International Journal of u- and e- Service, Science and Technology
Vol.8, No. 12 (2015), pp.23-34
http://dx.doi.org/10.14257/ijunesst.2015.8.12.03

A Distributed WebGIS Platform based on SaaS Architecture

Weng Yu, Xiaodong Jia and Yuan Jie

College of Information Engineering, Minzu University of China
dr_wengyu@126.com

Abstract

The emergence of cloud computing has changed the software development and deployment, providing users with great convenience. But even in the cloud computing environment, software development efficiency is particularly low. In order to improve the efficiency of software development, and make software release easier, we designed a layered service computing platform. With the help of basic services, the platform makes software development and publishing simple and efficient. To demonstrate our platform, we described the process of building a simple WebGIS system in this paper. This process represents the software development process in our platform. Experiment shows that our work is not only available but also simplifies the process of software development.

Keywords*: Services Computing, WebGIS, Service representation, SaaS*

1. Introduction

Since the emergence of cloud computing, it has brought great changes to the Internet community, and has greatly facilitated the users, because they no longer need to manage their own servers. But even with the cloud, we still need to develop the vast majority of software features in the software development, and also in the process of software release. In most cases, we need to build our own publishing environment, which resulting in a waste of resources. Therefore, an efficient system that can help us reduce the workload of software development and makes software release more convenient is necessary.

In this paper, we propose a service computing platform—SCP, in our platform, users only need to combine basic services that they need when they are developing software, and then submit these combined service to our platform. Platform will automatically publish the software, and automatically optimize resource allocation when services are running. Our platform provides users with a more efficient, flexible and simple approach to software development. At the same time, SCP has very good scalability, and support for complex service combination and service pipeline.

In our design process of SCP, the results of previous research gives us a lot of help, mainly in the following three aspects: First, the idea of cloud computing and some open source software in cloud computing, we built the SCP on them. Second, in terms of service representation and resource scheduling, the results of Mike P. Papazoglou *et al*. [1] and Lei Xu *et al*. [2] do us a great favor. Their method is very suitable for SCP. Of course, we have made some improvement. Third, in the aspect of service computing, Wei-Tek Tsai *et al*.[3]have done some researches, especially in cross-cloud service computing.

But our research differs from the work they do, we made some improvement in software development by adding service computing to cloud computing. In terms of service representation, our strategies focus more on service combination and service pipeline, so services in our platform are easily to be combined and expanded. In terms of resources schedule, our algorithm is dynamic which facilitated the users. And experiments have proved that our SCP is more suitable for building a scalable, efficient and flexible in size service computing platform in an organization or a company.

ISSN: 2005-4246 IJUNESST

Other parts of the paper are organized as follow. The second part describes the related work of predecessors. The third part is the description of the problem and the structure of the SCP. In the fourth part, we detailed the specific functions and realization of each part of the SCP. The relevant experiments are in the fifth part, and in the last part, we made a summary and briefed our future research plan.

2. Related Works

Cloud computing and service-oriented computing has gained rapid development in the past decade, especially in cloud computing, which is changing the entire computer industry, many people have done researches in these two areas [4].

Rajkumar Buyya *et al.* made an overview of cloud computing and described the architecture of cloud platform. They summarized the future research direction of DataCenter. At the same time, they described the vision of computing in 21th century [5]. Michael Armbrust, who gave a definition of cloud computing in their paper, explained the differences between cloud computing and traditional computing, they also detailed the opportunities and challenges faced by cloud computing [6]. Mike P. Papazoglou made an authoritative narrative on Service-Oriented Architecture—SOA, and detailed the structure of service computing [7]. Yi Wei and M. Brian Blake told the link between SOA and cloud computing, they pointed out that the two can be combined. Their idea inspired us, but they did not put it into practice [1]. Lionel Seinturier *et al*. proposed FRASCATI platform, which is a component-based platform that can be formulated for developing distributed service-oriented components, these components can also be combined into new applications. They use XML as configuration file, and increased some auxiliary processes in their platform [8]. WenAn Tan *et al*. proposed a highly available workflow scheduling algorithm, this algorithm made a compromise between execution time and resource consumption [9], but we focus on not workflow but service pipeline.

