CRASH ME INSIDE THE CLOUD: A FAULT RESILIENT FRAMEWORK FOR PARALLEL AND DISCRETE EVENT SIMULATION

Asad W. Malik, Imran Mahmood

Center for Research in Modeling, Simulation & Vision (Crimson)
Department of Computing
School of Electrical Engineering and Computer Science (SEECS)
National University of Sciences and Technology (NUST)
Islamabad, Pakistan
E-mail: asad.malik@seecs.edu.pk, imran.mahmood@seecs.edu.pk

ABSTRACT

The adoption of cloud computing has increased over the years due to its pay-as-you-go model. The other important feature that cloud provides is scalability achieved through virtualization. The dynamic burst of users is served through virtual machines. The cloud data center comprises of pool of commodity hardware systems used to provide computing capacity on request. The use of commodity hardware increases the failure rate. Therefore, the execution of Parallel Discrete Event Simulation (PDES) over fragile cloud environment can lead to erroneous simulation results. Moreover, simply replacing faulty VMs is not the possible solution for PDES. In this paper, we exploit the execution of PDES codes over cloud environment and propose a fault resilient framework that dynamically handles the VM failure without restarting the entire simulation. Our benchmarking results demonstrate the effectiveness of the proposed protocol, compared with the widely used and traditional PDES protocol i.e. Time Warp.

Keywords: Virtual Machine, cloud computing, PDES, Time Warp, Fault Tolerant

1 INTRODUCTION

Cloud computing enhances the concept of utility computing by providing services over the network. The cloud computing adoption has increased very rapidly, IBM created first cloud computing center at WuXi, China (Shuai, Shufen, Xuebin, and Xiuzhen 2010). The on demand provisioning of resources, attracts various applications especially computing intensive internet applications such as Facebook and Twitter (Linquan, Chuan, Zongpeng, Chuanxiong, C, and M 2013) (Pandey, Linlin, Guru, and Buyya 2010). However, the execution of Parallel Discrete Event Simulation (PDES) over a cloud environment has relatively been less explored. Inside a cloud, the workloads are handled through Virtual Machines (VMs). The cloud infrastructure dynamically manages users request and scales the resources accordingly through VMs. The VMs are prone to fail during execution (Jielong, Jian, Kwait, Weiyi, and Guoliang 2012). Although VMs are copies of each other but under certain circumstances VM can crash, reboot or hardware/ software failure can occur. In general, the cloud provides computing utility. Therefore, cloud users expect smooth and reliable execution of their tasks irrespective of fluctuating workloads or any other failures within the clouds.

Typically, cloud manages VM failures by keeping spare VMs that are then added to facilitate users. However, such a solution does not favor the execution of the PDES codes. The typical design of a PDES comprises

of thousands of processes executing over number of systems (Jafer, Liu, and Wainer 2012). The processes communicate with each other through messages. The messages are time stamped, and must be processed in increasing timestamp order. Therefore, each of the process maintains a local data structure to handle out of order execution. As a result, simply replacing faulty VMs with the new VMs are not desirable. Alternatively, running the PDES codes over a cloud environment can lead to a restart of entire simulation model. Therefore, such a mechanism would increase the execution time and add to the utility to be paid by the cloud end users (Begum and Khan 2011). As per our literature review, limited work exists that focuses on the execution of PDES over a cloud environment, but no work has been done to cover the VM failures. This paper presents a fault resilient protocol that handles the VM failure during execution of discrete event simulation over a cloud environment.

The reminder of the paper is organized as follows: Section 2 provides the necessary background information on PDES. The related work is discussed in Section 3. This is followed by our proposed framework in Section 4. In Section 5, we present the in-depth analysis of the proposed framework and compared with traditional Time Warp. Finally, we concluded the work in Section 6.

