Efficient Logical Process Mapping on Distributed Systems inside Cloud Data Center

Author1:AB
Author1:BC
Author1:CD

ABSTRACT

Cloud provides a shared computing space on a pay-as-you-go model. Due to this sharing, it is difficult to execute the task efficiently in terms of time. Several factors play its parts such as process scheduling and computing requirement of other processes sharing the system. Moreover, network usage is highly depended on processes, typically processes are categorized computing and communication intensive. Such sharing of platform often degrade the performance of parallel and distributed simulations (PADS). Execution of optimistic simulation can lead to a large number of rollbacks due to the imbalanced situation of physical systems. In this paper, we have analyzed the execution of Time Warp (TW) inside cloud environment and proposed a scheme to migrate the frequent communication processes. Thus, the objective is to reduce the overhead which is one of the sources of rollback especially due to unknown network conditions inside the cloud. The results section demonstrate the effectiveness of our proposed scheme.

KEYWORDS

Parallel and Discrete Event Simulation (PDES), Distributed Simulations, Time Warp, PADS, Cloud Computing, Load Balancing;

ACM Reference Format:
Author1:AB, Author1:BC, and Author1:CD. 1997. Efficient Logical Process Mapping on Distributed Systems inside Cloud Data Center. In *Proceedings of ACM conference (PADS' 18).* ACM, New York, NY, USA, Article 4, 7 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

The domain of Parallel and Discrete Event Simulations (PDES) has evolved over the years. Its purpose is to simulate the behavior of executing processes on Parallel or Distributed processors. Communication is established between these processes via either message passing or shared memory. Improving the overall performance of the simulation on heterogeneous platforms is one of the major research concern in this field.

Cloud computing provides its users with hardware and software resources and these resources can be remotely utilized by the users. Goal of cloud service providers is to improve the performance by enhancing the application portability and offer more diverse resource deployment options. Cloud computing data centers require a comprehensive resource allocation system to manage both computational and network resources. There are many optimal and suboptimal resource allocation techniques for cloud computing data centers [22]. With such appealing processing capabilities of cloud computing, many scientific research applications and paradigms use cloud computing architecture. Similarly, cloud computing presents a fascinating means to provide simulation applications to its users, specially the parallel and distributed simulations. As much as appealing this concept is, there are some challenges in execution of PDES on clouds[6].

- Performance degradation can be faced if PDES is executed on clouds, on the account of its multi-tenant environment.
- Compute or Data intensive application can effect the capability of other processes launched on the same physical node as the system is shared among different applications.
- Processes are placed at different locations on the clouds which is prone to congestion as more traffic will be generated.
- Cloud resources are shared by different applications that can arise inconsistency issues.
- A simulation that involves a lot of communication is more challenging to be performed on cloud platform.

In traditional distributed simulation, logical processes (LPs) are mapped on different systems/cores. These LPs communicate with each other by sending and receiving time-stamped messages. However, process mapping on different Processing Entities (PEs) in a cloud can greatly affect the performance of entire simulation system. Distributed simulation requires a synchronization algorithm to ensure that the distributed execution yields the same results as a sequential execution. Synchronization among processes is of significant importance. Moreover, some algorithms need to store information of processed events to undo out of order execution using rollback mechanism. S. Jafer et al[8] has presented a detailed discussion about some of the widely used synchronization techniques categorized as; Conservative Synchronization and Optimistic Synchronization.

