An Optimistic PDES Protocol Over Dynamic Cloud

By

Muhammad Umar Khan

NUST201260774MSEECS60012F

Supervisor

Dr. Asad Waqar Malik

Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Information Technology (MS IT)

In

School of Electrical Engineering and Computer Science,

National University of Sciences and Technology (NUST),

Islamabad, Pakistan.

(March 2016)

Approval

It is certified that the contents and form of the thesis entitled "**An Optimistic PDES Protocol Over Dynamic Cloud**" submitted by **Muhammad Umar Khan** have been found satisfactory for the requirement of the degree.

Advisor **Dr. Asad Waqar Malik**

Signature ___________________

Committee member 1 **: Dr Anis Ur Rehman**

Signature ___________________

Date ________________________

Committee member 2 : **Dr Muneeb Ullah**

Signature ___________________

Date ________________________

Committee member 3 : **Dr Omar Arif**

Signature ___________________

Date ________________________

Dedication

With affection and gratitude, I would like to dedicate this thesis to my mother and teachers who have been remain a continuous source of inspiration and motivation for me and support me all the way.

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name **Muhammad Umar Khan**

Signature ____________________

Acknowledgment

In the name of Allah the beneficent and merciful, on whom we all are dependent for eventual support and guidance. Foremost, I am highly grateful to ALLAH for His blessing that continue to flow into my life, and because of You, I made this through against all odds.

I would like to express my immense gratitude to my supervisor Dr. Asad Waqar Malik, who has guided and supported me throughout my thesis work and allow me to work in my own way and polish my skills. I highly appreciate his help in technical writing. His mentorship was predominant in sustaining versatile experience in my long term career goals.

I would also like to thank my committee members, Dr. Anees Ur Rehman, Dr. Muneeb Ullah and Dr. Omar Arif who continuously guide me in my thesis, and provide their valuable suggestions and encouragement.

To all my friends at GSM lab, Thank you for your understanding and encouragement in my many, many moments of crisis. Your friendship makes this degree a wonderful experience.

Table of Contents

List of figures

Abstract

Parallel discrete event simulation (PDES) simply known as parallel or distributed simulation is the execution of single discrete event simulation program on a parallel computers. PDES is largely accepted to promote interoperability and reusability of simulation components and to speed up the large scale simulations. In PDES, simulation is divided into small tasks which are placed on different processor for execution. Hence, the simulation system can be viewed as the combination of concurrent processes, each of them executing in a sequential manner on the separate processor and modeling some part of the physical system. All these processes communicate with each other by sending time stamped event messages. An event can be any update in the state of the simulation system at any simulation timestamp.

The events are scheduled in timestamp order. Events are causal related to each other and processed in timestamp manner. The other adopted techniques are conservative and optimistic. In conservative approach, the causality errors are not allowed and each event is processed after all the events which could affect it. In optimistic approach allow causality errors and use some detection and recovery mechanisms for the efficient processing of the events. Timewrap is the most commonly known protocol. In timewrap, whenever a causality error occurred a rollback mechanism initiated to recover the simulation to a state of correct computation. In order to roll back other processes, system sends the anti messages to other processes whenever a causality error occurs so that all the events that are processed prematurely can be undone. Thus, Roll back mechanism costs a lot of time in sending anti messages and state saving.

This thesis presents an efficient optimistic protocol to minimize the rollbacks in the simulation system. We optimize the timewrap protocol in a way that the processes which are communicating and sending messages more rapidly bring closer to each other to improve the efficiency and to minimize the number of roll backs. To analyze the performance, the PHOLD application is used which is the standard PDES benchmark. The PHOLD provides the easily understandable controls to analyze the performance and test the functionality simultaneously.

Chapter 1

1. Introduction

This chapter gives the basic idea of the concepts involved in this research. It also presents the background and motivation for this study. Moreover, it provides an idea of expected results, and methodology to get and evaluate the results. Finally, it presents the structure of this thesis document.

1.1 Introduction of Domain

Parallel discrete event simulation (PDES) can be expressed as the execution of DES project over numerous processors in parallel way [1]. Generally this is done to expand the execution and to scale DES to a bigger setups PDES significance is expanding because of the parallelism in the computational equipment with less concentrate on change of clock speed. The parallelism is used by developers to overcome the performance issues in simulations. However parallelization of simulation program requires skills and expertise for the development and execution of the PDES codes. Cloud computing provide the solution by hiding the details of the simulation from the users.

PDES program contain logical processes (LPs) which exchanges time stamped messages for communication. Each LP process these events in a time stamped manner to portray the changes in the simulation program. Synchronization of the parallel simulation is the major problem in the PDES. Events on each LP are executed in increasing time stamp order to ensure that no future event can affect the events in the past. Time wrap is the well known protocol of optimistic synchronization which uses roll backs to address the causality problem [2]. In timewarp the LPs are permitted to execute all the events it has received regardless of the time stamp. When the event have smaller time stamp than the earlier received events than the computation for those events have to be undone. If messages were sent to other LPs by this LP than those messages must be undone. Anti messages are the negative message used to undone these events.

Cloud computing is a service oriented technology where software is provided I the form of service through virtualized environment. Clients can use those resources from the remote locations to compute the tasks. Cloud computing provides the solution to the long standard problem of having a skill set of executing the complex application codes.

1.2 Problem Statement

In a multi-tenant environment like cloud, PDES systems can have number of rollbacks because computations of other users can slow the progress of some LPs resulting in number of straggler messages and rollbacks. In this thesis we will study the synchronization among different LPs and network performance during the simulation and propose a mechanism to handle the cloud environment effectively.

1.3 Purpose

The purpose of this research is to minimize the number of roll backs in PDES and increase the efficiency of the system.

1.4 Thesis Outline

Rest of the thesis report is structured as follows:

- Chapter 1: In this chapter, the introduction of research domain, problem statement, purpose, scope and objectives of this thesis are explained.

- Chapter 2: In this chapter, the major components of Parallel discrete events simulation and the two synchronization methods are explained.

- Chapter 3: In this chapter, the several related Synchronization algorithms with their pros and cons are reviewed.

- Chapter 4: In this chapter, our proposed scheme and simulation setup for the validity of proposed scheme is explained.

- Chapter 5: In this chapter, results are presented.

- Chapter 6: In this chapter, conclusion of entire work is briefly explained.

Chapter 2

2. Theoretical Background

2.1 Simulation with discrete events

Simulation is the art of mimicking the real world large and complex problems virtually on computers. It is the technique widely used to analyze the performance and behavior of the models. In simulations the computers are used to analyze and evaluate the models. Simulation is the efficient and cost effective way of designing the new systems according to the real word requirements [1]. Large and complex systems can be simulated to evaluate and can the optimized by analyzing their performance from simulations. Simulations provide the effective way of implementing the assumptions on the simulated environment before implanting them directly to the original system which can minimize the cost and risk of failure of the system.

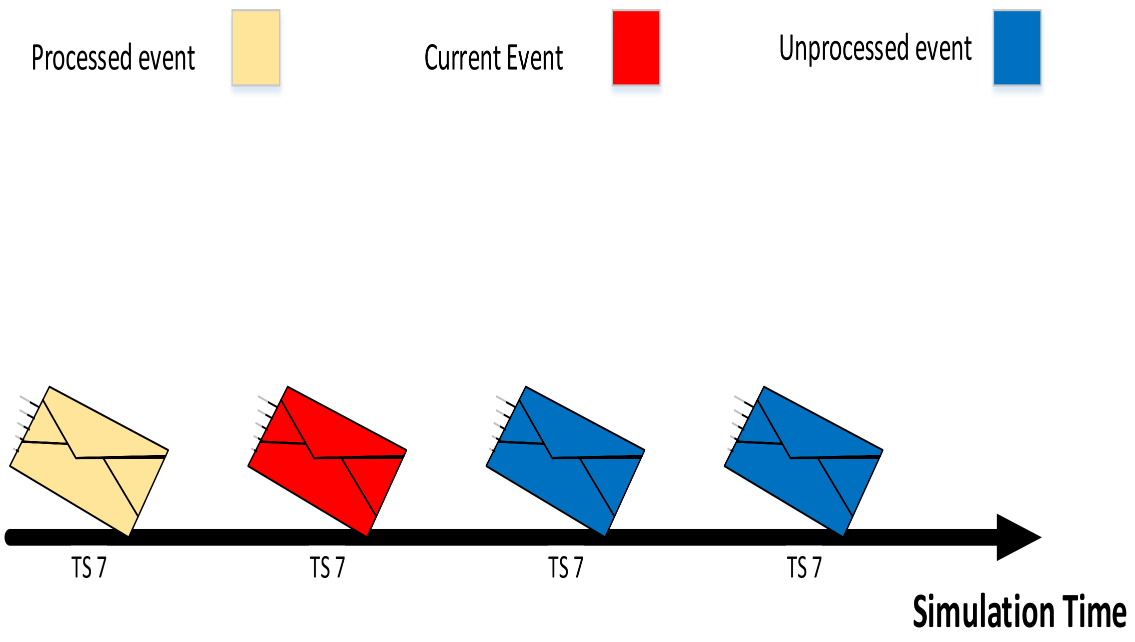


Figure 1 Simulation with Discrete Events

A simulation environment records state of the system over time. There can be a continuous or the discrete system. In continuous system the states of the program changes continuously over time. Whereas in discrete systems the state only modifies at a specific point in time [2]. In simulation progress of time is presented by the time progress function. Function classifies the simulation in two methods, event driven and time driven. The time is measured after small intervals in time driven method which gives the impression that the system is evolving continuously with time. Whereas time is measured in events which are the distinct points of time in simulation. Discrete systems are more appropriately simulated by event driven simulation method [3]. Moreover, these two methods can be combined for the simulation of more complex systems such as in simulation of computer networks, the fluid dynamics of the traffic can be captured by using continuous simulation method and the detail transactions of the network can be described by discrete events as shown in Figure 1.

In this thesis, our main focus is on discrete event simulation and its parallelization techniques. In discrete events simulation the event list is maintained which used a priority queue data structure and sorts the messages on time stamp on which they are scheduled. The current time of the simulation is specified by the clock variable. A loop can be used to process an event with the lowest time stamp and set the value of the clock with its timestamp. Each time when an event is processed the state of the simulation system changes which may result in generation of more events to be processed in simulated future. The loop makes sure that the simulation is running until the end of the simulation.

2.2 Parallel discrete event simulation

Parallel discrete events simulation is the execution of simple discrete event simulation program on parallel computers. By introducing the parallelism in the model, PDES can control the limitations of sequential simulations in memory space and execution time [4]. Hence the complex and large and time critical systems which require the immense computing resources can be efficiently modeled in PDES as shown in Figure 2.

