International Journal of Grid and Distributed Computing
Vol.3, No.1, March, 2010

A Trust Evaluation Model for QoS Guarantee in Cloud Systems*

Hyukho Kim, Hana Lee, Woongsup Kim, Yangwoo Kim

Dept. of Information and Communication Engineering, Dongguk University
Seoul, 100-715, South Korea
{hulegea, lhn1007, woongsup, ywkim}@dongguk.edu

Abstract

Cloud computing is a new paradigm in which dynamically scalable virtualized computing resources are provided as a service over the Internet. However, since resources are limited, it is very important that cloud providers efficiently provide their resources. This paper presents a trust model for efficient reconfiguration and allocation of computing resources satisfying various user requests. Our model collects and analyzes reliability based on historical information of servers in a Cloud data center. Then it prepares the best available resources for each service request in advance, providing the best resources to users. We also carried out experiments for reliability analysis with 4 data types, including an all data set, random data set, recent data set, and the data set within a standard deviation. As a result, using our trust model, cloud providers can utilize their resources efficiently and also provide highly trusted resources and services to many users.

Keywords: *Cloud Computing, Grid Computing, Virtualization, Trust Model*

1. Introduction

Cloud computing [1], [2] is a new computing paradigm composed of Grid computing and Utility computing concepts together. It provides dynamically scalable virtualized computing resources as a service over the Internet and users pay for as many resources as they have used. The Cloud itself is a network of virtualized servers or virtual data centers that can deliver powerful applications, platforms, and infrastructures as services over the Internet. Actually Cloud computing already has been used for web mail, blog, web hard storage service and web hosting services. However, due to limitations in software technologies and network bandwidth in the past, Cloud computing could not guarantee service levels and scope that needed to be delivered over the Internet. Nowadays, Cloud computing can provide various levels of service and functions over the Internet, as software and network technologies develop [3].

Cloud computing has various advantages as follows: 1) Improved performance, virtualized servers in a Cloud computing system will boot up faster and run faster, because each virtualized server has fewer programs and processes loaded into memory; 2) Lower IT infrastructure costs, instead of investing in larger number of powerful servers, users can borrow and use the computing resources of the Cloud to supplement or replace internal computing resources. They need no longer have to purchase servers

* This work was supported by a Ministry of Education, Science and Technology (MEST) grant funded by the Korea government(S-2009-A0004-00006, S-2009-A0104-0001).
This work was supported by Dongguk University Research Fund.
Yangwoo Kim (Corresponding Author)

to handle peak usage levels; 3) Unlimited storage capacity, the Cloud offers virtually limitless storage capacity. Whatever user needs to store, the user can; 4) Less maintenance, Cloud computing can reduce both hardware and software maintenance including data center space and electricity; 5) Improved compatibility, universal access to documents, collaboration, dynamically shared computing power, etc. However, Cloud providers should efficiently provide their resources to users because resources in Clouds are limited [4]. Hence, we propose a trust model for efficient reconfiguration and allocation of resources according to the various user requests. The reliability of computing resource means availability of a computing resource considering its performance and status. In order to provide resources with high reliability, we need an accurate way of measuring and predicting usage patterns of computing resources whose patterns are changing dynamically over time. Our trust model aims to reconfigure servers dynamically and allocate high quality computing resources to users. The proposed trust model in this paper uses the history information of nodes in the Cloud environment. This information consists of each node's spec information, resources usage, and response time. Then the model analyzes this information and prepares suitable resources on each occasion, and then allocates them immediately when user requests. As a result, Cloud system can provide the best resources and high-level services based on the analyzed information and it is possible to utilize resources efficiently.

The rest of this paper is organized as follows. In Section2, we present the trust model's formation and implementation. In Section 3, we explain the experiments and analyze experimental results. And finally, we conclude in Section 4.

