Journal of Engineering and Applied Sciences 11 (6): 1263-1268, 2016
ISSN: 1816-949X

An Efficient Scheduling Algorithm to Reduce the Number of Virtual Machine Migration in Cloud Environment

[1, 2]Abd Sura Khalil, [1]S.A. R Al-Haddad, [1]Fazirulhisyam Hashim,
[3]Azizol B HJ Abdullah and [4]Salman Yussof
[1]Department of Computer and Communication Engineering, UPM, Selangor, Malaysia
[2]Diyala University, Baquaba, Iraq
[3]Department of Computer Science and Information Technology, UPM, Selangor, Malaysia
[4]Department of Information Technology, UNITEN, Selangor, Malaysia

Abstract: As a result of rising demands to cloud environments, investigators are motivated to develop the system to be more profitable for the service providers and efficacious for the service's users. One of these developments is virtualization technology which utilized to enhance the resource utilization. However, the cloud operational expenses have been gradually increasing by the increasing amount of consumed electrical energy. Thus, lowering the consumed energy is required to consolidate virtual machines dynamically to be operated on minimal physical hosts' number utilizing live VM migration. However, using live migration can lead to various issues such as traffic congestion and security. In this study, genetic scheduling algorithm is developed to present more efficient VM assignment and hence more efficient system. The proposing mechanism is able to reduce the number of VM migration while maintaining the system efficiency.

Key words: Cloud computing, genetic algorithm, genetic operator, live vm migration, scheduling mechanism

INTRODUCTION

In datacenters, physical hosts are usually underutilized due to luck in efficient load balancing which back to choose the suitable scheduling and resource allocation techniques. Thus, to improve the energy efficiency, the utilization percentage in these hosts should be improved. Migrating the VMs or tasks to limited group of the datacenter's machines and switch off the rests is a common technique to improve the utilization which is called VM or workload consolidation (Maio *et al*., 2016). VM migration is widely utilized in consolidation since it able to move the state of running VMs among PMs, so the workload aredynamically adjusted (Sedaghat *et al*., 2016).

In spite of the considerable impacts the VM migration have on consumed energy, it has usually not been taken into account when presenting energy reduction mechanisms (Sun *et al*., 2016). Recently, for improving energy efficiency, researchers has started to focus on VM consolidation based on migration technique to lower the utilized PMs' number (Jung *et al*., 2016; Ye *et al*., 2015). Despite the benefits of this technique in improving energy efficiency, it can lead to many issues (Ahmad *et al*., 2015).

Significant performance degradation can be incurred when co-locating VMs caused by the shared resources' competition like networks and caches. The degradation level varies while VMs running various workloads are co-located.

Various migration overhead levels can be incurred via migrating VMs with various workloads. Hence, the workload performance is also degraded.

Another issue is security which considers the main concern and has not received its due attention (Huang *et al*., 2016). An attacker can gain control over VM during migration or recover migrating data by capturing the traffic. Besides, the traffic among the hypervisors can be sniffed or exposed to man in the middle attack or denial-of-service. Data integrity, confidentiality and essential credentials can be lost easily besides losing the available resources for that specific VM. After losing vital data, all VMs may be switched off by their host as an attack protection method (Ahmad *et al*., 2015).

Trying to avoid these issues by presenting performance improving techniques or adding a security system through VM migration can add extra burden to the system and degrade the system performance by consuming extra time for processing, creating a confliction with the advantage of using live VM migration which is moving VMs without disconnecting the client (Fernandes *et al*., 2014). Therfore, reducing the number of VM migration is the best solution to avoid or decrease the

Corresponding Author: Abd Sura Khalil, Department of Computer and Communication Engineering, UPM, Selangor, Malaysia

possibility of all the above issues, yet this reduction should be done in a way that does not affect the system efficiency. Improved scheduling technique can be proposed to allocate VMs efficiently with minimal migration and that is the core of this study.

Literature review: Lately, researchers concentrated mostly on reducing datacenter energy consumption producing different techniques to achieve their goals. One of these popular techniques is VM migration. They considered the benefits outcomes from migrations in term of load balance, energy consumption and preserve system performance through migration by used live migration. However, a few of them currently have addressed the issues result in using migration and tried to propose a new methods that can maintain the energy efficiency without depending on migration or at least decrease the number of needed VM migrations. Some of these mechanisms are discussed in this study.