2 PARALLEL DISCRETE EVENT SIMULATION

In the simulation domain, the term simulation is categorized into continuous and discrete event simulations (Fujimoto 2000). In continuous event simulation, the processes states are changed over the span of time; whereas the discrete event simulation is based on events generated at random intervals of time. The PDES paradigm is getting popularity among researchers due to its computational model that reduces the execution time as compared to the traditional sequential simulations. The PDES runs simulation in parallel over distributed systems (Fujimoto 2000). In the PDES, the tasks are divided into small units that are assigned to various processes executing on different or the same nodes. These processes are termed as Logical Processes (LPs). Each of the nodes holds multiple LPs to improve the system utilization and performance (Malik, Park, and Fujimoto 2010). The LPs communicate with each other by exchanging small size, time stamped Event Messages (M_E). The PDES is a collection of the logical processes distributed across network to efficiently utilize parallel execution. The typical execution of the PDES requires that each of the LP to store the receive events in its Input Queue (I_Q) for execution.

These events are executed based on the receive or timestamp order. In the receive order execution, events are executed in the order they are received; whereas, in the timestamp order, messages are executed in non-decreasing timestamp order. In the time stamped execution, messages are stored at the destination LP from where the smallest timestamp event is processed. Traditionally, there is a one to one relationship between the LP and the computing node. That is to say that, only a single LP resides on a machine; however, to utilize the parallel execution, a number of LPs are mapped to a single system and the outgoing messages are aggregated (Park and Fujimoto 2006). The LPs mapped on a single node, shares the I_Q; whereas the other data structures along with local time are managed locally at each of the LP. The main objective of the PDES is to utilize the parallel computing power to execute the LPs. In the PDES, the most challenging task is to produce the same results as produced by a sequential execution of the model. Therefore to achieve accurate results, higher synchronization is required among the LPs. In parallel simulation, synchronization is violated, if an LP received a message with timestamp smaller than its logical time. Therefore, to produce correct results, each of the LP must follow the Local Causality Constraint (LCC), where LCC is defined as processing of events in non-decreasing time stamp order (Wentong and Turner 2003).

Over the years, different techniques have been proposed to meet the LCC requirement. These techniques falls in two major classes: conservative or optimistic. The conservative approach, avoids the processing of the events until it is declared safe; whereas, optimistic simulation allows the LCC violation. However, a rollback mechanism is provided for recovery (Chandy and Misra 1981)(Jefferson 1985).

3 LITERATURE REVIEW

As discussed in section 1, no work exists that address the issues of VM failure while optimistic simulation is being executed over cloud environment. In recent efforts (Mancini, Wainer, Al-Zoubi, and Dalle 2012) (Al-Zoubi and Wainer 2010), cloud is used to host services that are responsible for the distributing tasks among smart phone nodes. The approach is adopted to simulate the master worker paradigm and utilizes the computing power available on the smart phones.
In an other instance Malik et al. noted in (Malik, Park, and Fujimoto 2010) that frequent rollback occurs due to the extra delay caused by the VM layer over the cloud environment. The authors developed a Time Warp Straggler Message Identification Protocol (TW-SMIP) to handle the frequent rollback issues by dynamically defining a barrier synchronization mechanism at each of the LP. In a similar work, Fujimoto et al. (Fujimoto, Malik, and Park 2010) suggested a master worker approach that aggregates the messages destined to same LP to efficiently utilize the available cloud bandwidth.
Aguilar et al. (Aguilar and Hernandez 2000) presented a fault tolerance protocol for parallel task execution by defining a buddy process . Each time a task is assigned to a process, a copy is send to a buddy process to handle the process failure. If the primary process fails, then the copies of the task on the buddy process are utilized. This is designed to handle process failure for parallel execution, but it generate a large amount of messages that are not even processed in the best case scenario. Therefore, such an approach is infeasible for the cloud environment as it requires a buddy process of each of the LP, due to which execution of thousands of LPs requires the same number of buddy processes. Agrawal and Agre (Agrawal and Agre 1992) presented an interesting approach to recover from the process failure by gathering generated messages from all of the other processes. Although the approach presented is quite effective but it is not designed for cloud environment.
Srikanth B et. at. (Srikanth and Kalyan 2015) has presented the challenges of executing PDES codes over cloud envirnoment. The challenges arises due to VM technologies and other scheduling policies that are not designed for PDES applications. So, they proposed a deadlock-free scheduling algorithm, tested on the Amazon EC2.
The area of PDES execution over cloud has relatively been less explored. The next section contains the detailed description of the proposed framework that handles the VM failure and continues the simulation execution without interruption.