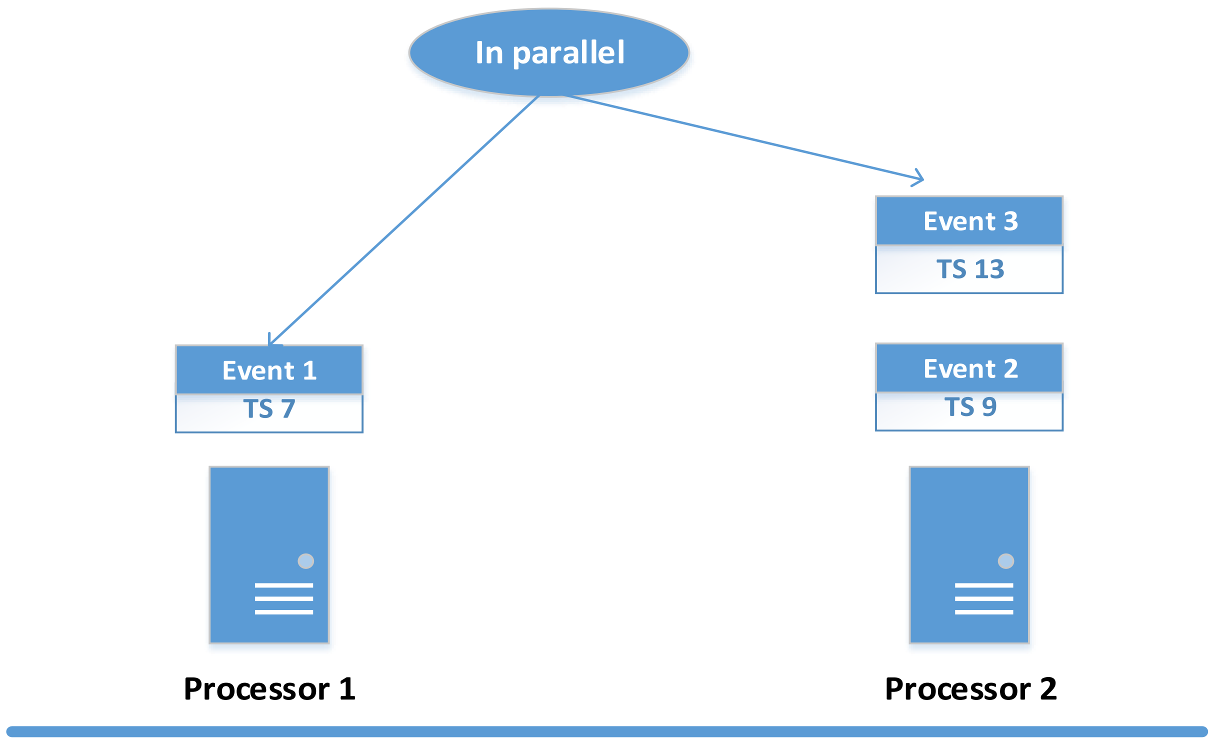


Figure 2 Parallel Discrete Events Simulation

Replicated trials is the simplest form of parallel simulations in multiple illustrations of a simulation program can be concurrently executed on parallel connected computers. This is a simple approach but its drawback is that it does not speedup the process and there is no memory efficiency because of the sequential execution.

Functional decomposition is the other type of parallel simulation in which different functions of the simulation program such as event handling and random number generation are assigned to different processors. The main problem of this form is that the simulation functions are tightly coupled and need to be synchronized among the components which can affect the parallelization.

A DES model works on an assumption that the system change states, only at the discrete points of time [5]. When an event is processed the simulation model changes its state. For instance in the simulating the communication network may use state variables to specify the communication links status, length of the message queues etc. A typical event can be receiving or forwarding a message at some node in the network or failure of any network component etc. our main focus is on the parallelization of DES programs to improve the efficiency of the simulation systems. The parallelization becomes difficult when we takes count of the operation of sequential DES program.

A sequential simulation uses three types of data structures (1) variables which specify the state of the simulation system. (2) List of events that contains all the unprocessed events which are being scheduled, (3) a clock to note the progress of the simulation. Every event change the state of the system which is being simulated and contain a timestamp to specify the time at which change has occurred. The main loop continuously process and remove the lowest time stamped event from the list. Event processing includes execution of some simulated code which reflect some change in the state of the system and in result more events are scheduled in the future to represent the causality relationship.

In this execution, it is important to process the event with the smallest time stamp. This is because an event with the larger time stamp can possibly modify the state variables of the event that have lower time stamp. Which results the system in which a future event can affect the past event that is unacceptable. These type of errors called causality errors as shown in Figure 3.

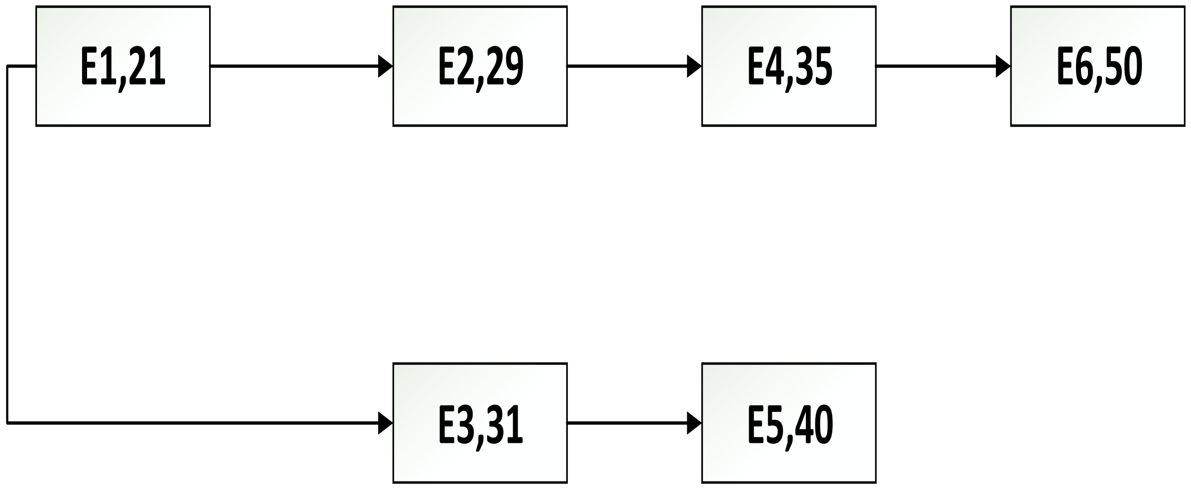


Figure 3 Causality Constraint

The parallelization of the simulation program involves in concurrently processing the events on different processors. Difficulty arises in direct mapping of above paradigm on to shared memory multiprocessor. Consider two events E1 and E2 with timestamps T1 and T2 with T2 > T1 are executing concurrently. If E1 modifies the state variable which is being read by E2 than E2 must be executed after E1 to ensure that there is no causality error. There must be some sequencing constraint to ensure the correct computation.

Existing technologies avoid such problems by making sure that no processes shall have direct access to the shared state variables. A physical system viewed as a combination of multiple physical processes which communicates with each other in simulated time. A physical process further have LP as shown in Figure 4. The communication between physical processes is done by sending times tamped events between logical processes. Every logical process represent some part of the state of the physical system it is representing and a clock variable which specifies the progress of the process. Causality errors can be avoided by adhering the local causality constraint that is all logical process should be processed in non decreasing timestamp order.it the responsibility of sequential simulation mechanism to ensure that no causality constraint should be violated during the execution of simulation program ion parallel computers. PDES is complex because the constraints on which the order is determined for the execution of the events are complex, highly data dependent and relative to each other.

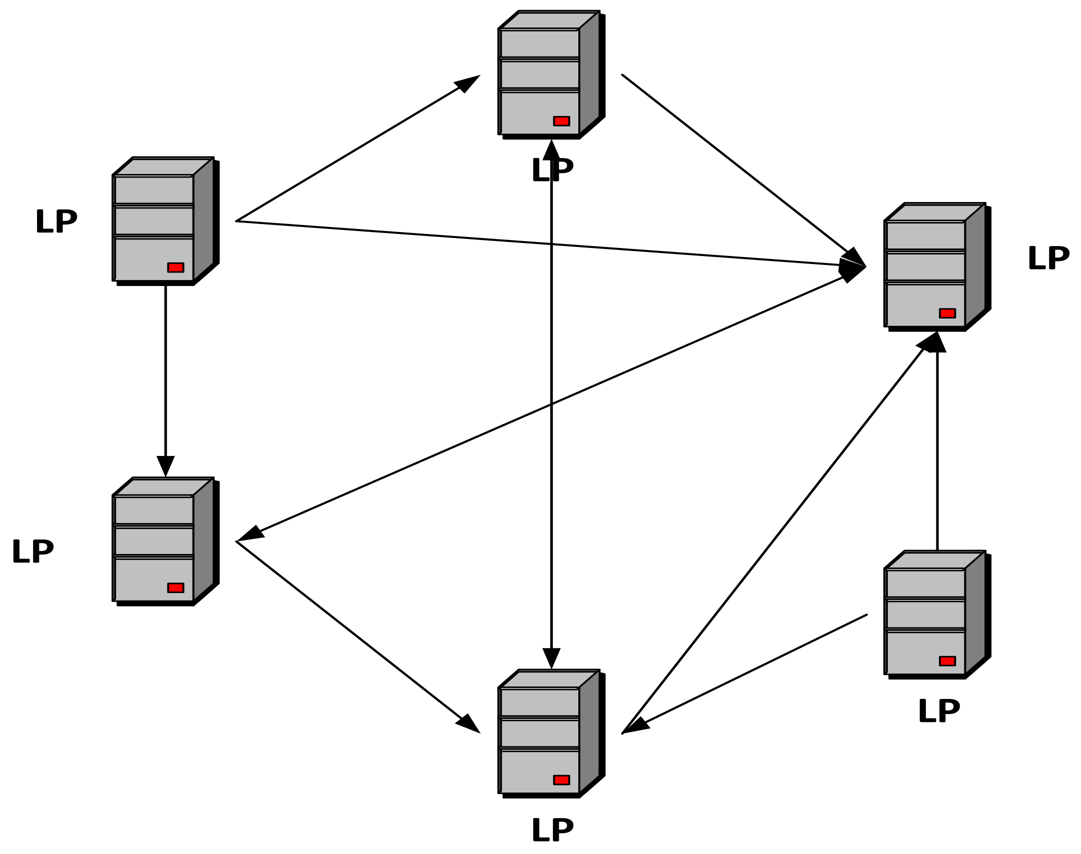
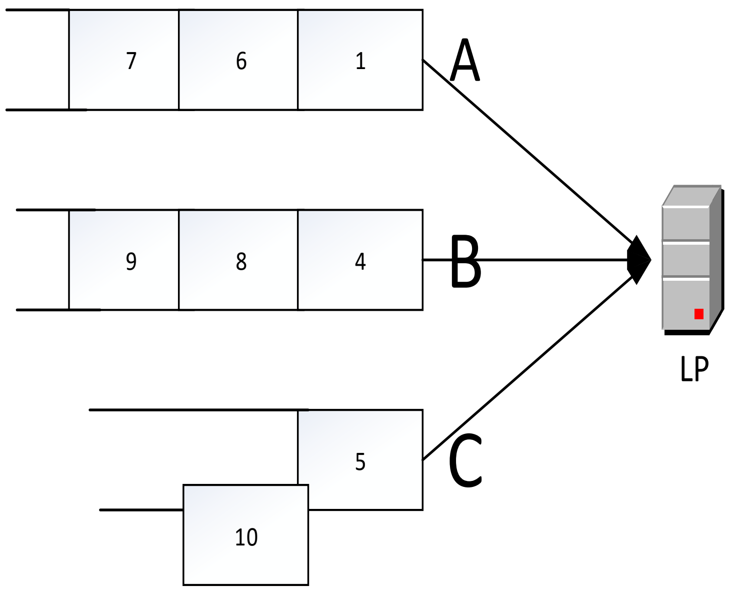


Figure 4 Logical Processors (LP)

PDES mechanisms is classified into two categories i) Conservative and ii) Optimistic [6].

2.2.1 Conservative mechanism

In this type of categories every possibility of causality error is avoided by following a strategy to process the events only when it is safe [7]. Initially the distributed simulation mechanisms follow the conservative approach. The major problem in this approach is to find out when it is safe to process an event. For example an event E1 with timestamp T1 can be processed, if a process can determine that there will be no such event in simulated future with time stamp smaller than T1.



Queue	Event
A	1
B	4
C	5
A	6
C	Block
A	7
B	8
B	9

Figure 5 Conservative Synchronization

In [8] and [9] authors designed algorithms of PDES in which it is statically specified that which process can communicate with which other process and on which link. To identify the time which is safe to execute an event, it is important that each message is transmitted according to their timestamp in chronological order. Every link is associated with a clock, which have the timestamp of the event that is at the queue's front. If the queue does not have any event then the clock has the timestamp of the event that has been processed latest. The process select the link with lowest clock value every time, if link's queue is not empty, the process updates its local clock to the link's clock and process the message. The order of the receiving events will be correct because each message in future will have the timestamp greater than a local clock because their arrival in chronological order. The process blocks the message if the queue is empty because it can later receive a message with greater timestamp than all the input timestamps. Process wait for a message to update the link's clock, before updating its local clock to ensure the chronology. This protocol ensure chronological integrity by processing events only in nondecreasing timestamp order as shown in Figure 5.

