A PARALLEL COMPUTING FRAMEWORK FOR CLOUD SERVICES

Ai Fang[1*], Xue Hu[2], Cong Chen[3], Zhuan Sheng[4]
[1]*School of Information Engineering, Suzhou University, Suzhou Anhui, 234000, China*
[2]*School of Computer and Information, Hefei University of Technology, Hefei 230009, China*
[3]*University of Electronic Science and Technology of China, Chengdu, Sichuan, 610054, China*
[4]*Beijing Institute of Control Engineering, Beijing 100190, China*

Abstract
In recent years, the requirements of computer performance for large data processing are getting higher and higher. From the perspective of users, in order to conveniently use big data processing platform, and put more focus on writing the parallel computing algorithm. This paper presents a study of cloud service, parallel computing, IronPython and virtualization, and designs client, service node and node calculation module by using Python for its ability of self-reflexive and observation and calling the library functions of which cannot be achieved using other languages. In this paper, the parallel computing framework based on Python can be easily implemented on the virtual machine of the cloud computing platform.

Keywords: Cloud Services; SOA; Python; Parallel Computing.

1. INTRODUCTION

Nowadays, along with the change of computer application technologies, people's requirements are imperceptibly changed, especially for processing power of the computer. The requirements for storage and speed in early 90s of last century driven the development of large computer data processing ability [1-3]. With PC computer software and hardware being accepted and the architecture of PC computer becoming more and more mature, more and more PC based supercomputers, such cluster systems have generally existed and occupied a very good position in terms of performance.

After the emergence of the new concept Service Oriented Architecture (SOA), the role of the Internet has been fully reflected due to its ability to establish relationships between the different platforms. For such a role, the computer resources management become what can be achieved through the internet. Using the functions provided by the Internet to achieve cross platform management of computer resources has become increasingly common [4].

However, in recent decades, parallel software technology has not made breakthrough progress due to the need to fundamentally solve these existing parallel programming problems, such as how to code the use of cross platform and how to compile and execute automation problems. The problem of control dependence and data dependence has become the first problem to be solved in parallel computing, and has been in the process of being discussed and solved by the researchers in the [4].

This paper presents a study of cloud service, parallel computing, IronPython and virtualization to design client, service node and node calculation module by using Python for its ability of self-reflexive and observation, by taking into account the case that users may face the restriction of computer performance and the installation and operational of database upgrade when they are in the use of parallel computing in the process. Combining cloud computing technologies and

parallel computing technologies can solve the current provides a lot of convenience for users, while the problems from the external environment do not need to be considered, which enables users to put more energy in the parallel computing algorithm [5-8].

2. RELATED WORKS

Service Oriented Architecture (SOA), the technology already of which has existed first to be known in 1983, but did not get attention, mainly refers to the dispersion in the Internet space service of an integrated in this architecture inside each service and other services to each other to send information, for an interactive [1]. The interaction not only as stated above, it also contains a mutual contact application service and service; or between service and service through the use of an agreement is the agreement as a basis for a specific combination of mutual cooperation in the way to achieve some requirements [9].

When users need parallel computing, the service nodes can compute resources for these users to complete the calculation, we call this kind of computing resources for parallel, doing so provides service in order to give more computation requests, and better finish the calculation. In this case, this kind of service can be regarded as parallel computing services, and has the advantages of large data processing and a lot of time and ability to service.

The open service grid system (OGSA) is the first time we have learned that it is in the relevant experiments, and this experiment is called Globus. The service oriented architecture is the most critical part of the grid architecture. OGSA is a kind of service oriented architecture which is more perfect than web Service, and it has dynamic characteristics, which can dynamically create and delete the service instance [10]. OGSA platform is able to better combine the existing technology, make full use of the existing technology and hardware and software resources, to better realize the idea of the grid. With the further research and application of OGSA, it will become more and more perfect, and will eventually become one of the most important architecture of distributed resource sharing and service.

