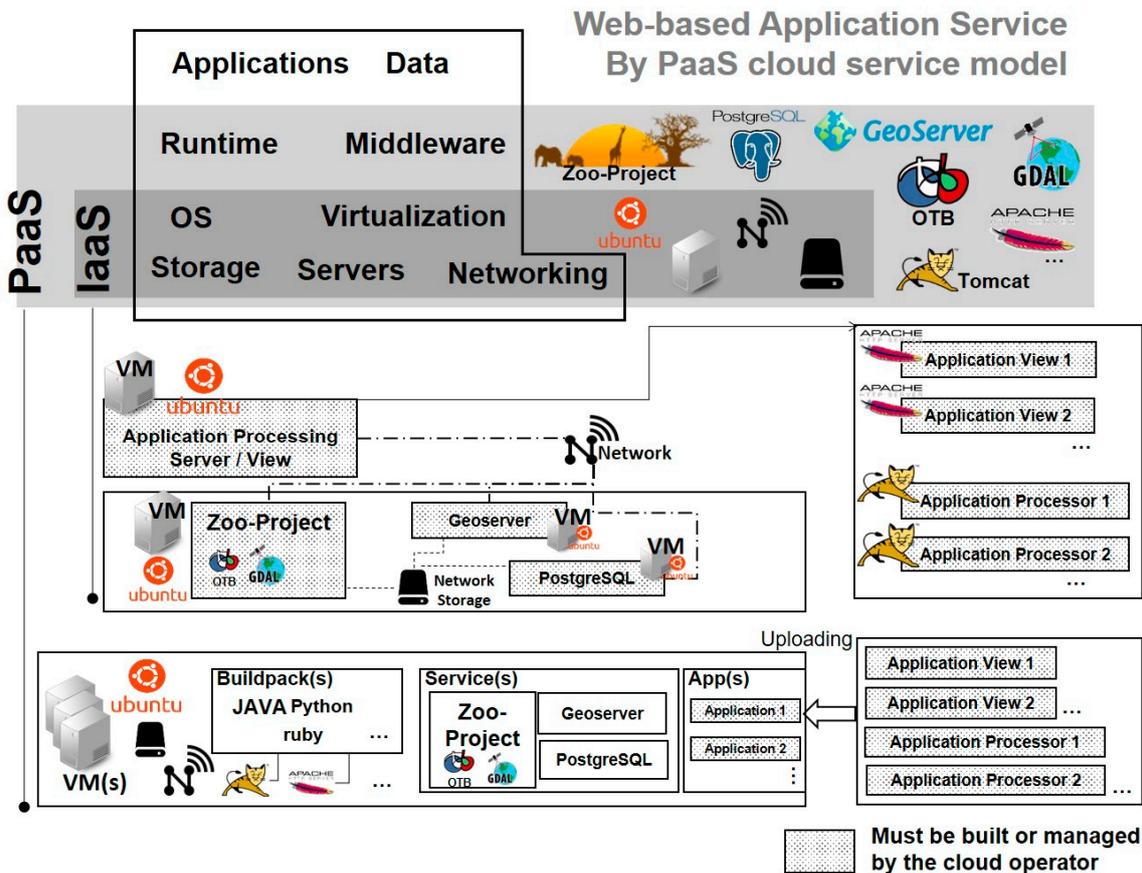


Figure 1. Basic architecture for cloud-based application services.

A basic architecture of actual open source solutions for geo-spatial information services in the cloud computing environments of IaaS and PaaS, respectively, is represented in Figure 1. This can be extended to SaaS services, which provide the geo-spatial image processing services directly to the Web, if additional features based on user requirements are added, and datasets for processing features are available. This schematic diagram depicts the main parts that the developers need to consider in providing cloud-based geospatial services. As can be seen, PaaS contains a software stack, as a middleware layer, composed of open source solutions for geo-spatial data processing. This study also considered the interoperable standards, provided by OGC WPS, of geo-spatial image processing services in the Web Processing Service (WPS) [22]. The WPS standard defines how to request the execution of a process, and how the process output should be handled. In particular, the Zoo-Project, which is a WPS implementation, programmed in C, Python, and JavaScript, provides a framework for creating and chaining WPS compliant web services, as well as a standard interface for OGC WPS 2.0, which consists of five standard interfaces, such as GetCapabilities, DescribeProcess, Execute, GetStatus, GetResult in Zoo-Project Team [23]. Yoon et al. [24] conducted a performance test of satellite image processing for OGC WPS 2.0, running on Zoo-Project in the OpenStack IaaS cloud, and showed test results for the other three requests, excluding GetStatus and GetResult, finding that the average response time increases rapidly as the number of users increases. Yoon and Lee [25] carried out a performance comparison test of a non-WPS case, synchronous process case of OGC WPS 1.0, and asynchronous process case of OGC WPS 2.0 with two kinds of image processing algorithms, such as

feature extraction and gradient magnitude computation, and presented the result that WPS 2.0 showed a high performance in the response time.

All middleware software or components, such as GeoServer, Zoo-Project, and the database management system, should be used, as instances, directly on the IaaS service. PostgreSQL [26], the open source database management software, serves the client by retrieving the processing status. The image database was not built in this study. Actual image processing was conducted by image files, and metadata of the image data, used in the experiment and information of the processing status, was stored to communicate with OGC WPS interfaces in this database management system. GDAL [27] is an open source software that helps to clip the processing zones and convert to an image on the Web after it has been processed using remote sensing algorithms in the Orfeo Toolbox [28]. Clipping is a function for extracting only those images that fall within a certain range from rectangular image data. GeoServer [29] was employed to manipulate the satellite image dataset and visualize the processing results. There are a number of things to consider, when applying PaaS, in order to make the actual on-demand approach. An app, which is a small application, developed by installing the web server and runtime environment, could be distributed using IaaS, PaaS or both types of services with IaaS. Since PaaS offers a web server and runtime environment, developers use the PaaS to build the right runtime environment and deploy the application in that environment and from the Web. The way to use virtual machines as hypervisors is to emulate hardware on top of the hypervisor, logically creating separate systems. On the other hand, the container arrangement provides an independent application performance environment by loading the required kernels and libraries. The container approach has advantages in terms of resource efficiency, although it is not independent of resource use, making it difficult to measure the amount of infrastructure resources. Application services of IaaS are deployed on a per-instance basis, whereas those of PaaS are deployed on a container-wide basis.