At the same time, we did some research on several internal structure of the cloud computing, such as resource scheduling [10, 11], virtual machine scheduling. And a number of other platforms also get our attention, so we do some researches on them [12-16].

Anton Beloglazov *et al*.designed a hierarchical computing architecture and a new virtual machine allocation algorithm which effectively reduce energy consumption of DataCenter [17]. Daniel Warneke and Odej Kao designed Nephele—a cloud-based platform for data processing framework, they used directed acyclic graph—DAG to describe the contact between jobs. And they designed a job scheduling algorithm, by contrast with Hadoop, Nephele indeed works [18]. Gunho Lee *et al*. reduced multi-job execution time by adjusting the execution order of jobs, meanwhile their algorithm have taken fairness of resource sharing in to account[19]. Fei Xu *et al.* compared the different performance evaluation model of DataCenter, studied the Live VM migration and management of VM performance, they also pointed out the research direction of DataCenter performance evaluation [20].

Lei Xu *et al*. proposed a cloud data center resource scheduling strategy which called Smart-DRS [2], this strategy is divided into two steps, the first step is to schedule virtual machines on the physical machine, and the second step is to turn off the physical machine [21]. They used projection of vector to determine the resource needs of applications, and then decide how to schedule virtual machines. But they only discussed the three-dimensional vector, which can only judge three kings of resources, but we have adopted a dynamic resource scheduling strategy, and before scheduling we use some strategies to determine the software release. Wei-Tek Tsai *et al.* proposed a layered, service-oriented cloud service architecture, this architecture applied SOA to cloud platform which enables applications to run across the clouds, they added intermediate

layer between cloud platform and application services in order to combine all kinds of clouds [3]. Our platform is to provide services, but also service-oriented.

3. Problem Description and Design Structure

There are two problems that we faced in cloud computing: First, software development is difficult and slow; second, not only software release is complex, but also a serious waste of resources after release.

To simplify software development and publishing, we have designed SCP. It provides users with a number of basic services. When user developing software, they only have to select basic services that provided from the SCP according to the needs of the software, and then combine these services together. Services can be used alone or as a service pipeline. After that, user defines the service data interface and upload the data. Finally, as long as user submits this application to the SCP, SCP will automatically complete the release and execution of the software.

SCP is a layered service computing framework, from top to bottom it is divided into five logical layers, they are: the application layer, static service layer, dynamic service layer, the virtualization layer and infrastructure layer. Figure 1 shows the overall architecture of the SCP.