2. Trust Model Implementation

Our trust evaluation model aims to configure the complex set of services dynamically in a cloud environment, according to the predictive performance in terms of stability and availability of all resources that are to be provided as cloud services. Therefore, it is very important to build an adequate trust model for prediction of service's performance and stability.

Our trust evaluating model in the cloud environment is set as follows: $<S, T, R, C, D, U>$, where S is the service consumer (request), T is a set of time slot units where a day is divided by n, such as $T = \{T_1, T_2, \ldots, T_n\}$, R is {a set of resources} $\cup$ {a set of services}, $C: R \rightarrow V$ denotes the capacity of each resource/service and means that the capacity of each resource is represented as a vector of Integer V, $D: S \rightarrow V$ denotes the demand function of how much resource a service demands to complete the request and is represented as a vector of Integer V, and $U: R \times T \rightarrow [0,1]$, where U denotes the degree of resource availability at a particular time slot based on the requested service's demand capacity and is represented as a percentage. According to the above model, our trust model focuses on predicting the operational availability (noted as U in the above model) during particular periods. Based on this prediction, we could estimate the operating availability for each resource on demand during particular time slots. So a system could configure services dynamically and distribute tasks efficiently in such way that a system minimizes task failure and task migration rate. To this end, we utilize the statistical usage history for the pre-evaluation of each resource in order to make the correct prediction of service availability in the Cloud environment.

In our trust model, we employ Probabilistic Latent Semantic Analysis (pLSA) [5] methodology to estimate the availability of each service/resource provision from the history of statistical usage data. Using pLSA, we can estimate system availability during specific periods and hence we can allocate resources with a minimum failure rate and support a more reliable cloud computing environment. We formalize the pLSA based trust model (equation 1) as follows.

$$P(r_i \mid s_j, u_i) = \sum_{z \ni Z} P(r_i \mid z, s_j) P(z \mid u_i) \quad (1)$$

where u_i is the service request, s_j is the resource ID, c_i is the minimum resource requirements for a service request u_i, r_t is the available capacity for the each time slice, and z is hidden space used for pLSA methodologies. If the expected satisfaction of users gets through this formula, the expected performance of the integrated services or functions can be anticipated, and the best services can be selected. For the purpose of performance prediction, various methods [6] are proposed and Petri Net [7], Process Algebra [8], And the Markov Chain [9] are the most common approach for performance prediction. However, these methodologies work under the assumption that the system clearly knows probabilities of the performance of service provision, which is almost impossible in a cloud environment. Therefore there are desperate needs for estimating resource based performance prediction methodologies specifically for a cloud environment. Therefore, in this paper, we only focus on the resources reliability analysis. In addition, even though pLSA methodology can provide accurate prediction performance, pLSA intrinsically requires huge computation power. That could give us a large burden of system overhead. So we have to filter out unnecessary candidate resources to save pLSA's computational overhead. To this end, we must also develop a utility model and corresponding ranking algorithm to filter out unnecessary candidate service providers. Our approach helps to reduce overall computational overhead and enable fast analysis of node performance prediction, and hence provide trustworthy resource allocation and service provisioning in a better way.

3. Experiments

This section describes how to configure the Cloud system, and algorithms for performance evaluation, performance metrics used, as well as the experimental results.

3.1. System Configuration

As you can see in Figure 1, the proposed Cloud system environment in this paper consists of virtual machines, physical machines, and an SLA (Service Level Agreement) manager. Physical machines are the basic available resources and it is assumed that there are a limited number of physical machines. The physical machines provide a set of virtual machines which are configured dynamically according to user requests. When the limited physical machines are provided to users from a pool of resources, the provided resources have two types; one is the dedicated resources and the other is the undedicated resources to give some extra margin in case of sudden request rise as shown in the slash regions of Figure 1. In this Cloud system environment, if a new user requests resources when all of the resources are already assigned, then the undedicated resources allocated to others are provided to the new users via dynamic

reconfiguration. At this time, providing the undedicated resources to new users shouldn't affect other dedicated resources' performance but it also should provide stable and high performance resources to new users. Therefore, in this paper we present mechanisms which sort high performance resources by analyzing the history information of the undedicated resources for providing highly trusted resources dynamically when the user requests arise.

