LOAD BALANCING ALGORITHM TO IMPROVE RESPONSE TIME ON CLOUD COMPUTING

Nguyen Xuan Phi[1] and Tran Cong Hung[2]

[1,2] Posts and Telecommunications Institute of Technology, Ho Chi Minh, Vietnam.

ABSTRACT

Load balancing techniques in cloud computing can be applied at different levels. There are two main levels: load balancing on physical server and load balancing on virtual servers. Load balancing on a physical server is policy of allocating physical servers to virtual machines. And load balancing on virtual machines is a policy of allocating resources from physical server to virtual machines for tasks or applications running on them. Depending on the requests of the user on cloud computing is SaaS (Software as a Service), PaaS (Platform as a Service) or IaaS (Infrastructure as a Service) that has a proper load balancing policy. When receiving the task, the cloud data center will have to allocate these tasks efficiently so that the response time is minimized to avoid congestion. Load balancing should also be performed between different datacenters in the cloud to ensure minimum transfer time. In this paper, we propose a virtual machine-level load balancing algorithm that aims to improve the average response time and average processing time of the system in the cloud environment. The proposed algorithm is compared to the algorithms of Avoid Deadlocks [5], Maxmin [6], Throttled [8] and the results show that our algorithms have optimized response times.

KEYWORDS

Load Balancing, Response Time, Cloud Computing, Execution Time.

1.INTRODUCTION

As load balancing algorithms have appeared to respond the cloud computing problems, load balancing on the cloud have been recently studied and suitable algorithms have been implemented in this area. In terms of load balancing, we often refer to some basic algorithms such as Round Robin algorithm, Weighted Roud Robin algorithm, Least Connection algorithm, Weighted Least Connection algorithm, Least Response Time algorithm. These are algorithms basically, which use methods such as round robin which choose the machine with the least connection to handle the task, choose the job has the smallest response time. There are also many other load balancing algorithms depending on the software or hardware load balancer used. The load balancing technique in cloud computing can be applied at different levels, depending on what the load balancing is. The problem is how to distribute those requests so that the response time is minimal and how to decide the number and characteristics of the virtual machines that handle the requests. These requests must be distributed to the available virtual machines for processing. Therefore, how to decide the number and characteristics of the virtual machine to handle these requests, and how to distribute such requests to optimize the response time, is the task that the solution load balancing must resolve.

Load balancing policies depend on how we want to perform. For example, if you want the data center to load physical hosts, ie, the physical host allocation policy for the virtual machine, such as a host allocation policy that has the least processing core used for a virtual machine. Such load balancing is called at the Host level. If you want to balance the load of virtual machines running the application, that is, each virtual machine divides resources received from the host for the task

DOI: 10.5121/ijccsa.2017.7601

or application service running on them, load balancing at this level is called virtual machine level. In order to meet the above requirements, the establishment of an efficient load balancing algorithm and how to use resources in a reasonable manner is the goal that cloud computing wants to achieve. [1], [2]. Load balancing techniques in cloud computing now look at different parameters such as performance, response time, scalability, throughput, resource utilization, fault tolerance, accommodation and related costs. In addition to energy efficiency, carbon emissions are also considered [3].

With such usability and performance, cloud computing has become an indispensable trend. In the future, the increase in the number of cloud users requires service providers to meet the needs of users with minimal response time. Therefore, load balancing methods in cloud computing are increasingly being developed, when number of servers or server configurations increasing is only temporary method. Effective use of resources on the "cloud" is a necessity. This is also a huge challenge in the field of cloud computing. In order to meet the above requirements, the establishment of an efficient load balancing algorithm and how to use resources in a reasonable manner is the goal that cloud computing wants to achieve [4], [5], [6], [7], [8].

Response time on cloud computing is very interested in the research. Shubham Sidana [9] et al. presented the NBST algorithm, balance the load by arranging the virtual machines on the basis of their processing power and arranging the cloudlets according to their Length i.e. number of instructions in the cloudlet. The list of virtual machines and cloudlets is then submitted to broker for the allocation. In the paper [10] Atyaf Dhari et al. proposed Load Balancing Decision Algorithm (LBDA) to manage and balance the load between the virtual machines in a datacenter along with reducing the completion time (Makespan) and Response time. The mechanism of LBDA is based on three stages, first calculates the VM capacity and VM load to categorize the VMs' states (Under loaded VM, Balanced VM, High Balance VM, Overloaded). Second, calculate the time required to execute the task in each VM. Finally, makes a decision to distribute the tasks among the VMs based on VM state and task time required. The authors compared the result of proposed LBDA with MaxMin, Shortest Job First and Round Robin. The results showed that the proposed LBDA is more efficient than the existing algorithms.

