An Approach of Advanced Optimistic Protocol Based on Time Dams

Zhongjie ZHANG, Jian HUANG, Zhijia WANG
Institute of Automation,
College of Mechatronics Engineering and Automation, National University of Defense Technology
Changsha, China, 410073
E-mail: 597289799@qq.com

Abstract—**Time Synchronization is a key technique which synchronizes the LPs (Logic processes) of PDES (Parallel and Discrete Event Simulation). After having described shortcomings of the existing algorithms for Time Synchronization, this thesis mainly proposes a new approach based on time dams. It introduces the concept of Time Dams Algorithm and presents its realization. Moreover, the performance of TD (Time Dams) Algorithm is compared with the one of TW (Time Warp) Algorithm through the use of PHOLD model in experiments. Finally, the experiment results state that TD performs better than TW under most conditions.**

Keywords-parallel and discrete event simulation; time sychronization; time dams algorithm

I. INTRODUCTION

PDES (Parallel and Discrete Event Simulation) becomes more and more popular due to its high performance of large scale discrete simulation. It divides the simulation system into many parts, namely LPs (Logic processes) which perform at different cores and use Time Synchronization to keep time order accurate. There are two classical approaches [1], conservative protocol keeping the time order seriously correct and optimistic protocol making LPs run as fast as possible and ensuring the time order by Rollback and anti-messages. Typically, TW (Time Warp) [3] algorithm is an absolutely optimistic protocol. However, it can not do well sometimes because of too much Rollback, which means a large amount of memory and computations are wasted. Even worse, if the time cost of Rollback is more than the one of which event is dealt with, error will never be corrected.

There are two ways to enhance TW algorithm [1]. One is to limit the optimism by setting barriers; the other is to artificially roll LPs back. Because artificial Rollback will waste some accurate computations, many enhancements on TW such as MTW (Moving Time Window) [4], BTB (Breathing Time Buckets) [1], and Wolf [5] are based on blocking. MTW sets a time window which can not be passed by LPs but it needs too many computations on GVT (Global Virtual Time) for the movement of window. BTB has a parameter called Event-Horizon like the lookahead of conservative Synchronization yet computed in an optimistic way, but it may be very slow for few events in one Bucket. Wolf will stop every LP the moment that error occurs, which can halt the spread of error but also reduce the simulation efficiency because of too many barriers.

This thesis will introduce a new algorithm, namely TD (Time Dams) algorithm building some dams on simulation time. Those dams divide every LP's LVT (Local Virtual Time) into some parts, provide barriers for GVT computation and limit any Rollback between two of them. Moreover, TD keeps the optimism of LP to some extent and does not cost many added computation.

II. TIME DAM ALGORITHM

Just as dams built on a river, we can also build some dams on LVT to divide them into many parts and when the system runs, it will be executed part by part. Those dams are also like some walls for Rollbacks to limit any error between two dams.

A. *The Idea of Algorithm*

The main intention of TD algorithm is to put one Rollback within the limit of two dams. Since every LP runs independently, Rollback occurs always companied with computations. To limit Rollback well means to reasonably limit computation, so that LPs will not lose too much optimism. Therefore, TD algorithm is designed as follows:

- Every LP's LVT is divided into many parts, whose length is constant.
- Every event belongs to one of those parts according to its time stamp.
- The LP can not do any work of another part until the current one is end.
- The end of one part means that all the events in it have been done and there will be no event or message sent to this part.
- Each LP can send new events or messages to any parts of any LPs.
- The work in a singal part is the same as the one in TW algorithm.
- The beginning of next part is the minimum time stamp of the events after current part, which means although the interval between two dams is a constant, the width of one dam is dynamical.

To keep the optimism of simulation to some extent, TD algorithm also keeps some risks of error computations, but in a macroscopic way, the simulation will perform chronologically in this way.

B. *Definition of Some Variables*

- LP (n): The nth LP in simulation.
- Part (n): The nth part of LVT.
- SN: The number of messages sent since the beginning of simulation.
- RN: The number of messages received since the beginning of simulation.
- DI: The interval between two Time Dams, also the length of one part which is a constant.
- MTS (i): The ith LP's minimum time stamp of the events in next part, which will be infinite if the event queue is empty.
- MinMTS: The minimum of MTS (1), MTS (2), MTS (3)… and MTS (n).