In conservative approach causality error is strictly avoided by applying strategies to determine when it is safe to process an event. Chandy et al[3] proposed a conservative synchronization mechanism that uses Null messages. In this mechanism, every time an LP sends an event message to a neighboring LP, it also sends a Null-message to all its other neighboring LPs. This Null message informs the neighboring LPs of a lower bound on the timestamp (i.e. its Simulation time plus its lookahead value) of any event message LP might send in the future. Neighboring LPs can thus determine a lower bound on the timestamp (LBTS) of all messages it might receive in the future (from all its neighbors). Optimistic Synchronization on the other hand defies the notion of Local Causality

PADS' 18, 2018, Italy

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

Constraint. It just proceeds with the execution of events and uses rollbacks to recover from events that are executed out of order. Jefferson presented a famous optimistic algorithm known as Time Warp [11]. In this synchronization technique for Distributed Simulations there is a local clock associated with each process. The time of that local clock is called Local Virtual Time (LVT). Each process compares timestamps of events with it own Local Virtual Time due to which it makes sure that no out of order message arrives[10]. If by any chance an out of order message arrives, necessary rollbacks are done and then simulation proceeds[21]. To be able to rollback changes, Time Warp mechanism keeps processed events in a different list causing memory overhead. A Global Virtual Time is computed that is the lowest timestamp of all unprocessed events, processed events and anti messages. Global Virtual Time (GVT) assures that processes do not go further away from it[2] and memory associated with all the events with timestamp lower than GVT can be regained [15].

Performance of Parallel and Distributed Simulations (PADS) degrades on account of the fact that logical processes exchange large number of *event messages* that are sent across the networks. A lot of network traffic is generated because of the recurring communication of processes. The goal of this work is to minimize this network traffic by an effective mapping of processes in cloud data center. A technique to overcome this situation is to perform other operations during communication, is known as Latency Hiding. We have suggested a migration technique that will not only reduce the amount of rollbacks but also the overall communication expense of the network of the cloud. The remaining paper is categorized as follows: Section II contains details about some of related works. Section III explains problem formulation. Our proposed methodology is explained in Section IV. Evaluation is done in Section V and conclusion is discussed in Section VI.

2 LITERATURE REVIEW

Some of the related work in this field is discussed here. Malik et al [16] presented a methodology to deal with recurrent rollbacks caused by unbalanced workload and crowded network. Fujimoto et al[7] highlighted key challenges in performing Parallel and Discrete Event Simulations over Cloud environment. Performance of a distributed simulation can degrade because of processes that frequently communicate with each other by message passing. Randomly placing processes over cloud can cause unbalanced workload, that is why some processes will compute execution immediately and some will last longer.

Perumalla et al[20] proposed a mechanism for processing optimistic parallel application using time warp. The mechanism supports a variety of synchronization techniques. It is designed in such a way that processes can dynamically choose between synchronization techniques. Alfred et al[19] proposed a technique for executing distributed simulations of large scale using a *master-worker* methodology while the simulation is performed on resources connected through network. Authors in [9] minimized synchronization delays by incorporating a model that uses multi-threading technology. A mechanism discussed in [1] is proposed to improve the efficiency of simulation applications by reducing the number of rollbacks.

Another attempt of the same nature is made by Wentrong et al[13] for augmenting large scale simulations performance, so that maximal simulation execution speed can be achieved. Idea is to make the system fully balanced by providing virtual machines with resources, dynamically. It also ensures that different nodes proceed with similar speed. Xiao et al[23] designed a procedure that increases the efficiency of High Level Architecture (HLA) systems by presenting load balancing under both communication and computation expenses.

In [24] Jingjing Wang et al. approaches to PDES by proposing a multi-threaded simulator. The objective of the author's work is to increase the performance of simulations by using a threaded model. Doing so will increase the performance of communicating processes compared to using a non-threaded model. Munck et al[17] discussed the problems with conservative synchronization's null message protocol justifying that it sends enormous amount of null messages causing overhead and then proposes a technique that integrates the fruitfulness of existing techniques.

In [5], writers Gabriele D'Angelo et al. discussed the challenges faced in executing parallel and distributed simulations in clouds as well as multi-core systems. Under the constraints of performance and utility, authors evaluated the existing PADS techniques for deficiencies and then proposed an adaptive way to overcome those deficiencies. The proposed system lessens the cost of communication and improves load balancing.