Within the scope of this article, we will focus on virtual machine load balancing. The scheduling policies used are time-share and space-share for virtual machines and tasks. The aim of the paper is to propose the improvement of the Throttled algorithm [8], based on the research and evaluation of three Maxmin algorithm [6], Avoiding congestion in load balancing algorithm[6], Throttled algorithm [8] to improve the response time and average processing time of the load balancing system in the cloud environment.

The paper is organized into the following sections: Part1 presents introduction to the load balancing algorithms. Part 2 presents the our proposed algorithm. Part 3 show the simulation results of the proposed algorithm. Part 4 conclusion.

2. PROPOSED ALGORITHM

The proposed equilibrium algorithm will be based on the basis that the authors [5], [6], [8] have done to optimize the average processing time and response time of virtual machines. When studying the load balancing algorithms [5], [6], [8] in the cloud environment, we find that the above works of the author group are extremely useful. To develop this algorithm, we tried to learn more and propose a new algorithm. Therefore, if the Cloud Manager can be further optimized in the load balancing algorithm by adding some parameters such as the expected completion time of each resource (vm) With lists of requests coming up, the result will be an optimal algorithm.

In the proposed algorithm, we will look at the parameters such as: the list of workloads of the system (Cloudlet), the queue list has been submitted for each virtual machine, percentage of utilization of virtual machine (represented by the number of requests for each virtual machine's requested queue and the expected cost of completing that queue), the expected completion time when there is a request for each virtual machine. With these parameters, we will select the virtual machine with the smallest scheduled completion time and the lowest level of usage for assigning tasks.

2.1. Theoretical basis

To measure the effect of load balancing can be based on many factors, but the most important are two factors: load and load performance. Load is the CPU queue index and CPU utilization. Performance is the average response time required by the user. The load balancing algorithm is based on input parameters such as the configuration of virtual machines, the length of the cloudlet tasks, the arrival time, the completion time of the tasks, and then the expected completion time. of each task, expected response time. Response time is the processing time plus the cost of the request or task transmission time, queued through the network nodes. Expected response time is calculated according to the following formula [4]:

$$Expected\ Response\ Time = F - A + T_{delay} \quad (1)$$

where:

F: time to complete the task.
A: arrival time of the task.
T_{delay}*: transfer time of the task.*

Because the algorithm that performs load balancing is that of DatacenterBroker, the level of the algorithm only affects the processing time in a local environment of a data center. Therefore the communication delay parameter can be omitted, so $T_{delay} = 0$.

Calculate expected task completion time [4]:

If the scheduling policy is Spaceshare-Spaceshare or Timeshare-Spaceshare, then the formula is defined by the formula (2), (3):

$$eft(p) = est + \frac{rl}{capacity * cores\,(p)} \quad (2)$$

Where capacity is calculated by the formula [4]:

$$capacity = \sum_{i=1}^{np} \frac{cap(i)}{np} \quad (3)$$

If the scheduling policy is Space share-Timeshare or Timeshare-Timeshare, it is determined by the formula (4), (5):

$$eft(p) = ct + \frac{rl}{capacity\ *\ cores\,(p)} \quad (4)$$

Where capacity is calculated by the formula [5]:

$$capacity = \frac{\sum_{i=1}^{np} cap(i)}{\max\left(\sum_{j=1}^{cloudlets} cores(j), np\right)} \qquad (5)$$

In formulas (2), (3), (4) and (5):

- *eft(p)* is the expected completion time of the Cloudlet p.
- *est* is the arrival time of Cloudlet p.
- *rl* is the total number of instructions the Cloudlet p must execute on a processor.
- *capacity* is the average processing power (in MIPS) of a core for Cloudlet p.
- *ct* is the current simulation time.
- *cores(p)* is the number of cores required by Cloudlet.
- *np* is the actual number of core that the host is considered.
- *cap* is the processing power of the core.