C. *A Simple Example*

Figure 1 shows a simple example of TD algorithm. There are 3 initial events in the system and DI is t. Simualtion runs as follows:

- At the beginning of simulation, e1 and e2 will be dealt with at once, and LP (1) will produce and send e4 to LP (2) after e1 is done. Then it will handle e3.
- After e2 has been done, LP (2) will produce and send e5 to LP (1). Because the time stamp of e5 is smaller than the one of e3, LP (1) will be rolled back. Despite being the next event of LP (2), e4 will not be dealt with because the current part is Part (1). After e5 has been done, LP (1) will produce and send e6 to LP (2). Although the time stamp of e6 is smaller than the one of e4, LP (2) will not be rolled back because e4 has not been handled.
- Simulation will not start Part (2) until Part (1) is totally finished. After Part (1) ends, both of them, GVT and the beginning time stamp of Part (2), will equal t', and then system will run Part (2).

Obviously, TD algorithm limits the span of Rollbacks. and does better than MTW algorithm and BTB algorithm because of the maximal possible GVT computation and the maximal possible keeping of optimism.

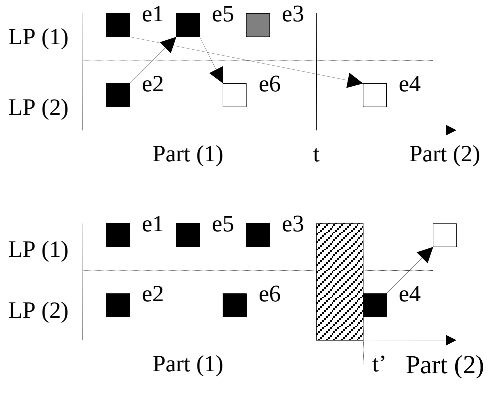

Figure 1. An example of TD algorithm

III. The realization of TD Algorithm

The main idea of TD algorithm is to build some dams on LVT. In TD algorithm, every LP should check the time stamp of every event in event queue. If the time stamp of an event is smaller than the one of next dam, LP will deal with it, or else, LP will have to break off and wait for the beginning of next part

A. *Classified LP Structure*

To realize the TD Algorithm, LPs will be classified into two types, Simulation LP and Judge LP.

Simulation LP is the LP which computes the simulation model and its job is to send, to receive and to deal with messages. Every Simulation LP has two cycles, an Event Cycle which deals with the events picked out of event queue and sends messages to other LPs and a Message Cycle which receives messages from other Simulation LPs and the Judge LP, and those two cycles will be realized by two threads. In addition, every Simulation LP has a memory shared with the Judge LP to record the number of messages sent to others and also the number of messages received from others. Those memories are very useful to the Judge LP's work.

Judge LP is a special LP which has a Message Cycle to communicate with Simulation LPs and its job is to manage the parts of LVT and to decide their ends and starts. The basic structure of TD system is shown in Figure 2.

B. *LP Management protocol*

To ensure that no error spreads from one part to another, a precise protocol between Simulation LPs and the Judge LP should be established. It is the protocol that manages the parts of LVT by controlling Simulation LPs to stop or to begin. When Part (n) ends, all the events of it have been done and all the messages of it have been received, so SN should be equal to RN.

In TD algorithm, if LP (i) running at Part (n) finds that all the events of Part (n) have already been done, it will stop working and send a report including MTS (i) to the Judge LP. After receiving reports from every Simulation LPs, the Judge LP will check their shared memories and send responses to them. There are two kinds of responses of the Judge LP, R1 and R2. If SN is not equal to RN, the Judge LP will send R1 to every LP; otherwise, it will send R2. MinMTS is calculated and included in R2.

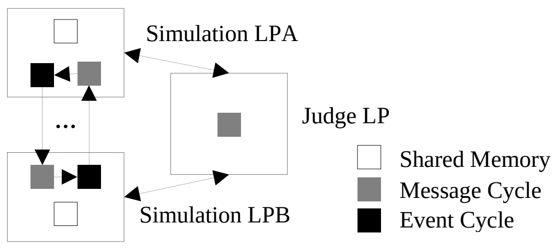

Figure 2. The basic structure of TD system

If LP (i) receives R1, it will just be restarted, and if it receives R2, it will be restarted and also deal with the next part of LVT.

The protocol is designed as two Petri Networks and shown in the Figure 3.The meaning of the Places and Transitions in Figure 3 is as follows:

1) *Simulation LP*

- a: Simulation LP is computing the simulation model.
- b: Simulation LP is waiting for the response from the Judge LP.
- c: Simulation LP is updating GVT and setting the next dam on LVT.
- 1: In the event queue, there is no event belonging to the current part and the Simulation LP has sent the report to the Judge LP.
- 2: Simulation LP has received R2 from the Judge LP.
- 3: Simulation LP has received R1 from the Judge LP.
- 4: Simulation LP has updated GVT and set the next dam on LVT.

