Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/090227

Neural Network PID Control based Scheduling Mechanism for Cloud Computing

Xiaoqi Xing[1,2,*], *Bin Liu*[1,2] *and Dongyi Ling*[1,2]

[1] Science & Technology on Reliability & Environmental Engineering Laboratory, Beijing 100191, China
[2] School of Reliability and Systems Engineering, Beihang University, Beijing 100191, China

Received: 29 May 2014, Revised: 27 Aug. 2014, Accepted: 29 Aug. 2014
Published online: 1 Mar. 2015

Abstract: With the progress of academia and industry, cloud computing is moving from theory to practice. Job scheduling is an important issue in the high performance cloud computing environment. An appropriate scheduling mechanism can efficiently reduce the response time and enhance the user experience. However, most of the traditional scheduling mechanisms focus on the response time or the security, and the load balancing strategy is static, which cannot satisfy the new trend of development of cloud computing. Based on the new features of multi-users and multi-tasks in the open computing environment, the paper proposes a fuzzy neural network PID (Proportional+Integral+Derivative) control based scheduling mechanism. The scheduler takes deviation between the current Quality of Service (QoS) of the system and the average QoS as input, and adjusts the task scheduling of different virtual machines (VM) in the cloud environment by using multi-input and multi-output fuzzy neural network PID control. In the scheduling mechanism, we evaluate the load status of the VMs in the cloud, and adopt the Neural Networks to automatically tune the jobs of VMs to get the best performance QoS of the system. Simulation shows the scheduling mechanism will not only improve satisfaction of customers but also achieve load balancing effectively.

Keywords: Cloud computing, fuzzy neural network, scheduling mechanism, quality of service

1 Introduction

With the development of computing technology, the demand to computing, data storage, and data transportation is increasing rapidly, which is hardly to satisfy for the traditional computing model. As the commercial implementation of grid computing [1], cloud computing came into reality under such circumstances. Cloud computing provide a variety of services for the users through the Internet, which is a complete new service-based resource providing model [2]. Cloud computing has a very wide range of user groups, which need to handle huge task and data volumes. So it is very critical to allocate and use the resource of the system properly, and effectively schedule the huge number of tasks that user submitted, which is targeted to provide the user a short response time and a small fee of the service. Most importantly, how to keep the load balance of each VM in the cloud is emphasis and difficulty of cloud computing technology. Cloud computing is a user-centered business model [3], what the users concern is not only the task processing time but also the cost of finishing the task. The cost of the task processing is mainly depends on the resource consumption, the market supply condition and some other factors as well. As to some time-sensitive tasks, the user will choose the servers with strong processing capabilities to finish them on time. And the user will choose the service with low fee to finish if the task is cost-sensitive. That's to say, not all users will choose the service with high service quality to finish their tasks in a short time for all kind of reasons, such the cost of the high quality service. Different user, different needs, and the target of the task scheduling of cloud computing is to provide a variety of service quality needs to satisfy all kind of demands, which is the most practical meaning of task schedule research.

Because of the significant advantages of new service model, cloud computing has received widely concern from not only the academic experts and scholars but also the industry giants. The technology has greatly developed even though it is justly proposed for only several years. At present, the IT giants all have released their own cloud

* Corresponding author e-mail: buaaxxq@163.com

computing solution, which include some well-known systems such as Amazon EC2 [4] platform, GoogleApp Engine [5] platform, Windows Azure Apache [6] platform, Blue cloud [7] platform from IBM, and Hadoop [8] platform from Apache.

CloudSim [9] is an open source web application that launches pre-configured machines designed to run many of the most common open source robotic tools. The CloudSim toolkit supports both system and behaviour modeling of Cloud system components such as data centers, VMs and resource provisioning policies. It implements generic application provisioning techniques that can be extended with ease and limited efforts. Currently, it supports modeling and simulation of Cloud computing environments consisting of both single and inter-networked clouds (federation of clouds). Developers can extend the platform to implement the task scheduling algorithm to satisfy your own needs.

The scheduling mechanism is much more difficult than others for its special features, which include very large scale of resources, virtualization of the machines, high degrees of reliability and the dynamic loads of tasks, the most important extendable of the system. To overcome such difficulties, the exports have proposed many algorithms. We will introduce some of them.

Heuristic intelligent scheduling algorithm [10], this kind of algorithm includes ant colony algorithm, genetic algorithm, and particle swarm optimization, which is the most matured algorithm to find the optimal solution. Because the task scheduling is a NP (Non-deterministic Polynomial) complete problem, by using this method, we can enhance the performance of the system. The self adaptivity is the advantage of the algorithm while the disadvantage is the algorithm is too complicated.

