International Journal of Grid and Distributed Computing
Vol. 9, No. 10 (2016), pp.89-98
http://dx.doi.org/10.14257/ijgdc.2016.9.10.08

Using Queue Model to Evaluate the Reliability in Cloud Platforms

Zhichao Liu[1*] and Ze Xiao[2]

[1]Network Information Center, Hunan Institute of Engineering, 411104, Hunan Province, China
[2]Department of Computer Technology and Application, Qinghai University, 810016, Qinghai Province, China
[1]ygfeng1974@126.com;[2]xiaoze1992@gmail.com

Abstract

In recent years, cloud platforms have been widely applied in various areas to provide flexible service for users. However, how to evaluate the reliability of cloud platforms still remains a challenging issue because of its complexity and dynamical characteristic. In this paper, we present a novel reliability evaluation model which is aiming at providing a technique to accurately evaluate the reliability of virtualized cloud platforms. The proposed model is deprived from the classical tree-structure based reliability model and incorporates some new mechanisms to overcome the shortcomings of previous models. Extensive experiments are conducted to investigate the effectiveness of the proposed model. The results shown that, comparing with the existing approaches, the proposed model can significantly improve the accuracy of reliability evaluation, especially when the target cloud platform is in presence of intensive workloads.

Keywords*: Cloud Computing; Reliability Model; Virtualization; Queue System*

1. Introduction

Cloud computing has emerged as a promising platform which allows users to deploy their applications in an environment with increased scalability, availability, and reliability [1-2]. So it has been widely considered as the next generation of distributed processing platform and network computing paradigm in the future [3-4]. To maintain desirable performance, cloud resource providers tend to equip their systems with more and more advanced IT devices, which consequently increases the system's complexity and makes it difficult to evaluate the reliability of large-scale cloud platforms if not impossible.

Recently, many cloud platforms are using virtualization technology to deploy their distributed resources [5]. From the perspective of cloud providers, virtualization technology provides an effective approach to extending the capability of their cloud infrastructures without increasing too many IT devices [6]; on the other side, it also increases the difficulty of reliability evaluation, because the lower-level physical resources are multiplexed by virtualized software through a multi-layer framework, which makes it almost impossible to accurately evaluate the reliability of virtualized resources [7-8]. Here, we summarize the challenges of evaluating the reliability of large-scale cloud platforms as following:

- Heterogeneity of resources results in various types of resource fault, which can hardly be taken into consideration in a single reliability model [7-8].
- Failure model is greatly influenced by the system workloads, which often fluctuate dramatically and being unpredictable during the runtime [9-10].

[*]Corresponding Author

- When using virtualization technology, multiple resources are interplayed with each other in a dynamical manner, which makes static reliability model inaccurate in practical cloud platforms [7-11].
- Due to the large-scale and the complexity of cloud platforms, reliability are difficult to model, analyze, and evaluate [12].

To address the above challenges, in this work we present a novel reliability evaluation model, namely *Reliability Model for Virtualized Clouds* (RMVC), which is deprived from the classical *Tree-structured Grid Reliability Model* (TGRM) proposed in [13]. TGRM has been proven to be effective for reliability evaluation in grid platform, in which resources are directly allocated to up-level applications. However, it can not be applied to cloud platforms, in which resources are firstly virtualized and then allocated to users. More importantly, the major shortcoming of TGRM is that it does not take into account the dynamic workload when calculating reliability. In our RMVC, we use the queueing model to describe the dynamic intensiveness of workloads, and present a technique to quantitatively evaluate the deadline-missing fault which is ignored by TGRM model.

The rest of this paper is organized as follows: Section 2 presents the related work. In Section 3, we analyze the shortcomings of TGRM, and present our RMVC. In Section 4, extensive simulations are conducted to verify the performance of the proposed model. Finally, Section 5 concludes the paper with a brief discussion of future work.