3. Open Source PaaS for a Geo-Spatial Image Service

3.1. Configuration for PaaS Application

The methods for configuring cloud computing services for PaaS can vary depending on the hardware conditions of service management and operation. In particular, PaaS is not only immediately applicable to general hardware, but it can also be configured above IaaS. PaaS actually works on top of hardware or virtualized resources in the following cases: PaaS could be offered on top of existing IaaS, which is not offered by the same provider; it could be offered by the same provider; and it could be built on top of a certain infrastructure, on top of which no IaaS services are offered. Thus, the strategies for PaaS application implementation are also diverse. The trial system with the geo-spatial data processing function was built on the PaaS cloud environment for testing in order to perform performance evaluations. To deploy PaaS, the open source scheme was employed in order to configure PaaS over IaaS, such that the IaaS-based cloud service was leveraged to compare the performance of the PaaS-based one. To deploy PaaS running on IaaS, all the services needed to support the geospatial application or service were manually deployed using virtual hardware in order to compare the processing performance. Both solutions for deploying IaaS and PaaS in this trial system are open-source; IaaS was realized via OpenStack [30], while PaaS, via Cloud Foundry [31] over the implemented IaaS cloud. Both are widely used as open-source projects for the development of cloud computing technology and services. The PaaS is a cloud computing environment, linked to the e-government framework, which is based on open source for the construction of public purpose Web services in Korea [32]. Cloud Foundry was adopted, along with many other open source PaaS products or solutions [33]. That is the main reason why Cloud Foundry was applied in this study. On the other hand, PaaS can be deployed in many IaaS environments, but OpenStack was applied because it is the most commonly used cloud computing environment. OpenStack is a collection of services, such as orchestration, block storage, networking, computing, measurement, authentication, and dashboard. The primary node configuration, which is deployed using OpenStack, consists of

a controller and a computer. The controller provides a function for controlling clients that call each hypervisor an API. In a hypervisor, one of the server virtualization types, virtual servers are separate because the OS and applications are virtualized on a unified basis. A product, named Nova, for the controller and computer functions in OpenStack is designed for horizontal expansion, regardless of whether a hypervisor is used, and provides the ability to interlink with legacy systems or third-party products. If necessary, the IaaS environment is configured to use block storage and object storage with additional storage nodes. A controller, a computer, and block storage are required to build the PaaS-based service for experimentation.

The physical servers used to build IaaS-based trial services, and the experimentation for the performance testing, are shown in Table 1. Storage is a recording service for storing data and programs, and there are three data accessing approaches offered by a cloud environment: block storage, object storage, and file storage. It has five servers, one is for the controller, two are assigned to the block storage service and two to the object storage service, as well as OpenStack computer services. Cloud Foundry is a platform that helps to deploy developer-run apps on his/her own computing infrastructure or an IaaS, minimizing the cost and complexity of configuring infrastructure for their apps. Cloud Foundry has subsystems that perform the following specialized functions: cloud balancing by three levels of BOSH, cloud controller, and router, supporting two types of virtual machines (VMs), such as the component VMs and the host VMs, and user authentication and authorization (UAA) servers. Its components communicate with each other by posting and sending messages, using the HyperText Transfer Protocol (HTTP)/HTTPS protocols and NATS in a cloud-native messaging system.

Table 1. Physical servers used in this experimentation.

Microprocessor for Server (OpenStack Service)	CPU (Core/Thread)	Memory (GB)	Storage (TB)
Intel Xeon E5-1607 (Controller)	4/4	8	1.5
Intel Xeon E5-1650 (Compute 1, Block Storage)	6/12	36	2
Intel Xeon X5650 (Compute 2, Block Storage)	6/12	24	2
Intel Xeon X5650*2 (Compute 3, Object Storage)	12/24	24	3
Intel Xeon X5550 (Compute 4, Object Storage)	4/8	48	3

The configuration for IaaS and PaaS in this study is illustrated in Figure 2. Necessary resources for the installation of Pivotal Cloud Foundry on OpenStack are as follows: 118 GB of RAM, 22 VMs or instances, 16 small VMs, three large VMs, three extra-large VMs, 56 vCPUs, and 1 TB of storage [34]. Cloud Foundry consists of 15 service units within seven components, such as the cloud controller, NATS messaging, Go-router, and UAA server. Networking should automatically support the Dynamic Host Configuration Protocol (DHCP) and be able to generate floating Internet Protocol (IP) addresses [35]. In addition, internal or external Domain Name System (DNS) servers should be deployed. With the PaaS service, several services and apps, such as geo-based image processing and the processed data manipulation for experimentation in the trial system, are uploaded and distributed on the Web. Access to them via a certain route needs to be supported. A separate deployment of DNS servers to a sub-domain, configured as a wild card, helps to connect the PaaS-based cloud. Cloud Foundry is installed using a command line interface (CLI), and a YAML file, a data serialization language [36] to minimize disk space or bandwidth requirements, was used for the configuration process. Pivotal [37], which heavily manages the Cloud Foundry project, provides an instance of the installation interface and calls it the Ops Manager. Through the Ops Manager, the Micro BOSH is created, an open source

tool chain for release and deployment management of large-scale distributed services, an instance required for Cloud Foundry installation. The instances created and managed by Micro BOSH are those in the PaaS service and have 22 instances, composed of 16 small VM for one vCPU with 1 GB RAM, three VMs for large VM of four vCPUs with 16 GB RAM, and three VMs for extra-large VM of eight vCPUs, with 16 GB RAM created in this study.

