remote sensing

MDPI

Article

Parallel Agent-as-a-Service (P-AaaS) Based Geospatial Service in the Cloud

Xicheng Tan [1,2], Song Guo [3], Liping Di [4], Meixia Deng [4], Fang Huang [5,6], Xinyue Ye [7], Ziheng Sun [4], Weishu Gong [4], Zongyao Sha [1,2] and Shaoming Pan [8,*]

1 International School of Software, Wuhan University, 37 Luoyu Road, Wuhan 430079, China; xctan@whu.edu.cn (X.T.); zongyaosha@163.com (Z.S.)
2 Engineering Research Center for Geo-Informatics and Digital Technology Authorized by National Administration of Surveying, Mapping and Geoinformation, Wuhan University, Wuhan 430079, China
3 Shanghai Academy of Spaceflight Technology, Yuanjiang Road 3888, Shanghai 201109, China; guosonghit@aliyun.com
4 Center for Spatial Information Science and Systems, George Mason University, Fairfax, VA 22030, USA; ldi@gmu.edu (L.D.); mdeng@gmu.edu (M.D.); zsun@gmu.edu (Z.S.); weishugong@gmail.com (W.G.)
5 School of Resources & Environment, University of Electronic Science and Technology of China, 2006 Xiyuan Ave., Chengdu 611731, China; hfhbhzp@uestc.edu.cn
6 Institute of Remote Sensing Big Data, Big Data Research Center, University of Electronic Science and Technology of China, 2006 Xiyuan Ave., Chengdu 611731, China
7 Department of Geography, Kent State University, Kent, OH 44242, USA; xinyue.ye@gmail.com
8 State Key Laboratory for Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079, China
* Correspondence: pansm@whu.edu.cn

Academic Editors: Sangram Ganguly and Prasad S. Thenkabail
Received: 30 January 2017; Accepted: 13 April 2017; Published: 19 April 2017

Abstract: To optimize the efficiency of the geospatial service in the flood response decision making system, a Parallel Agent-as-a-Service (P-AaaS) method is proposed and implemented in the cloud. The prototype system and comparisons demonstrate the advantages of our approach over existing methods. The P-AaaS method includes both parallel architecture and a mechanism for adjusting the computational resources—the parallel geocomputing mechanism of the P-AaaS method used to execute a geospatial service and the execution algorithm of the P-AaaS based geospatial service chain, respectively. The P-AaaS based method has the following merits: (1) it inherits the advantages of the AaaS-based method (i.e., avoiding transfer of large volumes of remote sensing data or raster terrain data, agent migration, and intelligent conversion into services to improve domain expert collaboration); (2) it optimizes the low performance and the concurrent geoprocessing capability of the AaaS-based method, which is critical for special applications (e.g., highly concurrent applications and emergency response applications); and (3) it adjusts the computing resources dynamically according to the number and the performance requirements of concurrent requests, which allows the geospatial service chain to support a large number of concurrent requests by scaling up the cloud-based clusters in use and optimizes computing resources and costs by reducing the number of virtual machines (VMs) when the number of requests decreases.

Keywords: geospatial service; Open Geospatial Consortium (OGC); remote sensing data processing; cloud computing; agent; parallel computing

1. Introduction and Background

With the development of Web service technology [1], Web service based applications were introduced in the field of geographic information. Because of the advantages of Web services and the

introduction and popularity of Service Oriented Architecture (SOA) [2–5], geographic information systems have developed into a new form: geospatial services. Researchers introduced Web services into the field of geographic information science and built a form of SOA suited to that field [6–8]. Since then, researchers have continued working to optimize and improve geographic science's SOA [9–12]. The ISO TC211 standard and the Open Geospatial Consortium (OGC) made great strides toward standardizing geospatial data and services and have significantly promoted the development of geospatial services. Today, the standard specifications of geospatial services (i.e., Web Map Services (WMS), Web Feature Services (WFS), Web Coverage Services (WCS), Web Processing Services (WPS), Catalog Service for the Web (CSW), Sensor Web Enablement (SWE), etc.) have been widely adopted [13–17]. These geospatial service specifications help the users publish Web-based maps, geospatial data, geoprocessing services, and sensor Web services. Moreover, CSW helps the users find specific geospatial services and make use of them.

Adhering to a standard for geospatial service based applications has several advantages such as (1) increasing the ability to share geospatial data and geoprocessing models within a distributed environment [18–23]; (2) allowing distributed geospatial services to be reused by other systems [9]; (3) enabling geoprocessing to be conducted in a worldwide network environment [24–26]; and (4) supporting aggregation of distributed geospatial services into composite services or geospatial service chains to be able to conduct more complex tasks [27].

The advantages listed above have caused the SOA approach—particularly for standard geospatial services—to be widely adopted and applied throughout the world.

However, because of the current architecture and configuration of traditional Web services, geospatial services are usually static. In other words, after a geospatial service has been installed and configured, it runs statically on only one server rather than having any dynamic computing capabilities [28–30]. This static architecture has seriously limited geospatial service applications and brings major challenges to traditional geospatial analysis.

The static, traditional SOA based geospatial service architecture causes the following serious problems in practical applications:

1. The fixed location of traditional SOA services reduces the efficiency of distributed geoprocessing services because they perform distributed geospatial data transfer inefficiently: the standard SOA-based geospatial data transfer services such as WCS and WFS are extremely inefficient when transferring large spatial datasets. This low distributed data transfer efficiency can cause the geoprocessing tasks to fail. Unfortunately, this situation is only becoming worse as the need to transfer high-resolution image datasets over the Internet increases.
2. As geospatial applications increase in scale, geospatial services will continue to expand on the Internet. There is a strong desire to chain these distributed geospatial services by combining them into more powerful and more complex geospatial services [31]. However, because of the high instability of the distributed network environment, such service chains will fail frequently due to various factors (e.g., a node may be powered off, disconnected from the Internet, or experience a machine failure).
3. The problems of inefficiency and instability in the existing atomic services accumulate in composite services, causing lower-quality, unstable service chains that weaken the possibilities for practical utilization of geospatial services.

A number of outstanding studies have been conducted to solve these geospatial service problems. Service migration methods have been proposed that can move a service over the network to reduce the data transfer volume required [32,33]. Rebuilding services to be compatible with parallel computing can speed up geoprocessing, achieving high-performance computing (HPC) geospatial services [34–36]. The moving code approaches are presented to achieve efficient distributed geoprocessing in Spatial Data Infrastructures (SDI) or sharing geoprocessing logic on the Web [37]. The cloud can be utilized to achieve better load balancing for geocomputing services [38–42], also, software-as-a-service approaches

are utilized to provide efficient and low cost public geospatial services in the cloud [43]. Li proposed distributed geospatial data compression methods to reduce the large volumes of geography markup language (GML) or image data to improve the efficiency of geospatial service data transfers [44]. The concept of quality of service (QoS) was proposed. QoS model factors can be extracted to evaluate a geospatial service or work toward enhancing geospatial service quality. QoS measurements include collection factors (i.e., efficiency, robustness, security, and so on) [45–47]. Many researchers have focused on geospatial service aggregation methods to combine existing atomic geospatial services into composite geospatial services that can address more complex tasks [48–50].