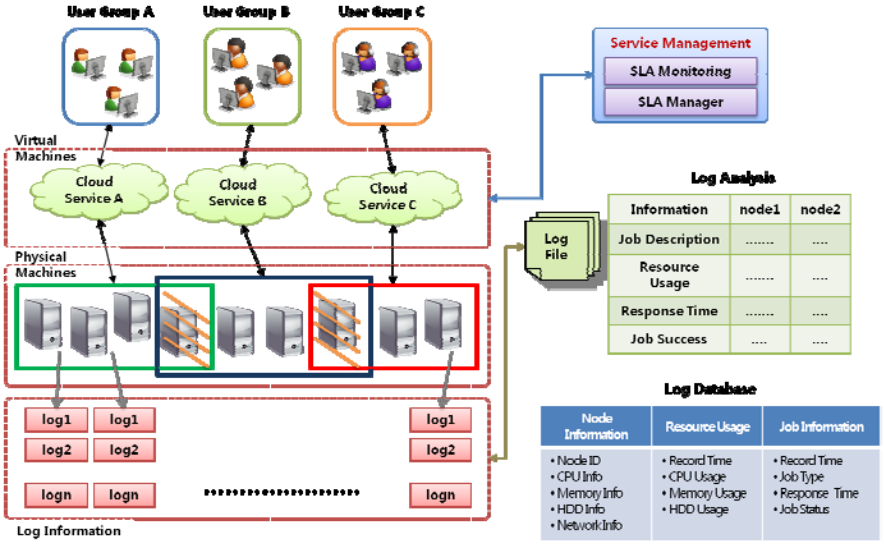

Figure 1. Overall System Configuration

3.1. Performance Metrics

We proposed mechanisms that sort high performance resources by analyzing the history information of the undedicated resources for providing highly trusted resources dynamically when the user requests arise. This is aimed to provide highly trusted resources to users based on the analysis of an idle server's history information together with the proposed algorithms. To maintain the current status of the system and provide the requested resources additionally, we analyze each node's log information at a regular interval so that we can sort and rank the resources and provide the best resources to a user as soon as possible. We designed and implemented the proposed algorithms with the Java programming language and for the experiments; we assumed that there exist 100 undedicated nodes in the slash regions shown in Figure 1. We also assumed that each node's log information is recorded at the same time interval which is 15 seconds. However, gathering real status log data from 100 real machines every 15 seconds is an unreasonable task so we randomly generated data for node specification (high, medium, low) and the resource usage, etc.

For performance analysis, we used resource/service usage information, which consists of node spec profile, average resource usage information, average response time, and average task success ratio. The node spec profile includes CPU type and frequency, memory size, hard disk capacity and the transmission rate of networking

devices. The average resource usage information consists of the current CPU utilization rate, memory status and available hard disk capacity. To analyze the average response time and the average task success ratio, we applied our measurement to four task execution types - addition, subtraction, multiplication and division. We also measured the response time for calculation with eight ciphers which are created through a random function. We also regard the task failure if the response time is zero and hence we calculated the average success ratio of tasks.

We used the expression (2) for calculating the average resource utilization in terms of a node's capacity.

$$R_{i,t,usage} = \sum_{j \in activity(i)} \frac{R_{j,t}}{R_{i,t,capacity}} / K_{i,t} \qquad (2)$$

where i is node ID, j is a single usage activity in the system, t is a specific period of a day, *activity(i)* is a set of all the resource usage activities at node i, K_i is the total number of resource usage activities at node i, K_j is the amount of resource usage from the activity j, and $K_{i,capacity}$ is the resource capacity of node i.