Shaw and Singh (2015) presented a proactive and reactive hotspot detection mechanism to decrease the VM migration number. Mostly, VM migration occurred in two situations: migrates all the host's VMs and shut it down when it is underutilized, besides for hotspot mitigation. In case the host is overloaded, then rather than directly migrated some of its VMs, the authors examined whether the migration was urgently demanded or not. To achieve right migration decision, they presented a load prediction algorithm. Once it was achieved, the algorithm searched for an appropriate destination host where the VM shifted based on its probable future load. They used cloudsim to accomplish their aims and compared their results with some previous study.

On the other hand, Khyyam Shahzad, Arif Iqbal Umer and Babar Nazir (Shahzad *et al*., 2015) addressed the impacts of frequent migrations on the netstudy congestion and operational cost. Their target was to lower the VM migration utilizing a scheduling mechanism to balance the loads. They modified NVMMP algorithm based on VM priority. Their suggested algorithm assisted in decreasing netstudy overhead via migrating exclusively non-critical VMs and leaving the critical ones in first phase. Critical VMs are the VMs that have shortest task accomplishment time while the non critical VMs have longest task accomplishment time left. In this way, they reduced the number of VMs migration and hence reduced netstudy congestion. The authors also used CloudSim to test their system as the previous ones.

While Baruchi *et al*. (2015) suggested postponing the live migration by utilizing the information extracted from identifying the VM studyload cycles. They claimed that their algorithm could reduced the amount of data transferred over the netstudy to 43% and decrease the live migration time to 74% compared to classical strategies of consolidation which performed live migration without taking the VM studyloads in consideration.

However, they did not address the effect of reducing the migration on lowering the consumed energy percentage. Thus, Seema Vahora, Ritesh Patel, Hetal Patel and Sandipkumar Patel by Rathor *et al*. (2015) explained how using VM migration could lead to tradeoff between energy utilized in migration versus energy utilized during studyload. Their aim was to lower the trade off via migrating VM efficiently to proper host. They found that it was significant and necessary to deal with three main VM situations: how VM allocated to host in a way did not make it overloaded in case of host overloaded which VM should be chosen for migration to where migrated VM allocated or reallocated. They claimed that by answering these three queries, VMs can be appropriately managed. Hence, the number of VM migration and the overall cloud computing system energy were decreased. Also, cloudsim is used to experiment their technique.

Proposed study: In this study, an explanation for Deoxyribonucleic Acid (DNA) and genetic algorithms is presented. In addition, the proposed combined approach of these two mechanisms is elaborated explaining how it will reduce the VM migration.

DNA computing: Molecular computing, or commonly known as DNA computing is a mechanism employed for processing parallelly a enormous amount of computations based on ground breaking (Boruah and Dutta, 2015). DNA is a double stranded sequence of four nucleotides usually called bases: Adenine (A), Cytosine (C), Guanine (G), besides Thiamine (T). In 1953, the known chemical structure of double DNA helix was discovered by Francis Crick and James Watson. The structure showed that DNA consists of a specific bond involving two linear bases sequences based on complementarily feature such as: Adenine bonds with Thiamine (A-T) or (T-A), Cytosine bonds with Guanine (C-G) or (G-C). These bonds are called Watson-Crick complementarily.

Every DNA strand has two various ends defined DNA polarity. The double helix is an anti-parallel bonding of couple complementary strands. The mechanism functions via encoding the issue to be processed in DNA language: A, T, C and G values. By using these bases, any conceivable issue can be tackled as in Turing machine tape. Chemically, any potential sequence can be generated in a test tube on huge amount of various DNA strands, then the proper ones can be selected utilizing genetic engineering methods (Wang *et al*., 2013). The

massive parallelism DNA feature is employed to produce and experiment the prospective combinations which are considered an optimal solutions to balance the load, hence reduce VM migration.

Genetic algorithm: Genetic mechanism considers a sort of optimal search algorithms that imitate genetic technique and biological evolution. The main notion involves mutate, chromosome, cross besides natural selection. A population of solutions is included via the mechanism and evolved via employing reproduction, mutation and other operators (Ye, 2015).