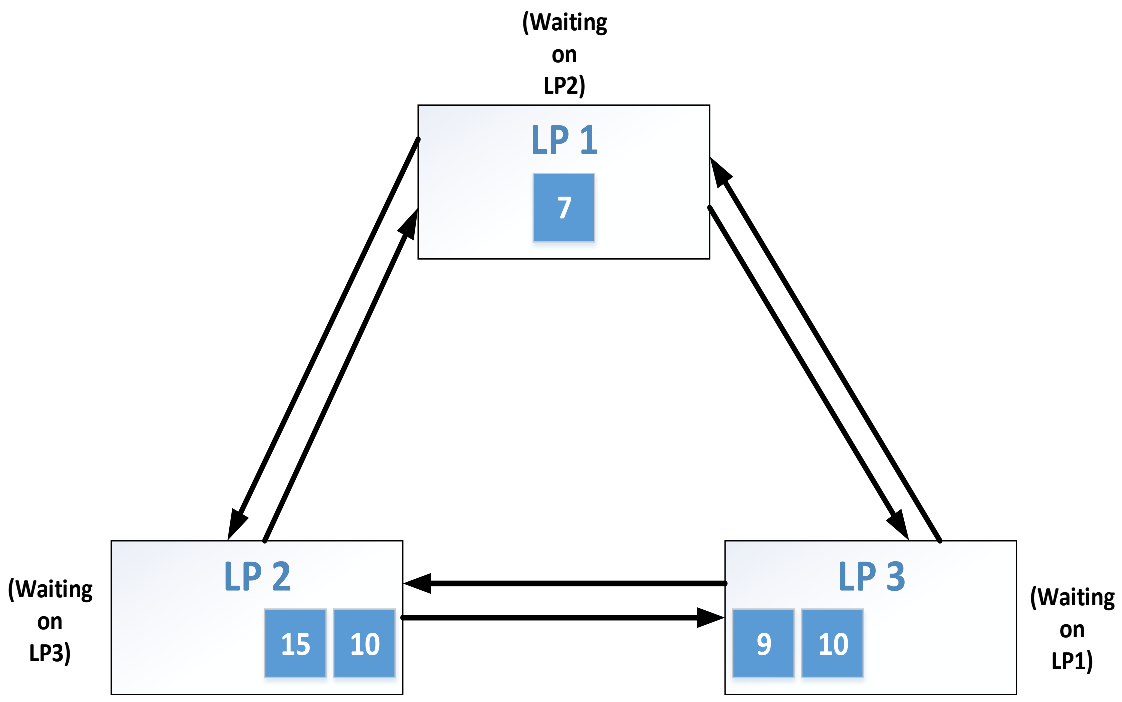


Figure 6 An example of deadlock

A cycle of blocked processes result in deadlock in the system and each process is blocked due to the small clock value. For example Figure 6 illustrates a deadlock scenario. All the processes are waiting on their incoming links because of the empty queue. Every process is on deadlock although other queues have messages that are waiting to be processed. Deadlocks are avoided by the null messages. Null messages does not have any involvement in the physical system and are only used for the synchronization of the system [10]. For example if LP_A sent time stamped null message is sent T_N to LP_B, it is just an indication to LP_B from LP_A that it a message with a time lesser than T_N will not be sent. The clock value of the incoming link is used to determine the timestamp of the unprocessed event that has to be removed next from the buffer of the link. This lower limit than can be used to identify the ts of the outgoing message. Every time the execution is done, the procedure sends a null message to each of the yield interface with this lower limit. Receiver than computes its lower bound and send it to its neighbor and so on.

2.2.2 Optimistic Synchronization

Optimistic synchronization works differently than conservative synchronization and allow the local causality errors to happen [11]. Event with smaller timestamp are allowed to execute as long as LP is able to identify the causality as shown in Figure 7. The problematic events are then reversed by rolling back. This approach is efficient than conservative approach and the topology of the simulation does not needed to be known.

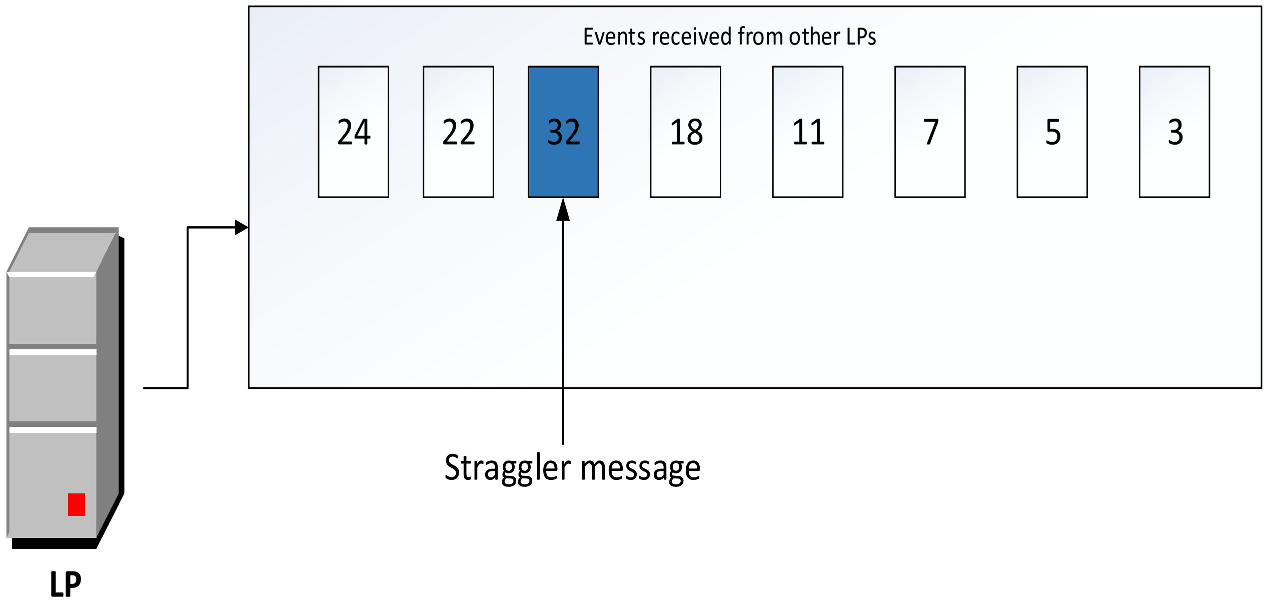


Figure 7 Optimistic Synchronization

Time wrap

Time wrap [12] [13] is the optimistic synchronization protocol for distributed simulations. In time wrap the local clock is set to the minimum received time of unprocessed events. Local clock is called Local virtual time (LVT). Process can execute the messages as long as it has input. During execution it may receive the event of smaller timestamp than LVT. In this case the process roll back in all the processed events with greater time stamp [14]. The event that caused the roll back called straggler. All the events that has been processed earlier are undone receiving a straggler.

This type of execution can affect two factors of the simulation that has to be rolled back. The state of the LP and other are the messages that are communicated to other processes [15]. State of the LP can be rolled back by saving states periodically and restoring them when roll back occurs. Events are rolled back by sending the corresponding anti messages which cancels the originally processed event. The anti message is used to annihilates the corresponding message exists in the queue. If an anti message is received for the message which has been processed earlier process receive the anti message of the message which has been processed earlier than the roll back must be started for that process.

The minimum of the timestamps of all the processes and the LVTs is known as Global Virtual Time (GVT). Events with lower timestamps than GVT can never be rolled back, so these events can be discarded to release the memory [16].

The above procedure can be understood as aggressive cancellation. The alternative of which is lazy cancellation [17]. Anti messages are not sent continuously in lazy cancellation. The process continue to process messages and when an anti message is scheduled it is compared with the messages in output queue, and if the same message is already queued than the anti message is discarded [18]. The anti message can only be sent if the same message was not regenerated earlier. Lazy cancellation may or may not improve the performance of the system.

Chapter 3

3. Literature Review

Cloud computing is a model where virtualized computing resources at remote location provide software as a service to the clients. Cloud computing provides the facility to clients to compute highly intensive tasks on remote resources and charge them on the basis of the usage [19]. Large scale parallel simulation require clusters and high performance computing machines which may become problematic because of their high costs. Cloud computing offers such facilities on lower costs and makes it more accessible to the simulation community. Moreover execution of parallel discrete events simulation over clouds opens different challenges in terms of performance. In this section we will focus on some of the techniques to improve the performance of PDES on cloud environment.

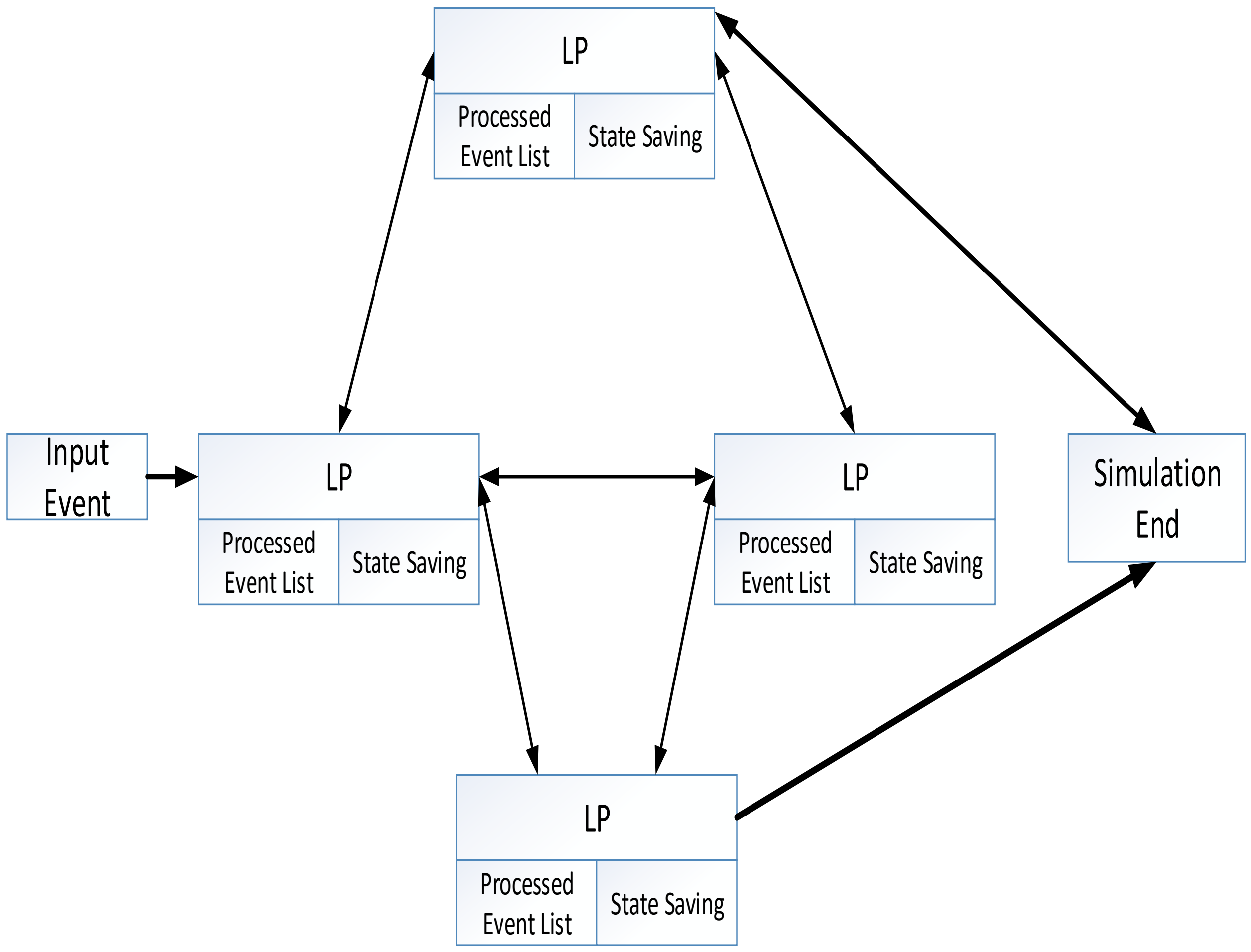


Figure 8 Time wrap framework for cloud computing environment

In [20] authors proposed a protocol to address the synchronization problem in optimistic synchronization. In cloud environment the existence of other user's computations may lead to increase the number of roll backs in traditional optimistic PDES i.e Timewrap. Some LPs which are used by some other user may slow down its progress as compared to the LPs that are lightly loaded which may become the cause of larger number of rollbacks and straggler messages. Furthermore cloud environment may have more communication delays as compared to the tightly coupled computing platforms which also become the cause of excessive straggler messages.