Trust-based scheduling algorithm [11], which is introduced to solve the problem that the isolation between the trust mechanism in resource management and the task scheduling methods. The algorithm finds the trust relationship between the task and resource by adding the trust into the cloud computing, which can guarantee the QoS of the scheduling algorithm.

Feedback dynamic algorithm [12], which is a negative feedback based dynamic load balance algorithm. This algorithm takes the response ability and the overload of each node into consideration, and adjusts the distribution of the task, to enhance the total throughput of the cluster by avoiding receiving too many requests even though it is overloaded already. However, collecting the state information of each node frequently bring too much burden to the system.

QoS-constraint scheduling algorithm [13]. This algorithm classifies the users into several groups according to the QoS demand of the service, and provides different kind of service to them.

These works have done well in their own area, however, in open system, it is very difficult to predict which task will add in or quit from the system at what time, and the mechanisms mentioned above all have to face the problem. All tasks scheduling can be taken as nonlinear control problems, so it is a must to make recognition to the control objects or do some linear process to the predicted object function. What's more, it is really tough for the open system to predict the QoS function of the applications, which is the top difficulty to feedback control precisely to the task scheduling. The ordinary control cannot achieve the desired effects to the nonlinear and time-varying systems because the control parameters cannot catch the variation of the tasks. To solve the problem, we propose a fuzzy neural network PID control based scheduling algorithm in the paper. The algorithm takes the advantage of the self-adaptive technology of the neural network parameters, and adjusts them to satisfy the control demand of highly complex systems.

The rest of the paper is organized as follows. In section 2, we will introduce the task scheduling model in cloud computing, chich includes structure of the model and the details of the scheduling. In section 3, the task schedule algorithm will be discussed. And we will prove the effectiveness of the machenism with the simulated experiment in section 4. At last, we make a conclusion and finish the paper.

2 Task Set and The Scheduling Model in Cloud Computing

2.1 Scheduling model in cloud computing

Before we explain how the mechanism works, a clearing understanding about the scheduling model is needed.

The structure of cloud computing includes three layers: application layer, platform layer, and infrastructure layer [14]. The application is faced to the user, which is used to provide interaction to the user and the service provider under the support of the system. What's more, the application layer is the interface for the user to submit the task and to receive the result of the task. The infrastructure layer is the collection of virtualized hardware resources and the related management functions, which is used to provide dynamic and flexible basic infrastructure service by implementing the automatic internal flow control and optimization of resource management. And the platform layer is the collection of the genetic and multiplexing software resources, which is used to provide the environment of developing, running, management, and supervision. According to the structure mentioned above, we adopt two levels scheduling model, as is show in fig1.

In Figure 1, the app level is the tasks that to be run on the virtual machines, each app in the first level is a task list, which includes different tasks, and each of the task has different resources requirement, including cpu, memory and io devices. In this model, the tasks in the task lists are unpredictable, so the resource requirements

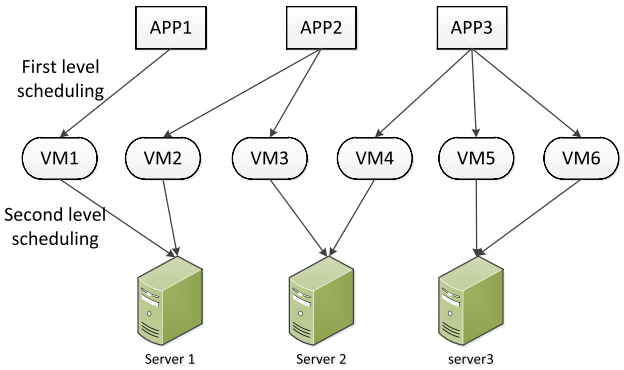


Fig. 1: Two levels scheduling.

are unpredictable as well. So the first level scheduling is to schedule the task on different virtual machines. Because the resource requirements are variable, so the workload of each virtual machine is different from time to time, the purpose of the first level scheduling is to balance the performance of different virtual machines. The method of the task scheduling is dispatching the new task to different virtual machine according to the performance of the virtual machines.

The virtual machine level in figure 1 is the basic computing unit of the cloud computing system. The virtual machines on each physical machine are pre-allocated, and the resources of each virtual machine are pre-defined as well. As described above, the tasks run on the virtual machine varies from time to time, so the resources utilization rate varies as the time goes, which leads the situation that some of the virtual machines are lack of memory, some of the virtual machines are urgent for computing capacity. The virtual machines run on the physical machines, the situation described above disturbs some of the resource utilization balance. So the second level scheduling is to adjust the virtual machine allocation on the physical machines, which is taken as virtual machine migration in practice. However, because of the low efficiency of virtual machine migration, the second level scheduling is not the secondary choice comparing to the first level scheduling. The second level scheduling only works when the structure of the computing system is not reasonable.