2. Related Work

With the growth of distributed computing platforms (*i.e.*, grid, cloud, cluster), it will be important that systems can provide high reliability. In particular, it will be critical to ensure that grid systems are reliable as they continue to grow in scale, exhibit greater dynamism, and become more heterogeneous in composition. In [14], the authors surveyed the work on grid reliability that has been done in recent years and reviews progress made toward achieving these goals and identified important issues and problems that researchers are working to overcome in order to develop reliability methods for large-scale, heterogeneous, dynamic environments. This work also illuminated reliability issues relating to standard specifications used in grid systems, identifying existing specifications that may need to be evolved and areas where new specifications are needed to better support the reliability.

In [15], Tian *et. al.,* presented a scheduling algorithm for workflow applications, in which finite-state continuous-time Markov process is applied for selecting a resource combination scheme which has the lowest expenditure under a certain credit level of the resource reliability on the critical path in the DAG-based workflow. In [16], the authors proposed CPN (Coloured Petri Nets) based modelling pattern formally described the process of task distribution and execution within the grid environment. Also it proposed a method for evaluation the grid service reliability based on the analysis of the model. In addition, an instance of the proposed model for a sample grid environment is constructed and analysed using CPN Tools. In [17], Zhang *et. al.*, studied the stability and reliability of grids, and described a quantitative investigation of the stability and reliability of grids with a focus on cascading failures under external disturbances. In [18], Thierion *et. al.*, proposed to simulate an ensemble of potential hydrological scenarios in order to support the forecaster's decision-making process. The developed applicative layer takes advantage of the computing capabilities of grid technology, significantly enhancing the reliability of grid resources. In [19], Tang *et. al.*, designed a hierarchical reliability-driven scheduling architecture that includes both a local scheduler and a global scheduler. The local scheduler aims to effectively measure task reliability of an application in a grid virtual node and incorporate the precedence constrained tasks' reliability overhead into a heuristic scheduling algorithm. In [20], Bawa *et. al.*, analyzed the combination of FTA (Fault Trace Archive) and GWA (Grid Workload Archive), and their results indicated that interactive jobs have a higher failure probability as compared to batch jobs.

Beside the above studies on grid reliability, reliability of cloud systems are also presented in recent years. For example, Qiu *et. al.*, proposed a reliability-based optimization framework, named ROCloud, to improve the application reliability by fault tolerance. Their first algorithm ranks components for the applications that all their components will be migrated to the cloud. The second algorithm ranks components for hybrid applications that only part of their components are migrated to the cloud. Both algorithms employ the application structure information as well as the historical reliability information for component ranking [21]. In [22], the authors designed and implemented an reliability enhanced storage scheme for Hadoop platforms, which is able to automatically conduct real time synchronization for data files duplicated among different Hadoop clusters.

3. Analysis and Design of Reliability Model

3.1. Background and Motivation

In Tree-structured Grid Reliability Model (TGRM), the target grid system is considered as a tree-like structure as shown in Figure 1. In such a system, service/resource integration, request, and management are controlled by the RMS (Resource Management System). A grid service is desired to execute a certain task under the control of the RMS. The RMS receives a service request from a user, and assigns the service task to available resources for execution. After the resources finish execution of the assigned subtasks, they return the results back to the RMS; and then the RMS integrates the received results into an entire task output, which is requested by the user.

The tree structure allows the common cause failures in shared communication channels to be modeled. Taking the Figure 1, as an example, the failure in channel L6 makes resources R1, R2, and R3 unavailable. This type of common cause failure was ignored by the conventional parallel computing models, and the star topology models. For small-area communication, such as a LAN or a cluster, such assumption that ignores the common cause failures on communications is acceptable because the communication time is negligible compared to the processing time. However, for wide-area communication, such as the grid system, it is more likely to have failure on communication channels. Therefore, the communication time cannot be neglected. In many cases, the communication time may dominate the processing time due to the large amount of data transmitted. Therefore, TGRM is a desirable model representing the functioning of grid services.