The basic service interface standard for high performance computing (HPC Basic Profile) is a kind of high performance computing grid access standard, by the open grid OGF (Open Grid Forum) for Web Service based on [10]. IBM, Microsoft and other companies and a number of research institutions to carry out some research behind together gives this standard, implemented in Web Service technology, when computing tasks will be submitted, to be authorized users to complete, and to manage the related resources.

3. CLOUD SERVICES ORIENTED PARALLEL COMPUTING ARCHITECTURE

Parallel computing services for cloud services Python parallel computing system corresponds to the application service layer in the cloud platform service hierarchy. Parallel computing is provided to users in the form of a service. Cloud services oriented Python parallel computing system is the use of Python to achieve the client and server, where the users can spend more time and energy on writing code, without needing to consider other external factors, especially the problems in the deployment environment. This paper first introduces the traditional parallel computing cluster, and focuses on the Python parallel computing framework for cloud services.

3.1 Traditional parallel computing cluster

Fig. 1 shows a network configuration diagram of a well-known parallel computing cluster. A computing cluster includes several functional modules as follows: client, this part mainly realizes the interaction with the user, accept and explain computing services to users, but also through the network environment and the service node complex communication links, to accept the results of the parallel computing nodes also forwarded the request [11]. The service node probably comprises the following: (1) the first to accept the request of the user for analysis then forwarding; (2) will accept the user's request for storage of statistics; (3) to interact with the control node, the cluster information dynamic monitoring; (4) communicating with the control node, can how to achieve the forwarding computation request; (5) and a control node by computing nodes allocation of communication resources monitoring, calculation and results of [1] transceiver request; (6) timely monitor node performance, improve the corresponding code; (7) can save results from the calculation nodes. Control node, in fact, this part in order to achieve specific monitoring. Computing nodes, the task of this module is very simple, is responsible for handling user requests from the service node and the final result to send [12].

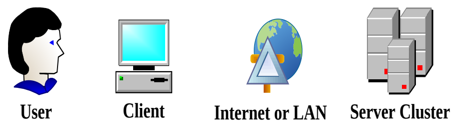

Fig.1. Traditional Parallel Computing Cluster

3.2 Python parallel computing framework for cloud services

The above section introduces the traditional parallel computing cluster, next we present an overall design of the cloud service-oriented Python parallel computing system from the business logic, knowledge oriented cloud services Python parallel computing framework, the logic chart of which is shown in Fig. 2.

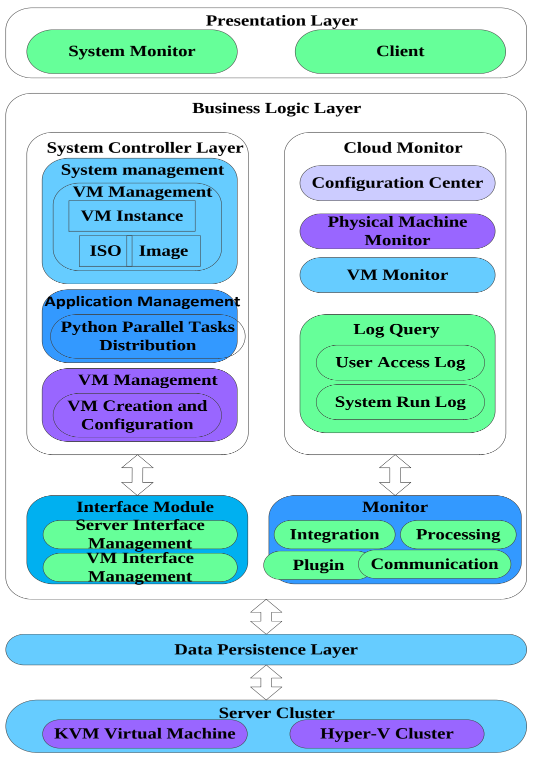

Fig.2.The Logical Architecture of Python Parallel Computing System For Cloud Services

3.3 overall system architecture and functional modules

According to logic of Python parallel computing system in Fig. 2, this section will refine the module functions, the each module and the corresponding overall architecture is shown in Figure 3. We next introduce the function modules of the whole system:

(1) For receiving the request from the client computing interface, when receiving a calculation request, client computing interface does not immediately forward which needs task queue request storage which is the achieved table structure in Python, finally wait to request the results of the query scheduling;

(2) For the request queue mentioned above, how to schedule the request in the queue, is based on user defined strategy to carry out, is not the same.as long as this is achieved through the corresponding priority, the use of the simplest and most common first come first served to achieve the specific requirements of the scheduling. That is, all requests from the client after the

arrival of the arrival time in accordance with the order and priority scheduling;

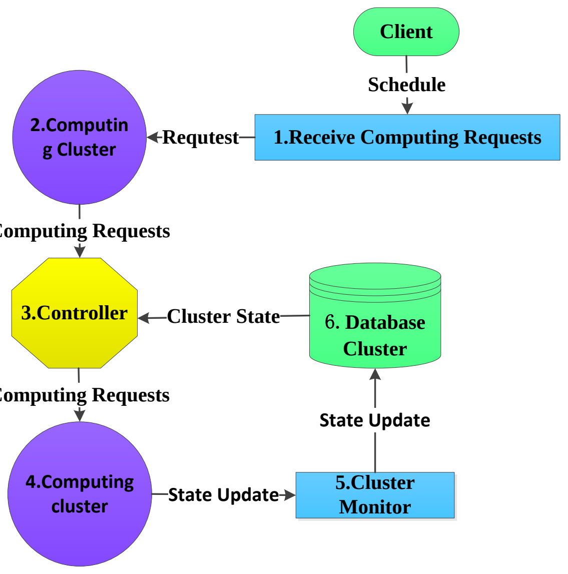

Fig. 3. System Module Function and Architecture

(3) The primary work of the controller is to monitor the calculation of node resources, and the dynamic monitoring, real time database cluster node information exists, then according to the user's request to query the state calculation, the node forward calculation request; the controller not only has said above also has function error handling ability, how to deal with the failure of the calculation request will be decided by the system strategy, a simple method of the first thought is the calculation request returns to the queue, or in a certain period of time after the drop, which can also be calculated node processing error report to the service node, so that the next time you can not this node sends a request to the controller; will not stop the calculation according to the status of the cluster in the cluster database update information, to send customized strategy Finally, the decision is forwarded to the computing cluster. Finally, the controller updates the status of all the computing clusters to the database in real time;

(4) Calculate the cluster receiving and calculate the request, mainly use the standard Web service interface to carry on the work of the cluster and the allocation of resources, and return the result;

(5) The cluster monitoring system is mainly used in computing nodes to install the plug-in program, and a monitoring server in the control node, to complete the monitoring of each cluster or node state, and real-time monitoring of the information in the database is updated, in order to ensure that the controller can make the distribution decision is calculated according to the cluster / latest node state information, and analyze the overall performance of computing cluster performance data to make a decision, which has been the most efficient use of computing resources;

(6) The whole cluster or clusters within each computing node state information is the information interactive resource monitor and data acquisition, dynamic storage in the database cluster, the cluster database not only saved the state now features of computing nodes, state information can also be recorded before, can be provided to other parts of the query read and write.

3.3.1 Client function module

Fig. 4 (a) shows the client module that is mainly responsible for the parallel program, creating the client, receiving a print transmitted request and used to calculate the results, and this module will parallel computing code forwarded to the client module. The client module, analysis from the user computing function call information and related data; and encoding, will then called from the compute nodes of Python computing services and obtain the corresponding results. Fig. 4 (b) is the client module function further detailed architecture, according to the above 4 (b) in the digital to functional description: (1) the client creates after the parallel computing code analysis function; (2) to obtain a function name as a parameter to the transmission module, transmission to the client library transmission module; (3) specific parameters; (4) and the calculation results to the client; (5) according to the specific code to obtain the parameters of parallel computing.