Timewrap is a well established protocol to resolve the synchronization issue in optimistic PDES that used roll backs to avoid local causality errors. Figure 8 shows the timewarp mechanism for cloud computing environment. Each of the LPs are allowed to process all events that they received. When the straggler is received all the events relevant to the straggler must be rolled back. TWSMIP address the challenges that arise when Timewrap is executed on cloud environment that are efficient utilization of resources, load distribution, efficient execution, fault tolerance and process synchronization. TWSMIP distribute status message across all the LPs residing on other processors. These status messages known as heart beat (HB) messages provide information about other LPs which may send the messages. These messages have information about sent messages to detect straggler message and to avoid roll backs. The number of HB messages can be reduced significantly by sending HB messages to only those processors which have communicated since the computation of last GVT value. Each process enters in HB phase after a fixed time and start sending HB messages to the processors to which it is communicating. The events are continuously processed by the LPs with the receiving of HB messages. When a straggler message is received through HB message LPs started to roll back to the time of straggler message. Anti messages also generated to stop other LPs from processing the false computations.

HB messages has two array fields one for the timestamp TS and other is for identification number of messages MID. LPs saves information of multiple messages in these arrays when it send messages to other LPs. Each LP stores timstamp and MID of all the messages it has communicated. So each LP has two list one that is maintained by itself which logs the messages that are arrived and other is for HB messages. These two lists are compared, if these two lists are not same than it means straggler messages is present in the simulation system. In this case computation stops and LPs are rolled back to the straggler time stamp. If straggler has TS higher than the local time then the LPs continues to process events till it reaches the time stamp of straggler. Figure 9 shows the straggler identification between two processes.

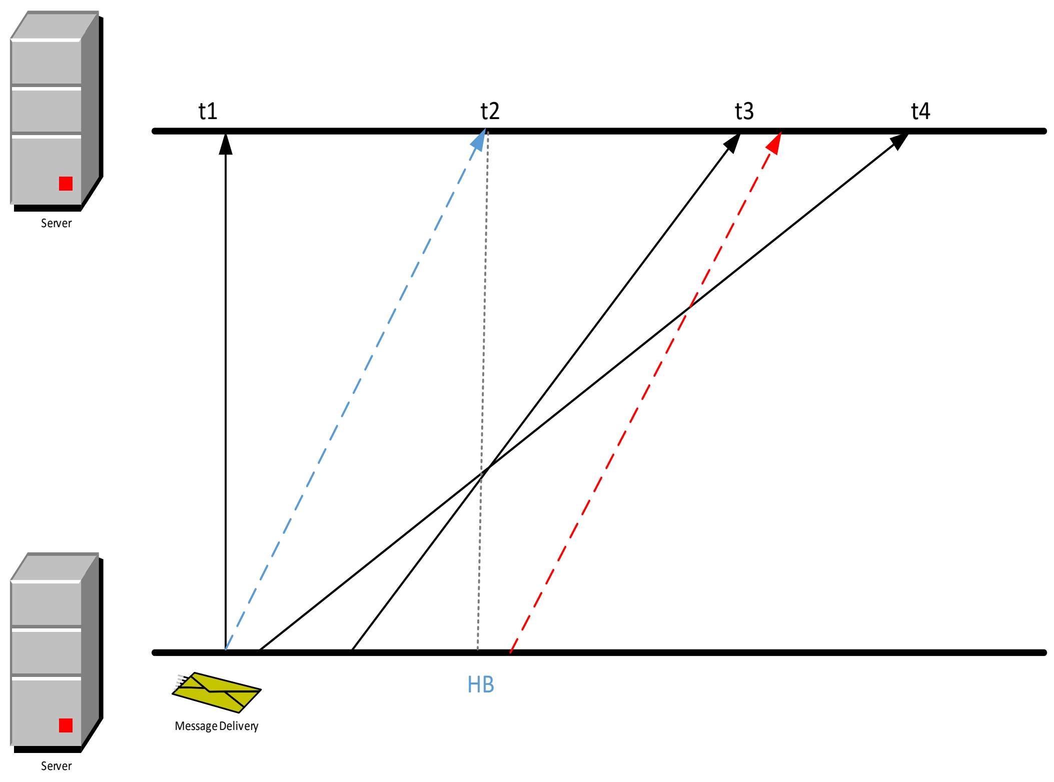


Figure 9 Straggler identification between two processes (LPi and LPj)

In [21] authors discussed the challenges for the execution of PDES on cloud environment. Cloud environment best performs in the conditions where high bandwidth is needed for communication. The problems arise for simulation applications which communicates by sending small messages and require the quick delivery of those messages. Cloud computing environment shares its resources among many users. Virtual clusters are assigned to the users but users does not have the access to the processors assigned to those virtual clusters. Execution of parallel simulation application can become problematic in such cases, specially those applications which uses optimistic synchronization techniques. Interactive simulation require the computations to meet the real time constraints so the response time can become critical in such applications.

To address these challenges authors adopted master worker architecture. Master worker paradigm in which large computations are divided into small tasks that are distributed to the multiple workers that works under the direction of the master. This mechanism is well suited for PDES in cloud environment. Despite sending the small messages among different LPs in traditional PDES, the master/worker mechanism aggregates the messages and send them as one unit which results in better utilization of the bandwidth. In master/worker paradigm the state of the LPs must be maintained so that it can be leased to a different worker. Furthermore the messages between the LPs must be recorded so that local causality constraint should be met.

The master/worker mechanism promotes the separation between the master and the worker. The master is further divided in three services as describe in**.** The services comprised of proxy, message state servers, and work unit state servers. This distributed environment helps to maintain scalability of the simulation system. The actual simulation is performed on the worker by contacting the backend services for the synchronization of simulation and the related messages. When the execution is completed the variables are uploaded to the state servers.

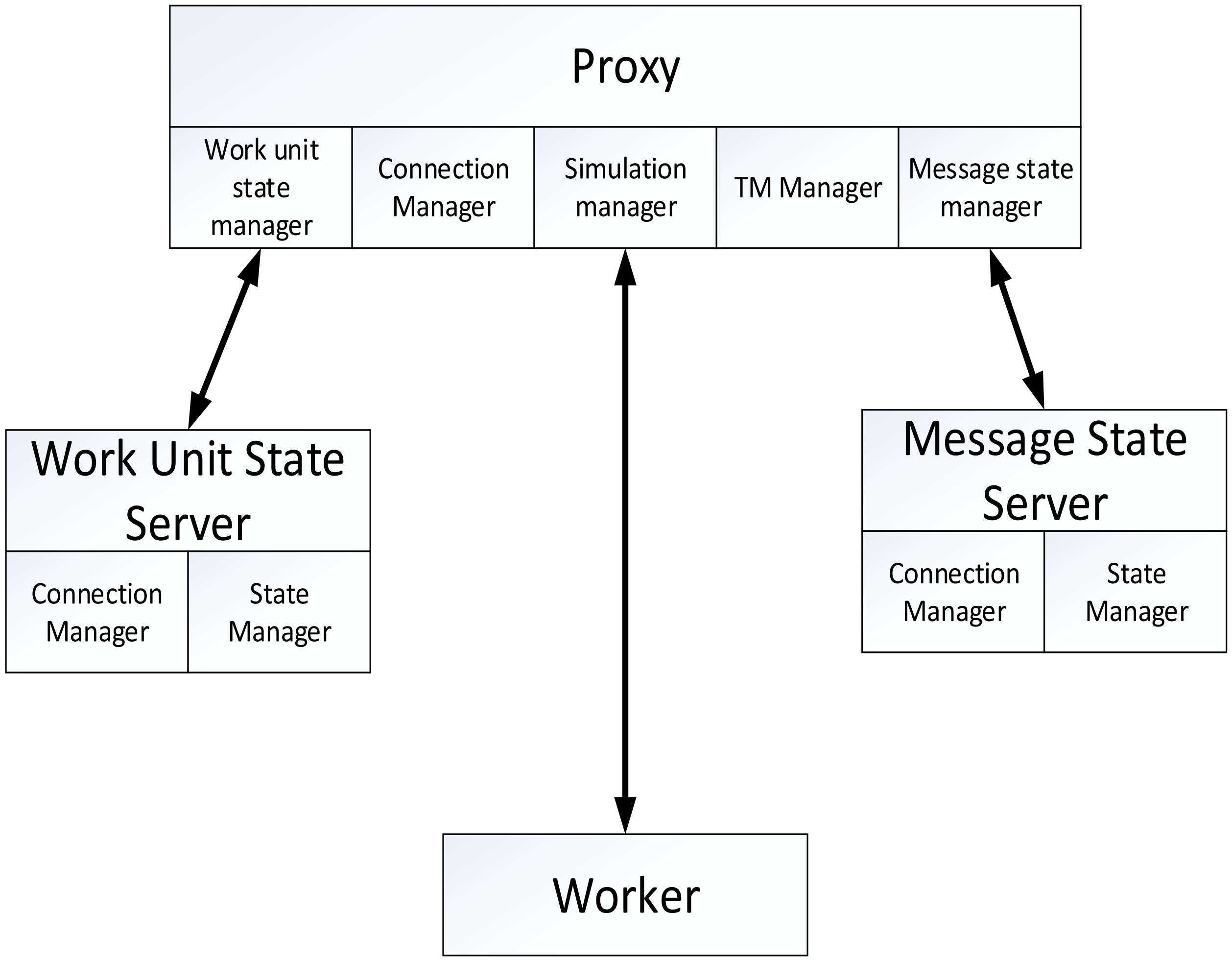


Figure 10 Aurora proxy internal components

The aurora proxy controls the simulation and all the services. The message state server and work unit servers are managed by associated managers in the aurora that record the metadata. These managers are i) state server manager, simulation package manager and message state server manager as shown in Figure 10.

Parallel applications heavily utilize resources for the execution making interaction between nodes nearly impossible. Hence, vacating the processors are important in parallel application so that owner of the machines can resume the interactive work either by checkpointing the the application or by migrating the processor to other hosts. In [22] authors presented a new approach of migration and check pointing for parallel applications. Transparency of the checkpointng from the application perspective is provided so that application must not be aware of migration of processes or the checkpointing. Virtual address is used so that processes in the application used same address till the end. Mapping of the original to current addresses is maintained by each process and these tables have to update upon migration.