The best node is defined as a utility function utilizing four performance measures – node spec profile, average resource usage, average response time, and average task success rate – described above. We used the expression (3) to estimate the best node (Higher G_i from (3) implies better performance).

$$G_i = w_{Rc} Rc_i + w_{Ru} Ru_i + w_T T_i + w_S S_i \qquad (3)$$

where G_i is the utility of the node i, w_x is the weight for each term x (performance measure) of G_i, Rc_i is the resource spec profile of node i, Ru_i is the average resource usage of node i, T_i is the average response time of node i, and S_i is the average task success ratio. To calculate G_i, we extract all the available data from our system usage history and convert them to a normalized format with domain range [0, 1] such that each piece of resource information has effects on the utility G_i in fair way. In addition, we either assigned the same weight to each of four performance measures, or placed larger value to one of four weight coefficients while the three other three performance measures were assigned equal. G_i has the maximum value 1, and when we calculate with the same weight, w is 0.25. Otherwise, G_i is calculated using the weight 0.2 from the other 3 information types if one information type has the weight 0.4. After we obtain G_i, we then have node ranking information, and can select a few best nodes. Then we can obtain $R_{i,t,usage}$ (see equation (2)) for such nodes and convert it to r_i used in our pLSA based trust model (equation (1)) to predict the degree of service availability.

3.1. Experimental Results

We used 4 different types of data sets for a node's log analysis including an all data set, random data set, recent data set and the data set within a standard deviation. However, the node's spec information is fixed and doesn't change so it is not affected by the various data types used.