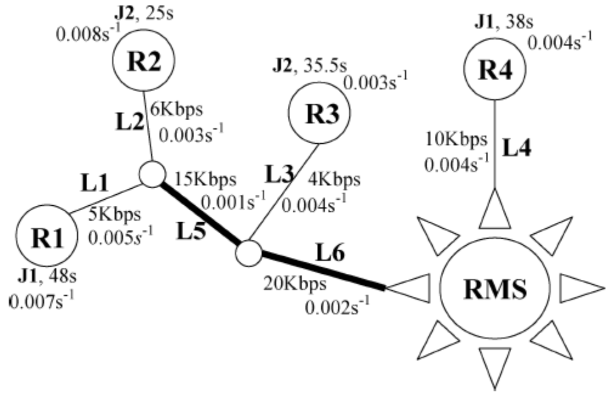

Figure 1. Tree-Structured Grid Reliability Model

In order to simplify the evaluation of reliability, TGRM make some assumptions as following: (1) RMS is completely reliable, which means there will be no faults in RMS;

(2) RMS receives tasks, assigns them to available resource, and integrates the received results for user's tasks. All the intermediate data are assembled on RMS, and then being sent back to users (3) Each available resource can only provide service for a single user's task. Based on the above assumptions, TGRM gives the expressions to calculating the reliabilities of the resources and the links as following:

$$RF_j^k = e^{-\varpi_k \cdot (\frac{c_j}{x_k} + \frac{a_j}{s_k})} \tag{1}$$

$$LF_j^k = e^{-\pi_k \cdot (\frac{c_j}{x_k} + \frac{a_j}{s_k})} \tag{2}$$

where RF_j^k is the probability that sub-task j is successful completed on resource k, LF_j^k is the probability that there is no occurrence of link fault during the execution of sub-task j, ϖ_k and π_k are the mean fault rates of the resource k and the links, c_j is the size of sub-task j, a_j is the amount of data transmitted between resource k and RMS when sub-task j is running, x_k is the computing speed of resource k, s_k the minimal bandwidth of all the network links between resource k and RMS. Therefore, the reliability of a task is that

$$\text{Reliability} = \max_{1 \le j \le m}[\max_{k \in w_j}(RF_j^k \cdot LF_j^k)] \tag{3}$$

where m is the number of sub-tasks, and w_j is the set of selected resources for executing the task.

3.2. Design of RMVC

In the TGRM, it is obviously that the third assumption is too conservative and unsuitable. For example, multi-cluster systems are a typical class of cloud system that generally used in scientific fields. In multi-cluster systems, a high-performance cluster exposes its service interface to the out-side users in form of *single-image*. So, a cluster should be considered as a single resource with paralleling computing capability. This is also true for those *Massive Parallel Processor* (MPP) systems. To be compatible with TGRM, we use M/M/1 queuing system to model those resources that can only provide service for a single task at a time, and M/M/C queuing system for those resources with parallel capability. As noted above, queueing system has been proven to be able to precisely describe the working and workload model of distributed resources. So, we use queueing system to construct the working and workload model of cloud resources. By this way, we incorporate workload-aware mechanism into the TGRM. More importantly, when using queueing model to describe the working of virtualized resources, we can evaluate some kinds of failures (such as deadline-missing failure) that cannot be obtained by conventional reliability model. Such an advantage is of significant importance for evaluating the reliability of virtualized resources.

In addition, TGRM ignores the waiting time of jobs when calculating the reliability. It may be feasible when resources are abundant or workload is in low-level, but not suitable to high-level workloads which are common in cloud platforms. Another problem of TGRM is that it does not take into consideration the deadline constraint, which is often required by real-time tasks. For those real-time tasks, if it cannot be completed before the deadline the results of the tasks are of less or no value for users. So, deadline-miss should be considered a sort of resource failure. To overcoming the above shortcomings of TGRM, we relax the third assumption of TGRM to provide supports for parallel serving resources. Also, we introduce a new type of fault, called *Deadline-Miss Fault* (DMF), into TGRM to support the reliability evaluation of real-time tasks. Therefore, Equation (1), (2) and (3) listed previously is modified as following:

$$RF_j^k = e^{-\varpi_k \cdot (\frac{c_j}{x_k} + \frac{a_j}{s_k} + d_j)} \tag{4}$$

$$LF_j^k = e^{-\pi_k \cdot (\frac{c_j}{x_k} + \frac{a_j}{s_k} + d_j)} \tag{5}$$

$$\text{Reliability} = \max_{1 \le j \le m} [\max_{k \in w_j} (RF_j^k \cdot LF_j^k \cdot DMF_j^k)] \tag{6}$$

where d_j is the deadline requirement of sub-task j, DMF_j^k is the probability that there is no deadline-miss when sub-task j is scheduled on resource k. As shown in (6), the key issue of our MRVC is to find an approach to calculating the DMF_j^k, which is greatly affected by the workloads on resources.