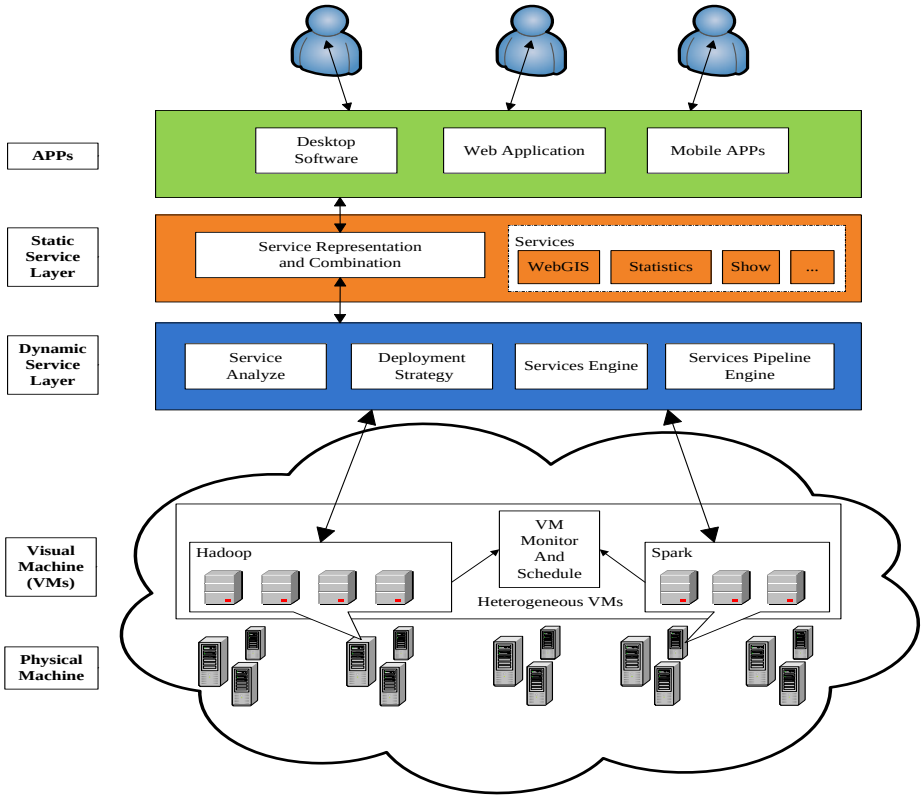

Figure 1. Overall Architecture of the SCP

The top layer is user-developed software or applications, which is related to specific requirement, so it will not be covered, but there is one thing need to be pointed out, these applications run on virtual machines—VMs.

The second layer is static service layer. This layer includes service representation and service combination. In order to get modularity of services, we defined a set of service config file to represent services in SCP, which was stored in format of XML.A service

config file defines parameters and returns of a service, and this file is a link between service and data. After users have finished these service config file, they should submit them to the next layer.

The third layer is a dynamic service layer, which consists mainly two parts, they are publishing model and service execution engine. Service publishing model will analyze submitted service config file firstly. In this process, we use an algorithm which likes a part of PageRank [22]. There are calls between services, so these services will form a map, the vertex is the service, and the edge is the calls between services. We set the weights of edges based on the frequency of the calls and the amount of data transmitted between the services. After the service analysis, the services will be deployed to the appropriate virtual machine (our VMs is heterogeneous, they have different number of CPU, different size of memory and so on). The other part of dynamic service layer is the service execution engine, due to our services are modular, so these services can be combined into a service pipeline or other service flows. Execution of these service flows is controlled by the service execution engine and service pipeline engine. Service Engine contains a scheduling algorithm, which is used to schedule services.

The fourth layer is a virtualization layer, and it contains the virtual machines and monitoring procedure. The virtualization layer contains a variety of heterogeneous virtual machines. They have different number of CPU, different sizes of memory, running different software, some of them running Hadoop MapReduce, some running Apache Spark, so they can provide external massively parallel computing power besides running basic services. Virtual machine monitoring procedures collect data for scheduling algorithm by monitoring there resource usage of virtual machines.

The last layer is the infrastructure layer. This layer contains all the hardware, including servers, redundant array of independent disks, and network equipment.

4. Details of SCP

In this chapter, we will detail the design process of the SCP, the overall presentation will be divided into five parts—service represents and combination, service pipeline, software and services publishing, run-time monitoring and scheduling, SCP construction and application development.

4.1. Service Represents and Combination

We do service representation and combination for two reasons: First, improving the reusability of individual service. In order to do this, we have defined a standard in the SCP, this standard abstract all services into a unified format, which can help us define the interface between services. If there is no this standard, service using will become very complicated. Secondly, service scan be combined, even if service can be used alone. Combination of services can make the services become more powerful and it can convenient the process software development.

Service represents and combination are defined by different XML files. Service represent file define basic service, and service combination file define the relationship of services, in other words, define software.

In SCP, we use an XML file to represent the service, and the file is called service represent file. Most software is a combination of SCP services, and services are the foundation of the software, so the services are very important. SCP only recognizes service represent file, which defines all the information of a service. When adding a new service to SCP, we only need to submit service represent file and the corresponding resource files to SCP. The basic items contained in service represent file are listed in Table 1.

Table 1. Items in Service Represent File

Item	Description
Service ID	String format
Service Name	String format
Service Description	String format
Service Class	String format
Service in parameters	String format, Key-Value type, can be multiple
Service out parameters	String format, Key-Value type, can be multiple
Service Type	May be: compute, display, IO
Service Data Size	In MB

These properties are the most basic properties of services, and service represent file must contain these properties. Among them, the in and out parameters are Key-Value format because it is good for services combination, but the value is not a simple String, it can be a structured String. Service type describes the function of a service. Computing services just do the calculation. It receives parameters, calculates and returns the result after calculation. Display service shows the data in an apposite way. IO services read or write the files.

Service represent file only represent the services in SCP, in order to develop software, we need service combination file.