To solve the problems of traditional SOA-based composite geospatial services, we proposed agent and cloud based service chaining method, which substitutes an agent for a fixed geospatial service, meaning that the service chain can still execute successfully when an individual atomic service becomes invalid, because the agent can act to replace the invalid service with an available service [51]. However, in that initial approach, the agent and the cloud-based services communicated in a non-standard manner (e.g., invoking ResultWFS and ResultWCS services to return results), which limited the agent's capabilities and applicability. Hence, we designed an Agent-as-a-Service (AaaS) based method to improve the efficiency, robustness, and to improve the ability for domain experts to collaborate [52]. The AaaS exhibits far better capacity than the traditional SOA based method and has a greater potential to handle disaster responses and environmental risks. However, AaaS performance is still compromised in some situations. For example, its execution time is still excessive when there are many concurrent requests, indicating that it may not meet some decision-making requirements. Moreover, in situations where many agents are trying to migrate to and execute on the same host node, the AaaS-based method may be unable to handle the load. Hence, both the scalability and performance of the initial AaaS method require further improvements.

In this paper, we will integrate parallel computing technology into the AaaS-based method to improve the scalability and performance. The proposed Parallel Agent-as-a-Service (P-AaaS) geospatial service can handle a larger number of concurrent requests with greater efficiency. The conducted experiments show the superiority of this improved method.

2. Geocomputing Resource Adjustment Mechanism for P-AaaS Based Geospatial Services

2.1. The P-AaaS Infrastructure

We initially designed the AaaS infrastructure in [52], and according to the designation, the AaaS infrastructure has five basic Web services, including Agent Registration, Agent Clone, Agent Migration, Service Generation, and Agent Life Detection. Users can register agents, conduct Agent migration and generate geospatial services by calling the five services. However, that agent was intended solely for sequential geoprocessing. Therefore, we optimized the existing AaaS infrastructure to fit a parallel computing paradigm.

This research develops a MPICH based P-AaaS approach. MPICH is a high performance and widely portable implementation of the Message Passing Interface (MPI) standard, and can support the redesign of the existing AaaS infrastructure with the least effort. There are alternative technologies such as Spark, which has better stability than MPICH. However, if the Spark is adopted, it is necessary to conduct a large redesign of the existing AaaS approach, including the data storage and the geospatial data publishing mechanisms. Hence, MPICH is adopted in this P-AaaS architecture.

Figure 1 shows how an agent migrates and converts into a parallel WPS service. With steps ① to ⑤, the agents will be searched, moved, registered, and transformed into WPS service. For the detailed procedure of Agent migration, see [52]. After migrating to the target cloud node, the agent sends message-passing interface (MPI) geocomputing programs to the shared folder of the master node of the virtual Linux cluster. As shown in Figure 1, the P-AaaS infrastructure has a new added service, PGEC, that the agent initiates to transfer the MPI parallel programs to the virtual Linux cluster nodes. This process is conducted by copying the parallel processing program from the AaaS node to

the folder of MPI on every VM in the Linux cluster. After the MPI parallel program has been sent to the target virtual machines (VMs), the parallel geocomputing environment is configured, and parallel geoprocessing can be launched.

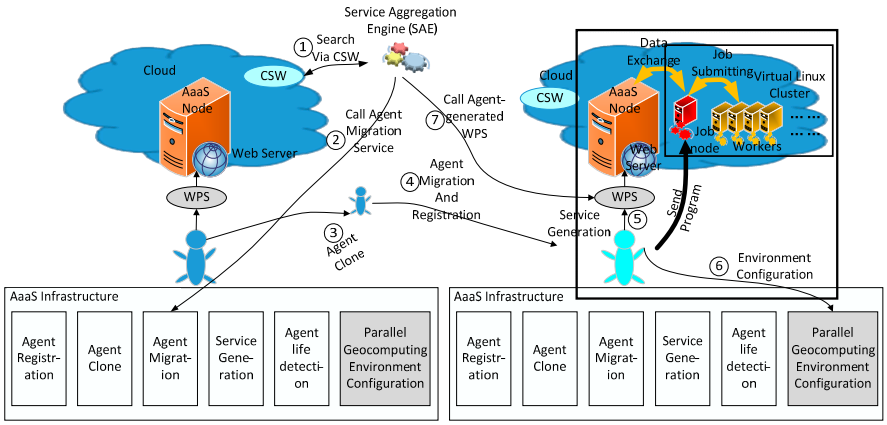

Figure 1. P-AaaS architecture.

2.2. Cloud Computing Resources Management Modules

To monitor and adjust the scale of the cloud computing resources dynamically, we designed the Geocomputing Environment Controller (GEC). The GEC includes the service environment Image Manager (IM), the Service Environment Image Configurator (SEIC), a Resource Monitor (RM), and Resource Adjustor (RA), as shown in Figure 2.

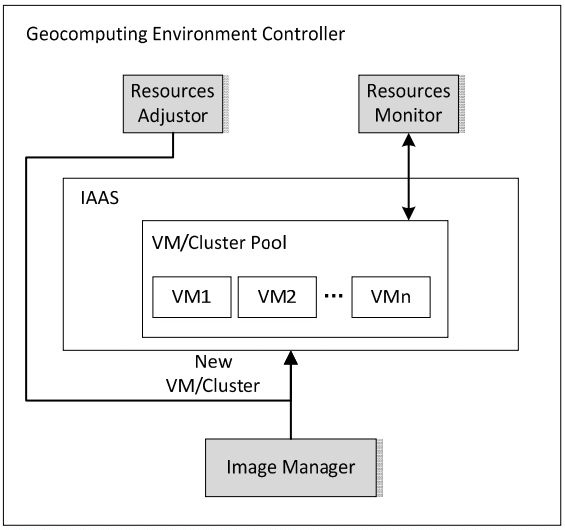

Figure 2. Computing resources management modules.

The functions of each of the four main parts are as follows:

1. The Resources Monitor (RM) monitors real-time performance data (i.e., CPU usage, RAM usage, network usage) from the virtual clusters and raises alerts when the cluster reaches threshold values.
2. The Resources Adjustor (RA) takes specific actions based on the various alerts from the RM to adjust the resources of a specific service. For example, if there is a lack of computing resources (e.g., CPU or RAM), the RA will launch new VMs to increase the cluster size. In contrast, if the

cluster contains more resources than the service needs, the RA will terminate some VMs to shrink the cluster.

3. The Image Manager manages image information; because different agents may need specific environments, the RA can search for a requested image via the IM and then launch new virtual machines or clusters as new computing resources.

2.3. Adjusting Geoprocessing Resources Dynamically

It is necessary to set the scale of a cluster; we can define the scale using experience and the efficiency that users and decision makers require. The higher the idle percentage, the higher the efficiency that can be obtained. In the procedures described above, after an AaaS virtual machine (VM) has been created and the appropriate agents migrated to that VM, the cloud Resources Adjustor will query the cluster's status and availability and then determine whether to create new VMs or terminate existing VMs. In this study, the number of VMs required is determined by the RA according to the number of concurrent requests being serviced. The rules for adjusting the parallel geoprocessing resources are as follows:

1. VM Creation Conditions. A predetermined number of fundamental VMs is assigned to handle group concurrent requests. For example, when the performance requirement is set to level 1, 24 two-core fundamental VMs are launched for every 15 concurrent requests. When the performance requirement is set to level 5, 28 two-core fundamental VMs are launched for every 15 concurrent requests. Consequently, as the number of concurrent requests increases, based on the performance requirement level, a corresponding number of additional VMs will be launched in the cluster.
2. VM Termination Conditions. Whenever the number of concurrent requests is reduced by 15, idle VMs will be terminated to reduce the scale of the cluster; a fixed number of fundamental VMs will be terminated from the cluster. For example, if the number of concurrent requests were to fall from 30 to 15, at the level 1 performance requirement, 24 fundamental VMs would be terminated. Similarly, at the level 5 performance requirement, 28 fundamental VMs would be terminated.