The capacity parameter defines the true capacity for processing tasks on each virtual machine. Clearly, capacity depends on the policy of scheduling computing resources on virtualized systems. Total processing power on a physical host is constant and depends on the number of physical cores and processing power of each cores. However, when this processing resource is shared for multiple tasks (cloudlet) simultaneously, each task requires a certain number of cores and if the total number of cores is greater than the number of physical cores, the virtual core concept appears, each virtual core will have lower power processor capabilities of physical cores. In other words, the capacity of a virtual core for a task is only equal to or less than the physical core and depends on the resource sharing policy. Capacity is the processing power of a virtual core. From this analysis and based on the resource sharing policy to develop the formula for capacity. Resource sharing policy is characterized by the scheduling mechanism in cloud computing. We have two levels of scheduling: virtual machine scheduling to share physical host machine resources and task scheduling to share virtual machine resources. There are two scheduling mechanisms, Timeshared and Spaceshared. In this paper, we will implement algorithms and simulations based on the Spaceshared - Timeshared policy in turn for virtual machines and tasks. Therefore, the basis for calculating the proposed algorithm will be based on formulas (4) and (5).

2.2. Algorithm design

Step 1: Initialize Data center Broker. The status table of the virtual machine and the state of the existing clouds. At the time of initialization no virtual machines were allocated the Cloudlet.

Step 2: When there is a request to allocate new virtual machine come Data center Broker, DatacenterBroker analyzes the status table. Then, calculate the total execution time of all existing cloudlets in the queue (of each virtual machine) and the expected completion time of the new cloudlet being prepared for processing. If the virtual machine has the smallest processing time expected, that machine is chosen to submit the next Cloudlet. If there is more than one, the first virtual machine is selected.

Step 3: Send the selected virtual machine ID to the Data center Broker then DatacenterBroker sends the cloudlet to the virtual machine allocate by that ID.

Step 4: Databroker notifies about new allocation and updates to virtual machine and cloudlet status tables.

Step 5: When the virtual machine completes the processing request and DatacenterBroker receives the Cloudlet response, it will update the Cloudlet's status table as completed and reduce a Cloudlet in the status table.

Step 6: Go back to Step 2.

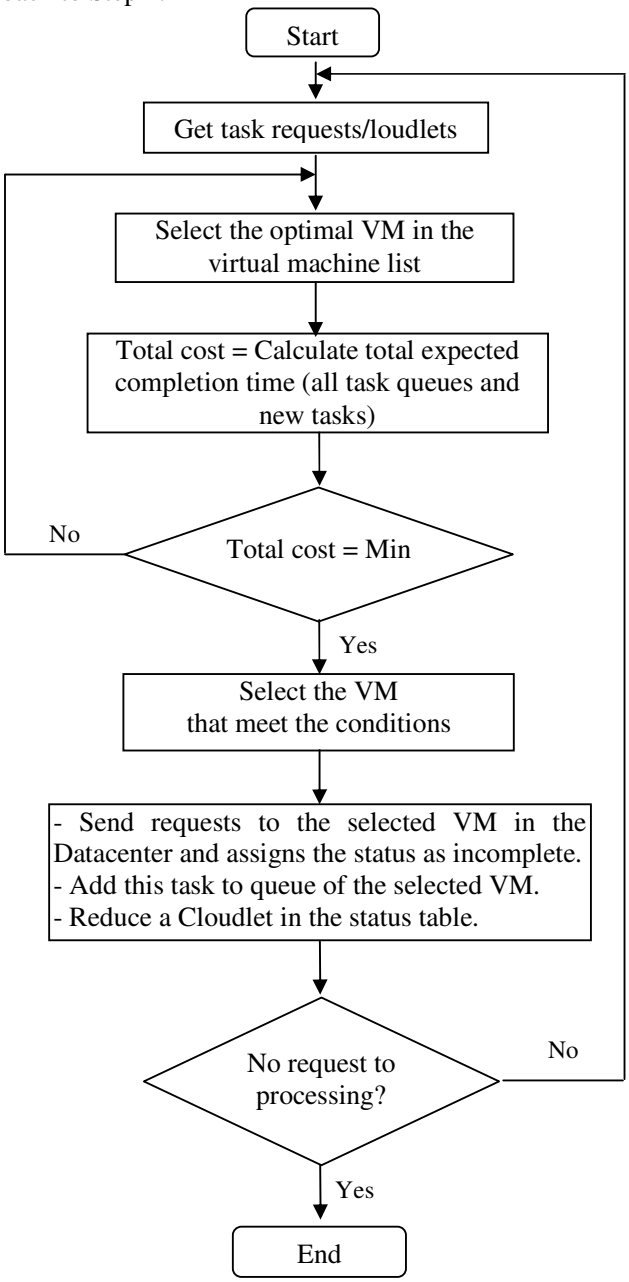

Figure 1. Flow chart of our proposed algorithm.