Service combination file define the combination of services. It is a XML file too. The items contained in service combination file are listed in Table 2.

Table 2. Items in Service Combination File

Item	Description
Software ID	String format
Software Name	String format
Service ID	String format
Service in parameters	It should be same as the in parameter of the corresponding service
Next service ID	Can be multiple, it is used to combine services
Service out parameters	It is an map

Software is defined by several service combination file, the software ID of these files are same, these file make a link between the parameters of these services. Service out parameters is a map from in parameter of current services to out parameter of next service, if there is no next service, it will be out parameter of the original service

4.2. Service Pipeline

In order to improve the execution performance of service flows, we use the strategy of service pipeline. It is particularly useful for data-intensive applications. In general, we combine services of a complex application into a service pipeline, which is similar to instruction pipeline. Service pipeline can speed up the execution of services.

Service line consists of three parts, they are: segment system, pipeline monitoring system and conflict management system. The core is the segment system, services executed in segment system. Pipeline monitoring system provides monitoring service for the segment systems, while providing information for conflict management systems.

Conflict management system is to deal with the conflicts occurred in the running of pipeline, and to improve performance by changing the segment system. The detail of each part will be described as follows.

Segment system is a container for service pipeline, which is the basis of the pipeline. Each node in segment system contains a service. Segment system consists of three parts: the input area, computing area and output areas. Computing area is our service, input area gets the input parameters, output area stores the result of computing are. Input area and output area are in memory, and these two areas have at least two child regions for cache. Figure 2 depicts the structure of the segment system.

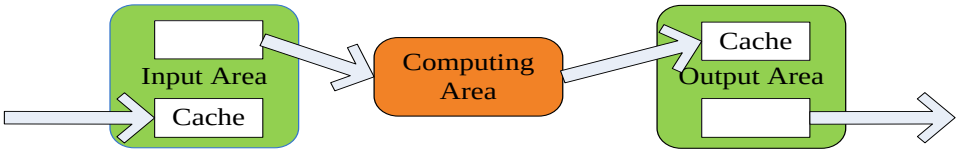

Figure 2. The Structure of the Segment System

In order to prevent the stop of service pipeline, we set up redundant buffer in input are and output area, the buffer can insure that computing area always has data to deal with. The size of buffer is fixed, but we can add new buffer if the speed of computing process is fast enough.

Pipeline monitoring system monitor the status of service pipeline and provide information for conflict management system. It works with conflict management system to prevent the stop of service pipeline, thereby improving the processing efficiency of the service pipeline. Information it has to be monitored is the size of input area and output area and the number of buffers. By obtaining this information, we can know the execution status of segment system.

Service pipeline is similar to the instruction pipeline, so the conflict processing is very important. Data conflicts are most likely to occur during the execution of the pipeline, for example, two adjacent services have to write data to the same block. We use several ways to deal with data conflicts. First, add bubbles, that is to say, add an empty task in the pipeline, so that the previous node has sufficient time to execute. Second, adding data path, this method is suitable for the case of two segment use the same data block. We add a data path between two segments, so that the data of one segment can be sanded to another segment directly, without further IO operations.

4.3. Software and Services Publishing

When publishing software and services, we use some simple strategies, which will achieve load balance of each VM.

When the service is released, based on the description of the service, SCP will allocate a suitable VM for it, and the VM is responsible for handling users' requests. SCP will assign IP addresses, and record the mapping information between IP and service ID.

When the software is released, user submits the services represent files and service combination files to SCP. Based on the service combination files, SCP will generate a directed acyclic graph to analyze services. Vertices in the graph are the services. The weights of edges are defined by transferred data size and frequency of the calls between the services. SCP will decide whether to deploy this service to an existing VM or a new VM by analyzing the weights of edges and the number of edges. The software information will be stored on a specific virtual machine too.