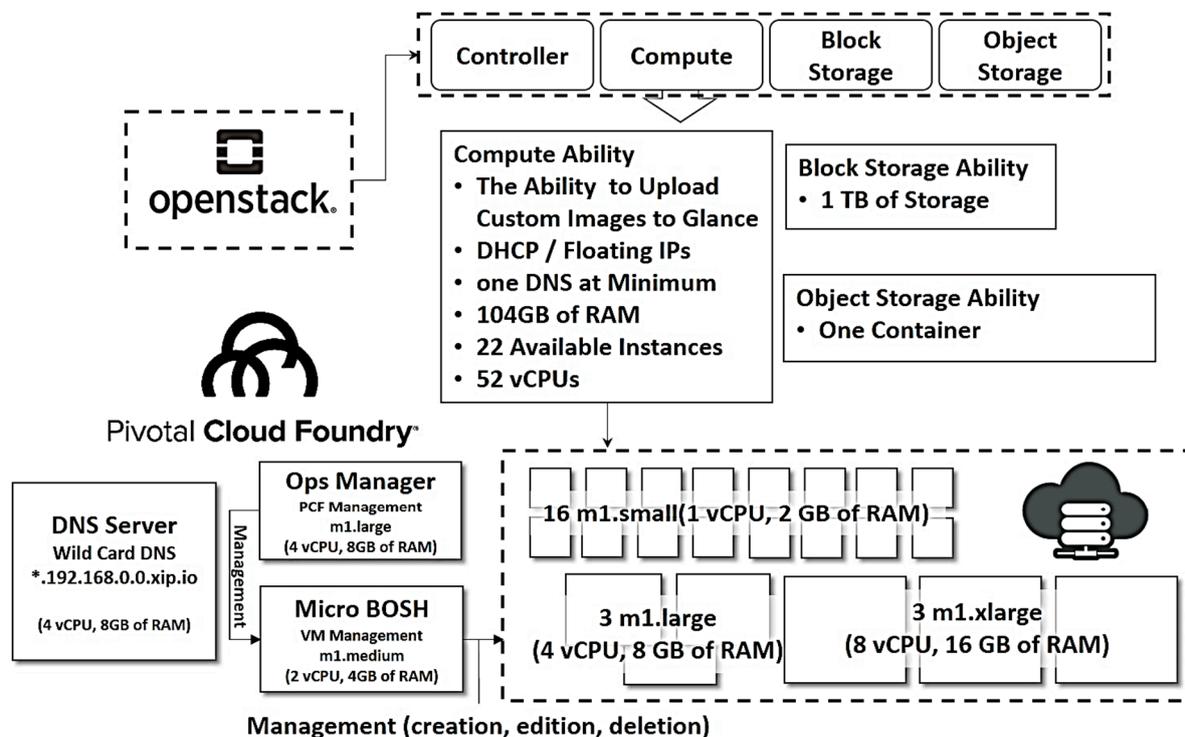


Figure 2. PaaS service implementation and instances using the open source of Cloud Foundry and OpenStack.

3.2. Geo-Spatial Image Processing Service for Experimentation

The focus of this study is basically to identify the performance of requests and responses to apps deployed through IaaS and PaaS computing environments. Thus, a simple and general algorithm with fewer input parameters was used in the experiments to lower the extra mathematical computing and processing time for the complex algorithms. Of course, other specialized algorithms and processing functions can be built in these cloud environments. OTB provides three types of edge detection filters, such as gradient, Sobel, and Touzi [38]. The gradient filter was applied in this study. The image used in the experiments was the Landsat 8 image, with the resolution of 30 m, taken on 3 March 2017, which was clipped as a rectangle boundary around the Seoul area. As for the final processed image, it was set as the OGC-based Web Map Tile Service (WMTS) type via GeoServer. It can also be imported and exported in the geotiff format using functions supported by a geo-spatial image processing service. In this case, the consuming time of importing the input image and generating the output image was not included in the response time. A geo-spatial image processing service based on PaaS cloud on IaaS eventually provides remote sensing image functions and analysis algorithms on the Web, and this experiment was carried out as one of the preliminary test studies.

The results of web applications deployed in IaaS and PaaS, built for configuration and performance measurements, are shown in Figure 3. It also shows what developers need to manage directly in IaaS and PaaS environments. As for the IaaS cloud, it is the configuration that developers install directly into the instance when they are provided with virtual machines that need to be configured. The virtual machines in this test system were created on OpenStack. As for the PaaS cloud, the

configuration assumes that the cloud provider provides the GeoServer, Zoo-Project, and PostgreSQL services. Since Cloud Foundry runs on IaaS, it must be deployed as an instance in order to be delivered as a service. Additionally, and it is necessary to add a Service Broker or register a service URL to be utilized as a service in Cloud Foundry. To register each middleware as a service, PostgreSQL added a service broker, and Zoo-Project and GeoServer were registered as URL-based services. Zoo-Project and GeoServer did not add a separate service broker because they could be serviced in the form of a simple RESTful. Instances for services should be deployed from the IaaS system separately, although this study leverages instances, configured in the IaaS configuration, to achieve the same performance for middleware systems. The cloud provider will configure a separate virtual machine for middleware in systems and provide it to users. Regardless of which cloud environment is used as an IaaS or a PaaS, most of them will be delivered to end-users in the style of SaaS. In this study, the same type of web application was distributed in each environment in order to derive results and perform processing functions for performance testing.