From the figure 1, we can find that, the first level scheduling is from user application to the virtual machines, which focus on the user's QoS demands of the task, and dispatch the task to proper virtual machines. The second level is from virtual machine to host resources, which is to find appropriate server from the host resources for the virtual machine to execute the task. During this process, which host to select is depended on the resource demand of the virtual machine and the current overload of the hosts. By using the two levels scheduling, the QoS demand of the task can be properly satisfied.

Usually, the user of cloud computing system cares about the cost of the system. The parallel computing needs more cpu cores or more memory to enlarge the throughput, however, the single thread of processing for a server is really a waste of resource, so we need more separated computing cores to reduce the cost of parallel computing.

To precisely describe the user's QoS demand of the task, we will consider the resource demands of the task, for example, CPU, memory, disk, and the bandwidth. Here we use membership function to present the effective value that gained by every QoS attribute and the effective value of the performance QoS attribute can be calculated by using the equation.

$$Time(i,j) = \begin{cases} 1 - \dfrac{time_{i,j} - deadline}{time_{i,j}}, & time_{i,j} > deadline \\ 1, & otherwise \end{cases} \quad (1)$$

$time_{i,j}$ is the time that the *i-th* task spends on *j-th* virtual machine, *deadline* is the expected finish time of the task.

The cost effective function is

$$Cost(i,j) = \begin{cases} 1 - \dfrac{cost_{i,j} - budget}{time_{i,j}}, & \cos t_{i,j} > budget \\ 1, & otherwise \end{cases} \quad (2)$$

$\cos t_{i,j}$ is the cost that the *i-th* task spends on *j-th* virtual machine, *budget* is the expected cost of the task.

According to the two equations, we can get the integrated effective function to show the comprehensive performance benefit value of the performance QoS.

$$performace(i,j) = \alpha \times Time(i,j) + \beta \times Cost(i,j) \quad (3)$$

Where $\alpha + \beta = 1, \alpha, \beta \in [0,1]$, the two parameters will tell the system which is more concerned by the user. If α is bigger, the user cares more about the execute time of the task, and if β is bigger, the user would save more money. The selection of the value of α and β depends on what the users care.

From the figure 1, we can find that, the first level scheduling is from user application to the virtual machines, which focus on the user's QoS demands of the task, and dispatch the task to proper virtual machines. The second level is from virtual machine to host resources, which is to find appropriate server from the host resources for the virtual machine to execute the task. During this process, which host to select is depended on the resource demand of the virtual machine and the current overload of the hosts. By using the two level scheduling, the QoS demand of the task can be properly satisfied.

2.2 Detail of task schedule

The detail of the task schedule focus on the second level of the schedule model, it emphasizes on the details of task scheduling on different server, from the microscopic point of view, we will discuss how the model works.

There are many different kinds of tasks in the open system, and each of them has different QoS demands,

which has a variety of weight to the system. Suppose there are n tasks $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_n\}$ in the cloud environment, and each of them has the weight $\omega_i (\in R_+)$ to indict the importance of the task, where the bigger of the weight is, the more important of the task. The value of weight can be defined in a variety of methods, for example, use the priority of the task directly, or just set by the user's personal preferences. If the weight is depended by more than one factor, then the value can be achieved by using analytic hierarchy process algorithm [15]. There are resource constrains between VMs, suppose the VM set of the cloud is $VM = \{vm_1, vm_2, \cdots, vm_m\}$, and the task that executed on $vm_j, (1 \leq j \leq m)$ is $\tau_{i,j}$. We set the QoS of task τ_i as $S_i, S_i \in [S_{i\min}, S_{i\max}]$, where $S_{i\min}$ is the worst QoS that the user can accept, $S_{i\max}$ is the best QoS that the task can be achieved in this system, and the corresponding task τ_i on virtual machine vm_j is $\tau_{i\min}$ and $\tau_{i\max}$. The relationship between the VM set $VM = \{vm_1, vm_2, \cdots, vm_m\}$ and the QoS of τ_i can be calculated by the QoS functions. In order to reflect the different relationships between QoS of different task, we will normalize the QoS of each task.

$$Q_i = \frac{S_i - S_{i\min}}{S_{i\max} - S_{i\min}}, \quad Q_i \in [0,1] \tag{4}$$

When $Q_i = 0$, then task τ_i will get the QoS of $S_{i\min}$, when $Q_i = 1$, task τ_i will get $S_{i\max}$. Now we will consider a single virtual machine, it is very useful for a single virtual machine scheduling for the system can be seen as many single virtual machine works on the same time. As to a single virtual machine vm_j, $\tau_{i,j}$, $\tau_{i,j\min}$ and $\tau_{i,j\max}$ can be simplified as τ_i, $\tau_{i\min}$ and $\tau_{i\max}$. The relationship between the single virtual machine and normalized QoS can be described as QoS function $\varphi_i : [\tau_{i\min}, \tau_{i\max}] \rightarrow [0,1]$, where $\varphi_i(\tau_i)$ is the QoS gained by τ_i on virtual machine vm_j, we can define the fairness of the QoS from the system task as follows.