Fig. 5 is the client library module further detailed architecture, according to figure 5 to give the functional description: (1) according to the analysis of parameters from the client and according to the analysis of the situation to acquire source code; (2) to obtain the source code, based on the analysis of the relevant parameters call function to obtain the original code; (3) for the code to call the library function is compressed to cpickle, to the transmission module; (4) through the WCF configuration, the calculation results using the relevant protocol is transmitted to the server and receiving server; (5) back the settlement results to the client module.

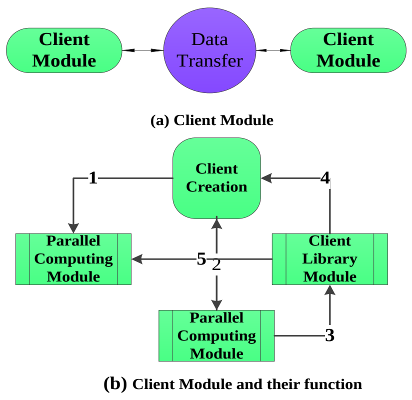

Fig. 4. Client Modules and Their Functions

3.3.2 Server function module

In figure 6 (a), the server module of the communication between each other is described, and the following modules are described in detail. Figure 6 (b) is a more detailed framework for the functionality of the service node module, which is described in the figure above:

(1) the calculation request is forwarded to the calculation data analysis module for data analysis, and the calculation data and source code are obtained;

(2) the data analysis on the calculation request, a detailed understanding of the request attribute, will calculate the request data added to the task of

transponder module task queue, waiting for the request to be forwarded;

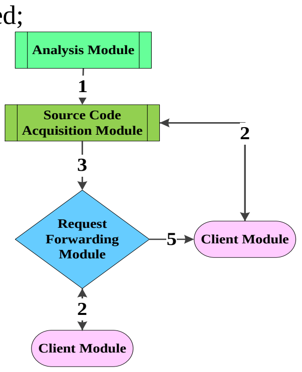

Fig. 5. Client Library Module Function Diagram

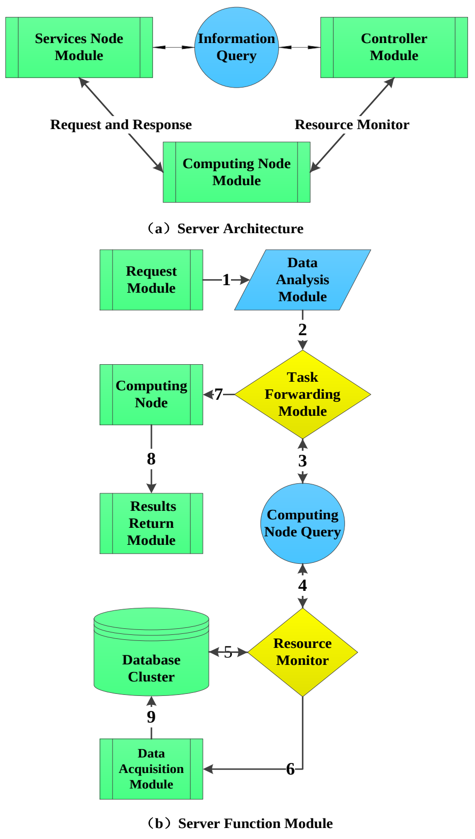

Fig. 6. Server Architecture and Its Functions

(3) the query request is sent to the query module of the computing node, and the module is queried according to the specific request;
(4) according to the concrete request, to carry on the corresponding inquiry to the database, and to obtain the desired result;
(5) the resource monitor will store the data from the data collector and store it in real time in order to ensure that the state of the computing node is the latest;
(6) the data acquisition request and the return of the specific information data are performed by the heartbeat between the resource monitor and the data collector;
(7) the task distribution module forwards the computation request and forwards it to the corresponding computing node through the corresponding query;
(8) the calculated results are given to the returning module on the computing node, and the result is returned.