2) *Judge LP*

- a: The judge LP is collecting reports from Simulation LPs.
- b: The Judge LP is checking the shared memory of every Simulation LP.
- c: The Judge LP is making and sending R2 to every Simulation LP.
- d: The Judge LP is making and sending R1 to every Simualtion LP.
- 1: All the Simulation LPs have sent reports to the Judge LP.
- 2: The Judge LP finds that SN is equal to RN.
- 3: The Judge LP finds that SN is not equal to RN
- 4: All the R2s have been sent.
- 5: All the R1s have been sent.

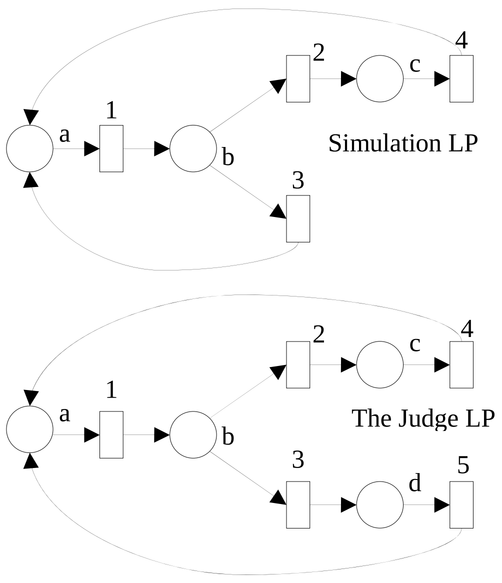

Figure 3. The Pretri Networks of two kinds of LPs

C. The GVT Computation

GVT is the time stamp across which any Rollback should not span [1]. MTS (i) is the minimum time stamp of LP (i) in the next part, and MinMTS is the minimum time stamp of MTS (1) to MTS (n), so MinMTS is the minimum time stamp of events in the next part. Before every LP receives R2, all the previous work and messages should have been finished, so no Rollback will span across MinMTS and GVT can be equal to MinMTS.

D. The Flow Path of Algorithm

1) *The Flow Path of Event Cycle in Simulation LP*

```
GVT=0     STOP=FALSE
WHILE ( !SIM_End )
    IF ( Event_Q.Size()>0 && !STOP )
        IF ( Event_Q.Begin.Time<GVT+DI )
            PROCESS ( Event_Q.Begin )
            Event.ERASE ( Event_Q.Begin )
            IF ( Send_Message( ) )
                SN++
            END
        ELSE
            MTS = Event_Q.Begin.Time
            SEND_REPORT( MTS )
            STOP = TRUE
        END
    ELSE IF ( !STOP )
        MTS = INF
        SEND_REPORT( MTS )
        STOP = TRUE
    END
END
```

2) *The Flow Path of Message Cycle in Simulation LP*

```
WHILE ( !SIM_End )
    IF ( Mes_Arrive && Mes.Type == R1)
        STOP = FALSE
    ELSE IF (Mes_Arrive &&Mes.Type == R2 )
        GVT = MinMTS
        STOP = FALSE
    ELSE ( Mes_Arrive )
        RN++
        PROCESS ( Mes )
    END
END
```

3) *The Flow Path of Message Cycle in the Judge LP*

```
WHILE ( !SIM_End )
    IF ( ALL_Reports_Come )
        IF ( SN != RN )
            SEND_RES( R1 )
        ELSE
            MinMTS = Min(MTS(1) …MTS(n))
            SEND_RES( R2 )
        END
    END
END
```

IV. Test and Analysis

This thesis applies PHOLD model to experiments testing and analyzing the performance of TD algorithm in different situations. Those experiments are realized by MPI and run at Acer-ASM3450 which has a FX-8100 8-core CPU (2.8 GHz) and 8G memory.

A. The PHOLD Model

PHOLD is a classical test model which supposes that the simulation is composed of N nodes and every node has one Simulation LP with one initial event. When PHOLD is running, Simulation LP will send a new event whose time stamp is LVT+lookahead+△T to any other Simulation LPs after one event has been done, and here the lookahead is 0.01 seconds and △T is a uniform distribution of random from 0 to 1 second.

B. Test

In the first experiment, we used both TW and TD whose DI is 2 seconds to run the PHOLD model for 10 seconds on 8 nodes. The map of LP (1) from real time to LVT was recorded and shown in Figure 4.