Suppose the part task on vm_j is τ_i, Q_i is the normalized QoS, and ω_i is the weight, if the following equation is true, we can get the proper task scheduling.

$$\frac{Q_1}{\omega_1} = \frac{Q_2}{\omega_2} = \cdots = \frac{Q_n}{\omega_n} = \overline{Q} \tag{5}$$

$\overline{Q}$ in the equation is the average QoS on a single task scheduling of virtual machines. ω_i is a monotonically increasing and continuous function. Then the task scheduling problem is to find the biggest value of $\overline{Q}$.

3 Task Scheduling Algorithm

3.1 System control structure

As mentioned above, there are two levels scheduling in the model. The model is to satisfy a variety of QoS demands. During this process, we need QoS monitor to collect and evaluate the QoS after the task is finished. Then we need some modules to control the task scheduling, as is shown in the following fig 2.

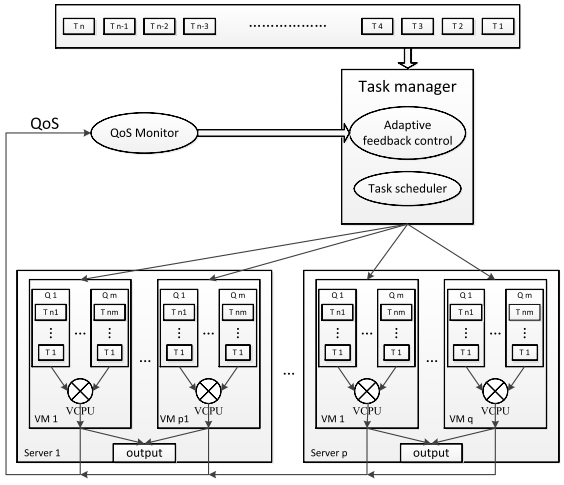


Fig. 2: System control structure.

As described in the figure 2, there are many servers in the system, where all kind of virtual machines can run on. Each of the virtual machine run on the servers maintains several task queues, and different queue holds different kind of tasks, which can be categorized with priority, QoS demands and some other criteria as well, just as described in section 2.3. The task on virtual machine will be executed with its own task scheduling mechanism, which depends on the operating system that run on it, we are not going to talk more about this part in this paper. At the entrance of the system, all the tasks are organized in a queue whose sequence is depended on the arriving time. The task manager of the system dispatch the tasks arrived on the system according to its own scheduling mechanism. Then the tasks will be scheduled to the virtual machine on the servers according to the result of the scheduling algorithm. once the task is finished, the server will calculate the QoS of the task according to the formula (1), (2), (3) mentioned in previous section, and the QoS of the task will passed to the QoS monitor of the system, after the evaluation of the module, the result will pass to the adaptive feedback control, which is used to readjust the scheduler of the system, and waiting fot the next scheduling process until all the tasks in the system are finished.

The QoS of the system will get the QoS value from each of the virtual machine, and calculate the value of $\overline{Q}$ by using equation (4) and (5). And the value will be returned to the task manager, and the task scheduler will adjust the task amount of each virtual machine to gain higher $\overline{Q}$.

3.2 Fuzzy neural network PID control of the scheduler

In this section, we will consider the task manager in fig2,

which use the fuzzy neural network PID to control the task scheduling.

For simple, we set $\omega_i = 1$, what's more, we will take the measured deviation into consideration. The task τ_i dispatched on virtual machine vm_j varies as time goes. We use $vm_j(k)$ to indicate the value of task τ_i in the time period $[t_k, t_{k+1})$, t_k is the *k-th* adjusting of the task scheduling, $Q_i(k)$ is the new QoS after the *k-th* task scheduling. We set the amount of task τ_i that dispatched on vm_j as $vm_j(0)$ at the moment of t_0.

Suppose there are n tasks in the queue, and tasks can be divided into p parts $\{\mu_1, \mu_2, \cdots \mu_p\}$. The $\{\mu_1, \mu_2, \cdots \mu_p\}$ is dispatched by the task scheduler to p virtual machines, and each of the virtual machine has an output, the output set can be presented as $\{e_1, e_2, \cdots e_p\}$, so the task manager is formed with p sub neural networks. And in the cloud system, there are t servers, and then the total of the system can be taken as a big fuzzy neural network with $p \times t$ small sub neural networks.