Ready message is used to identify that there should not be any message in the transit when the migration or the checkpoint is taken. When a checkpoint has to be taken each process is notified. Ready message is sent by each process to every other process to indicate that the channel from this process is clear. Checkpoint is to be taken when each process received the RM and the application is restarted. After restart the mapping tables are updated so that current addresses can be updated. Upon restart the receiving messages are checked from the buffer that there are not the same messages that were received earlier. If there is such a message present in the buffer it is retrieved and returned. One special process designated as a coordinator called deamon process which is responsible for dynamic load management, resource management or a user command migration.

The simulation applications are characterized by their size, scale, event interaction and the nature of computation. The desired features require highly scalable PDES engines for processing. In [23] a PDES engine is designed to process time warp based optimistic parallel applications. The engine runs on massive parallel architectures with minimal overhead.

The engine is designed with a unique approach to cater the wide range of synchronization techniques. It comprises of a unified architecture to support multiple types of simulations. The processes cake make use of many synchronization mechanisms and can dynamically select the required mechanism. Three priority queues containing LPs references are maintained on kernel of the engine. A queue that holds the LPs ordered on the earliest events that could be committed called Committable queue. Second queue holds the LPs ordered on earliest events that could be executed called Process able queue. Third queue holds the LPs ordered on earliest events that could be generated in future called Emittable queue. The engine uses internal LPs which are used for synchronization and parallel communication. Kernels also have flow control mechanism implemented on them so that the messages that are failed because of the full buffers could be sent again by the kernel processes. PHOLD benchmark application is used to study the performance. PHOLD helps in testing the performance and functionality. In PHOLD a fixed number of the messages distributed between LPs.

In [24] authors presented a thread based model ROSS-MT for PDES simulator for minimizing synchronization delays and message copying. In traditional PDES simulators processes are used that communicates by sending messages between each other. In ROSS-MT threads ae used instead of processes as shown in Figure 11. The motivation for using threads is that there is no need to transfer messages between threads because of their same memory image. Therefore a single input queue was maintained in place of queues for each thread, which records all the events from other threads. Furthermore a thread contains its own scheduler, event queue and memory manager for fossil collection.

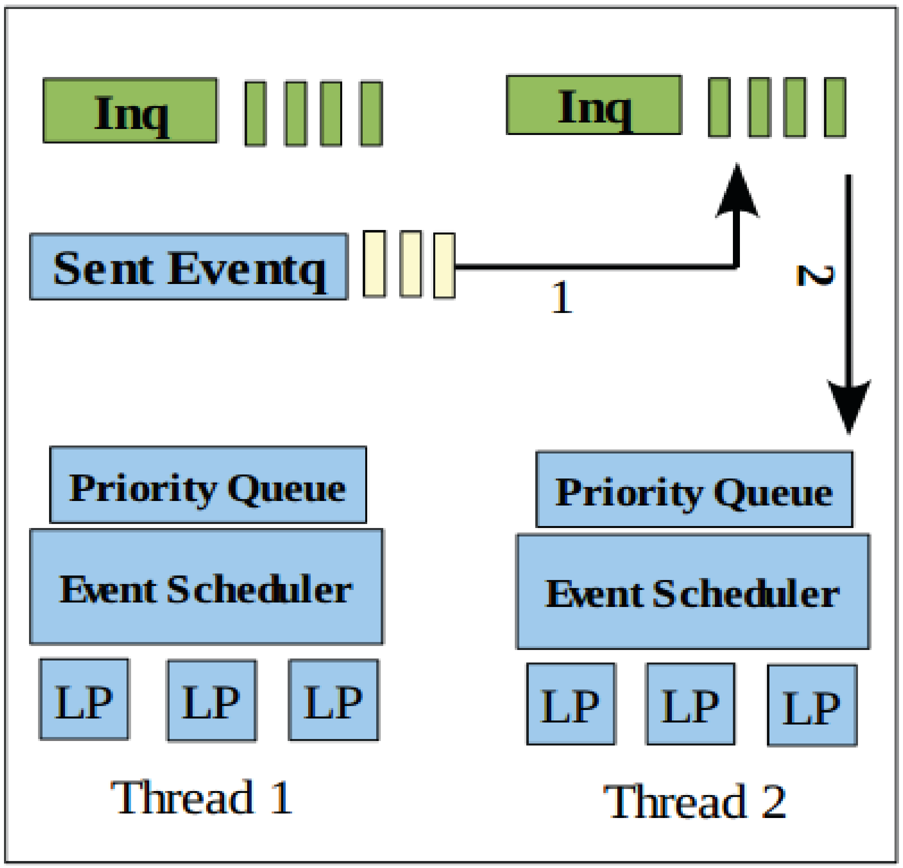


Figure 11 Multi threaded ROSS

A pointer is attached in input queue of the destination for communication. The sender threads keep copy of every message it has sent so that if rollback occurs anti messages can be sent. Threads on the receiver uses priority queue for processing the events, events are transferred from input queue to priority queue for processing. So synchronization delays are completely avoided. The results shows the substantial improvements in performance.

In [25] authors presented a master worker architecture (Aurora) for the execution of large scale PDES programs over computational resources connected through a network. Several researches have been done to utilize the massive computing power of the internet through public distributed projects. Projects based on volunteer distributed computing like SETI@home [26], distributed.net and community Grid were successfully completed. The idea of Aurora is to minimize the gap in general computing and the PDES by using web services. Web services allow interoperable communication to the application over the internet. Web services focuses more on interoperability over performance where as in aurora the priority id given to performance and interoperability is provided on the machine architecture level.

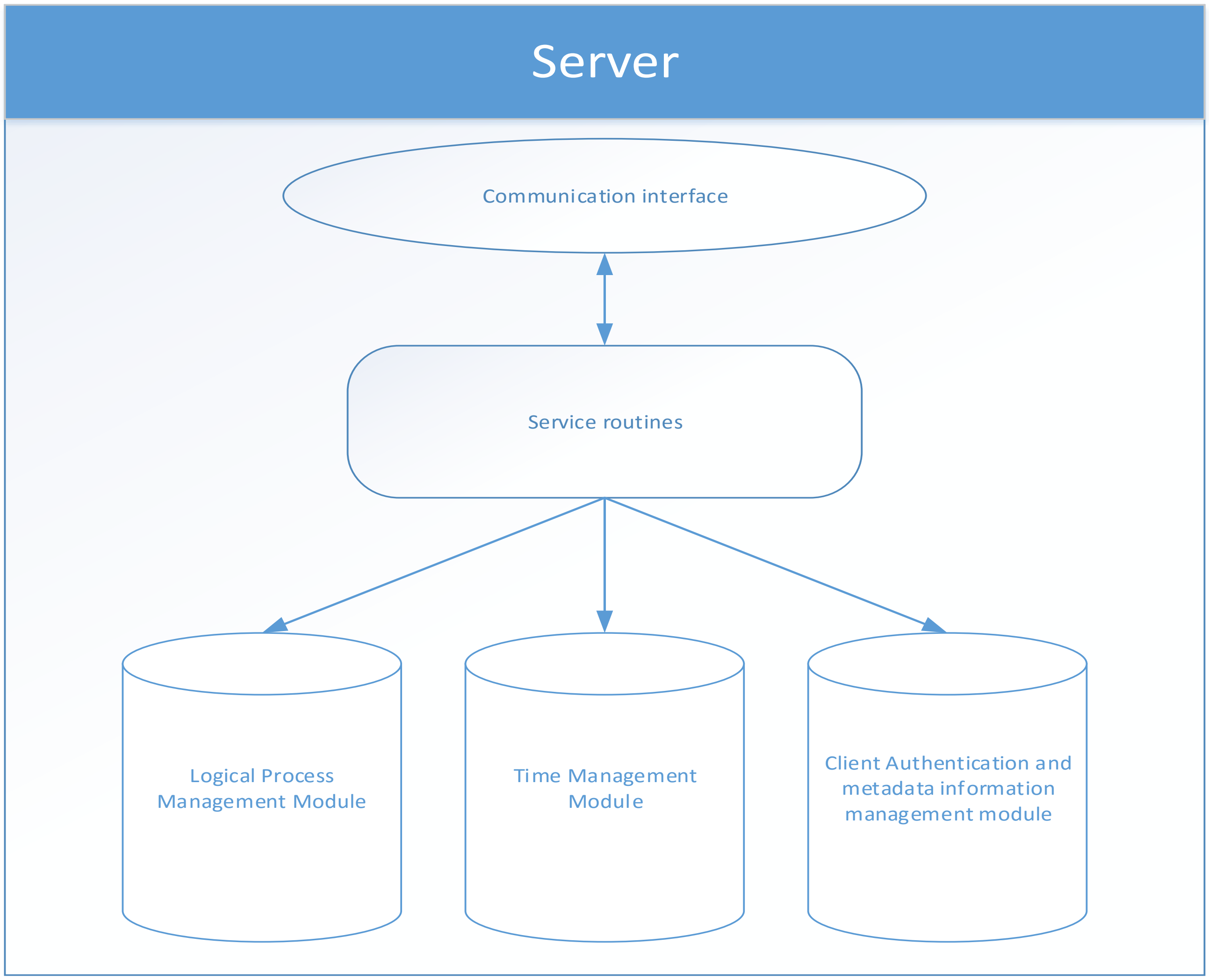

Server
Communication interface
Service routines
Logical Process Management Module
Time Management Module
Client Authentication and metadata information management module

Figure 12 Aurora Server

Aurora works on basic principles of PDES comprises of collection of LPs communicating by exchanging time stamped events. In aurora LPs are grouped into work units which is the unit of work transmitted between master and a worker. Master manages the work units and allocate them to each worker. Worker performs the executions and upload the results to the master and the cycle continues until the end of the simulation. Aurora works on the basis of conservative synchronization algorithm to execute only the events with ts order to avoid local causality constraint. Aurora uses the SOAP as the default communication mechanism to support the interoperability between heterogeneous play forms. The communication system shown in Figure 12 is invoked during the handshaking of client and server. The disadvantage of using the web services for PDES is the low performance [27]. To overcome this limitation gSOAP [28] toolkit is used which is designed with high performance and languages and machine interoperability as well. The management and control algorithms which are combined to make a master are managed by the aurora server. Server contains the three modules i) Logical process management ii) time management iii) Aurora client. The Figure 12 shows the interaction between different components of the aurora server. LP manager analyze all the LPs. LPs may be aggregated by the application to form a single work unit. Each work unit stored on server contains the state vectors for simulation variables. After the execution of the work unit the client returned it back to the server. State vectors of the work units are updated by the LP manager. The synchronization support is provided by the aurora time management module. A simple algorithm to compute the LBTS value of future messages is sufficient because the time management executions are done on the server. The client authentication and management module keep record of all the work units that are either available or not. The available work unit shows that no computations are going on by any client on the work unit and it can be leased to any client's request. A work unit when leased assigned a unique global key to allow the work unit to be issued more than once or it can be issued to multiple machines for fast execution. The workers in the architecture are maintained by aurora client. The work units are received by the aurora client from server through web service requests execute it and uploads the message buffer and state vectors back to the server. The aurora client send series of requests to the server for the available work units. State vector and the incoming messages are downloaded from the server when the client receives the availability of work unit from the server and start the execution. Once the execution is completed the state vector is uploaded back to the server.

In [29] a technique named Dynamic Local Time Window Estimate (DLTWE) is presented to control the optimism in PDES. DLTWE is particularly designed to increase the efficiency of simulation applications with highly variable time intervals between inter-LP events. In DLTWE each LP estimate the timestamp of the next outgoing event and keep sending these estimations to the neighboring LPs which update their local simulation time using these estimates. The occurence of the straggler messages cannot be rulled out because the timestamps are marely estimates, so LP must do rollbacks when received a straggler message. Moreover if the estimates are correct and accurate then there will be less number of rollbacks which results in higher efficiency.