Similarly, Weiwei Chen et al[4] proceeded with another method to improve parallel and discrete event simulations. They improved the performance of the overall simulation using multiple threads running simultaneously. This approach has the ability to decide at runtime whether to initiate a group of threads in parallel or not. In [14], existing methods are analyzed and compared by the author to his own method of optimizing task allocation and then he justifies how his technique puts him one up on other existing algorithms.

Vy Thuy Nguyen et al[18] proposed a criteria for the simulation of network containing large number of nodes. The criteria produces efficiency in the networks by giving high level parallelism. The gist of this work is to replace *node model* by *link model*, that is an LP portrays a link to the network and not a node and it will perform better.

We have proposed an approach that will migrate processes that most frequently communicate with each other, making Time Warp to execute efficiently over the clouds. The forthcoming section describes the problem formulation.

3 PROBLEM FORMULATION

In this section, we have devised and explained the problem that will help in understanding how we can minimize the delay experienced by the network. This is done by performing essential migration of those processes that communicate most frequently resulting in improved performance of the overall system. A simulation involves the participation of a number of Logical Processes (LP's) i.e.

$$LP_1, LP_2, \ldots\ldots LP_m \in LP$$

These logical processes are scheduled on a number of Compute Codes (CN's). A data center is a huge station containing a lot of

physical cores also called as Processing Elements (PE's). Each Processing Element is capable of scheduling multiple Virtual Machines. Logical processes are scheduled on these virtual machines. Compute Nodes are same as virtual machines. It is to be noted that these nodes may or may not belong to the same rack in a data center.

$$CN_1, CN_2, ...CN_m \in CN$$

Some of the assumptions we made are as under: We have assumed LP's to be representing an undirected graph that is:

$$G = (V, E)$$

Where G is an amalgamation of Edges E and Vertices V. In our assumption vertices represent LP's and edges represent paths through which those LP's will exchange messages.

Furthermore to layout the network we have supposed a mesh topology. Doing so will enable an LP to be able to communicate with every other LP on the network. There is a cost associated with every path through which a processes will communicate given by:

$$cost_{x,y} | \{weight(x, y) = weight(y, x)\}$$

Considering x to be source and y to be destination, weight from source to destination and destination to source should be the same.

Additionally, LP's can either be scheduled on the same compute node or on different compute nodes. The cost of communication between such LP's is represented by $Cost_{localNode}$. LP's could also be scheduled on different compute nodes but within one rack. In that case, cost of communication is maintained by $Cost_{localRack}$. In another case LP's could be scheduled on different nodes across different racks and then the cost for this situation is maintained by $Cost_{remoteRacks}$.

The cost of communication between LP's on the same compute node must be the lowest whereas the cost of communication between LP's on different racks must be the highest i.e.

$$Cost_{localNode} < Cost_{localRack} < Cost_{remoteRacks}$$

As cloud supplies its users with resources like processing power, memory and storage on pay-as-you-go basis. Users are charged on account of memory usage, amount of computation, amount of communication and usage of storage. The goal here is to lessen the cost of communication by migrating processes near to its communicating counterpart.

Assuming, if we have six compute nodes on a data center i.e. $CN_1, CN_2, CN_3 ... CN_6$ and an equivalent amount of LP's are launched on those compute nodes i.e.

$$\begin{aligned} \{lp_{1.w}, lp_{1.w+1}, ..., lp_{1.x}\} &\rightarrow cn_1 \\ \{lp_{2.x}, lp_{2.x+1}, ..., lp_{2.y}\} &\rightarrow cn_2 \\ &\dots \\ \{lp_{6.y}, lp_{6.y+1}, ..., lp_{6.z}\} &\rightarrow cn_6 \\ R_1 + R_2 + ... + R_k &\rightarrow DataCenter \end{aligned}$$

R stands for Racks that belong to a data center.
Additionally nodes pairs are placed on racks.