4 PROPOSED FAULT RESILIENT FRAMEWORK

In this section we discuss the fault resilient framework that is design to handle VM failure during execution of PDES codes. The proposed framework cater VM failures without interruptions. In our proposed framework, we made the following assumptions:

a. The communication channel is reliable
b. Balanced distribution of the LPs on the VMs and
c. Super Process (S_p) reside on actual host instead of the VM (role of super process is define below)

In the proposed framework, one of the logical process, acts as a super process (S_p) and a pool of VMs are kept separately (as shown in Figure1), that are used to replace the faulty VMs during the execution of the PDES. The pool of the VMs are maintained to handle the multiple failures during the execution. Each VM holds the same number of logical processes. We assume that the communication channel between the nodes are reliable. Therefore, the messages eventually delivered to the destination node. In our proposed framework, each of the LP sends its states to the S_p right after every Global Virtual Time (GVT) calculation phase. The framework locally maintains a snapshot of the I_Q along with the generated event list. However, running an optimistic simulation over a cloud environment demands a new GVT algorithm due to additional delays added by the VMs and other scheduling policies used inside data center (Srikanth and Kalyan 2015),

but in this study, we use the cut based GVT algorithm for fossil collection (Mattern 1993).
Our objective is to handle VM failures in such a way that PDES continues its execution without restarting the entire simulation. In this regard, following notations has been defined that is used in rest of the sections.

$$VM_1, VM_2 ..... VM_n$$
$$LP_1, LP_2 ..... LP_m$$
$$C_r1, C_r2 ..... C_rk$$

where VM represents the Virtual Machines, LP represents the Logical Processes and C_r represents the coordinating process. Each VM holds multiple LPs and one of the LP is designated as coordinating process (C_r). We can define tuple as:

$$< VM_i, (LP_1, LP_2, ...LP_j), C_rk >$$

In proposed framework, six types messages can be generated by an LP. The messages are notated as:

$$M_i =< M_E, M_A, M_G, M_S, M_{HB}, M_{PT} >$$

Where
M_E represents time stamped event message
M_A represents anti-message
M_G represents Global Virtual Time (GVT) message
M_S represents state info message
M_{HB} represents heartbeat message and
M_{PT} represents the messages that contains process table information

In the framework, the VM failure is identified through the missing consecutive heartbeat (M_{HB}) of messages. At each of the VM, a process with the smallest ID is designated as the coordinating process (C_r), responsible for generating the M_{HB} messages after some interval of time that is destined to the S_p. The selection of coordinating process at each VM is done only once at the time of the simulation initialization. Initially all of the LPs form a virtual ring, S_p initiate a token that moves between processes. The purpose of this token is to find the location of process at each of the VM. The structure of this token is given below:

$$Token_{message} = C_{color}, S_{id}, D_{id}, table < uid_{vm}, uid_{lp} >$$

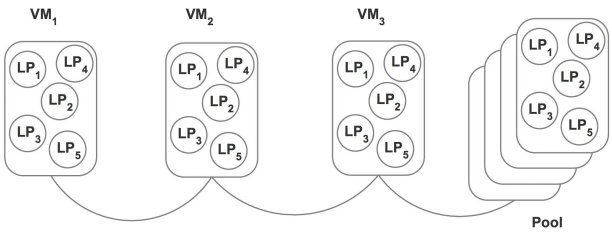

Figure 1: VM Pooling

Where S_id represents the source process identifier, D_id represents the destination process identifier, $table < uid_{vm}, uid_lp >$ is used to find the smallest process identification on each VM. The C_{color} is a single bit field, used to differentiate between the various phases of the process i.e. coordinator selection or coordinator information dissemination. The selection of the coordinating process is a two phase