According to Fig. 6 (b), the specific function of the service node is described in detail:
(1) the accepted calculation request to the Ironpython interface module;
(2) the Ironpython interface module completes the specific data and code loading by creating the Ironpython engine;
(3) the computing data which is loaded by the Ironpython engine is transmitted to the data analysis module of the, the data and the environment property are extracted, and the source code is restored;
(4) the calculated data and the source code are compiled dynamically, and the calculation is carried out;
(5) transfer the calculated results to the result return module;
(6) this function has been introduced at the service node.

3.4 Python parallel computing system flow

This section elaborates the SOA Python parallel computing process. Figure 7 is a Python parallel computing process, and the calculation of the specific process has been marked in the figure below, to describe the each step and to have an overall understanding of the system:
(1) the concurrent computation starts from a client who carries on the parallel algorithm programming, and calls the library function to carry on the computation attribute the compression, in order to calculate the request to be possible to transmit to the service node. Cloud computing services for Python parallel computing client library program through the complex extraction of the relevant functions needed to calculate and Python runtime environment. The data using the Python library to calculate these packaged data, function and status information, finally using the related protocol for transmission of parallel computing requests, which used the Web Service protocol to Parallel Computing Oriented cloud service request to the [10] service node is Python in parallel computing system;
(2) for cloud service Python parallel computing, service node in the system accepts the client's request after calculating, will place the request on the task queue, and the service node will calculate the node resources query to a control node in order to complete the calculation of sending request. The calculated node will receive the request task;
(3) the main control node has two functions, first is a dynamic monitoring of computing nodes, there is the service nodes to provide the necessary query help, computing nodes return state information to the service node. There is a resource monitor in the control node, which is responsible for collecting the status information of the computing cluster. Return the calculated node information from the client's request, and calculate the corresponding Python computing service on the corresponding computing node [13];
(4) the service node in the cluster receives the request from the client, and the request is forwarded according to the state information of the calculated node. The data from the client in the system is analyzed and forwarded to the computing node in the cluster. At the same time, the service node receives the information from the database of the control node, gets the processing power

of each computing node, and finally decides how to distribute the computing task [14];
(5) after compute nodes receiving the request from the client, Python computing service will use the IronPython library to calculate, and and send to a service node server after the results of the calculation is converted;
(6) because parallel computing is a concurrent operation, the service node will wait for the results of all calculations, and then use the Web Service protocol to transmit the data. Some of the client's Python library program will be analyzed by the return value from the server to add the results of the calculation to the runtime environment [15-17].

4. ARCHITECTURE ESTIMATION AND PERFORMANCE ANALYSIS

In this section, we will evaluate the system and test the performance of the Python parallel computing framework in this paper. The hardware and software environment and the performance evaluation criteria of the experiment are described in detail, and the experimental results are analyzed in detail.

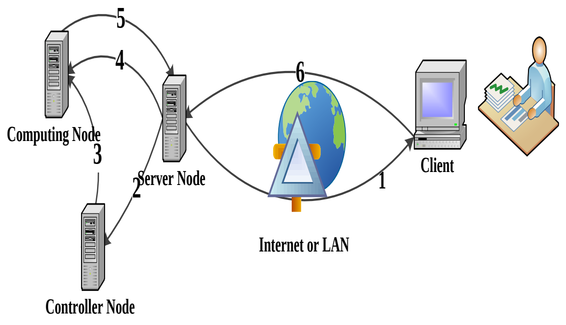

Fig. 7. Python Parallel Computing Flow Chart

4.1 Test environment configuration

In this paper, Python parallel computing system test environment consists of 2 blade servers running on the blade server KVM virtual machine, and service node and control node are installed in a blade server, other physical machine and virtual machine is used to compute nodes. The test machine configuration is shown in table 1.