After several evolution generations, the most proper solutions will be picked while the others will be abstracted. The procedure of genetic algorithm is (Wei and Tian, 2012):

- The population is initialized
- Each individual is evaluated with the function of fitness
- The best individual is selected and the termination conditions are determined. When the conditions are met, the algorithm is stopped. Otherwise, go on to point 4
- Individuals are selected for recombination
- The new individuals are mutated and new generation is produced. Back to point 2

MATERIALS AND METHODS

Proposed combination: In the proposed study, we depend on testing the requested tasks if it complex computational tasks or not. If the requested task is a complex computational one, then instead of migrating the VM to another host to process the tasks in case the exiting VMs of a host cannot handle the new task, we propose an improved mechanism that depends on dividing the new task to multi sub tasks. These sub tasks scheduled to different VMs to be processed depending on their deadlines and VMs studyload.

The proposed scheduling technique relies on improving the genetic mechanism by performing the generated sub tasks as DNAs. The genetic mechanism is applied using initial population of randomly generated DNAs. Figure 1 presents the procedure of the proposed mechanism. Generally, there are some organize steps that should follow when a new task (Tk) requested by the cloud user enters the cloud computing environment. First, the algorithm tests the VMs of a host if they have adequate capability for tasks processing. If they have then directly assigns the new Tk into the VMs. If they do not have then Tk is splitted to set of subtasks (Stk^{pl}) allocated into multiple VMs. The procedure of this process is explained as follow.

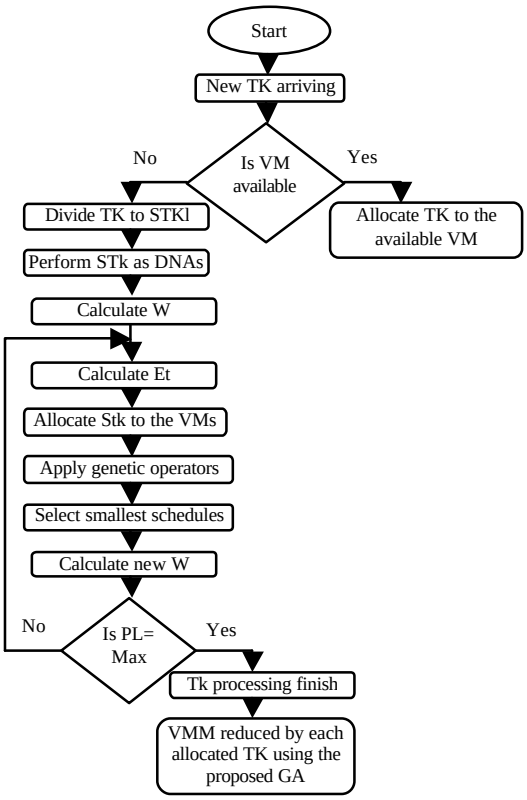

Fig. 1: Algorithm procedure

During allocation process, precedence level (Pl) is employed based on the genetic mechanism to schedule subtasks (STk^{pl}) of a Tk in which each one of them treated as an independent Tk. As the STk^{pl} need to be scheduled before the Tk deadline, task studyload (W) and Executing time ($Et(STk^{l}_{j})$) consider two significant parameters to deal with. The W over a PM is calculated as follow:

$$W = \frac{\sum_{a=1}^{b} TK_a}{Pt \times MIPS} \quad (1)$$

From Eq. 1, Pt is the total number of processing components presented in PM, MIPS is the million instruction per second employed to handle single Tk and Tk_a is the size in MIPS of a^{th} waiting subtask in the task queue of b length on a PM.

At first Pl, only one STk^{Pl} is presented to be allocated to VMs of PM. The scheduling value of allocated Tk is added into the list due to its precondition for the upcoming Pl which contains more than one STk^{Pl}_{j} as shown in Fig. 1.

At the second Pl, multiple STk^{Pl} are presented as DNAs generated using initial operator. The roulette wheel selection is employed to pick the best among the randomly generated DNAs in which the best is scheduled into VMs. To elaborate the procedure more in detail, the subtask sequence is firstly arranged in priority basis. Then, the Pl is assigned as 1 and the subtask sequence is chosen for Pl 1. In case only single Tk is available, assign it to the VM has minimum W.