In DLTWE each LP maintain three types of data structures i) sub-domain state, ii) event queue and iii) roll back history. The LPs communicate with other LPs by sending time stamped events. Each LP process these events and update the state variables. If the event is a straggler then roll back is performed. In this paper authors have presented two typed of roll backs i) simple roll back and ii) selective roll back. In simple roll back the events that were processed after the straggler are processed back and the LP's time is set to the previous time of the straggler. The messages that were sent during the roll back are cancelled by using anti messages and sending them to the corresponding LPs. On receiving anti message the roll back process is started on the LP. In selective roll back when LP received a straggler message it performs a refined analysis before roll back. The LP finds the events that were processed and are related to the straggler and roll back only these events which ultimately reduces the cost from the simple roll back. Roll back due to the straggler messages may decrease the efficiency of the system. So an LP should not update its local time same as the time stamp of the straggler message that may be received in future. In DLTWE each LP send the estimated ts of the next straggler to the neighboring LPs which is calculated with the help of current members of the events queue. The LP does not update its local time from the received estimate time stamp.

Chapter 4

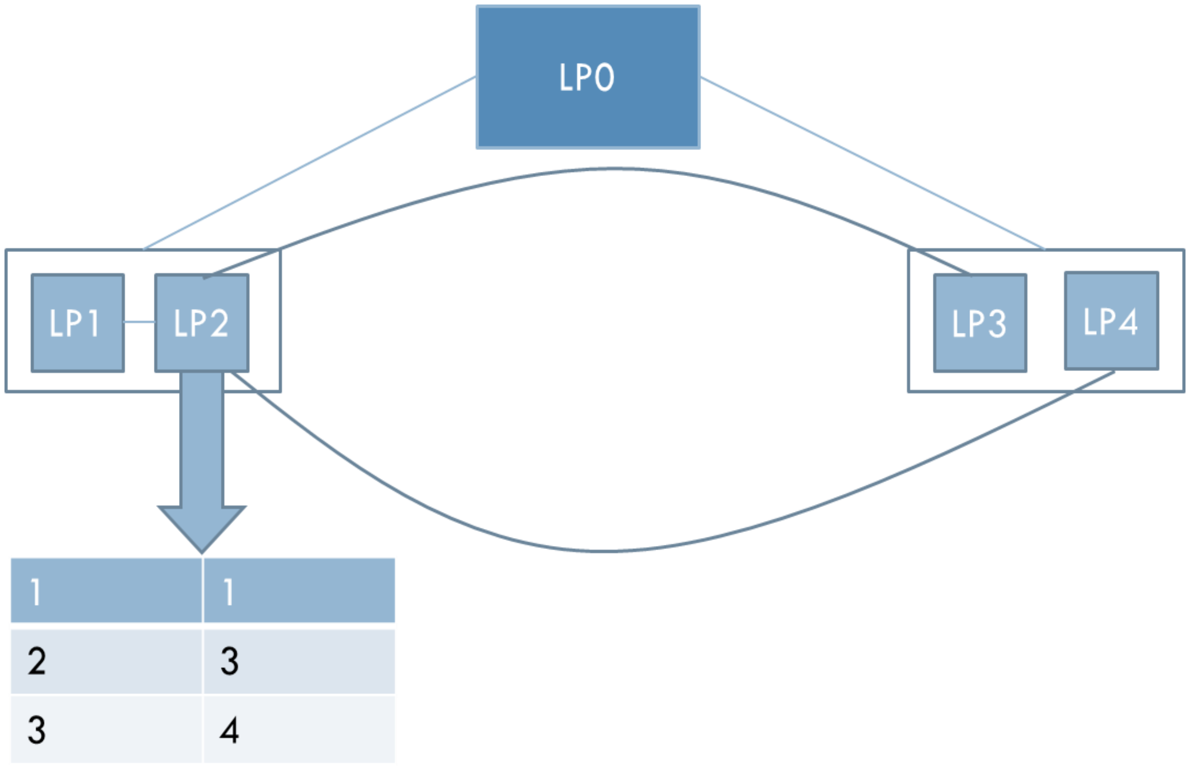
4. Proposed Scheme

Cloud computing provide the users with the facility of executing their tasks on virtualized resources at remote locations. Large scale high performance computing tasks need high performance resources for computations which ultimately increase the costs. In cloud computing environment clients does not need to worry about the physical systems, virtualized infrastructure is being provided to the clients to execute the complex tasks. Cloud has provided the cost effective solution to the HPC community.

PDES is a collection of LPs which communicates with each other by sending time stamped messages. The major problem in PDES is the synchronization of the LPs. Optimistic approach uses roll backs mechanism to ensure the synchronization in a PDES program. Time wrap is a well known approach used for the optimistic synchronization. Execution of time wrap over a multi tenant cloud environment may lead to a excessive number of roll backs due to the computation of other users on some LPs. Moreover because of virtualized nature of cloud environment some LPs may reside far from each other resulting in longer communication delays ultimately increasing the number of roll backs.

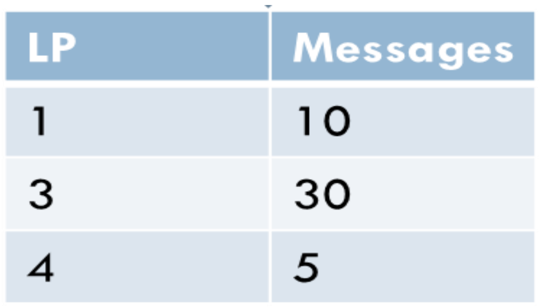
Our proposed methodology involves around the minimization of communication delays and eventually the number of roll backs in execution of PDES program over cloud environment. As discussed earlier the LPs communicate with each other by sending time stamped events with other LPs. Each LP maintains two queues one is to log all the message received from other LPs according to their timestamps and other is to maintain record of all the messages sent to other LPs. Each time when a message is received from some LP. It is compared with the time stamp of the last received message. If the time stamp of the received message is less than the last received message than LP started the roll back mechanism and send anti messages to other LPs to which it communicated earlier to undo all the false communication. Each time when an anti message is received the LP undo all the messages which were being processed earlier and have the time stamp less than the time stamp of the anti message

To minimize the roll backs each LP continuously monitors the communication with other LPs. After a fixed time LP goes to analyzing phase and analyze the number of messages sent to other LPs. The LP to which the maximum messages were sent is requested to swap with the neighboring LP, so that the distance between the more communicating LPs can be minimized which eventually result in decreasing the number of roll backs occurring during the execution. This methodology helps in reducing the communication delays, improve the efficiency of the system and ultimately decrease the number of roll backs.



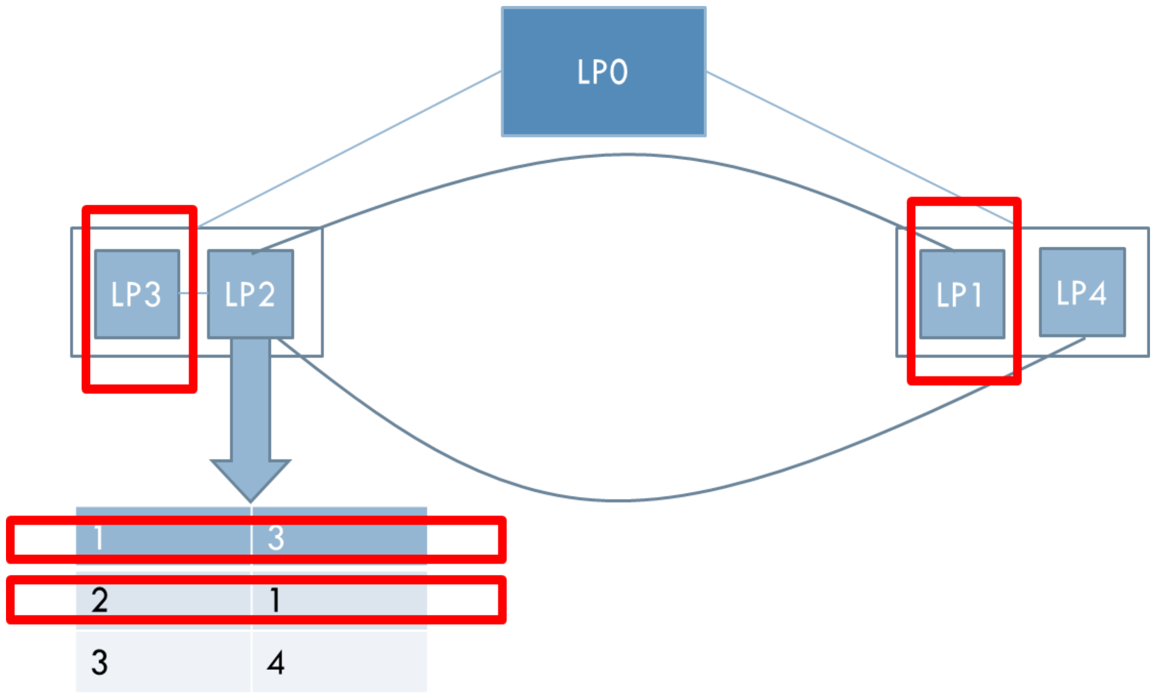
1	1
2	3
3	4

Figure 13 Message passing mechanism



LP	Messages
1	10
3	30
4	5

Figure 14 Number of messages sent from LP2 to other LPs



1	3
2	1
3	4

Figure 15 Migration of LP and table update

Each LP select random LPs from its routing table to send messages. In Fig 13 LP2 start sending messages to other randomly selected LPs from its routing table. LP0 is a leader selected to monitor the traffic pattern among LPs and helps later in migration of LPs. After some time simulation goes into a pause state to monitor the messages sent to other LPs. Each LP keeps record of the number of messages it has sent to other LPs. In Fig 14 a table of LP2 shows the number of messages which LP2 has sent to other LPs. This table is then sent to leader LP for analysis that to which LP it has sent more messages. In this case LP3 is the LP that has to be migrated. All the state variables of LP3 are sent to the nearest LP and tables of LP2 are updated that has been shown in Fig 15.

Chapter 5

5. Performance Evaluation

In this section the performance of the proposed methodology are presented.

5.1 Benchmark Application

For performance study we used PHOLD which is a standard benchmark for PDES applications. In PHOLD LPs distributes a fixed number of events to random destination with increased time stamps. PHOLD is used widely for performance evaluation. Model comprises of the objects that contains the events. An event is sent to other randomly selected object and later that object send another event to the third object and the simulation continues like this. The number of events in the simulation remains constant.