4.4. Run-time Monitoring and Scheduling

Table 3. Resource Schedule Algorithm

Resource schedule algorithm

```
Input VMs[];                // resource usage of VMs, get from monitoring program
Input Machines[];           //resource usage of physical machine, get from monitoring program
foreach(vm in VMs){
    if(vm> 70%)                                  // resource usage is more than 70%
    {
        aVM = FindAEasyVm();                     // find a VM that resource usage is low
        /* find a Service that resource usage is high from VM*/
        aService = FindAServiceFrom(vm);
        moveTo(aService, aVM);                   //move aService to aVM
    }
}
foreach(machine in Machines){
    if(machine > 70%)                            // resource usage is more than 70%
    {
        aMachine = FindAEasyMachine();       // find a VM that resource usage is low
       /*find a VM that resource usage is high from machine*/
    aVm = FindAVmFrom(machine);
        moveTo(aVm, aMachine);                   //move aVm toaMachine
    } else if(machine < 20%) {
        foreache(vm in VMFromMachine)        //for all the VMs in this machine
        {
            /*find a machine that resource usage is low*/
        aMachine = FindAEasyMachine();
            if(aMachine == null)                 //if there is no aspirate machine
              {
                    startANewMachine();          //turn on a new machine
              } else
              {
                    moveTo(vm , aMachine);       //move this VM to aMachine
              }
        }
    }
}
```

After the software is released, at run time, SCP will allocate and schedule services dynamically to meet the load balance of VM. Virtualization layer has a monitoring program to monitor resource usage of virtual machines and the physical machine, the information will be regularly sanded to the handler, and the handler will make the appropriate action. In the virtual machine and the physical machine monitoring, we use open source software, it will monitor the use of CPU, memory, IO and other resource usage of VMs and physical machines. After analysis, the handler will make the

appropriate action, such as: dispatching one service from one VM to another VM, adding or removing a VM, or if resource usage of one physical machine is low, it will move all the VMs of this machine to another machine, and turn off this one. In scheduling of resources, we used an algorithm, and it is described as Table 3.

4.5. SCP Construction and Application Development

SCP is a layered service computing framework, and it was built layer by layer. First we built the infrastructure layer. This layer is relatively easy to build, so not repeat it here. Then we constructed the virtualization layer, we have established heterogeneous VMs on physical machines, number of CPU, memory size, and disk capacity are all different. Then we installed different software, such as Hadoop MapReduce, Spark, OpenStack, etc. Finally we installed a resource monitoring software on it. Above virtualization layer is a dynamic service layer, we installed software release system and service execution engine. In the static service layer, we released a service library which contains basic services. Services of this library can be dynamically added and removed. Static service layer also contains software we have released.

Everything we do in the SCP is to make software development more convenient. As mentioned above, the application development of SCP is very simple, just three steps: First, select the required services. Secondly, combine these services together. Last, define the data interface between services and submit service combination file to SCP. In the next chapter, we will use an experiment to prove.

5. Experiment

In this section, we will describe the process of building a simple application—WebGIS with SCP. WebGIS is a geographic information system that showed in web pages. The reason why we use WebGIS is that some data of ours have geographical attributes. In order to view these data by region, we decided to display them on a map. We will describe the processes of building WebGIS application in following sections—service development, service combination and software release.

In service development section, we have to develop two services. A data access service is used to read data from HBase [23]. The other is display service, which is used to display data on a map. The parameter in service represent file of data access service are: database IP, table name, column family list and column names to query, the start key and the end key. The return value of data access service is a list of data. Service represent file of data access service is shown in Table 4. Another service is the display service. We used OpenLayers, which is a JavaScript library to build WebGIS system. The parameters of display service are: title to display, a data list, in which each piece of data includes at least the following information: latitude and longitude, the name of one point and the data to be showed in this point. What need to be pointed out is that each item of the data should be Key-Value format, but the value part can be text, image or video. Display service returns no data, because it only shows the data in web pages.

In the service combination section, just to combine the above two services, because there is no other services in this simple WebGIS application. Service combination file primarily configs the data exchange interfaces between the two services, and it is shown in Table 5:

In software release part, there are only two services, so we using the default publishing method. We deployed two services on two virtual machines, and then the software is released on another virtual machine. The display service was deployed in Tomcat and the display service was used as a separate frame which embedded in the software's HTML page. Data access service deployed on another virtual machine, and it only providing data to other services.