If more than one is presented at the next Pl, then the proposed improved scheduling is introduced. The random generator is activated to produce an initial DNAs population. A specified DNA is selected based on roulette wheel selection. Then, all STK^{Pl}_j existing in Pl 1 are assigned to set of VMs. Et (Stk^j Pl) of DNA is estimated which is equal to the total execution time of all the STk^{Pl}_j existing at Pl 1. Ft (STk^j Pl) is estimated as follow:

$$Et\left(STk_j^{Pl}\right) = \min_{i\in[o,j]}\left\{W_n^{STk_{ji}^{Pl}}\right\} \quad (2)$$

W of node n which is the node to which subtask is assigned in any schedule is varied when scheduling the subtask. The new studyload $W_n^{STk_{ji}^{Pl}}$ on node n is:

$$W_n^{STk_{ji}^{Pl}} = MEF + Ld_n^{STk_{ji}^{Pl}} \quad (3)$$

where, $Ld_n^{STk_{ji}^{Pl}}$ is load on n of subtask i at Pl. By specifying the VMs studyloads and the needed executing time for each subtask based on the original task deadline, the generated sub tasks scheduled to be allocated to the selected VMs based on W and Et(STk^j Pl) parameters. However, related to the proposed technique, various schedules will be generated due to different DNAs combinations. These schedules should be minimized to the most appropriate ones that outcome better allocation that assist in processing the full requested task in multiple VMs of a host and avoiding VM migration.

As mentioned before, the roulette wheel selection is employed to select a specified m schedules out of an initial DNAs. According to this, the second generation of schedules will be gained from the antecedents. In specific Pl, two schedules are selected from m schedules. Then, new two schedules are gained from them via applying genetic operators such as crossover and mutation on the selected first two schedules. Hence, m new schedules are gained for the second generation. Furthermore, we minimize the schedules range via picking the ones which have smallest Et and employ them for tasks allocation. Then, we update the next scheduling list according to them. These steps are applied for all utilized Pl. Finally, when Pl reaches the maximum level presented based the original requested Tk that schedule would represent the total execution time of the Tk. The process that is performed through the precedence level is demonstrated in Fig. 2.

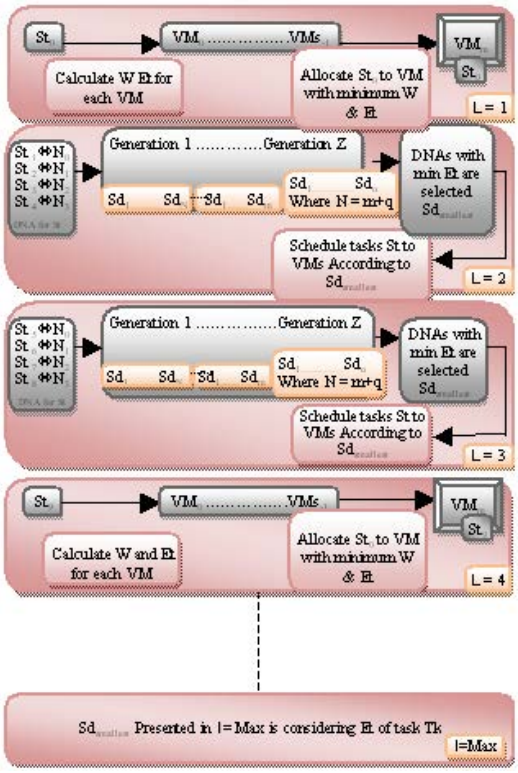

Fig. 2: Precedence level: process overflow

When VMs of a PM do not have enough capability to handle a complex computational user requested Tk in the cloud environment, the migration is taken over to find an appropriate VM that can process the new task. In case the migration process is performed frequently, then the system would face many issues such as traffic congestion and performance degradation. However, in the proposed method, this type of requested Tk in case cannot handle by VMs of a host is divided to set of STk allocated to multiple VMs depending on their capabilities (W and Et) instead of performing VM migrating to avoid task failing.