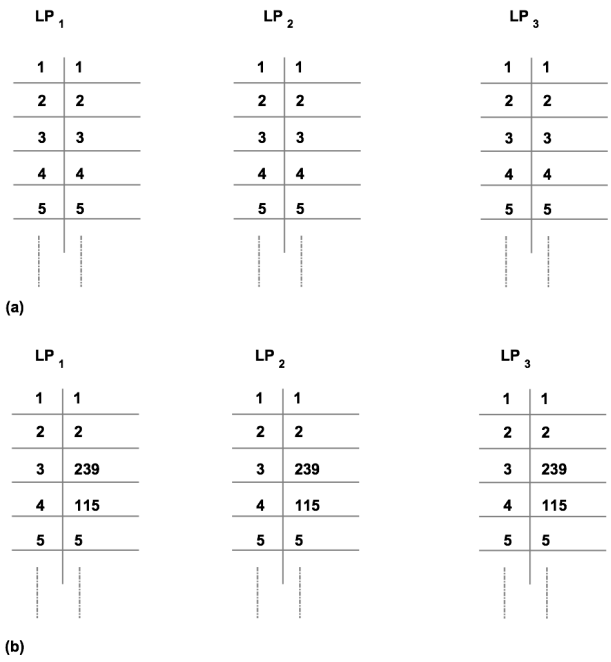

Figure 2: The process table a). With initial values b). After VM placement

process. Initially the color bit is set to zero and the token is moved from one LP to another. The token moves through all the LP hosted on VMs. The framework is implemented using the Message Passing Interface (MPI). Therefore, the next node is determined by adding 1 to its own process identification number.

On receiving a token, each of the LP performs a look up to its VM unique identification and compare to the logical process identification (ID). If the value is greater than its process ID, the LP replaces the value with its own ID in the token ring and forward to next process. The process continue until the token reaches the S_p, where the S_p set the color bit to 1 and the token is again disseminated to inform all LPs about their coordinating process. The coordinating process is responsible for sending M_{HB} messages to the S_p. The selection of the coordinating process is critical as it reduces the bandwidth consumption inside the cloud. Otherwise, all processes have to send the M_{HB} messages, and thus overburdens the communication network and increase the overall cost.
Along with the required data structure mentioned previously, each of the LP maintains a Process Table (P_T) that is used to send the messages to the other processes. The concept of P_T is similar to the routing table that handles the route failure by changing the next hop address. Similarly, in our proposed framework, the S_p notify about the VM failure to other LPs and send the updated P_T entry, as shown in Figure 2. Initially, the P_T is shown in Figure 2a; whereas after the failure, the faulty VM is replaced with another VM available from the pool. After replacing the faulty VM, process ID's are updated in the process table, as shown in Figure 2b. That is to say that LP_3, LP_4 are replaced with LP_{239} and LP_{115} respectively, before sending message to any LP, process performs a lookup operation on its P_T and returned ID is used as the destination node. The complete execution is shown in Figure 3. The failure is identify through multiple lost of M_{HB} messages, the S_P select a VM from available pool of VMs, share the states of failed VM with the new selected VM and instruct all the VMs to restart its execution to previous GVT value and resend the messages whose timestamp is greater than the previously calculated GVT value.

At the time of recovery it is important to cater all the received or transient messages. As depicted in Figure 4 the left portion shows a scenario where message is send before and received after GVT calculation phase. At the time of rollback to previous GVT state to cater faulty VM, such event messages can be missed out.