Table 4. A Service Represent File

Service represent file of data access service

```
<?xml version=’1.0’ encoding=’utf-8’>
<service>
    <31erviced>1</31erviced>
    <serviceName>
       HbaseReader
   </serviceName>
    <serviceDescription>
       Get data
 </serviceDescription>
    <serviceClass>edu.muc.hbase.GetData
  </serviceClass>
    <inputParam>
    <ip>10.10.167.210</ip>
    <tableName>
        Architecture
     </tableName>
    <startKey>a</startKey>
    <endKey>z</endKey>
    <columns>
          <familyName name = “info”>
                <column>lag</column>
                <column>lat</column>
          </familyName>
          <familyName name =”feature”>
              <column>text</column>
              <column>images</column>
              <column>videos</column>
          </familyName>
                ……
          </columns>
    </inputParam>
    <outputParam>
          <dataList>
          <data rowKey = “”>
              <familyName name = “”>
                  <column></column>
                  <column></column>
                  <column></column>
              </familyName>
                      ……
              </data>
              <data></data>
                      ……
          </dataList>
    </outPutParam>
    <servideType>IO</servideType>
    <dataSize>1MB</dataSize>
</service>
```

Table 5. A Service Combination File

Service combination file

```
<?xml version='1.0' encoding='utf-8'?>
<serviceCombine>
<appNum>1<appNum>
<appName>WebGIS</appName>
<serviceID>1</serviceID>
      <inputParam>
      <ip>10.10.167.210</ip>
      <tableName>architecture</tableName>
      <startKey>a</startKey>
      <endKey>z</endKey>
      <title>Architecture GIS Service </title>
      <columns>
            <familyName name = "info">
                <column>lag</column>
                <column>lat</column>
            </familyName>
            <familyName name = "feature">
                <column>text</column>
                <column>images</column>
                <column>videos</column>
            </familyName>
            ......
      </columns>
</inputParam>
      <outputParam>
      <title>ArchitectureGIS Service </title>
      <dataList>
          <data>
            <lag>lag</lag>
            <lat>lat</lat>
            <name>nationality</name>
            <columnFamily name = "feature">
                <cloumn>
                   <name>text</name>
                   <value></value>
                </column>
                <cloumn>
                   <name>images</name>
                   <value></value>
                </column>
                <cloumn>
                    <name>videos</name>
                    <value></value>
                </column>
                </columnFamily>
                   ......
            </data>
      <dataList>
</outputParam>
<nextServiceID>2</nextServiceID>
</serviceCombine>
```

After the software is released, it can be accessed from browser. The software is shown in Figure 3. After development of this application, we have data access service and WebGIS display service in SCP. Because the two services are data-independent, other applications can use these two services too, which is the advantage of SCP. As long as the service has been developed once, it can be used directly in other applications without duplicated development. We added a service query application in SCP, so users can search services. The more services in SCP, the more powerful SCP will be.

Figure 3. The WebGIS System

6. Conclusion

In this paper, in order to solve the problem of software development is difficult, inefficient and software release is complex, we propose a layered service computing platform—SCP, and described its overall structure and details of each part, including: service represents and combination, service pipeline, software and services publishing model, run-time monitoring and resource scheduling, and the last is construction and software development of the SCP. Our experiments prove that our platform is not only available, but also enhance the efficiency of software development.

However, our platform still has some defects, such as: When we are developing software, we need to write service represent file and service combination file manually, rather than using graphical design tools to do it. Software release and deployment section also not smart enough. We plan to solve these issues in the near future, and we are planning to add Container into our platform in order to improve the utilization of computer resources.

Acknowledgments

In the design and implementation of this platform, the members of our group helped us a lot in development and testing, and they gave some great suggestions. They are: Chengjie Li, Wenyi Cheng, Peng Guo, Ling Zhong, Ruiqi Wang, Haitao Ma and Wencheng Bai. Thanks them here.