RESULTS AND DISCUSSION

Experimental evaluation: In this study, number of VM migration is tested based on the number of different number of activated hosts and available VMs. CloudSim tool is used to perform the experiment. The number of available physical machines is ranged from 50, 100, 150 and 200. Each physical machine involves one CPU core with 1500 or 2000MIPS, 8G Ram and 1G available storage memory.These physical machines include 500 VMs with 1000 cloudlet. Each VM is provided with one CPU core with 250, 500, 750, 1000 MIPS.

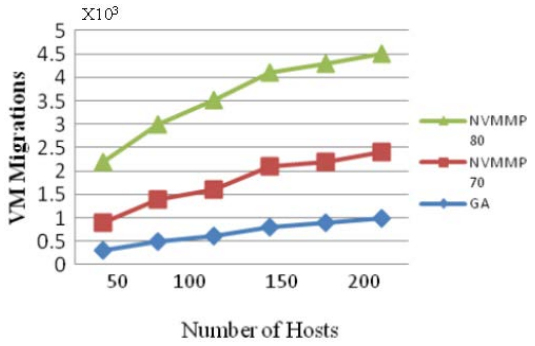

Fig. 3: Number of VM migration and comparison of VM

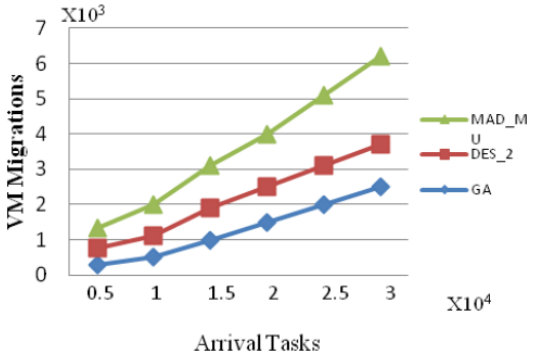

Fig. 4: Number of VM migration

In addition, the provided RAM for each VM equal to 250 MB. Dynamic various task requests are receiving to the cloud environment by different users. The results of this proposing study (GA) are compared with NVMMP70 and NVMMP80 techniques (Shahzad *et al*., 2015). As shown in Fig. 3, GA accomplishes less VM migration with different host range comparing to NVMMP70 and NVMMP80 techniques. The reason behind this result is that NVMMP technique depended on classifying the task to critical and non critical based on their processing done ratio to reduce migration. However, they did not consider the task deadline which can cause task failure if the processing time overrides the deadline. In the proposed improved GA, this issue is avoided by taking the task deadline in consider when executing the arriving task.

Another comparison is shown in Fig. 4. In this test the number of VM migration is calculated based on the number of the arrival tasks. The proposed GA is compared with MM and DES_2 techniques from Shaw and Singh (2015). In this experiment, the number of arriving task is important and the used range started from $(0.5\text{-}3)\times10^4$. As shown in the figure the proposed improved GA records maximum 2500 VM migration while MAD-MU technique reach to 6000 VM migrations by the maximum amount of the arrival tasks. In DES_2 algorithm, a load prediction mechanism is used to check if the migration is required or not and to predict future destination load and thus the migration is reduced. On the other hand, MAD-MU algorithm used a predicted utilization threshold to manage the migration process.

Generally, both techniques depended on predicting the utilization or studyload ratio of a server to lower the migration process. However, improved GA does not need to do any prediction mechanism or specify a utilization threshold as it divides the task to multi sub ones that need less execution time and less VM studyload. Hence, the number of VM migration is decreased.

CONCLUSION

One of the most important mechanisms in virtualization technology is live VM migration where it enables a VM to be migrated from a PM to another without notable execution interruption. More complicated policies can be implemented inside cloud system by employing this mechanism such as optimizing computational resources besides improving the quality-of-service. Nevertheless, it can cause several issues to VM application and service provider infrastructure such as netstudy congestion, security issue and sever performance degradation.