$$\begin{aligned} cn_1, cn_2 &\rightarrow R_1 \\ cn_3, cn_4 &\rightarrow R_2 \\ &and \\ cn_5, cn_6 &\rightarrow R_3 \end{aligned}$$

Assuming that communication occurred between process located on different racks is represented by m, communication occurred between processes on different nodes but same rack is represented by r and communication occurred between processes on the same node is represented by q. Assuming $m > r > q$

$$\begin{aligned} (cn_1, cn_3) &\Rightarrow (p_{1.r+r}, p_{3.j}) \rightarrow m \\ (cn_5, cn_6) &\Rightarrow (p_{5.l}, p_{6.m}) \rightarrow r \\ (cn_4, cn_4) &\Rightarrow (p_{4.x}, p_{4.x+1}) \rightarrow q \end{aligned}$$

Equations above represent how communication occurred between processes on different racks, on different nodes but same rack and on the same node.

The expense of the total communication is devised as:

$$f(x) = \{\sum_{i=1}^{m} Cost_{remoteRacks} + \sum_{i=1}^{r} Cost_{sameRack} + \sum_{i=1}^{q} Cost_{localNode}\}$$

Considering a three-tier architecture, communication happened between processes on different racks will require at least 2 hops as the message traverser through aggregate and access switches. Similarly, on same rack, the communication will require just one hop and on the same node, there are no hops as the communication is done locally.

$$min\{\sum_{i=1}^{m} Cost_{remoteRacks} + \sum_{i=1}^{r} Cost_{localRack} + \sum_{i=1}^{q} Cost_{localNode}\}$$

There is a way [12] to lessen the traffic on the network by introducing locality between communicating processes. More number of hops means more delay incurred. So:

$$delay_{localNode} < delay_{localRack} < delay_{remoteRacks}$$

Bringing processes who communicate more on the same node can lessen the traffic on the network, that is another goal of our work. The idea is to lessen delay incurred in communication between processes that transmit message most frequently by introducing locality between such processes. In traditional methodology, rollbacks can affect the performance of the simulation by congesting the network which will resultantly make the system costly. So the goal is:

$$min\{\sum_{i=1}^{m} del_{remRacks} + \sum_{i=1}^{r} del_{locRack} + \sum_{i=1}^{q} del_{locNode}\}$$

Reducing the amount of rollbacks means reducing the amount of anti-messages sent across the network, that in result will reduce the network usage and thus enhance the performance.

4 PROPOSED METHODOLOGY

As discussed earlier, Parallel and Discrete Event Simulation involves a number of LP's that communicate with each other by sending messages that has a time stamp associated with it. Problems arise while synchronizing the Logical Processes. To overcome this problem some technique were introduced like Time Warp. Time Warp overcomes the problem of synchronization by using rollbacks. But large number of rollbacks become a major problem when Time Warp is used on cloud environment. For synchronization purpose, it will generate a huge amount of rollbacks because of the network involvment [16]. Besides this, network performance is also degraded due to the placement of virtual machines in a dynamic manner as happens in the case of cloud environment.

Our procedure helps in overcoming delays experienced in communication, decreases the network usage and also the amount of rollbacks in case of Time Warp execution on clouds platform. Separate queues are associated with each LP for keeping a record of processed and unprocessed events. Events are initially executed in increasing order of their timestamps. Messages are also kept track of such that, if a message is received that has a timestamp lower than the local time of the Logical Process (that received the message) then the changes are rolled back. These rolled backs events are later considered for execution again. Rollbacks cause the system to sent anti messages which in turn causes other processes to do rollbacks. Following this strategy, there occur a lot of rollbacks, hence, the performance is degraded.

In our approach, one process acts as the master process (global coordinator) and all other processes acts as workers. The number of messages sent and received by a process are kept track of. The number of messages sent or received increases as more events are being executed. After a user defined unit of time each process creates a table that is in descending order of the number of sent and received messages. *"'N"* entries from the top of the table are shared with the master process. We calculated *"'N"* from dividing total number of processes by 2. Our methodology mainly follows the criteria of Alfred et al[19] and Fujimoto et al[7].