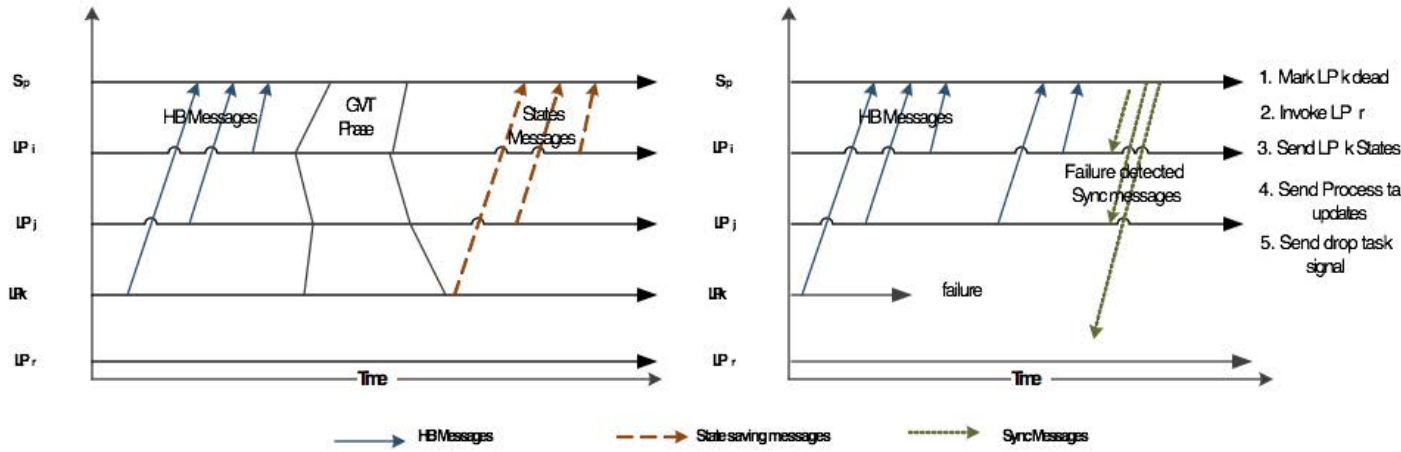

Figure 3: The proposed algorithm on timeline

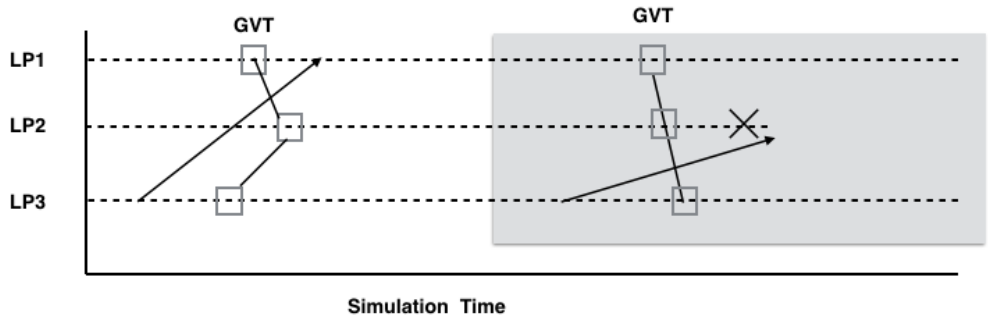

Figure 4: Two critical cases a) Left section: event message send before and received after GVT calculation b). right portion: destination process fail. The proposed framework handles such cases.

Therefore, in our proposed scheme, we incorporated an extra field inside every message that includes the last calculated GVT value. In this way our proposed framework determine the messages generated phase; in case of rollback to previous GVT value, the processes does not discard such events that are send before GVT phase. Moreover, the right portion of Figure 4 shows the a special scenario where destination process fails; such situation is handled through acknowledgement. At the time of rollback to previous GVT value, all events send before GVT must be acknowledged by the destination process, otherwise source process resend before marching forward. Similarly, multiple transmission of same messages are identified through seq. no. and discarded at destination process.