For the experiments we used 10000 events distributed across different number of LPs in different scenarios. The comparison was done with the performance of traditional time wrap protocol. We analyzed parameters as

i) Number of rollbacks

ii) Efficiency of the system

5.2 Results

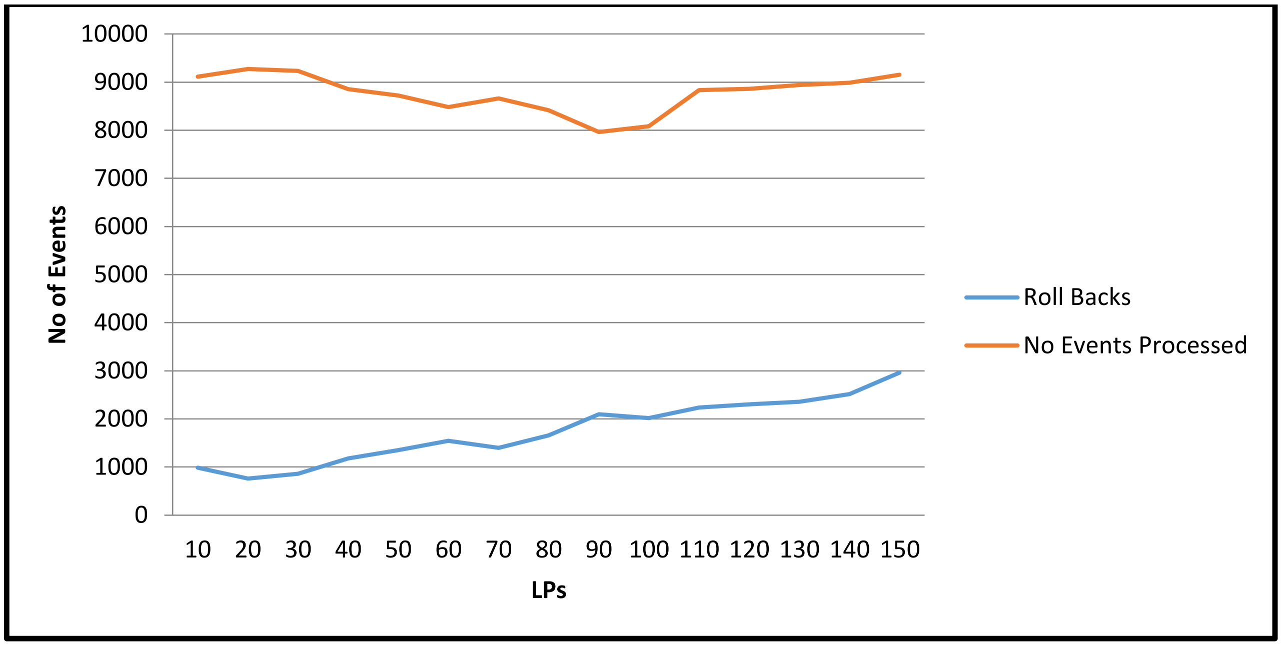


Figure 16 Roll backs in Traditional Time wrap

We first implemented traditional time warp protocol to justify the results and the comparison of our proposed approach. Fig 16 shows average number of committed events and the number of roll backs during the number of simulation experiments. The Y-axis represents the total number of events and X-axis represent the number of LPs involved in each simulation. The simulation was performed using LPs ranging from 10 to 150.

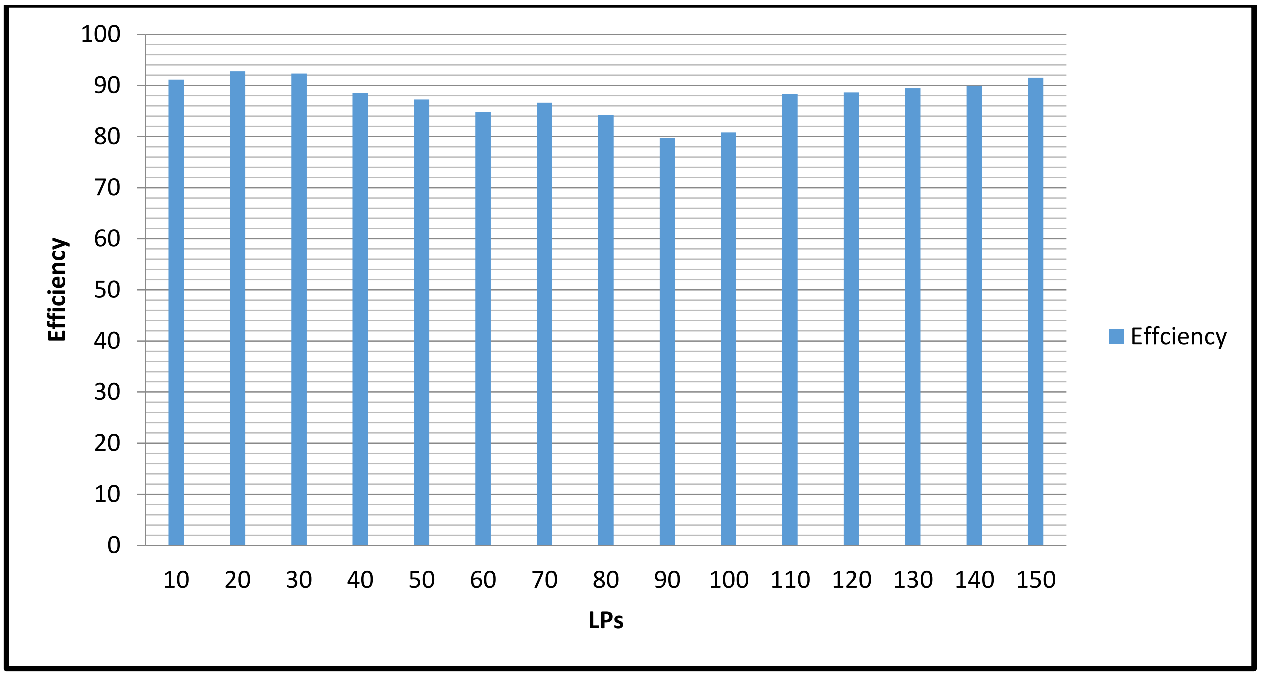


Figure 17 Effciency of Traditional Timewrap

Figure 17 shows the efficiency in percentage for the traditional timewrap protocol for the above performed experiments. Efficiency increases with the increase of number of LPs in most of the cases as shown in fig. efficiency was calculated on base of the results shown in Figure 16.

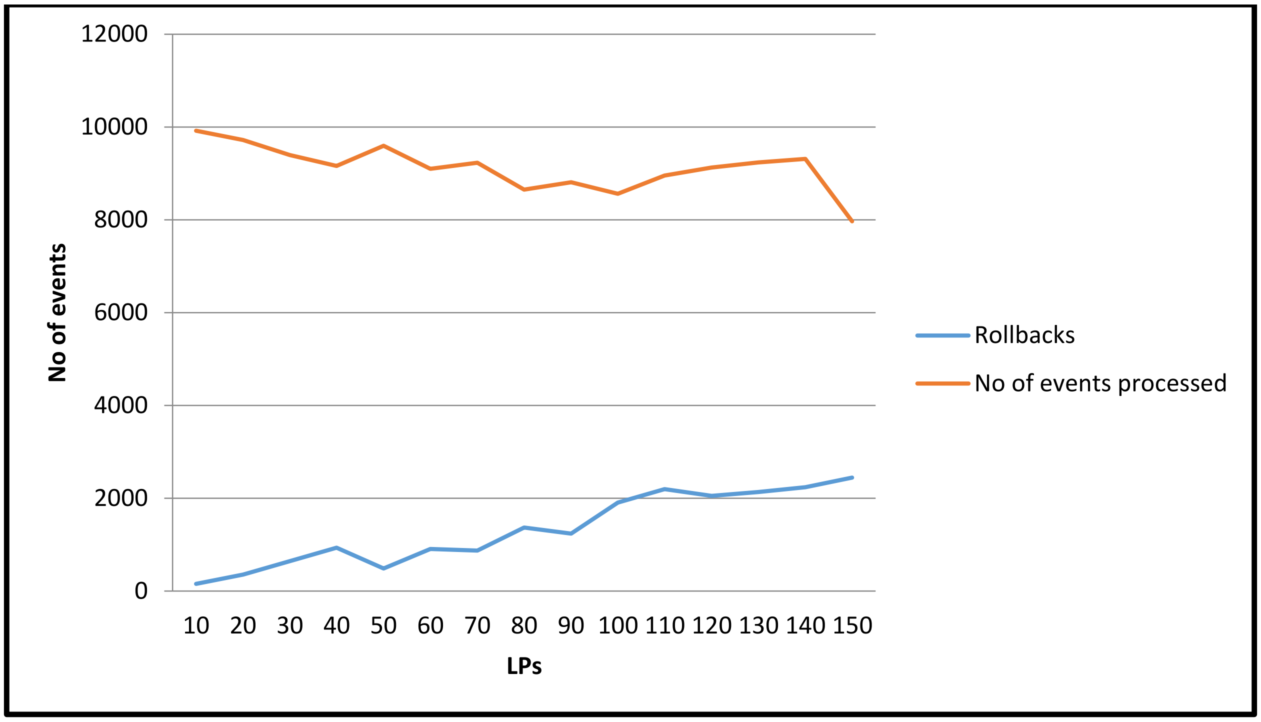


Figure 18 Roll backs in Proposed Technique

Figure 18 shows average number of committed events and the number of roll backs during the number of simulation experiments of our proposed approach. The Y-axis represents the total number of events and X-axis represent the number of LPs involved in each simulation. The simulation was performed using LPs ranging from 10 to 150.

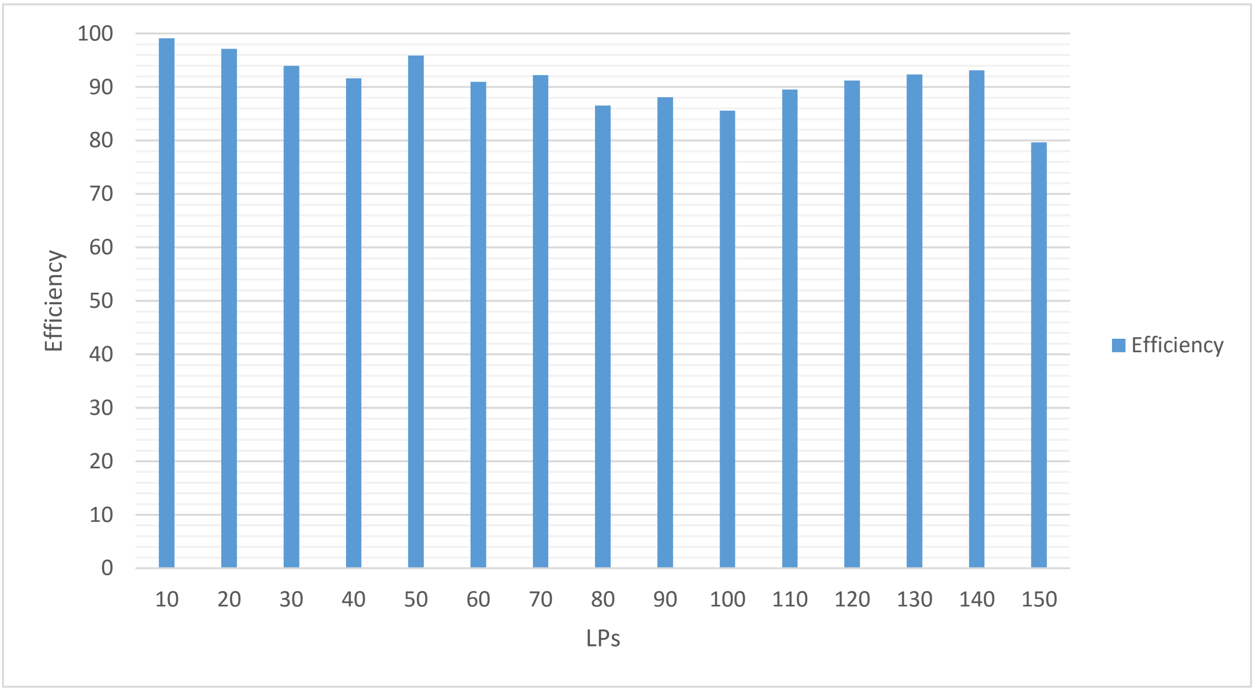


Figure 19 Efficiency of Proposed Technique

Figure 19 shows the efficiency in percentage for the traditional timewarp protocol for the above performed experiments. Efficiency increases with the increase of number of LPs in most of the cases as shown in fig. efficiency was calculated on base of the results shown in Figure 18.