After when the worker processes are done with sending their top *"'N"* entries, the master process on the basis of those entries draws a graph. Number of messages sent and received are added and given as weights to edges. An edge between two processes is added if their is any communication between them. If a processes has no communication with other processes, it is added to graph as an unconnected node. If there is a cycle in the graph it is eliminated by removing the lowest weighted edge.

After the graph is completed built, the master puts edges with the highest weight in a separate list one after one until all connected nodes are added to the list. Nodes that are still remaining i.e. those that were not connected to the graph are added to another list. Let's suppose processes x and z are currently located at node A and process y is located on another node B. Further, assume that processes x and y communicated with each another the most. Master process in such case will migrate process y to the same node as process x i.e. node A. Process z which in this case is not communicating with either x or y is migrated to node B. The master will always, in such case, swap two processes just like it did y and z. Processes that are added to the second list are considered expendable, meaning when a process needs to be migrated to its most communicating peer, master will pick any expendable process to pair up for swapping. The complete flow of procedure is show in figure 3.

Additionally key-value pairs are added to a routing table. A routing table is associated with each process that contains default values see Figure 2(a). Use of routing tables will help us in achieving transparency. Routing tables makes life easy by providing processes with the current id's of process in the destination. In the start, each routing table has default values but as soon as migrations are performed the routing table is updated with new id's of the processes, see Figure 2(b). This is vital because these tables are used to get the id's of destination processes.

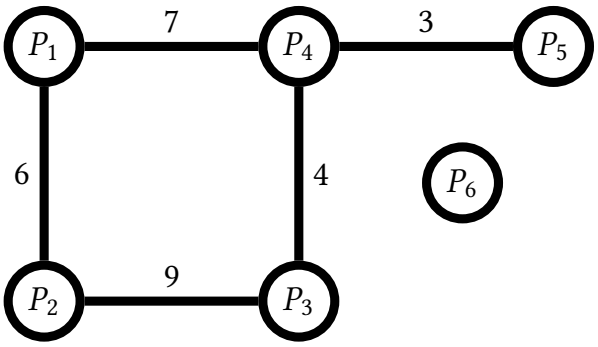
Process_Id	Updated Process_Id
325	325
456	456
986	986
391	391
235	235

(a) Routing table before migration

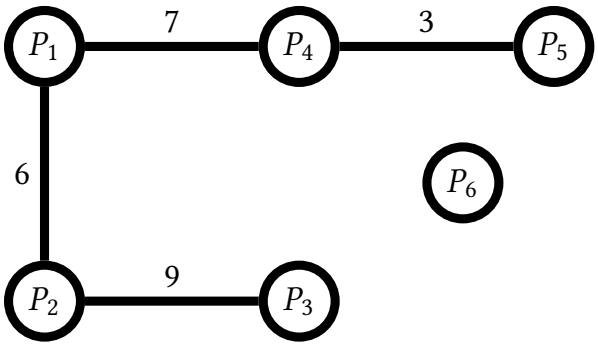
Process_Id	Updated Process_Id
325	235
456	986
986	456
391	391
235	325