In the fig3, the tasks are divided into n parts, this designed is just for faster dispatching the tasks, and the partition is just random. Suppose that the task of μ_1 is going to dispatched to virtual machine, first, it takes the feedback of the last round QoS as a parameter, and the feedback control will evaluate the performance of the virtual machine, then adjust the task schedule on the virtual machine. In the following of this section, we will demonstrate how the feedback control works.

In the above figure, the output of the system is the average QoS,

$$\overline{Q}(k) = \frac{1}{n}\sum_{i=1}^{n} Q_i(k) \tag{6}$$

Facing to the feedback of task τ_i, we can use deviation between the current QoS value of task τ_i and the average QoS value $\overline{Q}(k)$ of the system.

$$e_i(k) = \overline{Q}(k) - Q_i(k) \tag{7}$$

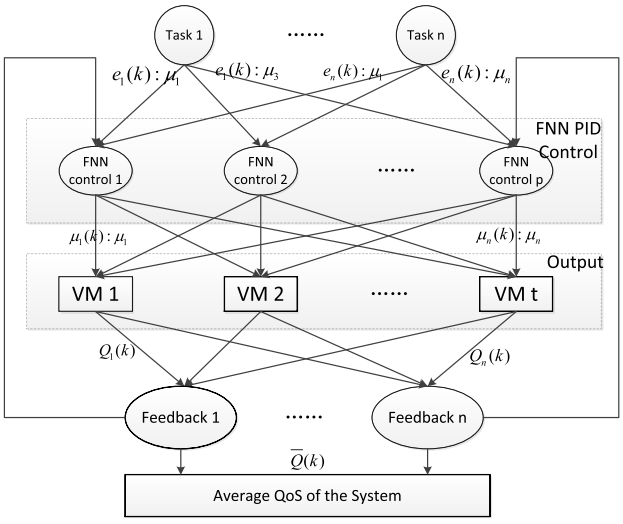


Fig. 3: FNN PID Controller.

Now we suppose at the beginning of the system, $e(k)$ is set to zero, and we have

$$e_i(k) = e(k) = 0 \tag{8}$$

So the input of the *s-th* sub fuzzy neural network can be calculated as follows

$$x(k) = \sum_{j=1}^{n} \omega_j vm_j(k) \tag{9}$$

Where, ω_j is the weight of the *j-th* input of the sub fuzzy neural network, $vm_j(k)$ is input the *j-th* virtual machine of the system in the *k-th* round.

After that, we will focus on the feedback of the system, which is used as a reference to increase or decrease the input of the virtual machines in the system. The feedback can generate by the following equation

$$f_i(k) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{t} \omega_j(k) Q_{i,j}(k) - \sum_{i=1}^{n}\sum_{j=1}^{p} \omega_j(k)(e_i(k) : \mu_j(k)) \tag{10}$$

In the above equation, $\omega_j(k)$ is the weight of the virtual machine, which is depended on its capability, $\mu_j(k)$ is the part of task τ_i which is dispatched to virtual machine k. and if $f_i(k)$ is smaller than 0, that's to say, the QoS of virtual machine is lower than the average, then in the next scheduling, the task will be decrease. And if $f_i(k)$ is bigger than 0, then the virtual machine has extra capability to do more work, so the task should be added.

In fact, the upper method to adjust the task scheduling is a little bit difficult, the condition goes even worse if the scale of the cloud grows fast, so we need another method which is less complicate and effective for big clouds. And this method is implemented in the loading balance module in the adaptive feedback control. Although the method is not so accurate as the upper method, however, it's much more suitable for large scale clouds.

By applying the load balancing module, we can evaluate the workload status of each virtual machine. The load balancing module is comprised of fuzzy logic workload measurement and neural network training component.

Fuzzy Logic Workload Measurement. In this part, we choose two parameters of the QoS of the service as the input fuzzy variables from virtual machine's performance: the CPU utilization and Memory Utilization. To compare the performance of different virtual machines, both fuzzy variables are normalized and characterized in the following formulas.

$$Light(x) = \begin{cases} 1, & if\ x < 0 \\ 1 - 2x, & if\ 0 \le x \le 0.5 \end{cases} \tag{11}$$

$$Medium(x) = \begin{cases} 2x, & if\ 0 \le x < 0.5 \\ 2 - 2x, & if\ 0.5 \le x \le 1 \end{cases} \tag{12}$$

$$Heavy(x) = \begin{cases} 2x - 1, & if\ 0.5 \le x < 1 \\ 1, & if\ x > 1 \end{cases} \tag{13}$$

Where, x is the utilization rate of the CPU or memory.

The following table demonstrates the inference rules which are used to compute the workload of each virtual machine in the system. Using load degree of each virtual machine can be obtained by using the Max-min inference and the center-of-fravity defuzzification processes.