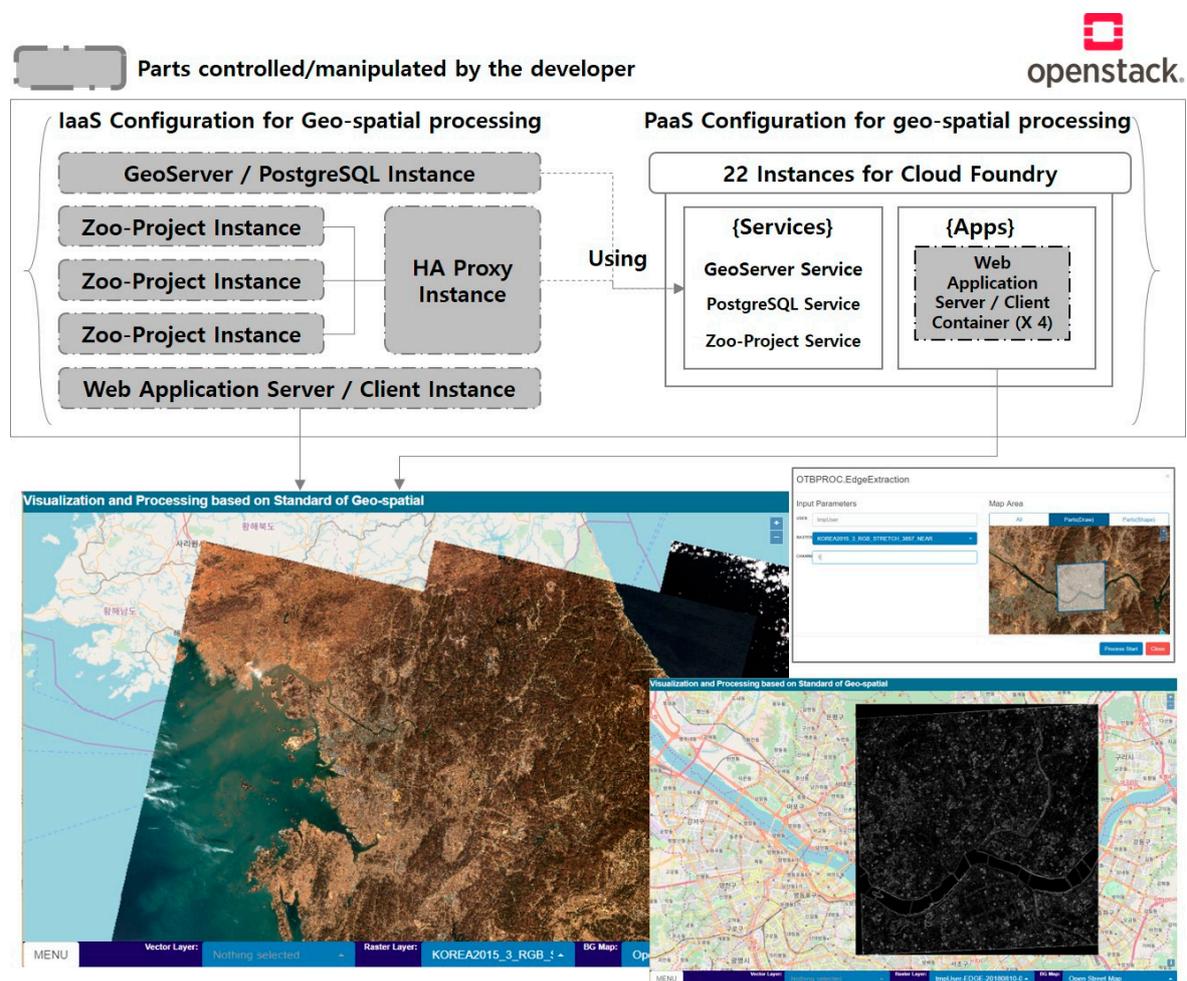


Figure 3. Configuration of the test case of the geo-spatial image processing service implemented in this study and an example of the processed result.

3.3. Experiment Conditions for a Performance Evaluation

A geo-spatial image processing service system using the interoperable web standards was developed through the deployed PaaS. This system supports the geo-spatial image web processing algorithms with OGC WPS 2.0. The performance evaluation in cloud computing environments may vary according to the measuring purpose or method, and a response time for each request, called

an image processing function, was chosen in this study. The open source tool used to evaluate performance was JMeter [39], which sets the numbers of the user within a certain interval to virtually match concurrent user overload tests. JMeter is used to simulate different load types on a server or network to test their strength or overall performance [40,41].

The control parameters for measuring performance are shown in Table 2. First, two cloud computing service cases were chosen from IaaS by OpenStack and PaaS on IaaS by Cloud Foundry on OpenStack. The number of threads simulates the number of concurrent users accessing the cloud services. The ramp-up period indicates the time duration between the thread launching starting and the thread execution ending. This means that if the number of the thread is 10, and the ramp up period is 100 s, as the experiment variables, it takes 100 s for all 10 threads to be completed. The loop count is how many times each thread will run. Middleware software or components in the geo-spatial image processing application of the deployed cloud computing environment for performance evaluations are shown in Table 3. The IaaS only offers VMs. The user is responsible for manually deploying the applications in the VMs, and PaaS is deployed by creating an internal container in the VM. The PaaS product, over the IaaS service in this study, already had HAProxy, a free and reliable solution for load balancing and proxying in Transmission Control Protocol (TCP) and HTTP applications.

Table 2. Experiment parameters for performance testing.

Test Parameter	Experiment Value		
Threads (unit: number)	300	500	700
Ramp-up Period (unit: seconds)	3600	3600	3600
Loop Count (unit: number)	10	10	10

Table 3. Open source solutions and middleware types for the geo-spatial image processing service system of IaaS and PaaS.

Instance/Container	Name	Index	IP/Router	Persistent Disk	VM-Type/Memory	Note
Container	Geo-spatial Processing App	0	WPS 2.192.168.0.0.xip.io	1 GB	1 GB	PaaS Service
		1		1 GB	1 GB	
		2		1 GB	1 GB	
		3		1 GB	1 GB	
Instance	Geo-spatial Processing App	0	172.16.0.5	80 GB	m1.medium	IaaS Service
Instance	GeoServer (Middleware)	0	172.16.0.11	100 GB	m1.medium	IaaS: IP PaaS: Service
Instance	PostgreSQL (Middleware)	1	172.16.0.8	100 GB	m1.medium	IaaS: IP PaaS: Service
Instance	HAProxy for Zoo-Project (Middleware)	0	172.16.0.15	None	m1.small	IaaS: IP PaaS: Service
Instance	Zoo-Project (Middleware)	0	172.16.0.3	100 GB	m1.large	Access via proxy
Instance		1	172.16.0.14	100 GB	m1.large	
Instance		2	172.16.0.16	100 GB	m1.large	

Load balancing is a common advantage of both IaaS and PaaS. For OpenStack IaaS, the load balancing capability is optional and additionally requires the implementation of LBaaS [42]. However, it is a basic function provided by the PaaS service. PaaS includes a load balance service at the same time as it is installed, so that load balancing can be applied immediately, depending on the status of containers for the distributed apps. Since IaaS does not include load balancing by default, separate HAProxy instances should be deployed. For the experimentation, PaaS in four 1 GB containers were

used to bring the memory to 4 GB, configured to the instance on IaaS. Instances with the Zoo-Project perform the actual geo-based processing functions. Thus, a HAProxy instance of load balancing for those functions was deployed. As for the virtual memory type, m1.small, m1.medium, and m1.large represent one vCPU with 2 GB memory, two vCPUs with 4 GB memory, and four vCPUs with 8 GB memory, respectively. There are various factors, such as network and computing resources, as well as hardware elements, relating to web stress performance measurements using JMeter. In order to minimize unexpected errors, three measurements were taken in the same test cloud environment, but all test systems were restarted prior to the measurement.

4. Results of Performance Tests

Performance evaluations of the 'DescribeProcess' and 'Execute' requests, among five OGC WPS 2.0 requests, such as GetCapabilities, DescribeProcess, Execute, GetStatus, and GetResult, were carried out. The results of performance testing for WPS, executing the operation in an IaaS environment without LBaaS and a PaaS environment by varying the number of threads, are shown in Figures 4–6. The mean values of the results of the three experiments for the 'DescribeProcess' request are presented in Figure 7. In Figures 4–6, the horizontal axis, representing the ramp-up period, shows the elapsed time in seconds. An experiment was performed, varying the ramp-up period by one hour. Since multiple experiments with different threads and variables could change the networking status and hardware conditions, instances on IaaS and apps on PaaS were adjusted to their initial state whenever each experiment was carried out in order to perform the experiment under the most similar environments possible. The vertical axis shows the response latencies, indicating the response time of each thread request in milliseconds. Parallel processing through HAProxy for adequately distributing the processing resources to the instances and log recording in the Zoo-Project confirmed that all three instances were being used for processing. Both IaaS and PaaS services represent that the response time increased over time, owing to a gradual increase in the load on the WPS servers, and PaaS services are more responsive than those running on IaaS.