In this paper, MRVC uses M/M/1 queueing system to model those resources that can only provide service for a single task at a time, and M/M/C queueing system for those resources with parallel capability (*i.e.*, cluster or MPP). Therefore, the approaches to calculate the DMF_j^k for these two types of resources are different. First, let $\boldsymbol{\psi}$ be the random variable representing the number of waiting tasks in R_k. According to queueing theory, the probability that there are m waiting jobs in R_k is

$$\begin{cases} \Pr\{\boldsymbol{\psi} = m\} = \begin{cases} \delta \cdot \dfrac{{\rho_k}^{m+C_k} \cdot C_k^{C_k}}{C_k!}, & m > 0 \\ \displaystyle\sum_{n=0}^{C_k} \delta \cdot \dfrac{(\rho_k \cdot C_k)^n}{n!}, & m = 0 \end{cases} \\ \delta = \left[\displaystyle\sum_{n=1}^{c_i} \dfrac{(\rho_i \cdot c_i)^n}{n!} + \dfrac{(\rho_i \cdot c_i)^{c_i}}{c_i} \dfrac{1}{1-\rho_i} \right]^{-1} \end{cases} \tag{7}$$

For M/M/C_k queueing system, the service rate is $C_k \cdot \mu_k$, which means the system can complete $C_k \cdot \mu_k$ tasks in a unit time. So, the amount of tasks that R_k can complete in period d_j is $C_k \cdot \mu_k \cdot d_j$. Therefore, the probability that R_k can guarantee a task's deadline d_j is equal to the probability that the waiting tasks in R_k is not more than $C_k \cdot \mu_k \cdot d_j - 1$. That is $DMF_j^k = \Pr\{\boldsymbol{\psi} \le c_i \cdot \mu_i \cdot d_j - 1\}$. Combing the above analysis, we can get that

$$\begin{aligned} DMF_j^k &= \Pr\{\boldsymbol{\psi} \le C_k \cdot \mu_k \cdot d_j - 1\} = \sum_{m=0}^{C_k \cdot \mu_k \cdot d_j - 1} \Pr\{\boldsymbol{\psi} = m\} \\ &= \sum_{m=0}^{C_k} \delta \cdot \frac{(\rho_k \cdot C_k)^m}{m!} + \sum_{m=1}^{C_k \cdot \mu_k \cdot d_j - 1} \delta \cdot \frac{{\rho_k}^{m+C_k} \cdot C_k^{C_k}}{C_k!} \end{aligned} \tag{8}$$

If resource R_k is modeled as M/M/1 queueing system, According to queueing theory, the probability that there are m waiting jobs in R_k is

$$\Pr\{\boldsymbol{\psi} = m\} = (1 - \rho_k) \cdot \rho_k^m, \quad m \ge 0 \tag{9}$$

For M/M/1 queuing system, the service rate is μ_k , which means the system can complete μ_k tasks in a unit time. Similar to the analysis the analysis of M/M/C model, we can obtain that

$$DMF_j^k = \Pr\{\psi \le \mu_i \cdot d_j - 1\} = \sum_{m=0}^{\mu_k \cdot d_j - 1} \Pr\{\psi = m\} = \sum_{m=0}^{\mu_k \cdot d_j - 1} (1-\rho_k)\cdot \rho_k^m \qquad (10)$$

3.3. Analysis of Task Execution Time

The set of all resource nodes and links involved in performing the given task form a task spanning tree. This task spanning tree can be considered to be a combination of minimal task spanning trees (MTST), where each MTST represents a minimal possible combination of available elements (resources and links) that guarantees the successful completion of the entire task. The failure of any element in a MTST leads to the entire task failure. For solving the graph traversal problem, several classical algorithms have been suggested, such as Depth-First search, Breadth-First search, *etc*. These algorithms can find all MTST in an arbitrary graph.