4.2 Performance evaluation criteria

In this paper, the speedup ratio is used to measure the performance of Python parallel computing system. The speedup of this system will have a very intuitive understanding of the efficiency of parallel computing. In simple terms, the intuitive meaning of speedup is that parallel computation is compared with serial computing, which can be expressed in the following formula (1).

$$E_n = \frac{M_1}{M_n} \tag{1}$$

Table 1. Configuration Information of Python Parallel Computing System

Facilities	Configuration Information	Number
Blade Server	OS：Debian 6.0.5 CPU: Intel(R) Xeon E3-1240 v2 3.40GHz 4 Chanel 16 Cores； Memory: 32GB； Disk: SATA 1T x 3； Disk: SSD 512G x 1； Gigabit Ethernet x 3； Power module x 2；	3
Virtual Machine	OS：WinServer2008 CPU：Dual-Core Memory：1GB Disk：100GB	5
Server	MySQL-Cluster Database Cluster 1 Management Node 3 Data Nodes 3 Access Nodes	3

One n that is in use in the calculation of the number of parallel CPU, M_1 said the cost of serial operation time, M_n said the use of parallel operation will take time, you can see that this ratio is large, so the efficiency will be high efficient [13] platform.
However, according to the law of Amdahl, the number of CPU used in parallel computing is not proportional to the speedup of the system, and the Amdahl law can be expressed in this way:

$$S_n = \frac{W_s+W_n}{W_s+\frac{W_n}{n}} \tag{2}$$

In the formula (2), W_n represents the operations that are performed in parallel, and then the W_s represents the sequential execution of operations in parallel computing. Also, W_s can be used as an acceleration ratio equal to 1 W_n. The above formula can be translated into the following formula (3):

$$S_p = \frac{W_s+W_n}{\frac{W_s}{1}+\frac{W_p}{P}} \tag{3}$$

Let

$$\mathrm{W} = W_s + W_p \tag{4}$$

Where W represents the total amount of work, therefore,

$$S_p = \frac{1}{\frac{W_s}{W}+\frac{W_p}{W\times\mathrm{P}}} \tag{5}$$

Since parallel computing can be made into N parts, then,

$$P_t = \frac{W_t}{W} \tag{6}$$

P_t represents the proportion of the operation of the t step in the whole algorithm operation, then the final speedup is:

$$\mathrm{S} = \frac{1}{\sum_{t=0}^{n}\left(\frac{P_1}{P_t}\right)} \tag{7}$$

The speedup is used for cloud services Python parallel computing system, its effect is mainly manifested in the calculation of the data request to launch, the serial server, open the costs can not be parallel that part largely determines the speedup of [18-20]. For this, the results will be analyzed in part.

4.3 Speedup test and result analysis

The 4.2 section describes how to evaluate the parallel computing efficiency, and the following is to introduce a calculation subject to the test. We are very familiar with the use of a sum of all the prime code, for its less than for a certain number of two (using the tot_primes function test n) and so_prime (n). Each time to give a number of functions tot_primes (n) call, the results will be stored in an array and transfer.

4.3.1The effect of execution time on the speedup ratio

When the input number is 100000, n=100000, respectively, compared with Parallel Python in serial parallel computing, parallel computing and a result, and obtain and record these time to relative speedup in comparison to get a conclusion. In Table 2, the first column represents the number of parallel computing requests, the second column represents using serial mode to solve the time required for the third column is to use the Parallel Python library to a parallel computing time required, fourth column represents the speedup of parallel computing ParallelPython library; fifth in the 16 nuclear cluster in the same way as the calculated results, the sixth column for the corresponding speedup of.

Fig.8 shows the speedup of the parallel computation for the two different system configurations: From Fig. 8 we can easily get the following 3 conclusions:

Table 2. Test Results

Amounts	4 Cores	16 Cores	Speedup	Cluster	Speedup
1	5.29	5.26	0.89	8.29	0.59
2	10.59	5.39	1.87	8.31	1.18
4	21.28	5.33	3.86	8.59	2.34
8	42.73	10.86	3.85	11.47	3.64
16	85.96	22.03	4.04	11.03	8.02
32	173.07	44.28	4.04	19.71	8.63
64	354.86	90.27	4.04	31.88	11.07
128	738.64	187.95	4.04	57.28	13.01
256	1584.63	406.58	4.04	114.73	13.87