References

[1] W. Yi and M. B. Blake, “Service-Oriented Computing and Cloud Computing: Challenges and Opportunities”, Internet Computing, IEEE, vol. 14, **(2010)**, pp. 72-75.
[4] H. Khazaei, J. Misic and V. B. Misic, “A fine-grained performance model of cloud computing centers”, Parallel and Distributed Systems, IEEE Transactions on, vol. 24, **(2013)**, pp. 2138-2147.
[5] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic, “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility”, Future Generation Computer Systems, vol. 25, no. 6, **(2009)**, pp. 599-616.
[6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz and A. Konwinski, “A view of cloud computing”, Communications of the ACM, vol. 53, **(2010)**, pp. 50-58.
[8] L. Seinturier, P. Merle, R. Rouvoy, D. Romero, V. Schiavoni and J. B. Stefani, “A component - based middleware platform for reconfigurable service - oriented architectures”, Software: Practice and Experience, vol. 42, **(2012)**, pp. 559-583.
[9] T. Wenan, S. Yong, L. Ling Xia, L. GuangZhen and W. Tong, "A Trust Service-Oriented Scheduling Model for Workflow Applications in Cloud Computing”, Systems Journal, IEEE, vol. 8, **(2014)**, pp. 868-878.

[12] H. Topcuoglu, S. Hariri and M. Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing", Parallel and Distributed Systems, IEEE Transactions, vol. 13, **(2002)**, pp. 260-274.
[17] A. Beloglazov, J. Abawajy and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing", Future Generation Computer Systems, vol. 28, **(2012)**, pp. 755-768.
[18] D. Warneke and K. Odej, "Exploiting Dynamic Resource Allocation for Efficient Parallel Data Processing in the Cloud", Parallel and Distributed Systems, IEEE Transactions, vol. 22, **(2011)**, pp. 985-997.
[20] X. Fei, L. Fangming, J. Hai and A. V. Vasilakos, "Managing Performance Overhead of Virtual Machines in Cloud Computing: A Survey, State of the Art, and Future Directions", Proceedings of the IEEE, **(2014)**.
[21] J. Baliga, R. W. A. Ayre, K. Hinton and R. Tucker, "Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport", Proceedings of the IEEE, **(2011).**
[22] B. C. Csáji, R. M. Jungers, and V. D. Blondel, "PageRank optimization by edge selection," Discrete Applied Mathematics, vol. 169, **(2014)**, pp. 73-87.
[23] L. George, "HBase: the definitive guide", O'Reilly Media, Inc., **(2011)**.
[2]X. Lei, C. Wenzhi, W. Zonghui and Y. Shuangquan, "Smart-DRS: A Strategy of Dynamic Resource Scheduling in Cloud Data Center", Cluster Computing Workshops (Cluster Workshops), IEEE International Conference, **(2012)**.
[3] T. W. Tek, S. Xin and J. Balasooriya, "Service-Oriented Cloud Computing Architecture", Information Technology: New Generations (ITNG), Seventh International Conference, **(2010)**.
[7] M. P. Papazoglou, "Service-oriented computing: concepts, characteristics and directions", Web Information Systems Engineering, WISE, Proceedings of the Fourth International Conference, **(2003)**.
[10] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph and R. H. Katz, "Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center", NSDI, **(2011)**, pp. 22-22.
[11] S. Dubey, V. Jain and S. Shrivastava, "An innovative approach for scheduling of tasks in cloud environment", Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), **(2013)**.
[13] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker and I. Stoica, "Dominant Resource Fairness: Fair Allocation of Multiple Resource Types", NSDI, **(2011)**, pp. 24-24.
[14] Z. Zhang, H. Jizhong, L. Bo, Z. Wei and M. Dan, "Lynn: A Multi-dimensional Dynamic Resource Management System for Distributed Applications in Clouds", Cloud and Service Computing (CSC), International Conference, **(2013)**.
[15] L. Mei, Z. Kailong, Y. Longhui and Z. Xingshe, "Research on resources scheduling technology based on fuzzy clustering analysis," Fuzzy Systems and Knowledge Discovery (FSKD), 9th International Conference, **(2012)**.
[16] M. Stillwell, "Dynamic Fractional Resource Scheduling for cluster platforms", Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), IEEE International Symposium, **(2010)**.
[19] G. Lee, B. G. Chun and R. H. Katz, "Heterogeneity-aware resource allocation and scheduling in the cloud", Proceedings of HotCloud, **(2011)**.

Authors

Weng yu, he is an associate professor of College of Information Engineering, Minzu University of China. His current research interests are distributed computing and service computing.

Yuan Jie, She is professor of College of Information Engineering, Minzu University of China. Her current research interests are Data mining and Computer network.

Jia Xiaodong, He is currently working toward the Master degree in computer science and technology at Minzu University of China. His current research interests are cloud computing and big data