However, MTST in graphs with a tree topology can be found in a much simpler way because each resource has a single path to the RMS, and the tree graph is acyclic. After the subtasks have been assigned to corresponding resources, it is easy to find all combinations of resources such that each combination contains exactly *m* resources executing *m* different subtasks that compose the entire task. Each combination determines exactly one MTST consisting of links that belong to paths from the *m* resources to the RMS. The total number of MTST is equal to the total number of such combinations *N*, which can be calculated by $\prod_{i=1}^{m}|w_i|$. It should be noted that a MTST completes the entire task if all of its elements do not fail by the maximal time needed to complete subtasks in performing which they are involved. Therefore, when calculating the element reliability in a given MTST, one has to use the corresponding record with maximal time. Gven the MTST and the times of their elements involvement in performing different subtasks, one can determine the PMF of the entire service time. We can obtain the conditional time of the entire task completion given the set of MTSTs is available as

$$T_{Exec} = \sum_{k \in w_j} \max_{1 \le j \le m} [\frac{c_j}{x_k} + \frac{a_j}{s_k}] \qquad (11)$$

Then, we can sort the MTST in an increasing order of their conditional task completion times T_{exec}, and divide them into different groups containing MTST with identical conditional completion time.

4. Experiments and Performance Comparison

4.1. Experimental Settings

The experiments are conducted by using CloudSim [23], which is a distributed resource management and scheduling simulator. The topology and IT-infrastructure settings are deprived from the test-bed DAS-2. The simulation cloud platform consists of twelve *Computational Elements* (CE_1,…,CE_{12}), each representing a high-performance cluster. The clusters are grouped into five groups by their geographical positions. Within each group, the clusters are connected by LAN ($Link_1$,…,$Link_{12}$). Then, they are connected by WAN ($Link_{13}$,…,$Link_{17}$) between groups. The failure of links follows exponential distribution with various parameters. According to the studies in [24], the failures of LAN link are different from the WAN's. So, we set that the LAN's fault rate limited in $[0.02, 0.04]$, and the WAN's fault rate limited in $[0.04, 0.12]$.

4.2. Evaluation on Accuracy

In this experiment, we mainly concentrate on the accuracy of our RMVC comparing with TGRM. As mentioned in Section 3.2, the differences between RMVC and TGRM are the reliability calculating formulas. So, the algorithms of reliability calculation and task scheduling in RMVC are as same as that in TGRM. In this experiment, we first calculate the reliabilities of tasks by using TGRM and RMVC respectively, then schedule the tasks onto resources. As to each task, there are three type of executing results: success, abort (resource fault or link fault), deadline-miss fault. The experimental results are shown in Figure 2, in which two curves represent the reliabilities calculated by TGRM and RMVC.

