Study of A Highly Efficient WPS Framework

Sun Wei
Geomatics College
Shandong University of Science and Technology
Qingdao, China
e-mail:83391860@qq.com

Liu Xiaoli, Li Chengming
Digital City & Smart City team
Chinese Academy of Surveying and Mapping
Beijing, China

Abstract— **The current WPS Platform exist some limitations such as insufficient extensible ability and low concurrent requests response efficiency. Aiming at these limitations, this paper proposes a new WPS frame based on task distribution, and uses task distribution system Gearman to reconstruct the WPS process core. The concurrent pressure contrast shows that using this new frame, it both can improve the extensible ability and the concurrent requests response efficiency, and make the WPS Service more open, extensible and efficiency.**

Keywords— WPS; task distribution; concurrent request; OGC

I. INTRODUCTION

With the continuous development of IT technologies in recent years, spatial information is playing an increasingly important role in people's daily life. However, owing to their limited variety of products, poor realtimeness, and incompatibility between data of different standards and formats, traditional geographic information system (GIS) service models are far from meeting users' requirements in acquiring, using and processing spatial information. In contrast, geographic information service models based on Web Services have broken free from the limitations on traditional service models and solved the issues concerning data sharing and interoperability. To standardize such data sharing and interoperability, experts around the world have jointly founded the OpenGIS Consortium (OGS), an organization committed to making standards for the sharing of geospatial data. The OGC Web Services team (OWS) is mainly responsible for setting standards for open spatial information Web services and has already published a series of geographic information service standards. Web Processing Service (WPS), as one of these stands, is mainly used for processing spatial data and providing certain forms of GIS processing services for service requesters via the Web.

Currently, the service capacity of most existing service platforms supporting WPS is limited and fixed. For instance, these platforms directly pack WPS-related services or provide only a handful of processing services. Despite many efforts to improve the expansibility of platforms by using dynamic languages to load processing functions, some problems still remain, such as difficult development and inflexibility of the platform framework. Moreover, with the soaring demands for geographic information services, WPS platforms faces the risk of being overloaded as the amounts of service requests from users located across the world will surely be astronomical. Under such circumstances, the standalone WPS platform is obviously incapable of handling service requests, thus limiting large user visits and efficient service provision.

This paper is supported by the geographic information public welfare special project (number: 201412003, 201512020)

In order to improve the expansibility of geographic information processing services and the responding efficiency under concurrent requests, this paper, by using ideas of parallel computing and service as reference, proposes an efficient task-distribution-based WPS framework. The task distribution software Gearman is used to reconstruct the WPS processing kernel based on the job server so as to achieve balanced load sharing among multiple back-end processes and the flexible expansibility. In this way, the efficiency in handling concurrent requests and the expansibility of WPS service platforms can be improved.

II. WPS STANDARD AND WPS SERVICE PLATFORM

A. WPS Standard

WPS is one of the latest standards developed by OGC. It is designed to provide the client with a series of interfaces required by GIS processing services via the Web, by which users can perform various operations on spatial information. WPS enables us to publish all GIS processing functions as a Web service, including all of its input and output parameters and how it is triggered.

WPS adopts XML for communication between the client and server. WPS service defines three basic operations, providing different levels of services such as information query and data processing. These three operations are GetCapabilities, DescribeProcess and Execute. GetCapabilities is used by the service user to request and obtain the metadata description of the available data processes. DescribeProcess is used for obtaining a description of a process including its inputs, outputs and format. Execute calls processes according to the parameters provided by the service user at the time of service request, and returns the outputs. The common calling procedures of WPS first obtain the identifier of the needed parameters and the description of inputs, and then call services to analyze the inputs and computing data. The data required by the processing can be either data sent directly by the service user to the server via the Web or the existing data on the server. WPS standard provides an identification mechanism for spatial reference data and standardizes input parameters and outputs.

© 2016. The authors - Published by Atlantis Press

B. *WPS Service Platform*

The advent of WPS has made possible the sharing of functions. Therefore, WPS and its implementation platforms are becoming a hot topic for GIS researchers around the world. Among the many studies abroad, 52 North WPS, a Java-based open source project by International Institute for Aerospace Survey and Earth Sciences (ITC) seems to be the most practical and mature WPS service platform at the moment. In China, there haven't been literatures documenting any mature WPS platforms. But the many attempts in service processing based on WPS standard have laid a solid foundation for the further implementation of WPS platforms. Most existing studies in China still focus on the implementation of WPS processing services and rarely touch upon upgrading expansibility of the WPS platform and its capacity of handling concurrent requests.