5 PERFORMANCE AND COMPARATIVE ANALYSIS

In this section, we analyzed the proposed framework in terms of the VM failure detection, replacement and the number of M_{HB} generated. We used the benchmark application model described in (Madisetti et al, 1993). The benchmark simulates the characteristics of load sharing in power grids. In the simulation model, the messages generated by each of the source can be categorized into self and propagating messages. The messages generated by source to itself are termed as self-messages and messages sent to other LPs are called propagating messages. Both of these messages are sent with increment in the timestamp ($local_{time} + look - a - head$). The electric power grid exhibits such behavior where load sharing requests are generated and

propagated to other stations. In the simulation, on receiving each LP generates a propagating message with a probability 0.5; otherwise, the LP sends a self-message. In case of the propagating message, neighbor node is randomly selected.

To analyze the proposed framework, an optimistic simulation protocol Time Warp is used to benchmark the proposed framework and series of experiments are performed on 64 VMs, and the results are obtained by varying the number of the VM failures. The proposed framework is based on an optimistic simulation protocol; therefore, we have also measured the total events generated, number of event rollback, and the event rate. Figure 5 shows the proposed framework behaves like traditional TW when no VM failure occurs. We analyzed the framework by varying the number of VM failures. The Figure 6 clearly shows that the number of the total events generated decreases with the number of VM failures. This is because the proposed framework discards the events and restores the states that are at the time of the previous GVT calculation. As the VM failure increases the number of events generated, rollback and event rate decreases. Similarly, the events dropped at other LPs due to VM recovery phase is increased with VM failures but after couple of reading it has been observed the rate stabilized. As cloud provides computing infrastructure as a utility, so more synchronization messages means more cost that user has to pay; therefore, we measured the number of messages that traversed the network to keep the S_P up-to-date. Figure 7, shows the number of M_{HB} messages generated on different VMs.

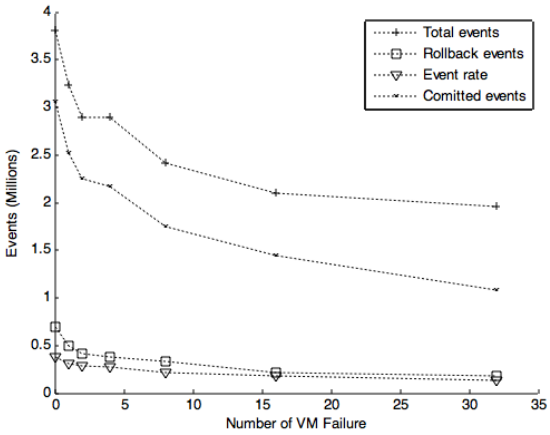

Figure 5: The proposed framework event analysis over VM failure

Figure 7 shows that the more the number of VMs participated in the simulation the more messages are traverse the network to update S_P, whereas, HB message rate plays a significant role in reducing control messages but at the same time, higher HB rate caused delay in VM failure detection. Figure 8 shows the efficiency comparison by varying the number of VM failures. This shows the efficiency gradually decreased as more VMs failure occur. With no VM failure, the efficiency achieved is 80 percent and with 32 VM failures, the efficiency decreased to approximately 30 percent.

6 CONCLUSION

In this paper, we proposed a fault resilient framework that dynamically handles VMs failure inside the cloud environment. The VM failure affects the entire simulation and lead to restarting the entire simulation. That is not acceptable under complex simulation scenarios, especially executing over the cloud environment that is based on pay-per-use model. The restart process simple means more cloud usage and result in over budgeting. Our proposed fault resilient framework is based on state saving at the S_P and snapshots of

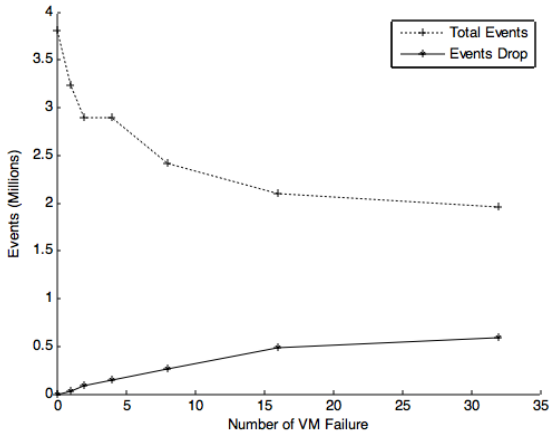

Figure 6: Packet drop due to VM failure.