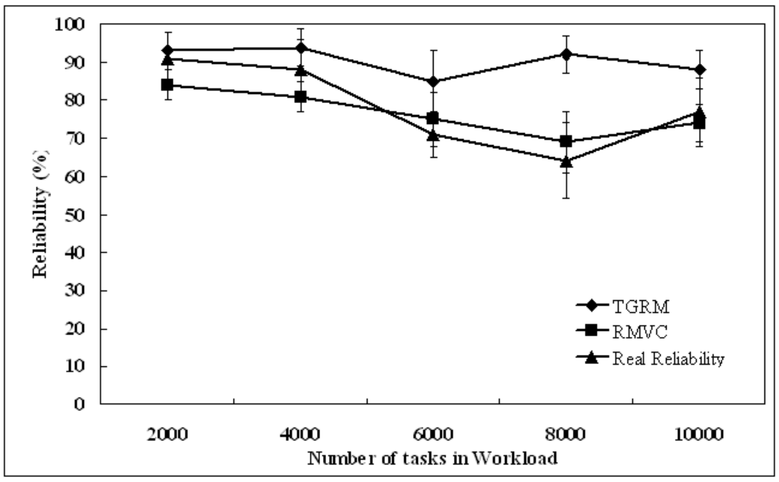

Figure 2. Comparison of the Accuracy of Reliability Evaluation

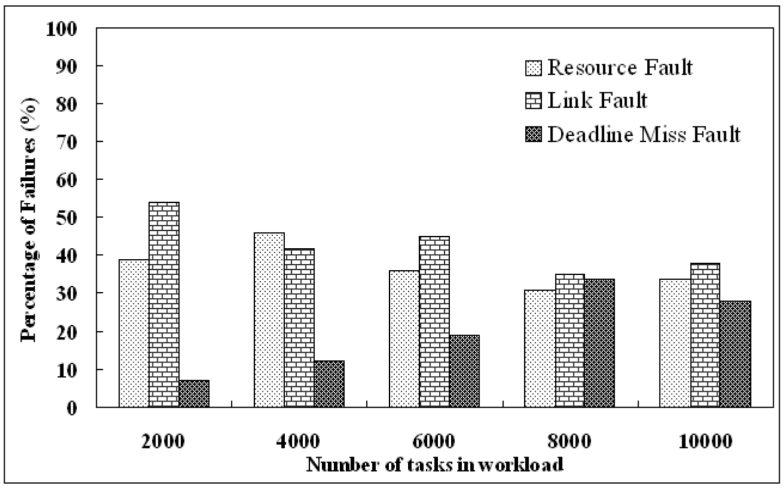

Figure 3. Percentages of Different Faults in the Simulation

We define the ratio of theoretic reliability to real reliability as the accuracy of TGRM or RMVC. As shown in Figure 2, the accuracy of TGRM is about 97% for the first 2000 tasks, and 93% for the second 2000 tasks. As to RMVC, they are about 92% and 91% correspondingly. However, the high accuracy of TGRM does not be maintained with the continuing of the simulation. In fact, TGRM's accuracy decreases dramatically for the third and the fourth 2000 tasks, which are only about 84% and 69%. On the contrary, RMVC's accuracy is always kept above 90%. In order to examine these results, we record all the causes of unsuccessful execution and their percentages, which are shown in Figure 3. As we can see, the percentages of resource fault and link fault are very high for the first 4000 tasks. However, percentage of deadline-miss fault keeps increasing with the continuing of the experiment, which becomes the highest (about 34%) when executing the 6000th - 8000th tasks.

The detailed observation of the simulation shows that most of the tasks are scheduled onto low-load resources at the beginning. With the increasing of new arriving tasks, the workload of resources gradually becomes heavier and heavier. As a result, the waiting time of tasks increased consequently, which causes higher deadline-miss fault. As TGRM does not take the waiting time into account, nor dose it consider deadline-miss fault as a type of fault, its accuracy inevitably decreases in presence of high deadline-miss fault just as shown in Figure 2. On the contrary, RMVC overcomes those TGRM's shortcomings. As a result, TGRM's accuracy can always be kept in high-level in various cases, which indicates that RMVC is more adaptive than TGRM in practice cloud systems.

4.3. Comparison of Scheduling Performance

In this experiment, we investigate the scheduling performance of TGRM and RMVC in term of *Mean Response Time* (MRT). Although the scheduling algorithm of both TGRM and RMVC are the same, their scheduling schemes (*Minimal-Task-Spanning-Tree, MTST)* for individual tasks are different, because their reliability evaluation formulas are different. The experimental results are show in Figure 4.