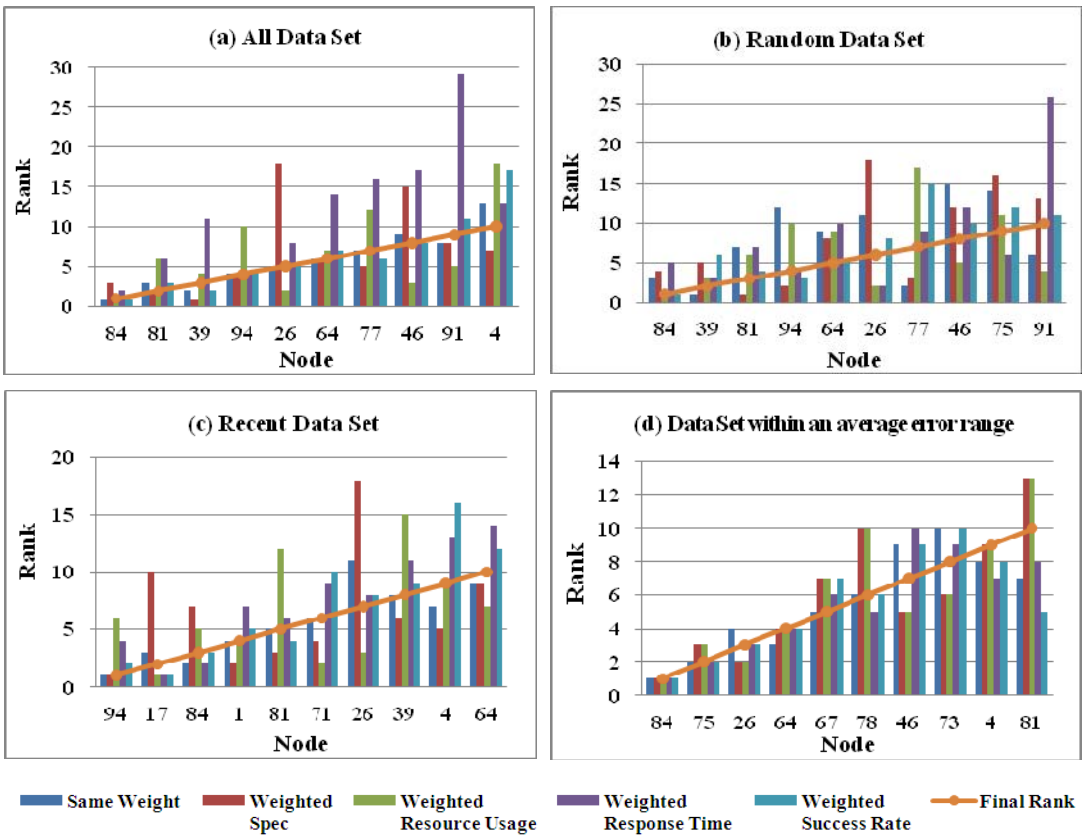

Figure 2. The Result of Experiments with the Different Data Set

(Case 1) All data set

The first experiment is aimed at analyzing the average resource usage and the task processing capability without considering specific time slot. As shown in Figure 2 (a), this is the result of experiments with all data gathered for 7 days, showing the top 10 ranked nodes selected by the log analysis algorithm. Each bar in the graph shows the ranking for each node when we changed the weight from 0.25 to 0.4 for each affecting factor. As shown (a), node 84 ranked the highest among all the nodes when we experimented with the all data set. Moreover, node 84 ranked high even when the weight of each factor was changed. On the other hand, node 91 ranked high overall even if it ranked low when the weight for response time was raised. This is because its basic spec is high and the current resources usage is low compared to the others.

(Case 2) Random data set

Figure 2 (b) shows the result of experiments with the random data set. A random data set consists of the specific amount of log data randomly extracted from the all data set. In this case we have similar result to case 1. This is because we selected data randomly from the all data set whose data is also randomly generated. Therefore, the random characteristic exists in both data sets. However, the result should be different if we

conduct the experiment in a real computing environment in which real log data is collected.

(Case 3) Recent data set

Each node cannot maintain a consistent usage pattern over time. That is, the usage pattern changes over time, and the recent data set may reflect the future usage pattern better while reducing time spent on analysis. For this experiment, we only used data gathered for the last 24 hours. Figure 2 (c) shows the top 10 ranked nodes that resulted from the experiments with the recent data set. The experimental results with the recent data set are little different from the previous two cases. In this experiment, node 94 which was out of the top 10 in previous cases ranked on top. It shows that node 94 can provide most stable performance based on the recent data set while node 84 whose rank was first in the previous two cases only ranked fifth because the recent resource usage increased.

(Case 4) The data set within an average error range

In this experiment, we conducted experiments using the data set whose data values are within the average error range excluding nodes which are unstable (for example, unexpected load increase or system shutdown, etc). The average error range is the range in which the standard deviation value is added and subtracted from the average value. We excluded some nodes that have substantial margins of error. As a result, we used 80 nodes out of 100 nodes, excluding 20 nodes. Results of the experiment with the data set whose values are within an average error range are shown in Figure 2 (d). We can see on this experiment that weight factors result in ignorable effects, compared to the previous cases. We have smaller log value deviation than other cases because nodes whose error range are beyond the average are excluded. As a result, node 84 ranked the top among all nodes because it had the best capacity on average.

An Overall Ranking

Based on the above analysis of 4 data sets, we made the overall ranking as shown in Figure 3. The final top ranked node 84 has 4GHz CPU, 4.096GB of memory, 300GB of hard disk capacity and 100Mbps of Ethernet network capacity. The average resource usage is also quite stable around 20%, and the change in resource usage is also small, under the average standard deviation whose value is 20. Although node 84 ranked relatively low in the experiment with the recent data set, it achieved the top rank overall because the average resource usage is low on average and its deviation is also small. On the other hand, the ranking for node 67 is not so good in terms of the basic spec information and the average resource usage, but it was able to rank 9th because the change in resource usage is smaller than others, below the standard deviation. Similarly, node 44 maintained a better ranking when analyzed with the recent data set because its' recent memory utilization is 30% less than that with older data.