Table 1: Fuzzy Inference Rules

Mem \ cpu	Light	Medium	Heavy
Light	Very light	Medium	Heavy
Medium	Light	Medium	Heavy
Heavy	Medium	Heavy	Very heavy

Once we have the workload measurement, we can easily get to know which virtual machine is overloaded, and we can use the following figure. The workload measurement system includes five layers: Input Layer, Hidden Layer X, Hidden Layer Y, Hidden Layer Z, and Output Layer.

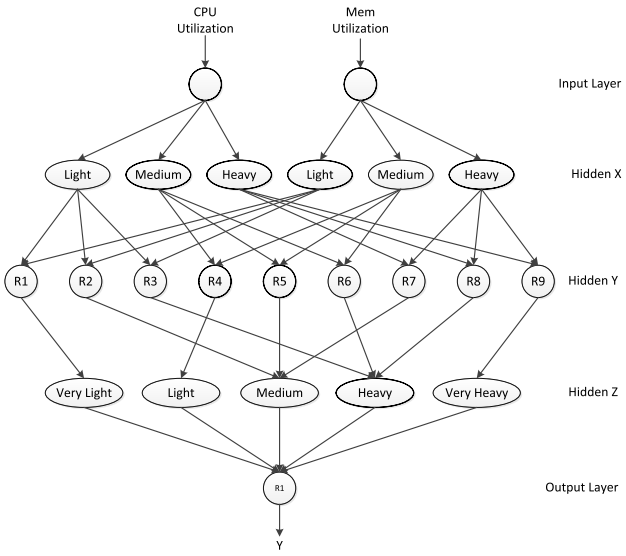


Fig. 4: Fuzzy logic workload measurements.

During the measurement of the system, the workload information of the virtual machines which includes CPU utilizations and Memory utilization as the input of the input Layer. The input are then individually fuzzified using the fuzzy sets defined in the table in the Hidden X layer. There are nine fuzzy rules defined in the Hidden Y layer in the system, which indicates the short slab of the virtual machine. For example, rule R1 indicates that the CPU utilization is light and the memory utilization is light, then the workload of the virtual machine is light. In this layer, the neuron computes the minimum value of the fuzzified degrees received form the Hidden X layer, which indicates the degree of match for the condition part of the rule. These minimum values is then send to the linked neuron in Hidden Z layer, and computed by the maximum operator to accomplished the Max_min rule inference process. The output layer will calculate the workload degree by the center-of-gravity defuzzification.

This method needs little computation, which is very suitable for the large cloud. By using such method, though we cannot make the accurate control as the FNN PID control, as to the huge cloud, which concerns more about the speed of the schedule, so this method should be a good choice.

3.3 The task scheduler of the system

In the previous part, we introduced how to decide which virtual machine is overloaded, once we have the result, we can adjust the task input of each virtual machine. In this part, we will present how the schedule the task between the virtual machines.

At the beginning of the establishment of the system, the number of the tasks τ_i that scheduled to virtual machine vm_i is bigger than $\tau_{i,j\min}$ and smaller than $\tau_{i\max}$. Then we can get the output QoS satisfy the needs $S_i \in [S_{i\min}, S_{i\max}]$. If $\tau_i = \tau_{i,j\min}$, we can get the QoS of $S_{i\max}$, if $\tau_i = \tau_{i,j\max}$, we will get $S_{i\min}$. When $\{Q_1(k), Q_2(k), \cdots, Q_n(k)\}$ is assured, the task that dispatched to virtual machine τ_i is also decided, which can be calculated in the following formula.

$$\tau_i(k) \propto S_{i\max} - \sum_{i=1}^{n} Q_i(k) \tag{14}$$

Because the task that gained by virtual machine τ_i is not indepent, if $S_{i\min} > 0$, we have

$$\left.\begin{aligned} \tau'_{i\max} &= \tau_{i\max} - \tau_{i\min} \\ \tau'_i(k+1) &= \tau_i - \tau_{i\min} \\ \tau'_{i\min} &= 0 \\ Q'(k+1) &= S_{i\max} - \sum_{i=1}^{n} Q_i(k) \end{aligned}\right\} \tag{15}$$

After new task schedule plan is generated, according to the formula (4) to formula, we can find the QoS of the system can be enhanced.

4 Experimental Evaluation

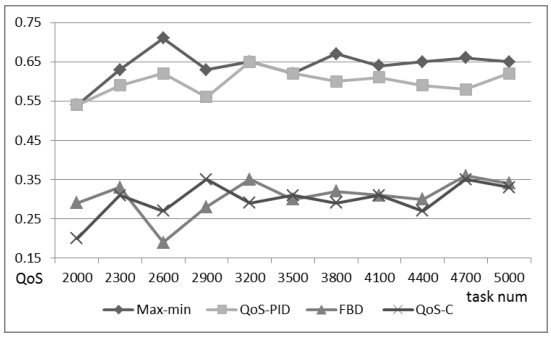
In the section, we will evaluate the effects of fuzzy neural network PID control based scheduling mechanism. We adopt the cloud simulation tools CloudSim 3 to simulate and implement the method described in section 2 and section 3, and employ network topology generator BRITE [16] 2.1b to simulate the network topology of the experiment. In the simulating experiment, we establish a datacenter, which contains 100 servers, each of which holds four or five virtual machines, and 2000-5000 tasks run in the system.