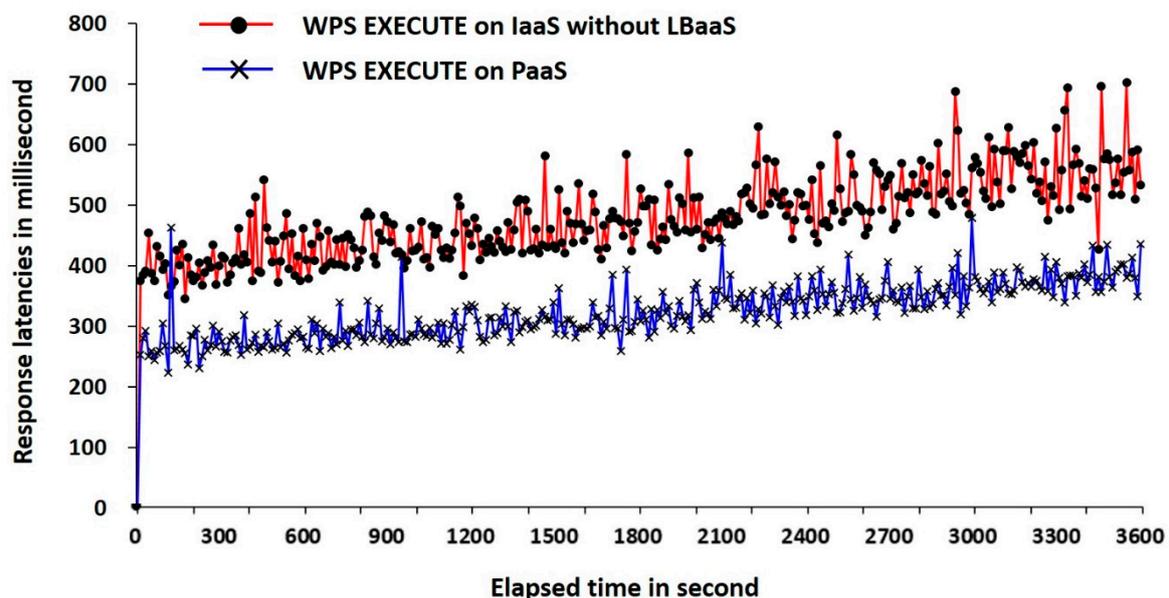


Figure 4. Performance test result for OGC WPS 2, executing an operation, with an edge detection algorithm, on PaaS and IaaS without LBaaS: a case of 300 threads for 3600 s.

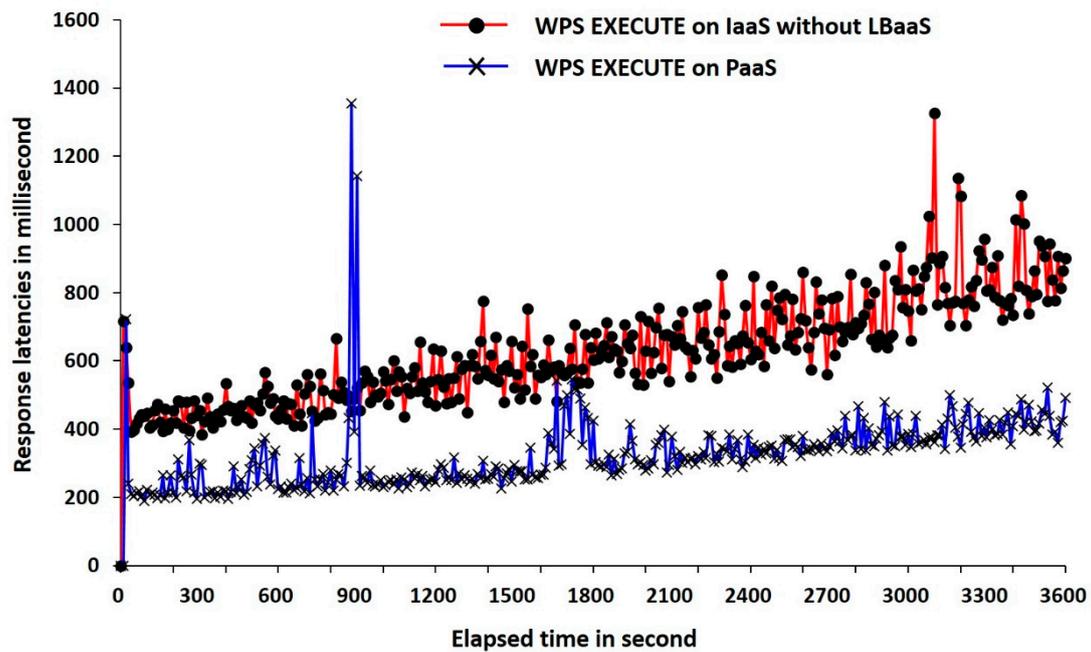


Figure 5. Performance test result for OGC WPS 2.0, executing an operation, with an edge detection algorithm, on PaaS and IaaS without LBaaS: a case of 500 threads for 3600 s.