Table 1. The final top 10 ranked nodes

Node	All Data Set	Random Data Set	Recent Data Set	Data set Within an average error range	Final Ranking
84	1	1	3	1	1
81	2	3	5	10	2
64	3	5	10	4	3
26	5	6	7	3	4
46	8	8	15	7	5
77	7	7	16	13	6
4	1	12	9	9	7
27	13	14	11	20	8
67	14	11	20	5	9
44	12	1	13	23	10

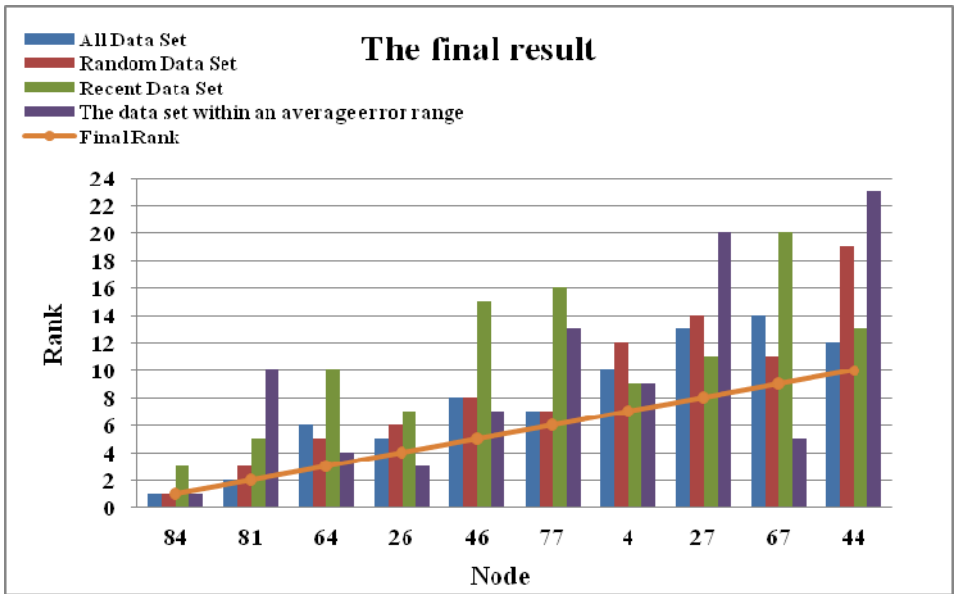

Figure 3. The Final Results of Experiment

As a result, we found that all the nodes within the top 10 kept higher and better rankings than others no matter whatever data set is used. In our experiments we collected a diversity of information at regular intervals such as the node's basic spec, resources usage, response time, and average task success ratio. Those saved history information were then classified as the all data set, random data set, recent data set, and data set within the standard deviation. Since we evaluated and ranked the undedicated nodes with the various data sets, dynamic resource allocation considering recent resource status as well as resource availability is possible for highly trustable Cloud services.

4. Conclusions

Author names and affiliations are to be centered beneath the title and printed in Times New Roman 12-point, non-boldface type. Multiple authors may be shown in a two or three-column format, with their affiliations below their respective names. Affiliations are centered below

each author name, italicized, not bold. Include e-mail addresses if possible. Follow the author information by two blank lines before main text. Cloud computing is a new computing paradigm composed of Grid computing and Utility computing concepts together. It provides dynamically scalable virtualized resources as a service over Internet and user pay as much as they used. Cloud itself is a network of virtualized servers or virtual data centers that can deliver powerful applications, platforms, and infrastructures as services over the Internet. Actually Cloud computing already has been used for web mail, blog, web hard storage service and web hosting services. However, due to limitations in software technologies and network bandwidth in the past, Cloud computing could not guarantee service levels and scope that needed to be delivered over the Internet. Currently, however, Cloud computing can provide various levels of service and functions over the Internet, as software and network technologies develop.