In the experiment, we considered 10 task circles of the task scheduling. And the parameter are settled in table 2. In the experiment, we choose six scenes to check our algorithm.

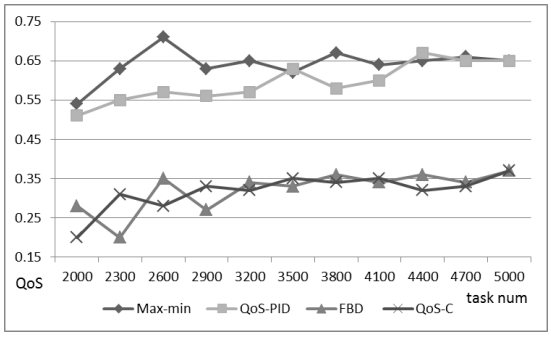
Table 2: Several Parameters

Scene	α	β	$\tau_{i\min}$	$\tau_{i\max}$
1	0	1	0	15
2	0.3	0.7	0	40
3	0.5	0.5	0	30
4	0.5	0.5	0	30
5	0.7	0.3	0	60
6	1	0	0	80

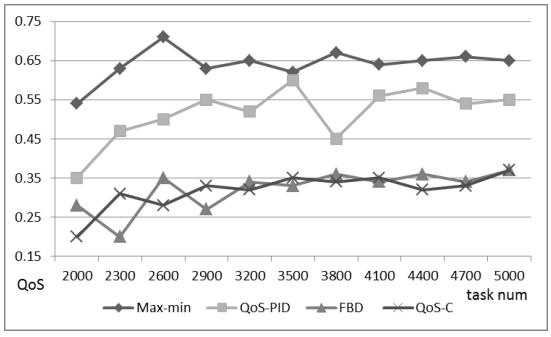
First, we will evaluate the influence of α and β to the average QoS value of the system under the different task schedule machenism. In the experiment, we have 3000 tasks to schedule. The result is show in the following figure.



(a) $\alpha = 0.3$ & $\beta = 0.7$



(b) $\alpha = 0.5$ & $\beta = 0.5$

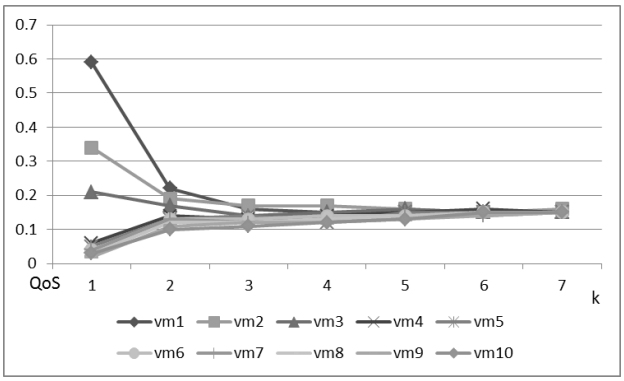


(c) $\alpha = 0.7$ & $\beta = 0.3$

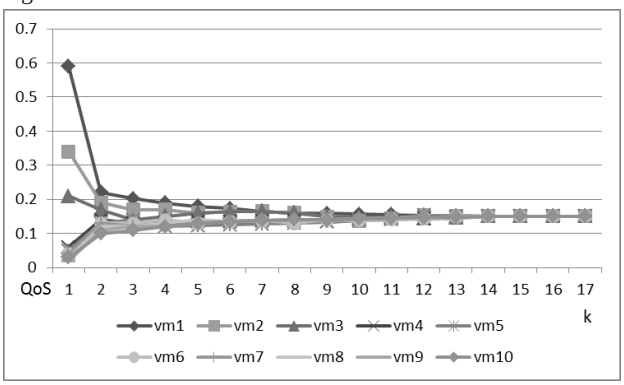
Fig. 5: QoS of different scheduling algrithm.

In the upper figures, the Max-min is the theoretically optimal results. QoS-PID is the algorithm that we propsed in this paper, the FBD is the feedback dynamic algorithm, and the QoS-C is QoS-constraint scheduling algorithm. from the result, we can see that the QoS-PID is the most closer to the theoretically optimal results. And the QoS-PID is the most stable one in the three algorithms.