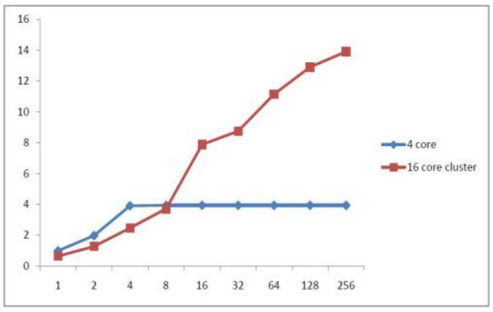

Fig. 8. Parallel Speedup of Different System Configurations

(1) the parallel effect of Parallel Python is very obvious, and through the comparison we also found a threshold, where the effect of parallel computing at this time did not reach is very obvious in the increase, but when more than request threshold, the speedup is stable for subjecting to local computer resources and for the reason that there is no way to get nodes;

(2) the Web service oriented Python parallel computing system, which is represented by a square line, has to consider other costs, such as web and networks. When the number of concurrent requests did not reach 8, the speed is not as quick as the local computing speed, but the back with the number of concurrent increase in speedup is gradually improved, and finally accelerated much higher than the local computation;

(3) with the concurrent requests becoming more and more, data transmission and the cost to start Web services become much smaller, this is the time to increase, so the proportion of which is changed, the proportion gradually reduced, under such circumstances, the speedup becomes very perfect, we may reach the memory of the theoretical value (16).

4.3.2. The influence of processing time of single computing data on speedup

Table 3 is the same for the cloud computing services Python parallel computing system cluster environment, the *n* different values, respectively, the calculation of the speedup is 10000, 200000 , 400000, 600000, respectively.

Table 3. Comparison of Speedup of N with Different Values

Amounts	10000	200000	400000	600000
1	0.19	0.39	0.59	0.79
2	0.18	0.79	1.24	1.71
4	0.27	1.64	2.51	3.32
8	0.29	3.21	3.69	6.33
16	0.82	4.49	8.01	10.21
32	1.59	6.78	8.81	10.21
64	2.09	9.11	11.21	12.41
128	3.31	10.19	12.87	13.79
256	3.42	11.77	13.87	14.80

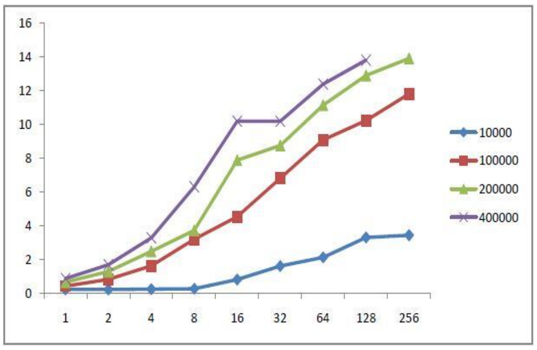

Fig. 9. Relationship between the Number of Concurrent Requests and Speedup

Fig. 9 shows the relationship between the number of concurrent requests and the speedup. The following conclusions can be drawn from Figure 8:

(1) When *n* is very small, do not use the cluster system for parallel computing operations, because you will find that in fact in other ways is the same effect, it is a waste of resources. Even at the same time there are 256 such parallel requests, the speedup is only 3.42, compared with the theoretical value of 16 there is still a big gap. According to the relevant law introduced above, this is because the parallel computing part and the whole operation time is very small, so that the proportion will be very small;

(2) From the top of the figure and the table we can also draw a conclusion of another, that if the number has changed the calculation units, and is becoming larger, the number of concurrent is in the same case, concurrent computing unit gradually increases the amount of calculation, the speedup will gradually improve.

To sum up, the number of concurrent clusters can also influence the performance of the above table through data analysis, and the final conclusion can be drawn that, when the number of concurrent requests did not reach the threshold, the time that cluster computing uses does not change basically, and this conclusion is more prominent in the case of *n*= 400000, this is because the entire platform itself has 16 nuclear, and these nuclear work is complicated, so that when the number of requests did not reach the threshold, the real advantage of the cluster can not be displayed.