Thus, the difference between the algorithms we propose versus the other algorithms is that: Put into the expected completion time of each resource (vm) for the cloudlet task in queue task. Based on this parameter, the algorithm will select the VM with the smallest expected completion time and the lowest percentage of utilization for task allocation. In the our proposed algorithm, we will

also look at specific parameters such as queue list (Cloudlet) of all system, the queue list submitted for each virtual machine percentage of utilization of virtual machine (expressed by the number of virtual requests submitted by each virtual machine and the expected cost of completing that queue), expected completion time when there is a request come to each virtual machine.

3. SIMULATION RESULTS

3.1. Data using simulation

The goal of this simulation is to compare, analyze, evaluate the response time and execution time of Throttled algorithms [8] and the proposed algorithm. Using CloudSim Cloud Simulator includes 1 datacenter. System parameter values are given in the Tables 1, Table 2, Table 3. The simulation scenarios implemented according to the VM scheduling policy and task policy is: SpaceShared-SpaceShared.

Table 1: Value of parameters in cloud setting.

Type	Parameters	Value
Datacenter	Number of datacenter	1
	Number of host	3
Host	Number of PEs per host	1-4
	MIPS of PE	1000-30000 MIPS
	Memory of host	5120-10240-12288
	Storage capacity	1024000-1044480MB (1000-1020GB)
	Bandwidth	10000MB
VM	Total Virtual Machine	3
	Virtual Machine Memory (RAM)	1024-3072
	Bandwidth	1024MB
Cloudlet/task	Total Cloudlet	10-60
	The length of the cloudlet	1024-20480
	Required PE number	1-3

Table2.Parameters of virtual machine

ID	Memory (Mb)	Bandwith (Mb)	Number of PE/core	Speed of PE (MIPS)
0	4069	1024	2	200000
1	2048	1024	1	100000
2	1024	1024	2	50000

Table 3. Parameters of cloudlets

ID Cloudlets	Leng of Cloudlet	Number of required PE	ID Cloudlets	Leng of Cloudlet	Number of required PE
0	2000	1	*15*	2000	1
1	3000	2	*16*	1000	2
2	4000	1	*17*	3000	1
3	3000	2	*18*	5000	2
4	2000	2	*19*	2000	1
5	2000	1	*20*	2000	1
6	1000	2	*21*	3000	2
7	3000	1	*22*	4000	1
8	5000	2	*23*	3000	2
9	2000	1	*24*	2000	2
10	2000	1	*25*	2000	1
11	3000	2	*26*	1000	2
12	4000	1	*27*	3000	1
13	3000	2	*28*	5000	2
14	2000	2	*29*	2000	1

3.2. Simulation results

In this experiment, we simulate a cloud of the following parameters: 30 cloudlet (task), 1 datacenter, 3 VM; with parameters in Table 1, Table 2, Table 3. Simulation on the Throttled [8] and our propose algorithm, response time results as follows:

Table 4. Results of throttled algorithm [8].

Cloudlet ID	VM ID	Time (ms)	Start (ms)	Finish (ms)
0	0	125	100	225
3	0	187.5	225	412.5
1	1	422.5	100	522.5
4	0	220	412.5	632.5
6	0	140	632.5	772.5
5	1	250	522.5	772.5
7	0	187.5	772.5	960
9	0	125	960	1085
2	2	1095	100	1195
10	0	220	1085	1305
8	1	642.5	772.5	1415
12	0	250	1305	1555
14	0	125	1555	1680
13	1	375	1415	1790
15	0	220	1680	1900
16	1	220	1790	2010
11	2	815	1195	2010
17	0	220	1900	2120
20	0	125	2120	2245
21	0	187.5	2245	2432.5
19	2	532.5	2010	2542.5
18	1	642.5	2010	2652.5
22	0	330	2432.5	2762.5
25	0	125	2762.5	2887.5
26	0	110	2887.5	2997.5

24	1	345	2652.5	2997.5
27	0	187.5	2997.5	3185
23	2	750	2542.5	3292.5
29	0	217.5	3185	3402.5
31	0	187.5	3402.5	3590

Calculate average execution time and average response time of all tasks (cloudlets):

- Average execution time: 319.3333 (ms).
- Average response time: 1911.667 (ms).