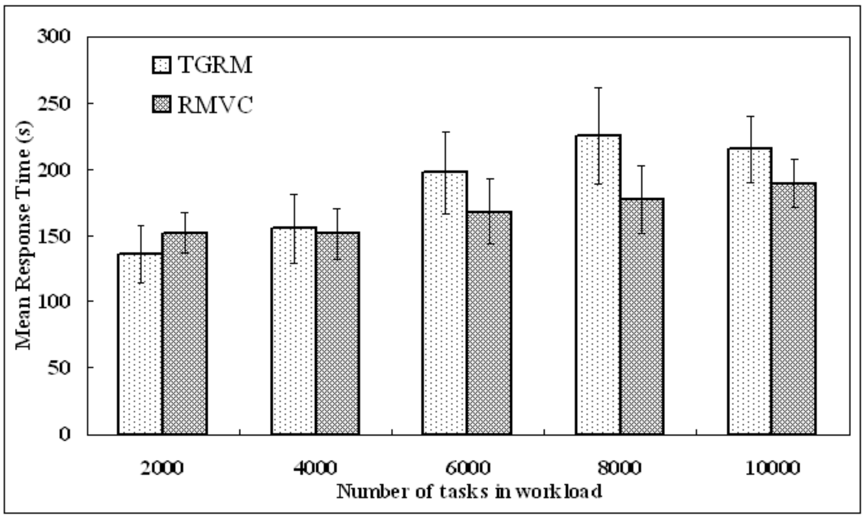

Figure 4. Percentages of Different Faults in the Simulation

The results show that the mean response time of RMVC is about 20% lower than TGRM since after executing the first 4000 tasks. The detailed observation of the simulation shows that the MTSTs generated by RMVC often obtains many low-load resources. As noted above, DMF_j^k is an important factor when calculating reliability in RMVC. The effects of DMF_j^k on the results of reliability evaluation are influenced by the resource's workload,. More specifically, when workload is low the effects of DMF_j^k is also lower, and vice versa. So, when the system is in face of the heaviest workload (during the 6000th – 8000th tasks), RMVC is more effective to be aware such heavy workload and generate better scheduling scheme.

5. Conclusion

In this paper, we propose a reliability evaluation model for virtualized cloud platforms (RMVC), which is deprived from the conventional TGRM. In the proposed model, queueing system is applied to describe the workload distributed resources. In order to evaluate the reliability of jobs with constrain to deadline, a new type of resource fault (Deadline-Miss fault) is introduced to the proposed model. In the simulations, we compare the performance of RMVC with TGRM in terms of accuracy. The experimental results show that RMVC can provide more accurate reliability evaluation when cloud platform in presence of high-level workload. This indicates that RMVC is more adaptive than TGRM in practice cloud systems. Also, simulations are conducted to examine the performance of reliability-based job scheduling. Experimental results show that RMVC is able to reduce the jobs' mean response time about 15% because its ability to generate workload-aware scheduling schemes. Currently, our RMVC mainly concentrates on the reliability evaluation when allocating resources. In some practice cloud systems, advance reservation is an effective technique that generally used to improve the reliability of co-allocation from multi-institutions. So, our future work is to take advance reservation into account when calculating resources' reliability. More specific, we plan to introduce another type of fault, namely Reservation-miss Fault, into our RMVC design.