III. DESIGN AND IMPLEMENTATION OF THE TASK-DISTRIBUTION-BASED WPS FRAMEWORK

A. *Design Thinking*

The main task of a WPS platform is to provide users with standard geospatial data processing services. In essence, the platform processes spatial data and returns outputs to users. Therefore, the many studies on parallel computing may shed some light on how to upgrade the expansibility of the WPS platform and its capacity of handling massive concurrent requests.

Parallel computing is a process of dealing with huge amounts of computation by using multiple types of computing resources. In the era of information explosion, the computing environment today is expected to assign workload and utilize computing resources more efficiently. Indeed, the computing power of a standalone server has multiplied than it was years ago. However, when dealing with a large amount of computing tasks, the server executes them in an orderly first come, first served manner, which inevitably prolongs the overall computation and hampers the responding efficiency. So, we decide to use parallel computing to solve this problem, that is, assign this large amount of computing tasks to other computing resources. In this way, constraints of storage and processing capacity on standalone computing resources can be solved.

As more and more applications of geographic information become popular with the masses, the number of Web-based geographic information services is soaring, rendering the existing standalone geographic information service platforms incapable of handling these huge numbers of concurrent service requests. It is possible to distribute these service requests evenly among multiple geographic information processing service platforms.

In practical work, however, this deployment operation itself consumes large amounts of resources and creates multiple interfaces, making it difficult for users to choose. So, it is still necessary to improve the expansibility of the WPS platform and its capacity of handling concurrent requests so that multiple tasks can be distributed in a rational manner and processed within one platform.

To this end, this paper, inspired by parallel computing, uses the open source task distribution software Gearma to reconstruct the processing kernel of WPS. The modified kernel is able to achieve balanced load sharing when handling concurrent service requests, and thus improve the responding efficiently of WPS.

B. *Design and Implementation*

Generally, the service procedures of a WPS service are as follows:

- The server of the platform receives a *GetCapabilites* request sent from the client and then returns the information about the services it provides from which the user can choose;
- When the user selects a specific process, the client sends a *DescribeProcess* request. The server receives this request and then returns input and output information needed for the process interface.
- Upon receiving the user's *Execute* request, the server uses the WPS engine to parse and transfer the parameters from the user to specific WPS processing modules for analysis and operation. The output results are sent back to the user.

During task execution, if multiple WPS services are to be distributed and processed, it is required to separate service presentation and service processing functions so as to ease service distribution between them. In this way, the processing and responding capability is increased while the convenience of services to users is still ensured. Based on the job server of Gearman, the WPS framework proposed in this paper is shown as follows.

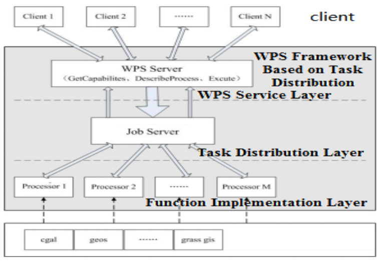

Fig. 1. WPS Service Frame based on Task Distribution.

It adopts a three-layer design. The top layer is the WPS service layer, that is, the presentation layer of WPS services. It is responsible for implementing the three interfaces of WPS services. The presentation layer receives requests from the client and returns service results outputted in the background. The bottom layer is the function implementation layer of WPS services, where specific task processing function modules compute and process services. Processing functions can be developed through the use of third-party libraries supporting the implementation of processing modules compiled by any language. The middle layer is the task distribution layer, where the job server distributes services, transfers them to the function implementation layer, and then sends the outputs to the service presentation layer.

In the figure, the solid arrow indicates the service transmission, the direction of the arrow points out where the service flows to, and the size of the arrow indicates the number of service requests. Service requests sent concurrently from multiple clients (or a single client) converge in WPS service layer, which parses them according to the WPS standard and then sends the parsing results of all these concurrent services to the job server. The job server distributes these service tasks to different processors in the function implementation layer according to service content. The generated outputs are returned to the job server in real time and then sent to the WPS service layer, where the outputs are parsed and fed back to the client. During the information feedback, the information from different processors can be transmitted either synchronously or asynchronously.