(b) Routing table after migration

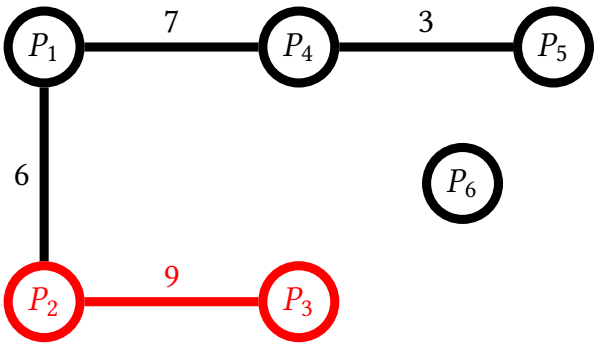
Figure 1: Routing table maintain



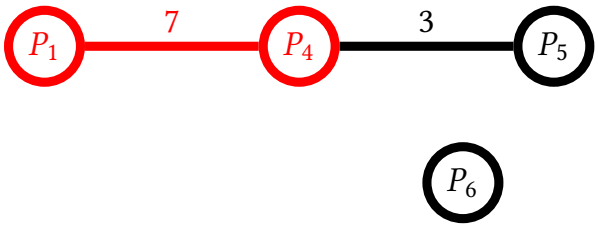
(a) Undirected graph is created based on communications



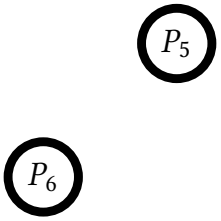
(b) Get rid of cycles



(c) Put the pair of nodes that communicate the most in a list



(d) Put the next most communicating pair in the list



(e) Processes that remain disconnected from the graph are to be added to the list of Expendables

Figure 2: Working of the system

5 PERFORMANCE EVALUATION

We have used PHOLD benchmark model to study the performance of the system. We compared the results of our systems with the standard optimistic Time Warp protocol on a cloud computing platform. Moreover, we used Amazon EC2 with about 20 instances. According to PHOLD model, a single instance is supposed to schedule many processes that should communicate with each other. Experiment was performed several times to obtain normalized values, then we took the average of the all results. Figure 3 depicts events committed using both scenarios, figure 4 describes the network usage, figure 5 shows the rollbacks happened and shown in figure 6 represents the efficiency of both techniques. The network usage is the amount of messages passed in between different racks. Our technique performs better as it outnumbers the traditional method by having more committed events. Furthermore, our technique not only has way less rollbacks but it also tones down the usage of network and has a better efficiency. Efficiency is the ratio of committed events to the total events. Therefore, results depicts that our algorithm augments the performance of Time Warp if used in cloud computing platform.