3. The P-AaaS Parallel Geoprocessing Mechanism

As additional job requests appear to be scheduled by the agent, concurrent jobs are submitted to the cluster via the Portable Batch System (PBS). The sub-tasks of these jobs are scheduled to VMs that have computational resources available. During geoprocessing task execution, the task is decomposed into smaller sub-tasks that share the same logic. These subtasks execute in separate worker threads in Single Instruction Multiple Data (SIMD) mode. The geospatial task decomposition and result combination is shown in Figure 3.

After the geoprocessing task has been decomposed logically into subtasks, the main thread sends equal numbers of subtask indices to the slave threads, and the worker threads then conduct the geoprocessing. Then the sub-results are sent to the master thread, which writes them into the combined result. Finally, the combined result will be copied to the shared folder of the AaaS node and then registered into the Geoserver as a data service (i.e., through WFS or WCS). We designed a parallel geoprocessing algorithm for agents as shown in Figure 4.

After completing the parallel environment configuration, when a WPS service is requested, the Agent will acquire the input data and save it to a folder shared with the login Linux node. After the required data have been copied to the login Linux node, the data are shared with all the other Linux worker nodes via NFS, thereby allowing all Linux worker nodes to acquire the needed data. Then, geocomputing starts by sending the input task-related parameters to the parallel program. The MPI program will then perform the geocomputing tasks in a parallel manner.

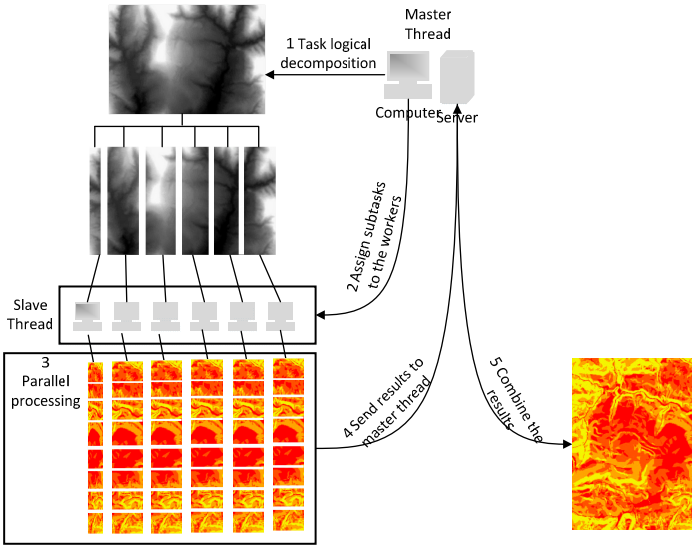

Figure 3. Geospatial task decomposition and result combination.

In this approach, a master/slave mode is adopted to conduct geocomputing tasks. The master thread decomposes the geocomputing task logically and sends the appropriate indexes of the decomposed task to slave threads via the MPI_send() function. The slave threads read the task data from the shared folder, extract the index number of the subtask, and start to conduct the computation. When a subtask completes, the slave thread save the result sub-tile data in the shared folder and sends the indexes of resulting sub-tile data to the master thread via MPI_send(). The master thread combines the finished resulting sub-tile data into the final result and copies it to the shared folder of the AaaS node. Finally, the Agent obtains the results and registers them into the Geoserver as the data services response.

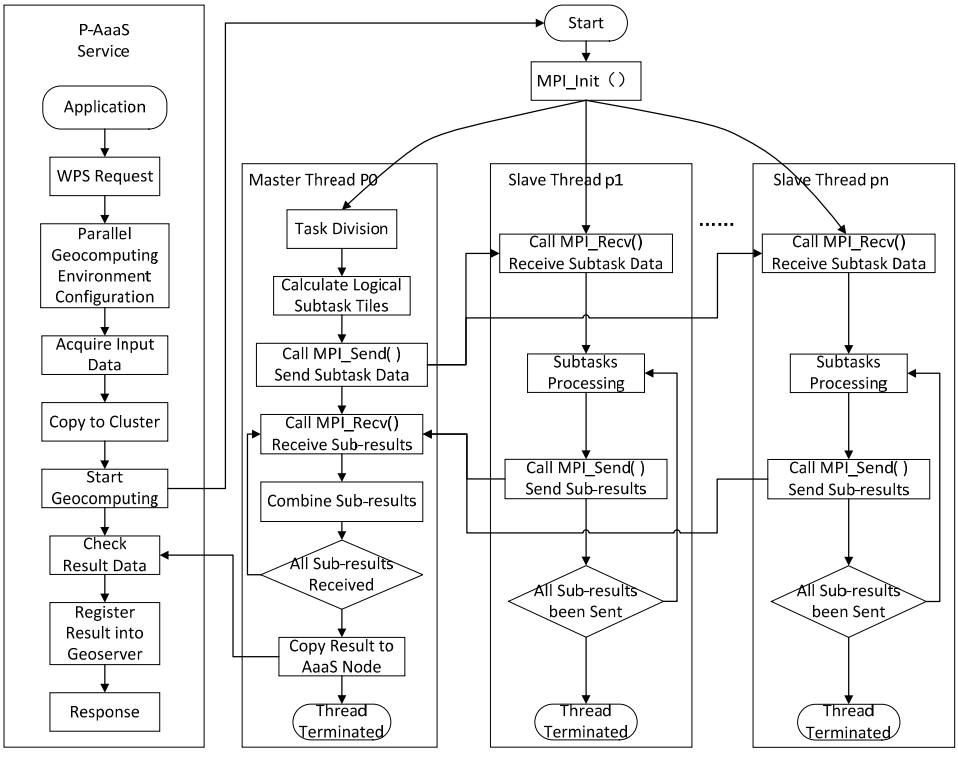

Figure 4. P-AaaS task scheduling mechanism.

4. P-AaaS Based Geospatial Service Chain Execution Algorithm

We described the details of the AaaS-based geospatial service aggregation mechanism in [52]. In this study, to improve the scalability and performance of the AaaS-based geospatial service, we redesigned the mechanism by which geospatial services execute, as shown in Figure 5.