The implementation of each layer is described as below.

1) WPS Service Layer

The WPS service layer mainly implements three interfaces, namely GetCapabilites, DescribeProcess and Execute. As for implementation, corresponding service contents are sent to the job server according to WPS Profile (description configuration information of the WPS Server) and Processor Profile (description configuration information of a process).

WPS Profile and Processor Profile are described using the YAML language. YAML is a data serialization format easily recognizable by computers and also a data description language similar to XML. With good interaction with scripting languages, YAML is often used for the implementation of configuration files. The section below will give an example of WPS Profile.

```
# WPS Profile
global: #
encoding: utf-8
version: 1.0.0
serverAddress: http://127.0.0.1/newmap/ogc/wps
lang: zh_CN
identification: #
title: NewMap WPS \
abstract: NewMap OGC WPS
fees: none
accessConstraints: none
keywords:[NewMap,NewMapServer4,OGC,WPS,GIS]
provider: # providerName: NewMapServer WPS
providerSite: http://www.newmapgis.com
individualName: newmap
positionName: newmap
addressDeliveryPoint: beijing
addressCity: beijing
addressCountry:China
addressAdministrativeArea: none
addressPostalCode: 100830
addressElectronicMailAddress:newmap@casm.ac.cn
phoneVoice:010-63880552
phoneFacsimile:010-63880552
```

2) WPS Service Layer

Task distribution layer is mainly implemented based on the task distributor of the open source software Gearman, i.e. Job Server. At present, Gearman is only supported on Linux. The implementation of functions is carried out by the daemon process Gearmand. Because Gearman provides the client secondary development interface (Client API) and the server secondary development interface (Server API) in various languages such as C/C++, Java, PHP, Python and Ruby, it is possible to use different languages to implement the interfaces connecting Gearman to the front-end WPS service layer and the back-end function implementation layer. This will not only make the development much easier by dividing the development work into smaller tasks but also greatly improve the expansibility of the WPS framework. In this paper, we use Python Client/Server API to build the links to the function implementation layer. Taking analysis operation for Buffer as an example, we parse the input parameters from the HTTP request according to the buffer.yaml description. At the same time, we call the Gearman Python Client API and use Gearman communication protocols to implement the Execute interface.

```
# input parameter
data = (input_geom, buffer_dis, output_format)
# construct Gearman client
gm_client = gearman.GearmanClient(['127.0.0.1:4730'])
# submit request
completed_job_request = gm_client.submit_job("buffer", data)
# check
check_request_status(completed_job_request)
```

3) Function Implementation Layer

Function implementation layer mainly implements the processor handling specific processing tasks. As mentioned previously, the Python-based Server API is used to develop the interface. Function algorithms are developed by using the third-party C-language-based open source library GEOS. In Python, ctypes is used to call the C language interface. The codes for constructing a processor are shown below.

```
# construct worker
gm_worker = gearman.GearmanWorker(['127.0.0.1:4730'])
# define wps
def task_listener_buffer(gearman_worker, gearman_job):
#register  wps
gm_worker.register_task('buffer', task_listener_buffer)
# wait
gm_worker.work()
```

IV. Experiment and Analysis

Taking the Buffer service as an example, a comparative stress test is performed.

Test subjects: the 3-layer task-distribution-based service platform vs. the traditional WPS platform handling service requests in order, i.e. the single-layer WPS service platform without the task distributor. Buffer algorithms of the two platforms are both implemented by using the third-party open source library GEOS.

Test data: simple polygons described using GeoJSON. The data content is:

{"type":"Polygon","coordinates":[[[492499.27669284556,2553443.80582409],[500907.33493374416,2551609.320389711

7],[494639.5096996198,2545647.2427279837],[492499.27669284556,2553443.80582409]]]}

Test environment: three groups of 10, 100, and 1,000 concurrent service requests are sent to each WPS platforms, respectively.

Test methods: the buffer radius is set as 1,000 meters. The traditional scheme adopts a WPS service responding to requests in order while the proposed 3-layer WPS framework uses 20 processors to handle service requests concurrently.

Results are shown in the figure below.