Table 5. Results of propose algorithm

Cloudlet ID	VM ID	Time (ms)	Start (ms)	Finish (ms)
0	0	125	100	225
1	0	235	100	235
3	0	220	225	445
4	0	220	335	555
5	0	220	445	665
6	0	110	555	665
2	1	565	100	665
7	0	187.5	665	852.5
9	1	297.5	665	962.5
8	0	407	665	1072
10	1	330	852	1182
12	0	250	1072.5	1322.5
14	0	125	1322.5	1447.5
13	1	375	1182.5	1557.5
15	0	220	1447.5	1667.5
16	1	220	1557.5	1777.5
11	2	815	962.5	1777.5
17	0	220	1667.5	1887.5

18	0	312.5	1777.5	2090
20	1	312.5	1887.5	2200
21	0	220	2090	2310
19	2	532.5	1777.5	2310
23	0	187.5	2310	2497.5
25	0	125	2497.5	2622.5
26	0	110	2622.5	2732.5
22	1	532.5	2200	2732.5
24	2	532.5	2310	2842.5
27	0	220	2732.5	2952.5
30	0	125	2952.5	3077.5
31	0	187.5	3077.5	3265

Calculate average execution time and average response time of all tasks (cloudlets):

- Average execution time: 284.65 (ms).
- Average response time: 1686.467 (ms).

The above results show that our proposed algorithm has better response times than the Throttled algorithm [8], as shown in Fig 2, Fig 3.

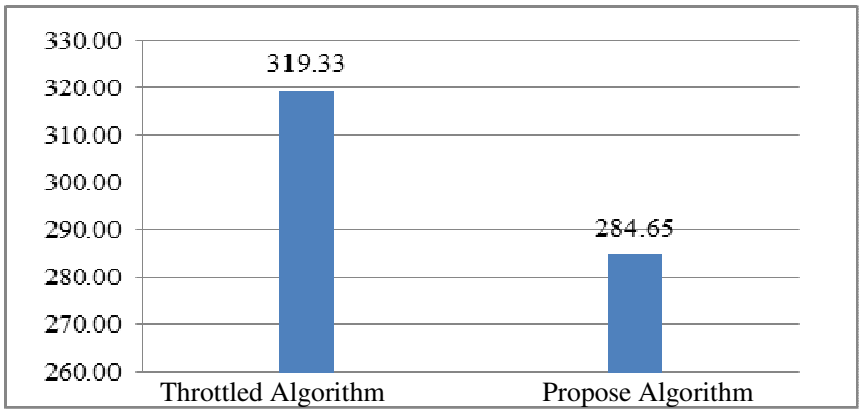

Figure 2. Comparison of execution time (ms): Throttled Algorithm [8] and Proposed Algorithm

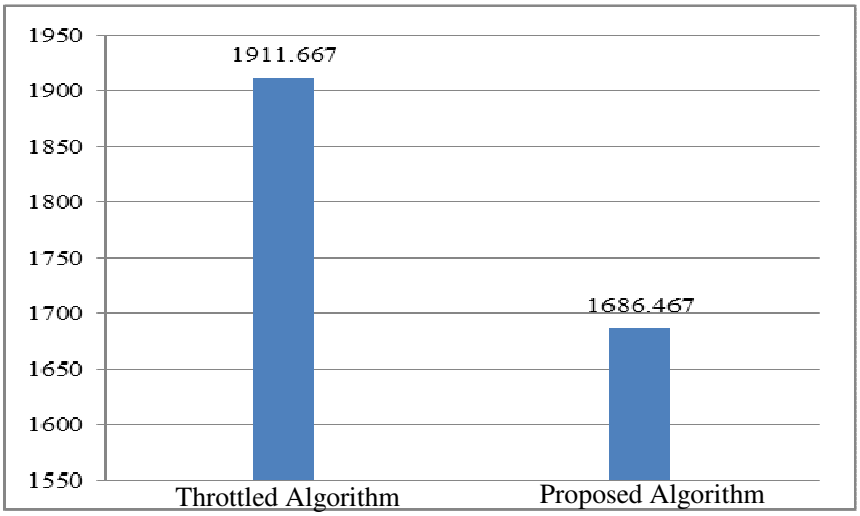

Figure 3. Comparison of response time (ms): Throttled Algorithm [8] and Proposed Algorithm

From simulation results and comparison charts between average excution time and average response time of Throttled algorithms [8] and our proposed algorithm, we see that average excution time and average response time of our proposed algorithm has been improved than Throttled algorithm [8].