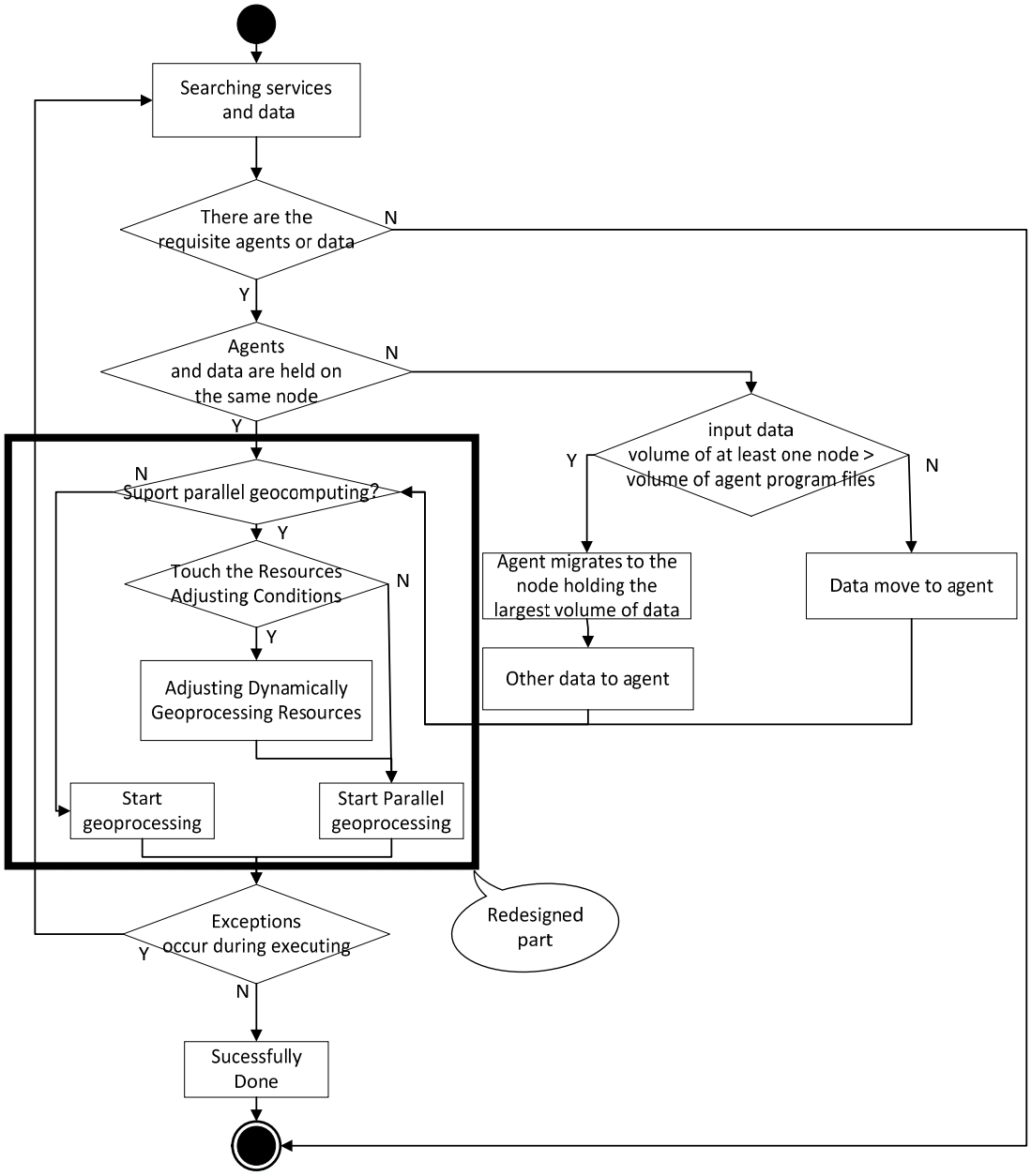

Figure 5. Execution mechanism of the P-AaaS based geospatial service.

The redesigned portion of Figure 5 shows that during the execution of P-AaaS based geoprocessing service, after the Agent migrates to the data, the procedure to conduct parallel geoprocessing operations is as follows:

1. Determine whether the geoprocessing service can be executed in parallel. This depends on whether the geoprocessing service supports parallel geoprocessing.
2. When computing resources are available for the task, it immediately begins conducting geoprocessing;

3. When more concurrent requests are submitted and have been processed by the geoprocessing resources adjusting mechanism as described in Section 2.3, the elastic geocomputing resources adjustment process will be conducted to accommodate the geoprocessing requirements.

After the geospatial service chain begins executing, agents that support parallel computing will execute with enhanced performance; however, if the service is intended to process only simple and fast tasks, it is unnecessary to parallelize them. In this manner, the P-AaaS method can not only utilize the mobile computing capability of existing agents but can also make use of the capacity of parallel computing. This approach optimizes the efficiency of the AaaS-based method and will act to improve the execution speed of the geospatial service chain.

5. Model and Experimental System

To illustrate the feasibility of the P-AaaS method described above, we used the same flood response model described in [52]. This model includes the requisite analyses during the flooding. Decision makers and participants can execute the model during the flooding or at any time to simulate the flooding response. To demonstrate the efficiency of the P-AaaS method, we need to parallelize the specific Agents, which are time consuming and can benefit from the parallelization. According to the execution time of the involved Agents, the flood submergence analysis and multi-spectral remote sensing data based crop extraction analysis consume the main part of the execution time. The other Agents handle smaller geospatial dataset or vector data processing, which are not time consuming in the model. Hence, they are still executed sequentially. However, if a very large vector data is involved, the execution time of the Agent might increase, and need to be parallelized. Consequently, we build two parallelized agents (i.e., parallel flood submergence analysis and parallel crop extraction analysis). The model is shown in Figure 6.

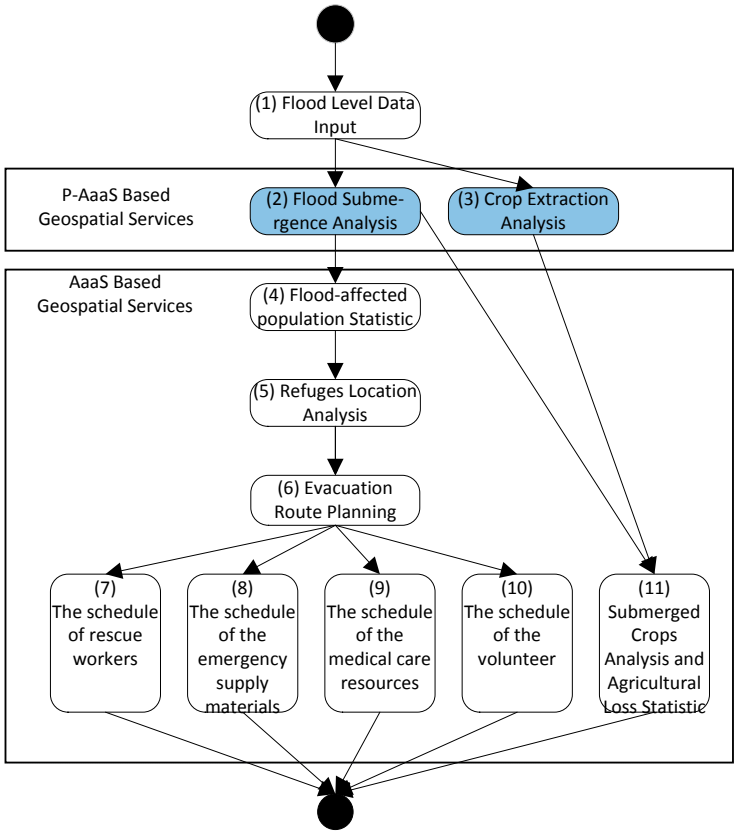

Figure 6. Flood response model.

We switched the cloud platform from Alibaba Cloud to QingCloud, a cloud platform in China that can provide global IaaS, PaaS, and SaaS services. QingCloud is featured by per second billing technology, and it opens the programming API to utilize the cloud resources efficiently. All these features helps the cloud users use the capability of cloud computing and reduce the cost. The Agents and data distribution of the use case are listed in Table 1.

Table 1. Agents and data distribution.