In the second experiment, we used both TW and TD whose DI is 0.5, 1, 2, 4 and 8 seconds to run the PHOLD model for 10 seconds respectively on 2, 4 and 8 nodes. We recorded the simulation time and presented it in Figure 5 and TW can be looked on as a special TD whose DI is infinite.

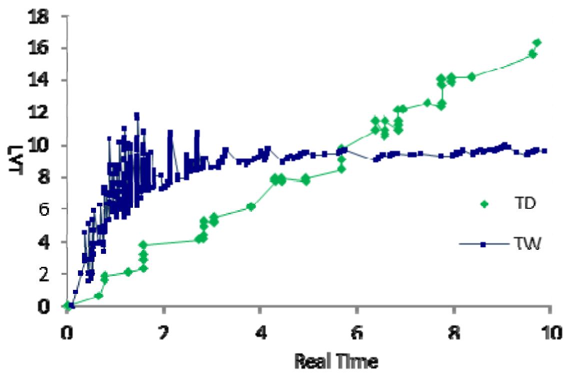

Figure 4. The map of LP (1) on 8 nodes

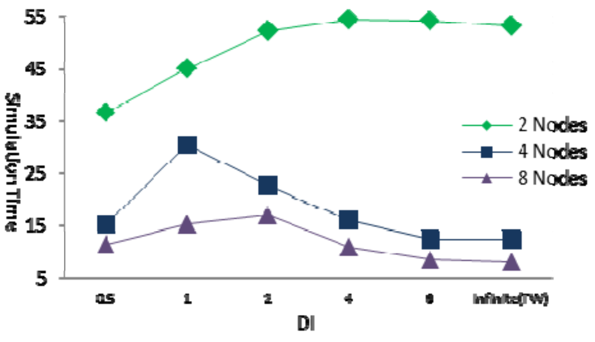

Figure 5. The map between Simulation Time and DI

C. Analysis

- Figure 4 illustrate that TW advanced much more slowly than TD because of too many long span Rollbacks. Despite TD algorithm handles fewer events than TW algorithm, it successfully limits the span and the number of Rollbacks and achieves a high performance on simulation time.
- Figure 5 illustrates that when the number of nodes is high, TD performs better than TW. Because the small number of nodes does not make too much Rollback, the computations for dams not only do not play an obvious role in limiting Rollback, but also reduce the speed of simulation.
- Figure 5 also illustrate that DI affects the efficiency of TD very much. The short DI will cost more computations for communication between the Judge LP and Simulation LPs, while the long DI means more Rollbacks. Therefore, DI should be decided with prudence.

V. Conclusion

This thesis has discussed a new approach of Optimistic Protocol — Time Dams Algorithm. We firstly introduce the concept and some rules of TD with which LPs have to comply, and then present the realization of the algorithm which includes the basic structure of TD system, the protocol between the Judge LP and Simulation LPs, the computation of GVT, and the flow path of them. Finally, we conduct two experiments and analyze their results proving that under most conditions, TD performs better than TW and does not lose too much optimism.

Acknowledgment

We are grateful to Professor HUANG Ke-di, Professor HAO Jian-guo and Doctor Gong Jian-xin for their insightful suggestions.

References

[1] R. M. Fujimoto, Parallel and Distributed Simulation Systems, Beijing: Publishing House of Electronics Industry, 2010.

[2] Huang Ke-di, Qiu Xiao-gang et al., "Modeling and Simulation Technology," Changsha: National University of Defence Technology Press, 2011.

[3] D. R. Jefferson B. Beckman et al, "The Time Warp operating systems", 11th Symposium on Operating Systems Principles, ACM: press, pp. 77-93, 1987.

[4] Sokol L. and B.K. Stucky, "MTW: Experimental results for a constrained optimisic scheduling paradigm," Proc. SCS Multiconference on Distributed Simulation, pp. 169-173, 1990.

[5] Madisetti V., J. Walrand, et al. "WOLF: A rollback algorithm for optimistic distributed simulation systems," Proc. The 1988 Winter Simulation Conference, IEEE press, pp. 296-305, 1988.

[6] Dickens P.M., and J. Reynolds, P. F., "SRADS with local rollback," Proc. SCS Multiconference on Distributed Simulation, pp. 161-164, 1990.

[7] Su Nian-le, Wu Xue-yang, Li Qun, Wang Wei-ping, Zhu Yi-fan, "Optimistic Parallel Discrete Event Simulation based on multi-core platform," Journal of System Simulation, Vol. 22, No. 4, pp. 858-863, Apri. 2010.