5. CONCLUSION

In this paper, we propose a parallel computing framework for cloud services, test and analyze the whole system of Python. First, the preparation and deployment of the hardware environment, then, network environment, and finally the deployment of the entire system software. Prior to the completion of the construction of the environment, before the test, we also introduced a standard for the experiment and analysis of the acceleration ratio, through the elaboration of a good understanding of the efficiency of parallel computing. Finally, a specific example is designed to test the performance of the system in detail.

6. ACKNOWLEDGEMENT

This work is partly supported by the following projects: Teaching research project (2016jyxm1026), Suzhou Regional Collaborative Innovation Center (2016szxt05) and the Natural Science Fund of Anhui Province (KJ2014ZD31).

7. REFERENCES

[1] M. Rodriguez and R. Buyya, “Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds,” IEEE Transactions on Cloud Computing, vol. 2, no. 2, pp. 222–235, April 2014.

[2] L. Chi, J. Liu, and Q. Hu. Evaluation and Test for Scalability of Numerical Parallel Computation, Journal of Computer Research and Development, vol.42, no.6, pp.1073-1078, 2005.

[3] D. Dang, Y. Liu, and X. Zhang et al., “A Crowdsourcing Worker Quality Evaluation Algorithm on MapReduce for Big Data Applications,” IEEE Trans. Parallel Distrib. Syst., vol.27, no.7, pp.1879–1888, July 2016..

[4] J. Li, S.Chen, H. Ma. Research of Service-oriented Architecture Reference Model and Its Application. Computer Engineering, 2006, 32(20): 100-102.

[5] T. Andrews, F. Curbera, H. Dholakia, et al. Business process execution language for web services. 2003:22-23.

[6] F. Curbera, R. Khalaf, N. Mukhi, et al. The next step in web services. Communications of the ACM, 2003, 46(10): 29-34.

[7] C. Mohan, G. Alonso, R. Gunthor, et al. Exotica: A research perspective on workflow management systems. Data Engineering Bulletin, 1995, 18(1): 19-26.

[8] Z. Cai, X. Li, and J. N. D. Gupta, “Heuristics for provisioning services to workflows in XaaS clouds,” IEEE Transactions on Services Computing, vol. 9, no. 2, pp. 250–263, 2016.

[9] I. Ang. Watching Dallas: Soap opera and the melodramatic imagination. Psychology Press, 1985, 56-58.

[10] Y.Zhou. A Python-Calculation Grid Based on Service-Oriented Parallel Computing. Shanghai Jiaotong University, 2008,12-67.

[11] S. Xue. The Comparison Research of Cloud Computing and Grid Computing, 2010, 2-9.

[12] T. Fang. Research on grid application of parallel computing. Guangdong University of Technology, 2006, 45-47.

[13] Y. Li. Design and implementation of parallel computing platform based on MPI. Northeast Normal University, 2007, 2-10.

[14] W. D. Liu. A distributed data flow model for composing software services. Stanford University, 2003,12-14

[15] A. Fuggetta, G. P. Picco, G. Vigna. Understanding code mobility. Software Engineering, IEEE Transactions on, 1998, 24(5): 342-361

[16] F. Curbera, R. Khalaf, N. Mukhi, et al. The next step in web services. Communications of the ACM, 2003, 46(10): 29-34.
[17] Y. Ye, S. Ying, and W. Li et al.. Research of SOA and Its System Building. Computer application research, 2005, 22(2): 32-34.
[18] R. Perrey, M. Lycett. Service-oriented architecture. Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on. IEEE, 2003: 116-119.
[19] G. Li. Research on cloud computing platform architecture based on Virtualization Technology. Journal of Jilin Architectural and Civil Engineering Institute, 2011, 28(1): 79-81.
[20] J. Wen. Establishment and performance analysis of parallel computing platform. School of computer science, Guangdong University of Technology, 2007, 34-45.