Cloud Node ID	Owner	Cloud Region		Agent	Data	Retained Data Volume
		Region Name	Location			
Cloud 1	*FCDRO*	PEK2	Beijing1	None	Flood depth, safe area, FDZ data	0.5 GB
Cloud 2	*MLR*	PEK2	Beijing1	None	1 m DEM, topographic map	162.25 GB
Cloud 3	*MT*	PEK3A	Beijing2	None	Transportation and road data	0.5 GB
Cloud 4	*BS*	GD1	Guangzhou	None	Population and economic statistics data	0.32 GB
Cloud 5	*MA*	PEK3A	Beijing2	None	Landsat TM multispectral remote sensing data, agricultural multispectral sample library	16.5 GB
Cloud 6	*ARO*	AP1	Hong Kong	Parallel flood submergence analysis; flood-affected population evaluation; refuge location analysis and evacuation route planning; scheduling of rescue workers, emergency supply materials, medical care resources, and volunteers; parallel crop extraction analysis, submerged crop analysis and agricultural loss evaluation.	None	0

To build the P-AaaS based geospatial service system, six Windows Server 2008 R2 cloud VMs were created. The P-AaaS infrastructure, Geoserver 2.7, Openlayers 3.0, Grass GIS 6.44, and 52° North WPS were installed in the Windows image to publish the geospatial data and conduct geographic analysis. The CentOS instance was created in the cloud cluster, and MPICH was used to build the parallel geoprocessing program.

Figure 7 shows the prototype system of P-AaaS-based flood response. After users click the "START ANALYSIS" button, the flood response analysis task begins to execute according to the flood response model described above. When the task is completed, the submerged corps data, evacuation routes, and refuge locations are created and overlapped on the flood affected area as shown in Figure 7. Then, users can query the requirement level for food and fresh water, rescue workers, emergency supply materials, medical care resources, and volunteers by clicking on the refuge location area. These geoprocessing results can help decision makers make sound flood response decisions rapidly.

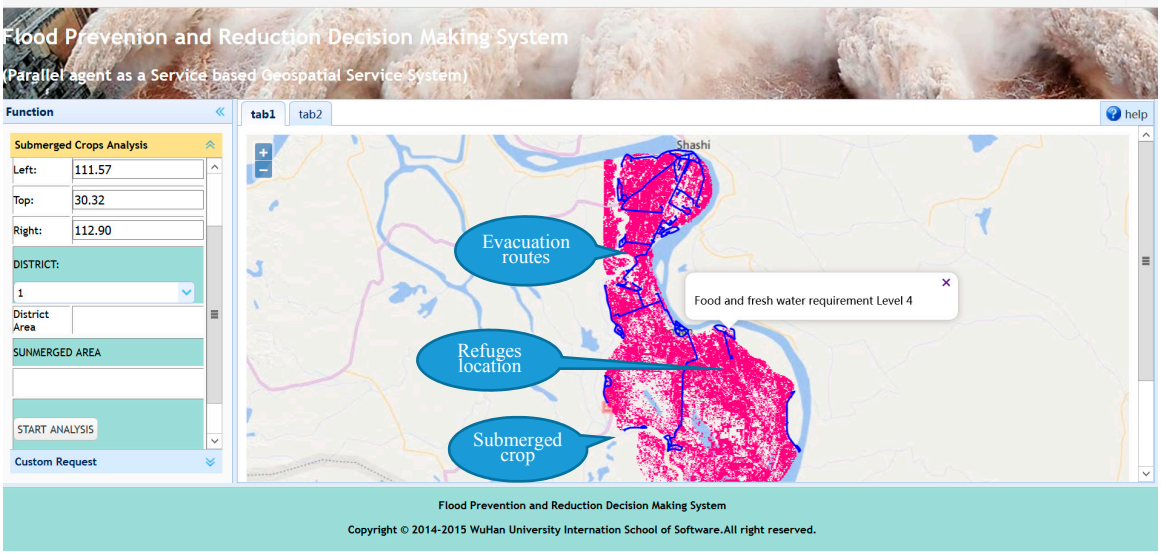

Figure 7. Flood response results.

6. Discussion

The P-AaaS geospatial service execution performance and data acquisition is compared to the existing AaaS service; the new characteristics of the P-AaaS method are also tested. The configurations of the AaaS and P-AaaS approaches are listed in Table 2.

Table 2. System configurations for the tests.

Approach	Machine Configuration	Deployment of Data and Agents
1	Six QingCloud Windows VMs, each with eight VCPUs with 2.6 GHz, 64 GB of RAM, a 50-GB disk and DTR of 10 Mbps	As shown in Table 1
2	Six QingCloud Windows VMs, each with four VCPUs with 2.6 GHz, 32 GB of RAM, 50-GB disks and DTR of 10 Mbps Virtual Clusters: QingCloud Virtual Clusters, each node with dual-core CPUs with 2.1 GHz, 32 GB of RAM, can increase or decrease Linux nodes dynamically	As shown in Table 1

6.1. Performance for Different Concurrent Requests

We conducted performance comparisons between the AaaS-based method and traditional SOA methods in [52]; these comparisons showed that the AaaS-based method had far better performance than the SOA methods. Therefore, we do not compare AaaS or P-AaaS with SOA methods here, but instead compare the performances of the AaaS-based method and the P-AaaS based method, and the performance requirement is set to level 4. The results are shown in Figure 8.

Although the AaaS-based method improved performance over the traditional (i.e., SOA-based) methods, the data shows that the performance of the AaaS-based method is not promising in some situations. For example, it requires approximately 5 h to execute when there are more than 15 concurrent requests. When there are 60 concurrent requests, the AaaS method's execution time increases to more than 50 h. This indicates that the AaaS-based method may not be suitable for meeting some decision-making requirements when there are many concurrent requests. In contrast, the P-AaaS method required only approximately 2.66 h to service 15 concurrent requests. When the number of requests increased to 30, 45, and 60, the P-AaaS method's execution times were approximately 2.7, 4.8, and 5.8 h, respectively, indicating that the P-AaaS based method achieves dramatically better performance than the AaaS-based method when there are many concurrent requests. Figure 8 also

shows that the data acquisition time comprises the dominant portion of the execution time for the P-AaaS based method. Moreover, as the number of concurrent requests increases, the data acquisition time increases accordingly. This occurs because when there are multiple concurrent requests, the IO performance of the data server degrades accordingly. In future work, we plan to enhance the data acquisition capability of the proposed approach.

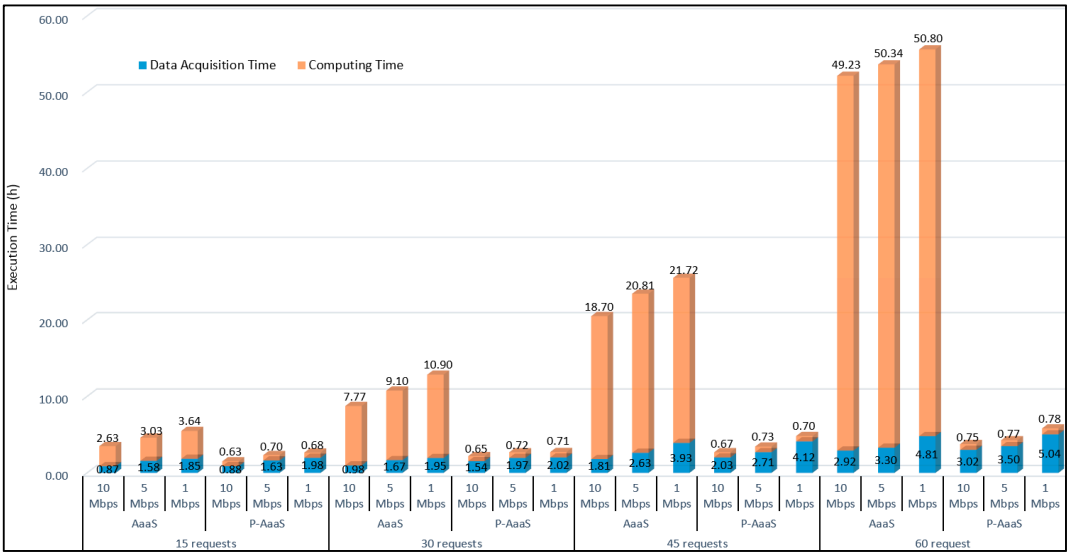

Figure 8. Execution times of the AaaS-based and P-AaaS based methods.