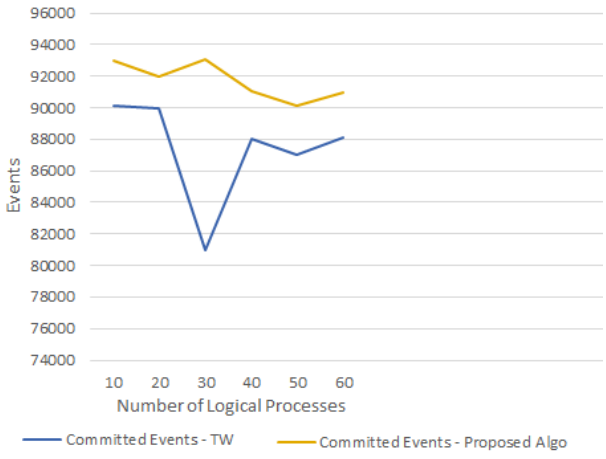


Figure 3: Events Committed

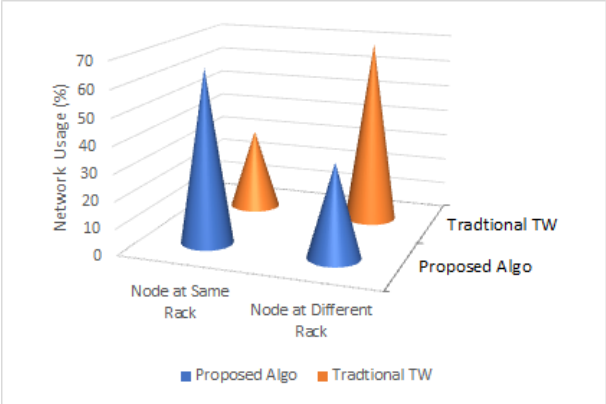


Figure 4: Network Usage

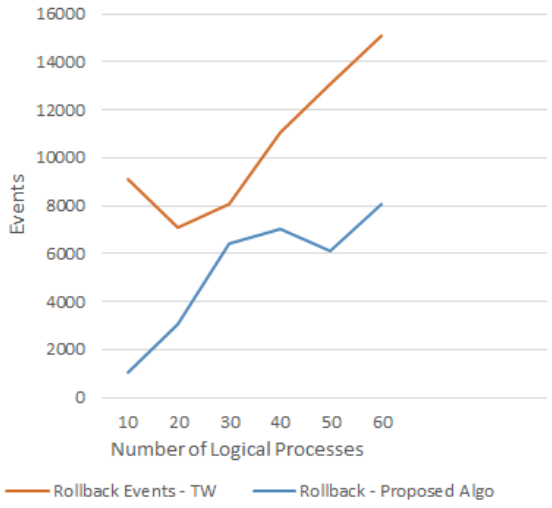


Figure 5: Rolled-back Events in both cases

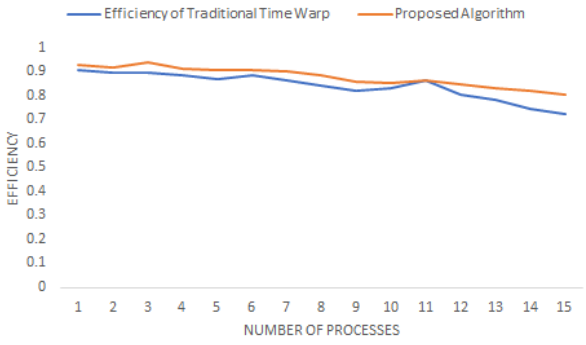


Figure 6: Efficiency Comparison

6 CONCLUSION

In cloud computing platform a single resource is shared by multiple remote users. Different users may have different workloads. Executing Parallel and Discrete Event Simulation on such platforms may cause performance issues because of unbalanced distribution of workload. Some criteria should be adopted that can handle changes in dynamic manner so that the overall system remains balanced and well-performing. In PDES, *event messages* are used as a source of communication between Logical Processes. These *event messages* can affect the performance negatively as it can clog the network.

Our approach, as compared to the existing standard Time Warp (that does not work better on cloud platform due to the single-resource-multiple-users trait) works better on the cloud environment. We have devised a methodology in a way that it decreases the communication across the network thus, improving the performance of the simulation. In future, we are looking forward to include mechanisms that will support fault-tolerant and autonomous migration of processes.

REFERENCES

[1] P. Bauer, J. Lindén, S. Engblom, and B. Jonsson. 2015. Efficient inter-process synchronization for parallel discrete event simulation on multicores. *Proceedings of the 3rd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, ACM* (2015), 183–194.

[2] C. D. Carothers and R. M. Fujimoto. 2000. Efficient execution of time warp programs on heterogeneous, NOW platforms. *IEEE Transactions on Parallel and Distributed Systems* 11, 3 (2000), 299–317.

[3] K.M. Chandy and Misra J. 1979. Distributed simulation: A case study in design and verification of distributed programs. *IEEE Transactions on software engineering* (1979), 440–452.

[4] Weiwei Chen, Xu Han, Che-Wei Chang, Guantao Liu, and Rainer Dömer. 2014. Out-of-Order Parallel Discrete Event Simulation for Transaction Level Models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33, 12 (2014).

[5] Gabriele D'Angelo and Moreno Marzolla. 2014. New trends in parallel and distributed simulation: From many-cores to Cloud Computing. *Simulation Modelling Practice and Theory* 49 (2014), 320–335.

[6] Richard M Fujimoto. 2016. Research Challenges in Parallel and Distributed Simulation. *ACM Trans* 26, 4 (2016), 1–29.

[7] R. M. Fujimoto, A. W. Malik, and A. Park. 2010. A. Parallel and distributed simulation in the cloud. *SCS M&S Magazine* (2010), 1–10.