4. CONCLUSION

Our algorithm is proposed from the Throttled algorithm [8]. In the Throttled algorithm [8], the authors pay attention to the amount of load that virtual machines are making. In the proposed algorithm, in addition to concentrating on the load, the researcher is able to perform the tasks / requirements of the virtual machine. In the cloud environment, the distribution of load between virtual machines is heterogeneous in terms of processing power, so that each virtual machine can have different processing time costs. For efficient load balancing, choose which virtual machines cost the least processing time to assign tasks. Our proposed algorithm was improved and inherited from the throttled algorithm [8] and was tested in the Cloudsim cloud computing environment and used in the Java programming language. In this article we use the same schedule as Spaceshared - Timeshared with virtual machines and tasks. From Figures 2 and 3 we find that the response time and average processing time of the algorithm are significantly improved compared to the Throttled algorithm. In the future, we will consider consider the security of the load on cloud computing.

REFERENCES

[1] Jasmin James, Dr. Bhupendra Verma (2012), "Efficient vm load balancing algorithm for a cloud computing environment", *International Journal on Computer Science and Engineering (IJCSE),* pp. 1658-1663.

[2] Hiren H. Bhatt and Hitesh A. Bheda (2015), "Enhance Load Balancing using Flexible LoadSharing in Cloud Computing", *International Conference on Next Generation Computing Technologies (NGCT)*, pp.72-76.

[3] Mayanka Katyal, Atul Mishra (2013),"A Comparative Study of Load Balancing Algorithms in Cloud Computing Environment", *International Journal of Distributed and Cloud Computing*, Volume 1 Issue 2, pp.5-14.

[4] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov ,C´ esar A. F. De Rose and Rajkumar Buyya (2010), "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", *Software: Practice and Experience (SPE)*, Volume 41 Number 1, pp.23-50.

[5] Rashmi. K. S, Suma. V, Vaidehi. M (June 2012), "Enhanced Load Balancing Approach to Avoid Deadlocks in Cloud", *Special Issue of International Journal of Computer Applications on Advanced Computing and Communication Technologies for HPC Applications – ACCTHPCA*, pp.31-35.

[6] Rajwinder Kaur, Pawan Luthra (2014), "Load Balancing in Cloud System using Max Min and Min Min Algorithm", *International Journal of Computer Applications Proceedings on National Conference on Emerging Trends in Computer Technology (NCETCT- Number 1)*, pp.31-34.

[7] Navtej Singh Ghumman, Rajwinder Kaur (2015),"Dynamic Combination of Improved Max-Min and Ant Colony Algorithm for Load Balancing in Cloud System", *6th International Conference on Computing, Communication and Networking Technologies (ICCCNT).*

[8] Hafiz Jabr Younis (2015), "Efficient Load Balancing Algorithm in Cloud Computing", *Islamic University Gaza Deanery of Post Graduate Studies Faculty Of Information Technology.*

[9] Shubham Sidana, Neha Tiwari (2016), "NBST Algorithm: A load balancing algorithm in cloud computing", *International Conference on Computing, Communication and Automation (ICCCA)*, pp. 1178 – 1181, IEEE Conference Publications.

[10] Atyaf Dhari and Khaldun I. Arif (2017), "An Efcient Load Balancing Scheme for Cloud Computing", *Indian Journal of Science and Technology*, Vol 10 (11), IEEE Conference Publications.

Authors

Nguyen Xuan Phi was born in Vietnam in 1980.
He received Master in Posts & Telecommunications Institute of Technology in Ho Chi Minh, Vietnam, 2012, major in Networking and Data Transmission. Currently he is a PhD candidate in Information System from Post & Telecommunications Institute of Technology, Vietnam. He is working at the Information Technology Center of AGRIBANK in Ho Chi Minh city, Vietnam. His main research areas are load balancing on cloud computing, optimizing the performance of cloud computing.

Tran Cong Hung was born in Vietnam in 1961.
He received the B.E in electronic and Telecommunication engineering with first class honors from HOCHIMINH University of technology in Vietnam, 1987. He received the B.E in informatics and computer engineering from HOCHIMINH University of technology in Vietnam, 1995. He received the master of engineering degree in telecommunications engineering course from postgraduate department Hanoi University of technology in Vietnam, 1998. He received Ph.D at Hanoi University of technology in Vietnam, 2004. His main research areas are B – ISDN performance parameters and measuring methods, QoS in high speed networks, MPLS. He is, currently, Associate Professor PhD. of Faculty of Information Technology II, Posts and Telecoms Institute of Technology in HOCHIMINH, Vietnam.