6.2. Performance Adjustability

The prototype system also has the capability to adjust performance by setting the performance requirement level. If higher performance is required, the user can raise the performance level to level 5. Conversely, when high performance is not required, the user can reduce the performance level to level 1. We variously set the performance level between 1 and 5 and tested the system with 30 concurrent requests and a 1-Mbps network. Under these conditions, the execution times of the tests are illustrated in Figure 9.

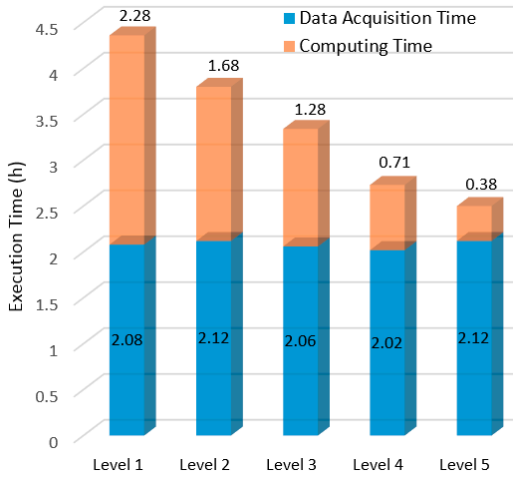

Figure 9. Performance of P-AaaS based method at different performance requirement levels.

As Figure 9 shows, under different performance requirement levels, the P-AaaS method can meet users' needs by adjusting its performance. When the performance level is set to level 1, the average

computing time for all requests is approximately 2.28 h, and the total execution time is 4.3 h. When the performance requirement level is set to level 5, the average computing time falls to approximately 0.4 h, and the total execution time is 2.5 h. This adjustable feature is advantageous for different applications. For example, during a disaster (e.g., a flood or typhoon response), a higher performance requirement level is necessary, but for applications that are not urgent such as science data analysis, a lower performance requirement level is recommended, which saves cloud computing resources. The figure also shows that during execution of the P-AaaS based method, although it can save time in transferring distributed spatial data, its data acquisition time would be far below that of the traditional SOA-based method. This feature was demonstrated in [52], but data transfer time is still a dominant factor here. Again, even in this highly concurrent system, the efficiency of disk IO and geospatial data services (i.e., WFS and WCS) still requires more optimization.

6.3. Resources Requirement Analysis

It is obvious that the sequential AaaS-based method requires fewer computing resources, but when many concurrent requests occur, its performance is poor. In comparison, the P-AaaS based method requires more computing resources but can obtain a much better performance. Figure 10 shows how the P-AaaS based method utilizes cloud platform capabilities to launch new VMs dynamically according to the number of concurrent requests. When more requests and higher performance requirements are anticipated, the system can launch more VMs in the cluster and when fewer concurrent requests occur or lower performance requirements are needed, the system can terminate unused VMs in the cluster. Hence, when there are few concurrent requests (e.g., less than 15), idle computing resources can be used by other applications. This approach also helps geospatial service vendors save on capital investments—which is critical for smaller institutes that would like to be able to offer or use high-performance geospatial services.

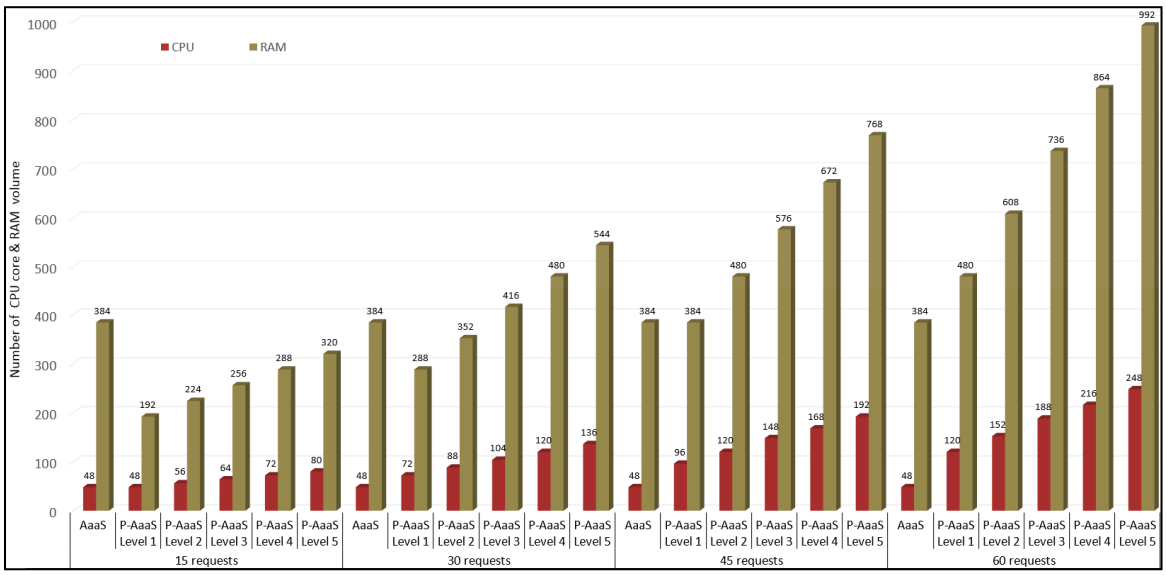

Figure 10. Computing resource requirements of the P-AaaS based method.

6.4. Transferability and Generalization Analysis

Transferability and generalization capability is usually a significant factor to evaluate the superiority of a method. The proposed P-AaaS method in this paper is built based on the OWS specifications, which makes the P-AaaS method has superior transferability. Moreover, the P-AaaS works on Windows and Linux systems, which are supported by the popular cloud platforms. Hence, the P-AaaS can also be moved to other IaaS cloud environments (e.g., Amazon AWS, Azure, etc.).

To generalize this method in other scenarios (e.g., earthquake, tsunami, etc.), we need to build field related parallel Agent and aggregate them into the service chain according to the models. On the other hand, the P-AaaS method still needs more optimizations. In this research, not all tasks are parallelized, and we determine which Agent need to be parallelized empirically. However, if the method is going to be adopted widely, more efforts need to be put on the management and description of the Agents and build the rules of choosing parallel Agent or sequential Agent automatically. We will continue to do more work on the generalization of the P-AaaS approach to make it easier to be implemented in more geospatial applications.