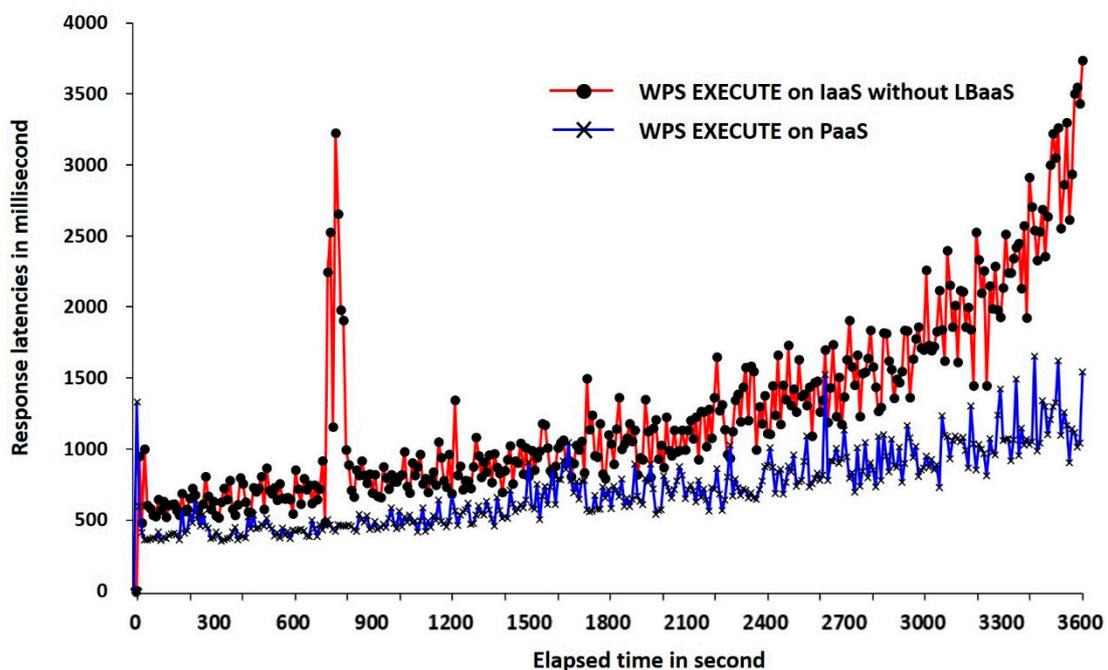


Figure 6. Performance test result for OGC WPS 2.0, executing an operation with an edge detection algorithm, on PaaS and IaaS without LBaaS: a case of 700 threads for 3600 s.

The asynchronous processing scheme, supported by OGC WPS 2.0, was used, which immediately sends an answer to the client when a request is made to execute. The possible asynchronous calls might prevent the WPS service from having very long connections, as well as clients from being blocked in waiting for an answer. Generally, an asynchronous processing scheme is applicable to any condition in which linking to local resources while a remote request is being processed is not desirable. Server virtualization virtualizes the entire hardware environment, while containers virtualize the application

execution environment. Each container can run applications without affecting other containers, and containers reduce performance degradation because of their low virtualization overhead. Since the Zoo-Project performs the actual processing, there is no actual burden on the geo-spatial image processing service. When an execute processing is requested, a jobID that responds to that request is sent to the service app and is stored, then, the processing status is checked periodically until the experiment is finished. As such, the more concurrent accesses, the higher the response to an execution request becomes.

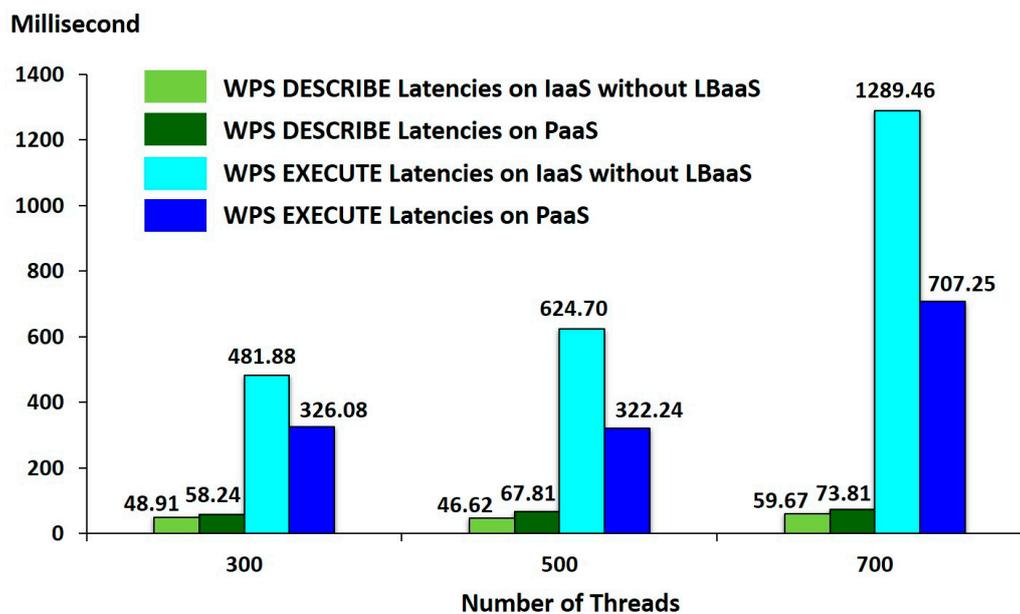


Figure 7. Performance test summary of OGC WPS 2 execute and describe operations with an edge detection algorithm for PaaS and IaaS without LBaaS.

Response latencies for 300 threads, for IaaS without LBaaS and PaaS, are shown in Figure 4. The result for 500 threads, in which PaaS shows almost the same trend as that in the case of 300 threads, but the response latencies for the IaaS service without LBaaS are higher than in the case of 300 threads, is shown in Figure 5. The result for 700 threads is shown in Figure 6. Moreover, the rate of growth of the IaaS service without LBaaS is much higher than that of one of PaaS services. The response latencies of PaaS are relatively stable, while it seems that they become doubled.

The average value of the five performance measurement experiments of each multi-thread, with 300, 500, and 700 threads, is shown in Figure 7. This chart also shows the error rate of responses to the OGC WPS 2.0 ‘describe’ request. On requesting the describe, the processing name as a parameter was sent, and the response result was the data path and the channel variable value for the edge detection algorithm applied in this experiment. These variables were entered into the execute request to perform the actual data processing. Unlike in the case of the execute request, the result of the OGC WPS 2.0 describe request was a single operation that disappears from the memory when a request is responded to. The average performance time required for the describe operation is considerably less than for the execute operation, as shown in Figure 7. In sum, the average of the execute response rates for PaaS was significantly lower than that for IaaS, as shown in Figure 7. Contrary to the execute operation case, the describe results show that the response to IaaS without LBaaS is slightly faster than that to PaaS. The reason is as follows: The app running on the IaaS cloud was configured as a single instance of m1.medium with two vCPUs and 4 GB of RAM, and the apps on the PaaS cloud consisted of four containers, each with 1 GB of capacity in the instance of m1.xlarge, with eight vCPUs and 16 GB of RAM. Assuming that a single request is given in an instance of an IaaS environment and a single

container of a PaaS environment, a ‘describe’ request can respond quickly in higher-performance configurations of the IaaS.