[8] S Jafer, Q Liu, and Wainer. 2013. Synchronization methods in parallel and distributed discrete-event simulation. *Simulation Modelling Practice and Theory* (2013), 54–73.

[9] D. Jagtap, N. Abu-Ghazaleh, and D. Ponomarev. 2012. Optimization of parallel discrete event simulator for multi-core systems. *In Parallel & Distributed Processing Symposium (IPDPS), IEEE* (2012), 520–531.

[10] D. R. Jefferson. 1985. Virtual time. *ACM Transactions on Programming Languages and Systems (TOPLAS)* (1985), 404–425.

[11] D. R. Jefferson and H. A. Sowizral. 1985. Fast concurrent simulation using the Time Warp mechanism. *Proceedings of the SCS Distributed Simulation Conference* (1985).

[12] W. Jian, W. KwameLante, G. Kartik, and XenLoop. 2008. A Transparent High Performance Inter-vm Network Loopback. *Proceedings of the 17th International Symposium on High Performance Distributed Computing* (2008), 109–118.

[13] ZengXiang Li, Xiaorong Li, Long Wang, and Wentong Cai. 2014. Hierarchical Resource Management for Enhancing Performance of Large-scale Simulations on Data Centers. *Proceedings of the 2nd ACM SIGSIM Conference on Priciples of Advanced Discrete Simulation* (2014), 187–196.

[14] Weiwei Lin, Chen Liang, James Z. Wang, and Rajkumar Buyya. 2014. Bandwidth-Aware Divisible Task Scheduling for Cloud Computing. *Journal of: Software: Practice and Experience* (2014).

[15] Y. B. Lin and E. D. Lazowska. 1990. Reducing the state saving overhead for Time Warp parallel simulation. *University of Washington, Department of Computer Science* (1990).

[16] A. Malik, A. Park, and R. Fujimoto. 2009. Optimistic synchronization of parallel simulations in cloud computing environments. *IEEE International Conference on Cloud Computing* (2009), 49–56.

[17] S. De Munck, K. Vanmechelen, and J. Broeckhove. 2014. Revisiting conservative time synchronization protocols in parallel and distributed simulation. *Concurrency and Computation: Practice and Experience* 26, 2 (2014), 468–490.

[18] Vy Thuy Nguyen and Richard Fujimoto. 2016. Link Partitioning in Parallel Simulation of Scale-Free Networks. *IEEE/ACM 20th International Symposium on Distributed Simulation and Real Time Applications* (2016).

[19] Alfred J. Park and Richard M. Fujimoto. 2012. Efficient Master/Worker Parallel Discrete Event Simulation on Metacomputing Systems. *IEEE Transactions on Parallel and Distributed Systems* 23, 5 (2012), 873–880.

[20] K. S. Perumalla. 2007. Scaling time warp-based discrete event execution to 104 processors on a Blue Gene supercomputer. *Proceedings of the 4th international conference on Computing frontiers, ACM* (2007), 69–76.

[21] R Rönngren, M Liljenstam, R Ayani, and J Montagnat. 1996. Transparent incremental state saving in Time Warp parallel discrete event simulation. *ACM SIGSIM Simulation Digest* 26, 1 (1996), 70–77.

[22] Mohamed Abu Sharkh, Abdallah Shami, and Abdelkader Ouda. 2017. Optimal and suboptimal resource allocation techniques in cloud computing data centers. *Journal of Cloud Computing* 6, 1 (2017), 6.

[23] Xiao Song, Yaofei Ma, and Da Teng. 2015. A Load Balancing Scheme Using Federate Migration Based on Virtual Machines for Cloud Simulations. *Mathematical Problems in Engineering* 2015, 506432 (2015).

[24] Jingjing Wang, Deepak Jagtap, Nael Abu-Ghazaleh, and Dmitry Ponomarev. 2014. Parallel Discrete Event Simulation for Multi-Core Systems: Analysis and Optimization. *IEEE Transactions on Parallel and Distributed Systems* 25, 6 (2014).