7. Conclusions

In this paper, we proposed a performance-adjustable P-AaaS based geospatial service approach that can address the performance problems of the sequential AaaS-based method. Although the performance of the AaaS-based method is much better than the traditional SOA-based methods, we still found that the sequential AaaS-based method experiences performance challenges when handling many concurrent requests. To solve these problems, P-AaaS utilizes cloud-enabled parallel computing technology to optimize the efficiency of the agents introduced for the AaaS-based method. P-AaaS approach also provides a new manner to aggregate the cloud-based parallel remote sensing data processing with the geospatial service chain. The P-AaaS based method not only inherits the capabilities of the AaaS-based method (i.e., avoiding the transfer of large volumes of data, agent migration, intelligent service conversion, improving domain expert collaboration, etc.), but also, based on the experiments described in this paper, shows how the remote sensing data involved composite geoprocessing service chains or workflows can obtain higher performance. It is usually not easy to obtain high performance geoprocessing in a distributed environment, but, using the method proposed here, when domain experts utilize a distributed geospatial service concurrently, this method provides an acceptable level of geoprocessing performance, which is critical during emergency and natural disaster responses, particularly when large-volume remote sensing data or big spatial data are involved. The experiments also demonstrated that the P-AaaS method can adjust the performance of the geospatial service chain when facing different performance requirements. The performance-adjusting ability of the proposed method is useful for those users who have different performance requirements. Moreover, in the future, our efforts will focus on accurately controlling the execution time of the geospatial service chain or geoprocessing models by adjusting the performance. This will enable domain experts to accurately specify the performance requirements of their geospatial services, which will improve the efficiency of emergency responses, making them more predictable and controllable. Finally, the experiment demonstrated that the P-AaaS based method can automatically change the scale of the computing resources employed based on the number of concurrent requests and on performance requirements. Consequently, geospatial services can be provided in a green and less expensive manner. Our future efforts will also involve researching the temporal-spatial rules that govern changes in the number of requests and using that information to adjust the available cloud computing resources in advance, offering users an even better experience.

Finally, data acquisition time is still a dominant factor when there are many concurrent requests because of the capabilities of data services (i.e., WCS, WFS, and the Geospatial server (e.g., Geoserver)). Therefore, our future work will address data service efficiency through scaling the data server by adopting big data processing methods (e.g., HDFS and NoSQL) to optimize data acquisition performance and obtain a superior cost performance.

Acknowledgments: We thank the editors and the reviewers for their outstanding comments and suggestions, which greatly helped to improve the technical quality of the manuscript. This study was supported in part by the National Science Foundation of China (NSFC) (Nos. 51277167, 41371371, 41671382, and 41271398), Shanghai Aerospace Science and Technology Innovation Fund (No. SAST2016006), and the Key Laboratory of Spatial Data Mining & Information Sharing of the Ministry of Education, Fuzhou University (No. 2016LSDMIS06).

Author Contributions: Xicheng Tan and Shaoming Pan contributed to design the main idea, implemented and developed the proposed methodology, and wrote the paper; Liping Di, Meixia Deng and Fang Huang contributed some ideas; Ziheng Sun, Weishu Gong and Song Guo contributed to the discussion of the methodology and results; Zongyao Sha and Xinyue Ye revised the paper.

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

References

1. Papazoglou, M.P.; van den Heuvel, W. Service oriented architectures: Approaches, technologies and research issues. *VLDB J.* **2007**, *16*, 389–415. [CrossRef]
2. Fang, Y.; Lee, B.; Chou, T.; Lin, Y.; Lien, J. The implementation of SOA within grid structure for disaster monitoring. *Expert Syst. Appl.* **2009**, *36*, 5784–5792. [CrossRef]
3. Rotem-Gal-Oz, A.; Bruno, E.; Dahan, U. *SOA Patterns*; Manning: Shelter Island, NY, USA, 2012.
4. Pasley, J. How BPEL and SOA are changing web services development. *IEEE Internet Comput.* **2005**, *9*, 60–67. [CrossRef]
5. Schroth, C.; Janner, T. Web 2.0 and SOA: Converging concepts enabling the internet of services. *IT Prof.* **2007**, *9*, 36–41. [CrossRef]
6. Karantzalos, K.; Bliziotis, D.; Karmas, A. A scalable geospatial web service for near real-time, high-resolution land cover mapping. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 4665–4674. [CrossRef]
7. Castronova, A.M.; Goodall, J.L.; Elag, M.M. Models as web services using the open geospatial consortium (OGC) web processing service (WPS) standard. *Environ. Model. Softw.* **2013**, *41*, 72–83. [CrossRef]
8. Kussul, N.; Shelestov, A.; Skakun, S.; Li, G.; Kussul, O.; Xie, J. Service-oriented infrastructure for flood mapping using optical and SAR satellite data. *Int. J. Digit. Earth* **2014**, *7*, 829–845. [CrossRef]
9. Di, L.; Zhao, P.; Yang, W.; Yue, P. Ontology-driven automatic geospatial-processing modeling based on web-service chaining. In Proceedings of the 6th Annual NASA Earth Science Technology Conference, College Park, MD, USA, 27 June 2006; pp. 27–29.
10. Yue, P.; Di, L.; Yang, W.; Yu, G.; Zhao, P. Semantics-based automatic composition of geospatial web service chains. *Comput. Geosci.* **2007**, *33*, 649–665. [CrossRef]
11. Deng, M.; Di, L. *Facilitating Data-Intensive Research and Education in Earth Science—A Geospatial Web Service Approach*; LAP LAMBERT Academic Publishing GmbH: Saarbrücken, Germany, 2010.
12. Kalluri, S.; Gundy, J.; Haman, B.; Paullin, A.; Van Rompay, P.; Vititoe, D.; Weiner, A. A high performance remote sensing product generation system based on a service oriented architecture for the next generation of geostationary operational environmental satellites. *Remote Sens.* **2015**, *7*, 10385–10399. [CrossRef]
13. Di, L. A framework for developing Web-service-based intelligent geospatial knowledge systems. *Geogr. Inf. Sci.* **2005**, *11*, 24–28. [CrossRef]
14. Di, L.; Chen, A.; Yang, W.; Wei, Y.; Mehrotra, P. The development of a geospatial data grid by integrating OGC web services with globus-based grid technology. *Concurr. Comput. Pract. Exp.* **2008**, *20*, 1617–1635. [CrossRef]
15. Di, L.; Yu, G.; Shao, Y.; Bai, Y.; Deng, M.; McDonald, K.R. Persistent WCS and CSW services of GOES data for GEOSS. In Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Honolulu, HI, USA, 25–30 July 2010; pp. 1699–1702.
16. Bielski, C.; Gentilini, S.; Pappalardo, M. Post-disaster image processing for damage analysis using GENESI-DR, WPS and grid computing. *Remote Sens.* **2011**, *3*, 1234–1250. [CrossRef]
17. Hu, C.; Guan, Q.; Chen, N.; Li, J.; Zhong, X.; Han, Y. An observation capability metadata model for EO sensor discovery in sensor web enablement environments. *Remote Sens.* **2014**, *6*, 10546–10570. [CrossRef]
18. Du, W.; Chen, N.; Yan, S. Online soil moisture retrieval and sharingusing geospatial web-enabled BDS-R service. *Comput. Electron. Agric.* **2016**, *121*, 354–367. [CrossRef]
19. Chen, Z.; Lin, H.; Chen, M.; Liu, D.; Bao, Y.; Ding, Y. A framework for sharing and integrating remote sensing and GIS models based on Web service. *Sci. World J.* **2014**, *2014*, 354919. [CrossRef] [PubMed]