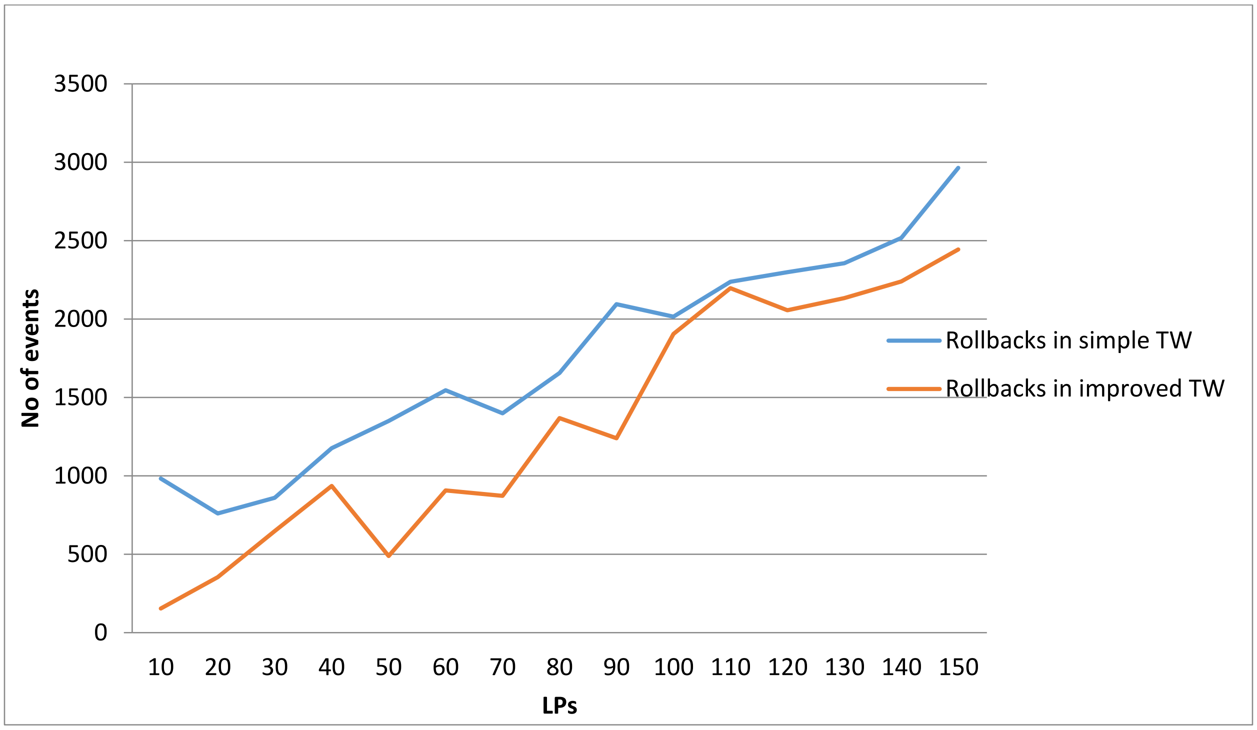


Figure 20 Roll backs comparison between traditional Timewarp and Proposed Scheme

The comparison was done from the results that we produced from traditional time warp and our proposed approach. Figure 20 shows the comparison of number of roll backs occurred in traditional time warp and our proposed approach. It can be seen that number of roll backs significantly reduced in our proposed approach.

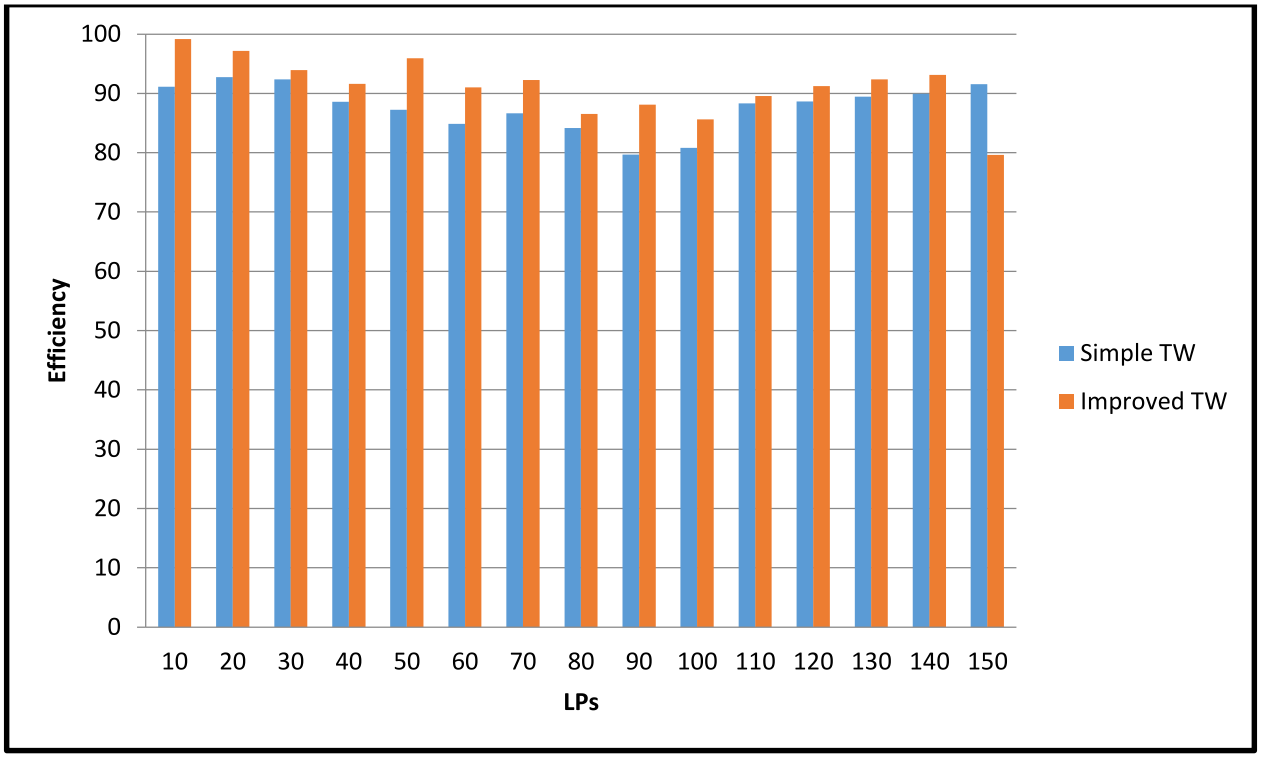


Figure 21 Efficiency comparison between traditional Timewap and Proposed Scheme

Figure 21 shows the comparison in terms of efficiency for the traditional timewrap and our proposed technique and it was observed that our proposed technique reduces the number of roll backs thus ultimately improves the overall efficiency of the simulation. Which results in better performance of overall simulation execution.

Chapter 6

6. Conclusions and Future Directions:

6.1 Conclusion:

In a multi tenant environment like cloud computing environment resources are shared among many users. The work load of every user may vary over the time. During the on running execution of parallel simulation applications new resources may become available and the existing resources may become heavily loaded due to other users work load. An ideal program must adapt these changes for the better utilization of resources and to increase the performance of the system. PDES program communicate through messages between the processes which results in increase in network cost and message failure due to the traffic on the network. Traditional Time wrap program on cloud environment can perform poorly because of the multi user shared nature of the cloud environment which leads to uneven processing.

In this thesis we studied the options regarding the improvement of the performance of simulation in cloud environment. We designed a new process migration protocol to decrease the number of roll backs by decreasing the gap between the LPs, which also improve the overall efficiency of the system. The gap between the LPs are minimized by migrating the LPs nearest to each other by assessing the communication between them. Experiments shown significant results with less number of rollbacks and improved efficiency of the simulation program.

7. References

[1] Liu, Jason. Parallel Discrete-Event Simulation. John Wiley & Sons, Inc., 2009.

[2] Overeinder, B. J., L. O. Hertzberger, and P. M. A. Sloot. "Parallel discrete event simulation." (1991): 19-30.

[3] Zeigler, Bernard P., Herbert Praehofer, and Tag Gon Kim. Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems. Academic press, 2000.

[4] Livny, Miron. "Study of parallelism in distributed simulation." SCS Conf. Distributed Simulation. 1985.

[5] R. M. Fujimoto, "Parallel discrete event simulation," in *Communications of the ACM 33.10 (1990): 30-53*.

[6] Reynolds Jr, Paul F. "A spectrum of options for parallel simulation." Proceedings of the 20th conference on Winter simulation. ACM, 1988.

[7] Shchur, Lev, and Liudmila Shchur. "Parallel Discrete Event Simulation as a Paradigm for Large Scale Modeling Experiments." (2015).

[8] R. E. Bryant, "Simulation of packet communication architecture computer systems," (1977).

[9] Chandy, K. Mani, and Jayadev Misra. "Distributed simulation: A case study in design and verification of distributed programs." Software Engineering, IEEE Transactions on 5 (1979): 440-452.

[10] Nicol, David M., and Paul F. Reynolds Jr. "Problem oriented protocol design." Proceedings of the 16th conference on Winter simulation. IEEE Press, 1984.

[11] Jafer, Shafagh, Qi Liu, and Gabriel Wainer. "Synchronization methods in parallel and distributed discrete-event simulation." Simulation Modelling Practice and Theory 30 (2013): 54-73.

[12] Jefferson, David R. "Virtual time." ACM Transactions on Programming Languages and Systems (TOPLAS) 7.3 (1985): 404-425.

[13] Jefferson, David, and Henry Sowizral. Fast Concurrent Simulation Using the Time Warp Mechanism: Part I, Local Control. Rand Corporation, 1982.

[14] Rönngren, Robert, et al. "Transparent incremental state saving in Time Warp parallel discrete event simulation." ACM SIGSIM Simulation Digest. Vol. 26. No. 1. IEEE Computer Society, 1996.

[15] Barnes Jr, Peter D., et al. "Warp speed: executing time warp on 1,966,080 cores." Proceedings of the 2013 ACM SIGSIM conference on Principles of advanced discrete simulation. ACM, 2013.

[16] Carothers, Christopher D., and Richard M. Fujimoto. "Efficient execution of time warp programs on heterogeneous, NOW platforms." Parallel and Distributed Systems, IEEE Transactions on 11.3 (2000): 299-317.

[17] Lin, Yi-Bing, and Edward D. Lazowska. Reducing the state saving overhead for Time Warp parallel simulation. University of Washington, Department of Computer Science, 1990.

[18] Perumalla, Kalyan S. "Parallel Discrete Event Simulation." (2014).

[19] Guo, Haifeng, et al. "The application of utility computing and Web-services to inventory optimisation." Services Computing, 2005 IEEE International Conference on. Vol. 2. IEEE, 2005.

[20] Malik, Asad Waqar, Alfred Park, and Richard M. Fujimoto. "Optimistic synchronization of parallel simulations in cloud computing environments." Cloud Computing, 2009. CLOUD'09. IEEE International Conference on. IEEE, 2009.

[21] Fujimoto, Richard M., Asad Waqar Malik, and A. Park. "Parallel and distributed simulation in the cloud." SCS M&S Magazine 3 (2010): 1-10.

[22] G. Stellner, "CoCheck: Checkpointing and process migration for MPI.," in *Parallel Processing Symposium, 1996., Proceedings of IPPS'96, The 10th International. IEEE, 1996*.

[23] Perumalla, Kalyan S. "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.

[24] Jagtap, Deepak, Nael Abu-Ghazaleh, and Dmitry Ponomarev. "Optimization of parallel discrete event simulator for multi-core systems." Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International. IEEE, 2012.

[25] Park, Alfred, and Richard M. Fujimoto. "Aurora: An approach to high throughput parallel simulation." Principles of Advanced and Distributed Simulation, 2006. PADS 2006. 20th Workshop on. IEEE, 2006.

[26] Anderson, David P., et al. "SETI@ home: an experiment in public-resource computing." Communications of the ACM 45.11 (2002): 56-61.

[27] Chiu, K., M. Govindaraju, and R. Bramley, Investigating the Limits of SOAP Performance for Scientific Computing, in Proceedings of the 11 th IEEE International Symposium on High Performance Distributed Computing. 2002, IEEE Computer Society.

[28] Van Engelen, Robert A., and Kyle A. Gallivan. "The gSOAP toolkit for web services and peer-to-peer computing networks." Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on. IEEE, 2002.

[29] Bauer, Pavol, et al. "Efficient inter-process synchronization for parallel discrete event simulation on multicores." Proceedings of the 3rd ACM Conference on SIGSIM-Principles of Advanced Discrete Simulation. ACM, 2015.