References

[1] R. Clarke, "User requirements for cloud computing architecture", Proc. of IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. Melbourne, Australia, **(2010)**, pp.625-630.
[2] S. Bera and R. Misra, "Cloud Computing Applications for Smart Grid: A Survey", IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 5, **(2012)**, pp. 1477-1494.
[3] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Future Generation Computer Systems, vol. 25, no. 6, **(2009)**, pp.599-616.
[4] A. Polze and P. Troeger, "Trends and challenges in operating systemsufrom parallel computing to cloud computing", Concurrency and Computation-Practice & Experience, vol. 24, no. 7, **(2012)**, pp. 676-686.
[5] F. Lombardi and R. Di Pietro, "Secure virtualization for cloud computing", Journal of Network and Computer Applications, vol. 34, no. 4, **(2011)**, pp. 1113-1122.
[6] A. Beloglazov and R. Buyya, "Energy Efficient Allocation of Virtual Machines in Cloud Data Centers", Proc. of IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Melbourne, Australia, **(2010)**, pp. 577-578.
[7] Z. Zibin, T. C. Zhou, M. R. Lyu and I. King, "Component ranking for fault-tolerant cloud applications", IEEE Transactions on Services Computing, vol. 5, no. 4, **(2012)**, pp. 540-550.
[8] B. Yang, F. Tan and Y. -S. Dai, "Performance evaluation of cloud service considering fault recovery", Journal of Supercomputing, vol. 65, no. 1, **(2013)**, pp. 426-444.
[9] C. Wang, Q. Wang, K. Ren, N. Cao and W. Lou, "Toward Secure and Dependable Storage Services in Cloud Computing", IEEE Transactions on Services Computing, vol. 5, no. 2, **(2012)**, pp. 220-232.
[10] B. Yang, F. Tan and Y. S. Dai, "Performance evaluation of cloud service considering fault recovery", Journal of Supercomputing, vol. 65, no. 1, **(2013)**, pp. 426-444.
[11] J. Li, B. Li and T. Wo, "CyberGuarder: A virtualization security assurance architecture for green cloud computing", Future Generation Computer Systems, vol. 28, no. 2, **(2012)**, pp. 379-390.
[12] B. Addis, D. Ardagna, B. Panicucci, M. S. Squillante and L. A. Zhang, "Hierarchical Approach for the Resource Management of Very Large Cloud Platforms", IEEE Transactions on Dependable and Secure Computing, vol. 10, no. 5, **(2013)**, pp. 253-272.
[13] Y. S. Dai and G. Levitin, "Reliability and performance of tree-structured grid services", IEEE Transactions on Reliability, vol. 55, no. 2, **(2006)**, pp. 337-349.
[14] C. Dabrowski, "Reliability in grid computing systems", Concurrency and Computation-Practice & Experience, vol. 21, no. 8, **(2009)**, pp. 927-959.
[15] G. Z. Tian and J. Yu, "Towards critical region reliability support for Grid workflows", Journal of Parallel and Distributed Computing, vol. 69, no. 12, **(2009)**, pp. 989-995.
[16] M. A. Azgomi and R. Entezari-Maleki, "Task scheduling modelling and reliability evaluation of grid services using coloured Petri nets", Future Generation Computer Systems, vol. 26, no. 8, **(2010)**, pp. 1141-1150.
[17] L. Chang and Z. Wu, "Performance and reliability of electrical power grids under cascading failures", International Journal of Electrical Power & Energy Systems, vol. 33, no. 8, **(2011)**, pp. 1410-1419.

[18] V. Thierion and P. A. Ayral, "Grid Technology Reliability for Flash Flood Forecasting: End-user Assessment", Journal of Grid Computing, vol. 9, no. 3, **(2011)**, pp. 405-422.
[19] X. Tang and K. Li, "A hierarchical reliability-driven scheduling algorithm in grid systems", Journal of Parallel and Distributed Computing, vol. 72, no. 4, **(2012)**, pp. 525-535.
[20] R. K. Bawa and R. Singh, "Correlation based combined analysis of FTA and GWA for reliability and performance study of grid environment", International Journal of Grid and Distributed Computing, vol. 7, no. 2, **(2014)**, pp. 53-68.
[21] W. Qiu and Z. Zheng, "Reliability-based design optimization for cloud migration", IEEE Transactions on Services Computing, vol. 7, no. 2, **(2014)**, pp. 223-236.
[22] T. Yeh and H. Lee, "Enhancing availability and reliability of cloud data through syncopy", Proc. of IEEE International Conference on Internet of Things, **(2014)**, pp. 126-133.
[23] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Software-Practice & Experience, vol. 41, no. 1, **(2011)**, pp. 23-50.
[24] J. Li and D. Liu, "K-pancyclicity of k-ary n-cube networks under the conditional fault model", IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 6, **(2012)**, pp. 1115-1120.

Authors

Zhichao Liu received the M.S. degree in computer science from the Xiangtan University in 2010. Currently, he is an advanced network engineer in HP High-performance Network Centre in Hunan Institute of Engineering. His research interests include cloud computing, distributed resource management.

Ze Xiao is currently studying in Qinhai University majoring in Computer Science and Technology. His research interests include distributed systems and cloud computing.