20. Lin, H.; Yu, B.; Chen, Z.; Ge, R. A geospatial web portal for sharing and analyzing greenhouse gas data derived from satellite remote sensing images. *Front. Earth Sci.* **2013**, *7*, 295–309. [CrossRef]
21. Han, W.; Di, L.; Zhao, P.; Shao, Y. DEM explorer: An online interoperable DEM data sharing and analysis system. Environ. *Model. Softw.* **2012**, *38*, 101–107. [CrossRef]
22. Li, X.; Di, L.; Han, W.; Zhao, P.; Dadi, U. Sharing geoscience algorithms in a web service-oriented environment (GRASS GIS example). *Comput. Geosci.* **2010**, *36*, 1060–1068. [CrossRef]
23. Yang, C.; Raskin, R. Introduction to distributed geographic information processing research. *Int. J. Geogr. Inf. Sci.* **2009**, *23*, 553–560. [CrossRef]
24. Chen, J.; Xiang, L.; Gong, J. Virtual globe-based integration and sharing service method of geospatial information. *Sci. China Earth Sci.* **2013**, *56*, 1780–1790. [CrossRef]
25. Chen, N.; Di, L.; Yu, G.; Gong, J. Geo-processing workflow driven wildfire hot pixel detection under sensor web environment. *Comput. Geosci.* **2010**, *36*, 362–372. [CrossRef]
26. Chen, A.; Di, L.; Wei, Y.; Bai, Y.; Liu, Y. Use of grid computing for modeling virtual geospatial products. *Int. J. Geogr. Inf. Sci.* **2009**, *23*, 581–604. [CrossRef]
27. El Hadad, J.; Manouvrier, M.; Rukoz, M. TQoS: Transactional and QoS-aware selection algorithm for automatic web service composition. *IEEE Trans. Serv. Comput.* **2010**, *3*, 73–85. [CrossRef]
28. Jensen, J.L.; Bohonak, A.J.; Kelley, S.T. Isolation by distance, web service. *BMC Genet.* **2005**, *6*. [CrossRef] [PubMed]
29. Curbera, F.; Duftler, M.; Khalaf, R.; Nagy, W.; Mukhi, N.; Weerawarana, S. Unraveling the web services web: An introduction to SOAP, WSDL, and UDDI. *IEEE Internet Comput.* **2002**, *6*, 86–93. [CrossRef]
30. Foster, I. Service-oriented science. *Science* **2005**, *308*, 814–817. [CrossRef] [PubMed]
31. Yue, P.; Gong, J.; Di, L. Augmenting geospatial data provenance through metadata tracking in geospatial service chaining. *Comput. Geosci.* **2010**, *36*, 270–281. [CrossRef]
32. Cai, H.; Peng, C.; Deng, R.H.; Jiang, L. A novel service-oriented intelligent seamless migration algorithm and application for pervasive computing environments. *Future Gener. Comput. Syst.* **2013**, *29*, 1919–1930. [CrossRef]
33. Byun, E.; Kim, J. Dynagrid: A dynamic service deployment and resource migration framework for WSRF-compliant applications. *Parallel Comput.* **2007**, *33*, 328–338. [CrossRef]
34. Yang, C.; Raskin, R.; Goodchild, M.; Gahegan, M. Geospatial cyberinfrastructure: Past, present and future. *Comput. Environ. Urban Syst.* **2010**, *34*, 264–277. [CrossRef]
35. Huang, F.; Liu, D.; Tan, X.; Wang, J.; Chen, Y.; He, B. Explorations of the implementation of a parallel IDW interpolation algorithm in a Linux cluster-based parallel GIS. *Comput. Geosci.* **2011**, *37*, 426–434. [CrossRef]
36. Huang, F.; Zhou, J.; Tao, J.; Tan, X.; Liang, S.; Cheng, J. PMODTRAN: A parallel implementation based on MODTRAN for massive remote sensing data processing. *Int. J. Digit. Earth* **2016**, *9*, 819–834. [CrossRef]
37. Matthias, M.; Lars, B.; Johannes, B. Moving code in spatial data infrastructures—Web service based deployment of geoprocessing algorithms. *Trans. GIS* **2010**, *14*, 101–118.
38. Yang, C.; Goodchild, M.; Huang, Q.; Nebert, D.; Raskin, R.; Xu, Y. Spatial cloud computing: How can the geospatial sciences use and help shape cloud computing? *Int. J. Digit. Earth* **2011**, *4*, 305–329. [CrossRef]
39. Li, Z.; Yang, C.; Huang, Q.; Liu, K.; Sun, M.; Xia, J. Building model as a service to support geosciences. *Comput. Environ. Urban Syst.* **2014**. [CrossRef]
40. Tan, X.; Di, L.; Deng, M.; Fu, J.; Shao, G.; Gao, M.; Sun, Z.; Gong, W.; Ye, X.; Sha, Z.; Jin, B. Building an elastic parallel OGC web processing service on a cloud-based cluster: A case study of remote sensing data processing service. *Sustainability* **2015**, *7*, 14245–14258. [CrossRef]
41. Kussul, N.; Mandl, D.; Moe, K.; Mund, J.-P.; Post, J.; Shelestov, A.; Skakun, S.; Szarzynski, J.; van Langenhove, G.; Handy, M. Interoperable infrastructure for flood monitoring: SensorWeb, grid and cloud. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2012**, *5*, 1740–1745. [CrossRef]
42. Li, W.; Song, M.; Zhou, B.; Cao, K.; Gao, S. Performance improvement techniques for geospatial web services in a cyberinfrastructure environment—A case study with a disaster management portal. *Comput. Environ. Urban Syst.* **2015**, *54*, 314–325. [CrossRef]
43. Elsaghir, Z.; Elghety, H.S.; Abdelaziz, M. Data quality in software as a service implementation of public geographic information system. *IJECCE* **2012**, *3*, 1063–1066.
44. Li, H.; Zhu, Q.; Yang, X.; Xu, L. Geo-information processing service composition for concurrent tasks: A qos-aware game theory approach. *Comput. Geosci.* **2012**, *47*, 46–59. [CrossRef]

45. Wang, X.; Zhu, J.; Zheng, Z.; Song, W.; Shen, Y.; Lyu, M.R. A spatial-temporal QoS prediction approach for time-aware web service recommendation. *ACM Trans. Web* **2016**, *10*, 1–25. [CrossRef]
46. Khanouche, M.E.; Amirat, Y.; Chibani, A.; Kerkar, M.; Yachir, A. Energy-centered and QoS-aware services selection for internet of things. *IEEE Trans. Autom. Sci. Eng.* **2016**, *13*, 1256–1269. [CrossRef]
47. Sun, Y.; White, J.; Eade, S.; Schmidt, D.C. Roar: A QoS-oriented modeling framework for automated cloud resource allocation and optimization. *J. Syst. Softw.* **2016**, *116*, 146–161. [CrossRef]
48. Wang, Q.; Wang, J. Intelligent Web map service aggregation. In Proceedings of the 2009 International Conference Computational Intelligence and Natural Computing, Wuhan, China, 6–7 June 2009; Volume 2, pp. 229–231.
49. Yue, P.; Guo, X.; Zhang, M.; Jiang, L.; Zhai, X. Linked data and SDI: The case on web geoprocessing workflows. *ISPRS J. Photogramm. Remote Sens.* **2016**, *114*, 245–257. [CrossRef]
50. Zhou, Z.; Cheng, Z.; Ning, K.; Li, W.; Zhang, L. A sub-chain ranking and recommendation mechanism for facilitating geospatial web service composition. *Int. J. Web Serv. Res.* **2014**, *11*, 52–75. [CrossRef]
51. Tan, X.; Di, L.; Deng, M.; Chen, A.; Huang, F.; Peng, C.; Gao, M.; Yao, Y.; Sha, Z. Cloud- and agent-based geospatial service chain: A case study of submerged crops analysis during flooding of the Yangtze River Basin. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2015**, *8*, 1359–1370. [CrossRef]
52. Tan, X.; Di, L.; Deng, M.; Huang, F.; Ye, X.; Sha, Z.; Sun, Z.; Gong, W.; Shao, Y.; Huang, C. Agent-as-a-service-based geospatial service aggregation in the cloud: A case study of flood response. *Environ. Model. Softw.* **2016**, *84*, 210–225. [CrossRef]