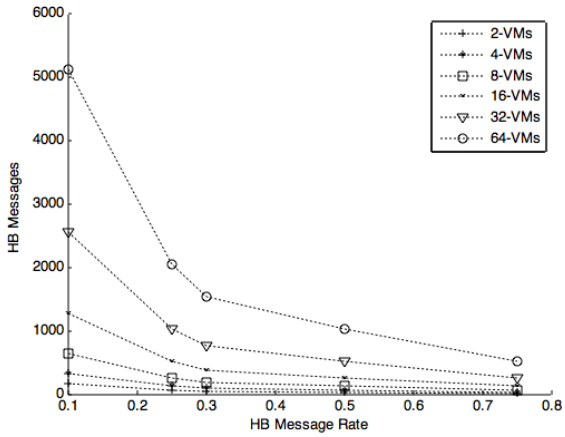

Figure 7: HB generated by varying VMs

processed event list locally, after every GVT calculation. The experimental section showed that the overhead is compared to restarting the entire simulation.

REFERENCES

Agrawal, D., and J. R. Agre. 1992. "Recovering from Multiple Process Failures in the Time Warp Mechanism". *IEEE Transactions on Computers* vol. 41, pp. 1504–1514.

Aguilar, J., and M. Hernandez. 2000. "Fault tolerance protocols for parallel programs based on tasks replication". In *8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 397–404. IEEE.

Al-Zoubi, K., and G. Wainer. 2010. "Rise: Resting heterogeneous simulations interoperability". In *Proceedings of the Winter Simulation Conference (WSC)*, pp. 2968–2980. WSC, ACM/IEEE.

Begum, S., and M. K. Khan. 2011. "Potential of cloud computing architecture". In *International Conference on Information and Communication Technologies*, pp. 1–5. ICICT.

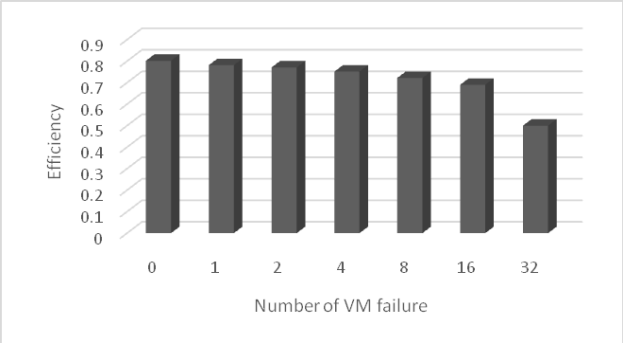

Figure 8: Efficiency comparsion

Chandy, K. M., and J. Misra. 1981. "Asynchronous distributed simulation via a sequence of parallel computations". *ACM Communication* vol. 24, pp. 198–206.

Fujimoto, R. M. 2000. *Parallel and distributed simulation system*. Wiley Interscience Publication New York.

Fujimoto, R. M., A. W. Malik, and A. J. Park. 2010. "Parallel and distributed simulation in the cloud". In *Magzine of the Society for Modeling and Simulation*, pp. 1–10. SCSC.

Jafer, S., Q. Liu, and G. Wainer. 2012. "Synchronization Methods in Paralle and Distributed Discrete-Event Simulation". *Simulation Modeling Practice and Theory* vol. 30, pp. 54–73.

Jefferson, D. R. 1985. "Virtual Time". *ACM Transactions on Programming Languages and systems* vol. 7, pp. 404–425.