RECOMMENDATIONS

This study proposed a new method to reduce the number of required live VM migration by improving an efficient genetic scheduling algorithm. CloudSim is the simulation tool that used to test the algorithm due to its ability to support on-demand virtualization. Testing the study using CloudSim proves that the proposed algorithm presents better results in reducing the number of VM migration comparing to some recent study in this area based on the number of the host and the number of arrival tasks. The future study includes expanding and improving the proposed scheduling technique to achieve better results and cover other performance metrics such as decreasing processing time and testing netstudy traffic.

ACKNOWLEDGEMENT

The study is supported by the University Putra Malaysia Grant GP-IPS/2015/9462000.

REFERENCES

Ahmad, R.W., A. Gani, S.H.A. Hamid, M. Shiraz and F. Xia *et al*., 2015. Virtual machine migration in cloud data centers: A review, taxonomy and open research issues. J. Supercomputing, 71: 2473-2515.

Baruchi, A., E.T. Midorikawa and L.M. Sato, 2015. Reducing virtual machine live migration overhead via workload analysis. IEEE. Latin Am. Trans., 13: 1178-1186.

Boruah, K. and J.C. Dutta, 2015. Twenty years of DNA computing: From complex combinatorial problems to the Boolean circuits. Proceedings of the 2015 International Conference on Electronic Design, Computer Networks and Automated Verification (EDCAV), January 29-30, 2015, IEEE, India, Assam, ISBN:978-1-4799-6207-5, pp: 52-57.

Fernandes, D.A., L.F. Soares, J.V. Gomes, M.M. Freire and P.R. Inacio, 2014. Security issues in cloud environments: A survey. Int. J. Inf. Secur., 13: 113-170.

Huang, T., Y. Zhu, Y. Wu, S. Bressan and G. Dobbie, 2016. Anomaly detection and identification scheme for VM live migration in cloud infrastructure. Future Gener. Comput. Syst., 56: 736-745.

Jung, J., K. Kim and H. Kim, 2016. On the necessity of VM migration: Simulation on datacenter network Resources. Wirel. Personal Commun., 86: 1797-1812.

Maio, D.V., R. Prodan, S. Benedict and G. Kecskemeti, 2016. Modelling energy consumption of network transfers and virtual machine migration. Future Gener. Comput. Syst., 56: 388-406.

Rathor, V.S., R.K. Pateriya and R.K. Gupta, 2015. An efficient virtual machine scheduling technique in cloud computing environment. Int. J. Mod. Educ. Comput. Sci., 7: 39-46.

Sedaghat, M., R.F. Hernandez and E. Elmroth, 2016. Decentralized cloud datacenter reconsolidation through emergent and topology-aware behavior. Future Gener. Comput. Syst., 56: 51-63.

Shahzad, K., A.I. Umer and B. Nazir, 2015. Reduce VM migration in bandwidth oversubscribed cloud data centres. Proceeding of the 2015 IEEE 12th International Conference on Networking, Sensing and Control (ICNSC), April 9-11, 2015, IEEE, Abbottabad, Pakistan, ISBN:978-1-4799-8069-7, pp: 140-145.

Shaw, S.B. and A.K. Singh, 2015. Use of proactive and reactive hotspot detection technique to reduce the number of virtual machine migration and energy consumption in cloud data center. Comput. Electr. Eng., 47: 241-254.

Sun, G., D. Liao, V. Anand, D. Zhao and H. Yu, 2016. A new technique for efficient live migration of multiple virtual machines. Future Gener. Comput. Syst., 55: 74-86.

Wang, Z., D. Huang and C. Tang, 2013. Fast parallel algorithm to the minimum edge cover problem based on DNA molecular computation. Appl. Math., 7: 711-716.

Wei, Y. and L. Tian, 2012. Research on cloud design resources scheduling based on genetic algorithm. Proceedings of the 2012 International Conference on Systems and Informatics (ICSAI), May 19- 20, 2012, IEEE, Beijing, China, ISBN:978-1-4673-0198-5, pp: 2651-2656.

Ye, H., 2015. Information technologies optimization of resource scheduling based on genetic algorithm in cloud computing environment information technology. Metal. J., 6: 386-391.

Ye, K., Z. Wu, C. Wang, B.B. Zhou and W. Si *et al*., 2015. Profiling-based workload consolidation and migration in virtualized data centers. IEEE. Trans. Parallel Distrib. Syst., 26: 878-890