The error rate results, which were obtained through the table of the summary report of the JMeter tool, are shown in Table 4. Error rates which represent failure rates are expressed as a percentage regarding the result of incomplete or unclear responses to requests. Due to the asynchronous processing functionality of WPS 2.0, the response to the request is successful, but not necessarily the completion of the process. When there is too much concurrent processing or application overloading, the execute response value may fail. In error rates for the PaaS service, both the describe and execute operations are found to be less than 0.4% of any threads. This means that 28 requests of 70,000 requests by 700 threads have failed. Error rates of IaaS without LBaaS are higher than those of PaaS and are highly proportional to the number of threads. One of the reasons is that the response time is greater and, thus, the load becomes greater over time. This shows that as the load gets higher, when the number of threads is increased, the error rate also gets higher, except in one case of a PaaS execute with 500 threads. The maximum error rate rises to about 1.8% with 700 threads, which is a failure of 126 times. In comparison, the error rates for IaaS are 4.7 times higher than those for PaaS.

Table 4. Error rate of two operations in percent.

Request	Number of Threads		
	300	500	700
IaaS (without LBaaS) DESCRIBE Error rate	0.87	0.96	1.73
PaaS DESCRIBE Error rate	0.20	0.28	0.36
IaaS (without LBaaS) EXECUTE Error rate	0.47	1.32	1.74
PaaS EXECUTE Error rate	0.23	0.14	0.30

5. Discussion

The uses of open source cloud platforms of developing services that process geo-based images in a web environment are in the early stages. Therefore, experimental studies on the applicability and extensibility of cloud computing over geo-spatial processing are needed. This is the motivation for this research. Among many possible research topics, the focus in this study is to evaluate whether it is better to build a certain PaaS product over a PaaS or IaaS service using the generic variables, such as a multi-thread, ramp-up period and loop count, as well as the measurement of the error rates, regarding the completeness of requests in the cloud service.

The image processing function selected for the experiment is an edge extraction algorithm that performs a simple function, which does not require other functions linked with minimum user input variables. The image processing function applied in this study is one of the most frequently used satellite image processing functions. Since both IaaS and PaaS cloud services are basically web-based, OGC WPS 2.0 standards were applied in the experiment for performance evaluation. As for the experiment processes and results affecting the difference between the vendor and implementation, this experiment excluded, as much as possible on a technical basis, the influencing factors from these. Comparing the commercial systems with the open source systems is not the main scope of this study. This experiment did not apply LBaaS optional load balancing. If LBaaS were applied, it would have contributed greatly to improving the performance in the IaaS cloud. This study conducted experiments under limited conditions and simple variables. Thus, further experiments are needed to measure performance under more practical conditions, using large amounts of data in distributed servers to indicate which cloud environment becomes an effective way. For instance, a study comparing the performance of the PaaS cloud to the service applied by IaaS with LBaaS will also produce meaningful results, as in the case of applying open source tools, including Cloud Foundry and OpenStack. An experiment is necessary to identify how the optional LBaaS service applied to IaaS can contribute to improving performance when it is linked with a PaaS load balancing scheme.

6. Concluding Remarks

Cloud computing can be utilized in a vast range of fields. SaaS services can actually be deployed and provisioned through the use of PaaS and IaaS services. In order to make widespread use of actual cloud computing technologies in scientific application services, including geo-spatial data processing, the SaaS cloud service can be built on top of many prior studies in order to produce some valuable results. In particular, developers can expect to use cloud technologies for the application fields of service development, as the PaaS cloud provides support for middleware solutions, as well as other distributed computing resources. A great variety of applications can be developed and deployed through PaaS via the exploitation of middleware services and the building of the right and stable environment for the execution of these applications. Among the many issues related to open cloud computing applications, the performance evaluation was carried out to determine the need for PaaS in terms of geo-spatial image processing services.

As for the results, the execute request to OGC WPS 2.0 for an edge detection algorithm shows that a PaaS service via IaaS and PaaS clouds is better overall than IaaS without LBaaS. By applying an edge detection function, the performance of the WPS execute operation of a PaaS-based implementation is better than that of the IaaS without LBaaS. On the contrary, IaaS is better than PaaS regarding the ‘describe’ operation delay time. Under the same experiment conditions and variables for the performance tests, the error rate of most PaaS implementation cases was lower than those of the IaaS without LBaaS. Practical studies are needed to apply and link this technology to geo-spatial data application fields. These experimental results can be regarded as a reference point for the utilization of this technology in geo-spatial image processing services with the analysis functionalities of a large volume of data sets.

Author Contributions: K.L. conceptualized the research objectives, drafted the manuscript and provided revisions. K.K., under K.L.’s supervision, performed the system implementation and experiments.

Funding: This research was financially supported by Hansung University for Kiwon Lee. Additionally, 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) for Kwangseob Kim.

Acknowledgments: We thank the anonymous reviewers for their careful reading of our manuscript and their many insightful comments, corrections and suggestions.

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

References

1. Mell, P.; Grance, T. *The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology*; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2011; pp. 1–7.
2. Platform as a Service (PaaS) Market Worth \$6.94 Billion by 2018. Available online: <https://www.marketsandmarkets.com/PressReleases/platform-as-a-service-paas.asp> (accessed on 26 June 2018).
3. Platform as a Service (PaaS) Market to Observe Strong Development by 2022. Available online: <https://www.reuters.com/brandfeatures/venture-capital/article?id=16656> (accessed on 26 June 2018).
4. Choosing the Right Cloud Service: IaaS, PaaS, or SaaS. Available online: <https://rubygarage.org/blog/iaas-vs-paas-vs-saas> (accessed on 26 June 2018).
5. Yang, C.; Huang, Q.; Li, Z.; Xu, C.; Liu, K. *Spatial Cloud Computing: A Practical Approach*; CRC Press: Boca Raton, FL, USA, 2013; 309p.
6. Kang, S.; Lee, K. Auto-Scaling of Geo-Based Image Processing in an OpenStack Cloud Computing Environment. *Remote Sens.* **2016**, *8*, 662. [CrossRef]
7. Infrastructure as A Service—5 Important Benefits. Available online: <https://statetechmagazine.com/article/2014/03/infrastructure-service-5-important-benefits> (accessed on 27 June 2018).
8. Costache, S.; Dib, D.; Parlavantzas, N.; Morin, C. Resource Management in Cloud Platform as a Service Systems: Analysis and Opportunities. *J. Syst. Softw.* **2017**, *132*, 98–118. [CrossRef]