The Cloud system consists of many commodity servers and provides virtualized resources to users. However, it needs to reconfigure virtualized resources dynamically when the user requests increase unexpectedly. So we proposed the trust model which analyzes the history information of each node and allocates reliable resources according to user requests. It can efficiently utilize the limited resources in the Cloud environment and provide reliable Cloud services to users. It also has the advantage of providing the requested resource immediately because it prepares and selects highly efficient nodes by analyzing the history information of each node. We experimented on reliability analysis with a diversity of data sets, including the all data set, random data set, recent data set, and data set within the standard deviation. By doing so, we can increase the reliability of overall Cloud system by providing highly trustable computing resources.

References

[1] R. Buyya, Chee Shin Yeo, and S. Venugopal, “Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities”, Proc. Of the 10th IEEE International Conference on High Performance Computing and Communications, 2008, pp. 5-13.
[2] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner, “A Break in the Clouds: Towards a Cloud Definition”, ACM SIGCOMM Computer Communication Review, Volume 39, Issue 1, 2009, pp. 50-55.
[3] Hong Li, Jeff Sedayao, Jay Hahn-Steichen, Ed Jimison, Catherine Spence and Sudip Chahal, “Developing an Enterprise Cloud Computing Strategy”, Korea Information Processing Society Review, Volume 16, Issue 2, 2009, pp. 4-16.
[4] Mladen A. Vouk, “Cloud computing — Issues, research and implementations, Information Technology Interfaces”, 30th International Conference (ITI 2008), 2008, pp. 31–40.
[5] Thomas Hofmann, “Probabilistic Latent Semantic Indexing”, Annual ACM Conference on Research and Development in Information Retrieval, Proceedings of the Twenty-Second Annual International SIGIR Conference on Research and Development in Information Retrieval (SIGIR-99), 1999, pp. 50-57.
[6] Agnieszka SzymańskaKwiecień, Jan Kwiatkowski, Marcin Pawlik and Dariusz Konieczny, “Performance Prediction Methods”, Proceedings of the International Multiconference on Computer Science and Information Technology, 2006, pp. 363–370.
[7] Petri Nets World, http://www.informatik.uni-hamburg.de/TGI/PetriNets/
[8] J. C. M. Baeten, “A brief history of process algebra”, Theoretical Computer Science archive Volume 335, Issue 2-3,2005, pp. 131–146.
[9] Gokhale and Trivedi, “Analytical Models for Architecture-Based Software Reliability Prediction: A Unification Framework”, IEEE Transactions Volume 55, Issue 4, 2006, pp. 578–590.

Authors

Hyukho Kim

He received the B.A. degree in Information and Communications Engineering from Daejeon University in 2003. He received the M.S degrees from Dongguk University in 2005. Since 2005, he has been with Dongguk University as a Ph. D student. His current interest areas are “Grid Computing”, “Cloud Computing” “P2P Computing” and “Parallel and Distributed Computing”.

Hana Lee

She received the B.A degree from Dongguk University in 2008. Since 2009, she has been with Dongguk University as a M.S student. Her current interest areas are “Grid Computing”, “Cloud Computing” and “Parallel and Distributed Computing”.

Woongsup Kim

He received the B.A. degree in Computer Engineering from Seoul National University in 1998. He received his M.S. degree in Computer and Information Science from University of Pennsylvania, and Ph.D. degrees in Computer Science from Michigan State University in 2001 and 2006, respectively. Since 2007, he has been an Assistant Professor of Department of information communications engineering at Dongguk University. Research interests include software engineering, semantic web and service oriented architecture.

Yangwoo Kim

He received the M.S and Ph.D degrees from Syracuse Univ. in 1986 and 1992 respectively. Since 1996, he has been with Dongguk University as a professor. His current research interest areas are "Grid Computing", "Cloud Computing" "P2P Computing" and "Parallel and Distributed processing".