Now let's consider the convergence process of the task scheduling. All QoS converge to the same steady value of the system.



(a) QoS of Fuzzy Neural Network PID control based schedule algorithm



(b) PID control based schedule algorithm

Fig. 6: QoS comparison of different schedule algorithm.

We can find that the fuzzy neural network PID control based schedule mechanism has a quick convergence speed. The control speed is very important in the cloud computing environment, for that the tasks in cloud computing vary fast, faster speed means that the system has better adaptability.

5 Conclusions

We designed and implemented a fuzzy neural network PID control based scheduling mechanism for cloud computing, which includes adaptive feedback control, task scheduler and the QoS monitor. The QoS monitor collects precise workload information from each virtual

machine in the cloud. And the component sends its results to the adaptive feedback control module, which dynamic adjusts the task scheduling. Once the feedback control has a better solution to the task scheduling plan, passes it to the task scheduler. Then the scheduler will dispatch the task to the virtual machine of the system. As the experiment shows that, when different α and β is selected, the QoS will come to a balance as the task number raises. And the experiment prove that, the new algorithm has better QoS output and faster convergence speed.

References

[1] FOSTER I, ZHAO Y, RAICU I, et al. Cloud computing and grid computing 360-degree compared. *In Proceeding of IEEE Grid Computing Environments Workshop*, 2008, pp.1-10.

[2] Liu Peng. Cloud Computing, Beijing: *Publish House of Electronics Indestry*: 2007, pp.2-3,

[3] Almorsy, M.; Grundy, John; Ibrahim, A. S. Collaboration-Based Cloud Computing Security Management Framework. *Cloud Computing (CLOUD), 2011 IEEE International Conference*, pp.364-371

[4] NEIGER G, SANTONI A, LEUNG F, et al. Intel virtualization technology: Hardware support for efficient processor virtualization. *Intel Technology Journal*, 2006, Vol.10(3): pp.167-177.

[5] AKIOKA S, MURAOKA Y. HPC benchmarks on Amazon EC2. In Proceeding *of 24th IEEE International Conference on Advanced Information Networking and Applications Workshops*, 2010, pp.382-390

[6] GOOGLE. Google App Engine. http://code.google.com/appengine, 2006.

[7] IBM. IBM Blue Cloud. http://www.ibm.com/cloud-computing/us/en/

[8] Narayan, S. Bailey, S. Daga, A. Hadoop Acceleration in an OpenFlow-Based Cluster. *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion*. 2012, pp.535- 538

[9] Buyya, R. Ranjan, R. Calheiros, R. N. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. High Performance Computing & Simulation, 2009. *HPCS "09. International Conference*. 2009, pp.1-11

[10] PANDEY S, WU L, GURU S, etal. A particle swarm optimization-based heuristic for scheduling workflow applications in Cloud computing environments. In Proceeding *of 24th IEEE International Conference on Advanced Information Networking and Applications*, 2010, pp.400-407

[11] HU Z, WU K J, HUANG J S. An utility-based job scheduling algorithm for current computing Cloud considering reliability factor. *In Proceeding of 3rd International Conference on Software Engineering and Service Science*, 2012, pp.296-299.

[12] LI J Y, QIU M K, LIU J W, et al. Feedback dynamic algorithms for preemptable job scheduling in Cloud systems. In Proceeding *of IEEE international Conference on Web Intelligence and Intelligent Agent Technology*, 2010, pp.561-564.

[13] LI L Q. An optimistic differentiated service job scheduling system for cloud computing service users and providers. In Proceeding *of 3rd International Conference on Multimedia and Ubiquitous Engineering*, 2009, pp. 295-299.

[14] Armbrust M. Above the clouds: a berkeley view of cloud computing. EECS Department, University of California, Berkeley, 2009

[15] Fung BCM. Wang K, Chen, R. Privacy-preserving data publishing: a survey of recent developments. *ACM Computing Surveys*. 2010, **42**, pp. 1-14

[16] Xiangqian Chen, Kia Makki, Kang Yen, Pissinou, N. A New Network Topology Evolution Generator Based on Traffic Increase and Distribution Model. *Networking, 2007. ICN '07. Sixth International Conference on*. 2007, pp. 56

Xiaoqi Xing is a PhD student in control science and engineering, School of Reliability and Systems Engineering, BeiHang University.His research interests include Computer control technology, software testability and software fault inject.

Bin Liu is a PhD and a professor in BeiHang University School of Reliability and Systems Engineering. His research interests include system engineering, software testing and software reliability.

Dongyi Ling is a PhD student in BeiHang University School of Reliability and Systems Engineering, Beijing. She received the MS degrees in BeiHang University School of Software, Beijing. Her research interests include software architecture analysis, component-based software modeling, testing, and software reliability evaluation.