9. Wu, D.; Liu, X.; Hebert, S.; Gentzsch, W.; Terpenney, J. Democratizing Digital Design and Manufacturing using High Performance Cloud Computing: Performance Evaluation and Benchmarking. *J. Manuf. Syst.* **2017**, *43*, 316–326. [CrossRef]
10. Duan, Q. Cloud Service Performance Evaluation: Status, Challenges, and Opportunities—A Survey from the System Modeling Perspective. *Digit. Commun. Netw.* **2017**, *3*, 101–111. [CrossRef]
11. Lee, K.; Kang, K.; Kim, K.; Chae, T. Cloud-based Satellite Image Processing Service by Open Source Stack: A KARI Case. *Korean J. Remote. Sens.* **2017**, *33*, 339–350. [CrossRef]
12. Senyo, P.K.; Addae, E.; Boateng, R. Cloud Computing Research: A Review of Research Themes, Frameworks, Methods and Future Research Directions. *Int. J. Inf. Manag.* **2018**, *38*, 128–139. [CrossRef]
13. Yue, P.; Zhou, H.; Gong, J.; Hu, L. Geoprocessing in cloud computing platforms—A comparative analysis. *Int. J. Digit. Earth* **2013**, *6*, 404–425. [CrossRef]
14. Noor, T.H.; Zeadally, S.; Alfazi, A.; Sheng, Q.Z. Mobile cloud computing: Challenges and future research directions. *J. Netw. Comput. Appl.* **2018**, *115*, 70–85. [CrossRef]
15. Rodero-Merino, L.; Vaquero, L.M.; Caron, E.; Muresan, A.; Desprez, F. Building safe PaaS clouds: A survey on security in multitenant software platforms. *Comput. Secur.* **2012**, *31*, 96–108. [CrossRef]
16. Kritikos, K.; Kirkhamb, T.; Kryza, B.; Massonet, P. Towards a security-enhanced PaaS platform for multi-cloud applications. *Future Gener. Comput. Syst.* **2017**, *67*, 206–226. [CrossRef]
17. Best Free and Open-Source PaaS Tools for Developers. Available online: <https://blog.g2crowd.com/blog/platform-as-a-service-paas/best-free-open-source-paas-tools-developers/> (accessed on 26 June 2018).
18. Open Source IaaS Software Offering a Real Alternative to Commercial Clouds. Available online: <https://www.linuxlinks.com/iaas/> (accessed on 26 June 2018).
19. 10 Open Source Tools for Cloud Infrastructure and Management. Available online: <http://www.pcquest.com/10-open-source-tools-for-cloud-infrastructure-and-management/> (accessed on 26 June 2018).
20. Kim, K.; Lee, K. Linkage Base of Geo-based Processing Service and Open PaaS Cloud. *J. Korean Assoc. Geogr. Inf. Stud.* **2017**, *20*, 24–38. [CrossRef]
21. Kozhirbayev, Z.; Sinnott, R.O. A performance comparison of container-based technologies for the Cloud. *Future Gener. Comp. Syst.* **2017**, *68*, 175–182. [CrossRef]
22. Web Processing Service. Available online: <http://www.opengeospatial.org/standards/wps> (accessed on 10 December 2017).
23. ZOO-Project Documentation Release 1.5. Available online: <http://www.zoo-project.org/new/Resources/Documentation> (accessed on 10 January 2018).
24. Yoon, G.; Kim, K.; Lee, K. Performance Testing of Satellite Image Processing based on OGC WPS 2.0 in the OpenStack Cloud Environment. *Korean J. Remote Sens.* **2016**, *32*, 617–627. [CrossRef]
25. Yoon, G.; Lee, K. Performance Test of Asynchronous Process of OGC WPS 2.0: A Case Study for Geo-based Image Processing. *Korean J. Remote Sens.* **2017**, *33*, 391–400. [CrossRef]
26. PostgreSQL. Available online: <https://www.postgresql.org/> (accessed on 20 December 2017).
27. GDAL. Available online: <http://www.gdal.org/> (accessed on 20 December 2017).
28. Orfeo Toolbox Open Source Processing for Remote Sensing Images. Available online: <https://www.orfeo-toolbox.org/> (accessed on 10 December 2017).
29. GeoServer. Available online: <http://geoserver.org/> (accessed on 26 June 2018).
30. OpenStack Documentation: Overview. Available online: <https://docs.openstack.org/install-guide/overview.html> (accessed on 20 February 2018).
31. Cloud Foundry Open Source Cloud Application Platform. Available online: <https://www.cloudfoundry.org/> (accessed on 28 June 2018).
32. The Government of South Korea Creates an Open PaaS with Cloud Foundry. Available online: <https://www.altoros.com/blog/south-korea-adopts-cloud-foundry-as-its-paas/> (accessed on 27 June 2018).
33. South Korea Standardizes IT Service Delivery with Cloud Foundry. Available online: <https://www.cloudfoundry.org/blog/south-korea-standardizes-it-service-delivery-with-cloud-foundry/> (accessed on 27 June 2018).
34. PCF on OpenStack Requirements. Available online: <https://docs.pivotal.io/pivotalcf/2-1/customizing/openstack.html> (accessed on 26 June 2018).
35. OpenStack Networking Guide. Available online: <https://docs.openstack.org/mitaka/networking-guide/> (accessed on 26 June 2018).

36. YAML: YAML Ain't Markup Language. Available online: <http://yaml.org/> (accessed on 27 June 2018).
37. Pivotal Documentation. Available online: <https://docs.pivotal.io/pivotalcf/2-0/customizing/openstack.html> (accessed on 20 February 2018).
38. Chapter 8 Basic Filtering. Available online: <https://www.orfeo-toolbox.org/SoftwareGuide/SoftwareGuidech8.html> (accessed on 8 August 2018).
39. JMeter. Available online: <http://jmeter.apache.org/> (accessed on 26 January 2018).
40. Top 11 Open Source Performance Testing Tools for Load and Stress Testing. Available online: <https://www.joecolantonio.com/2017/07/18/open-source-performance-testing-tools/> (accessed on 28 June 2018).
41. Web Performance Testing: Top 12 Free and Open Source Tools to Consider. Available online: <https://techbeacon.com/web-performance-testing-top-12-free-open-source-tools-consider> (accessed on 28 June 2018).
42. Load Balancer as a Service (LBaaS). Available online: <https://docs.openstack.org/mitaka/networking-guide/config-lbaas.html> (accessed on 26 June 2018).



© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).