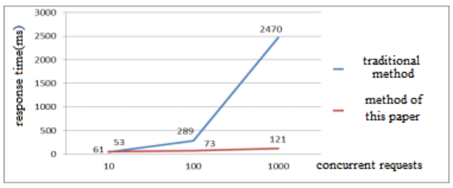

Fig. 2. Comparison of WPS concurrent request response between traditional method and the method of this paper

It can be seen from Figure 2 that when handling 10 concurrent service requests, the response time of the traditional method (53ms) is slightly smaller than that of our WPS platform (61ms); when the number of concurrent requests increases to 100, the response time of our WPS platform (73ms) is relatively stable while that of the traditional method grows significantly (to 289ms); in case of 1,000 concurrent requests, the gap in response time between the two platforms grows to be surprisingly large. It takes the traditional WPS platform nearly 2.5s (2470ms) to return the output results of the 1,000 service requests. The long waiting time means bad user experience. The response time of our 3-layer WPS platform, by contrast, is only 121ms, hence a significantly better efficiency.

The findings from the test results above are quite obvious. In handling a few concurrent service requests, the traditional WPS platform has the power to respond fast enough, whereas the proposed 3-layer is a bit slower owing to the additional time incurred by network communication in the task distribution layer. However, as the number of concurrent service requests increases, the response time of traditional WPS platform grows linearly because it has to execute the services in order. Meanwhile, our proposed WPS platform assigns the huge amounts of concurrent service requests to a number of processors. Because the number of service responses to be handled by each processor remains generally stable, the whole WPS platform maintains a high responding efficiency.

From the results and analysis of the experiment, we can infer that the processing platform constructed by the task-distribution-based WPS framework will be superior and responding even faster when it comes to providing services upon concurrent requests and long-running processes (i.e. time-consuming tasks like image re-sampling and data re-projection).

V. Conclusion

WPS is one of the standards released by PGC. It has solved many constraints on the existing GIS processing services and enabled users to analyze various types of spatial information via the Web. Although common WPS service platforms are able to implement a large amount of processing services, their services and functions are limited, fixed and suffer poor expansibility. Moreover, as the user base of geographic information services grows, WPS platforms are faced even greater pressure posed by numerous concurrent services. The low responding efficiency of standalone platforms has limited the volume of visits and the provision of highly-efficient concurrent geographic information processing services. Through analysis of existing WPS service platforms, this work adopts certain ideas of parallel computing and service, proposes an efficient WPS framework based on task distribution, and uses Gearman to reconstruct the processing kernel of WPS based on the job server. Consequently, it not only supports the implementation of geographic information processing and function development interface under various languages, improves the service expansibility and difficulty of development, but also separates WPS service presentation and processing functions, thus boosting the service efficiency. The balanced load sharing between processes of multiple back-end services is achieved through task distribution, which has upgraded the responding efficiency in cases of concurrent service requests. At last, a comparative stress test is given to verify the feasibility of the proposed framework and the capacity of the WPS platform built with it.

Although the proposed task-distribution-based WPS framework has improved the expansibility of geographic information processing services and the efficiency in handling concurrent requests, however, given the limited time and technical capabilities, we implement the task distribution function thereof by using the open source software Gearman. With the continuous development of parallel computing and GIS processing services, we believe that the task distribution development framework based on universal message queue will also be able to well handle task distribution. Therefore, we will focus on developing an efficient WPS framework based on universal message queue, strive to improve the responding efficiency through independent research and development so as to provide users with open, convenient and efficient geographic information processing services via multiple channels and at various levels.

References

[1] Open Geospatial Consortium，Inc．About OGC: OGC Vision, Mission, Strategic Goals [EB/OL],http://www.opengeospatial.org/ogc/vision，2015.

[2] Open Geospatial Consortium，Inc．Standards：Web Processing Service [EB/OL],http://www.opengeospatial.org/standards/wps，2015.

[3] Open Geospatial Consortium，Inc．OpenGIS Web Processing Service Specification （version0.4.0,2005） [EB/OL],http://portal.opengeospatial.org/files/?artifact_id=13149&version=1&format=pdf，2015.

[4] Gearman. [EB/OL], http://www.gearman.org/，2015

[5] 52 North WPS[EB/OL], http://52north.org/,2015

[6] The Future of GIS：Open Source GIS[EB/OL],http://opensourcegis.org/，2011.

[7] YAML[EB/OL],http://www.yaml.org/,2015