Jielong, X., T. Jian, K. Kwait, Z. Weiyi, and X. Guoliang. 2012. "Survivable Virtual Infrastruture Mapping in Virtualized Data Centers". In *5th IEEE Conference on Cloud Computing*, edited by IEEE, pp. 196–203. IEEE.

Linquan, Z., W. Chuan, L. Zongpeng, G. Chuanxiong, M. C, and L. F. C. M. 2013. "Moving big data to the cloud: An online cost minimizing approach". *IEEE Journal of Selected Areas in Communications* vol. 32, pp. 2710–2721.

Malik, A. W., A. J. Park, and R. M. Fujimoto. 2010. "An Optimistic Parallel Simulation Protcol for Cloud Computing Envirnoments". In *Magzine of the Society for Modeling and Simulation International*, edited by SCSC, pp. 1–9. SCS.

Mancini, E., G. Wainer, Al-Zoubi, and O. Dalle. 2012. "Simulation in the Cloud Using Handheld devices". In *12 International Symposium on Cluster, Cloud and Grid Computing*, pp. 867–872. IEEE/ACM.

Mattern, F. 1993. "Efficient Algorithms for Distributed Snapshots and Global Virtual Time Approximation". *Journal of Parallel Distributed Computing* vol. 18, pp. 423–434.

Pandey, S., W. Linlin, S. M. Guru, and R. Buyya. 2010. "A Particle Swarm Optimization-based Heuristic for scheduling workflow applications in Cloud computing envirnoments". In *24th IEEE Conference on Advanced Information Networking and Applicaitons*, edited by IEEE, pp. 400–407. IEEE.

Park, A. J., and R. M. Fujimoto. 2006. "An Approach to High Throughput Parallel Simulation". In *Proceedings of the 20th workshop on principles of advanced and distributed simulation*, pp. 3–10. IEEE Computer Society, IEEE Computer Society.

Shuai, Z., Z. Shufen, C. Xuebin, and H. Xiuzhen. 2010. "Cloud Computing Research and Development Trend". In *2nd International Conference on Future Networks*, edited by ICFN, pp. 93–97. ICFN.

Srikanth, B. Y., and S. P. Kalyan. 2015. "Efficient Parallel Discrete Event Simulation on Cloud/Virtual Machine Platforms". *ACM Trans. Model. Comput. Simul* vol. 26 (1), pp. 1–26.

Wentong, Y. Z., and S. J. Turner. 2003. "Causal order based time warp: a tradeoff of optimism". In *Proceedings of the Winter Simulation Conference*, Volume 1, pp. 855–863. WSC.

AUTHOR BIOGRAPHIES

ASAD WAQAR MALIK is an Assistant Professor at NUST School of Electrical Engineering and Computer Science, Pakistan. He did his PhD from College of Electrical and Mechanical Engineering, NUST. He worked as a visiting scholar at Georgia Institute of Technology, and North Dakota State University, USA. He is the co-director of Center for Research in Modeling and Simulation (CRIMSON). His research interests include parallel and distributed simulation, cloud computing, Internet of Things (IoT) and large-scale networks. His email address is asad.malik@seecs.edu.pk.

IMRAN MAHMOOD is currently working as an Assistant Professor at Department of Computing, School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Pakistan. He earned his Master's degree in Software Engineering in 2007 and PhD in Computer Systems in 2013 from KTH-Royal Institute of Technology, Sweden. He has worked in University of Engineering and Technology, Lahore before moving to NUST. His scholarly interests are in modeling and simulation of complex systems and he leads the Center for Research in Modeling and Simulation (CRIMSON) as director in NUST. He has supervised research projects as lead researcher at the Center for Simulation & Visual Analytics Research and worked in collaboration with Swedish Defense research agency (FOI) during the Masters and Doctoral research. He has published quality research papers and delivered different workshops, lectures and invited talks. He has also served in IT industry earlier at various leading roles, at different private and governmental organizations.