Performance Optimization of Throttled Time-Warp Simulation

Seng Chuan TAY and Yong Meng TEO
Department of Computer Science
National University of Singapore
3 Science Drive 2
Singapore 117543
email: teoym@comp.nus.edu.sg

Abstract

While constraining the speculation in Time Warp (TW) tends to decrease the number of false event executions, it also introduces an opportunity cost when the processors are not fully utilized. To obtain good runtime performance such a trade-off must be optimized. This paper studies the implications of regulating speculation in TW, and develops an analytic framework for optimizing the elapsed time of TW simulators. By aggregating the effect of three time components, namely computation time, communication time and processor idle time, our analytic framework optimizes the degree of speculation for better performance. Our experiments on a Fujitsu AP3000 distributed-memory parallel computer simulating several applications show that the predicted performance metrics deviate from the measured values by less than 8%. Both the analytical and experimental results have ascertained that speculation without rollbacks may not produce the best elapsed time. Instead, a controlled degree of causality error is preferred in most practical cases.

Keywords: *performance modeling and optimization, probabilistic model, opportunity cost*

1 Introduction

Time Warp (TW) [2] adopts an optimistic approach to execute simulation events. As the TW mechanism does not enforce strict time-ordered event execution, it can lead to excessive rollback overheads [3]. A throttling scheme aims to better match the parallelism exposed by the TW mechanism with the parallelism exploited at run time through dynamic control of TW optimism. While promoting speculation in TW increases the rollback risk, constraining speculation reduces the opportunity for parallel events processing. This indicates the existence of an optimal point for the simulator to attain the best performance.

This paper is based on an analytic performance framework presented in [8]. The proposed model considers processor idle time, communication overhead, costs of state-saving, global virtual time (GVT) computation and cascading rollback. The rest of this paper is organized as follows. Section 2 summaries the throttled TW algorithm developed earlier. Section 3 addresses our time analysis of throttled TW, and identifies an optimized speculation for TW mechanism. We model the elapsed time using three time components, namely, *computation time*, *communication time* and *processor idle time*, where the idle time is caused by insufficient executable events and the delay in GVT computation. By aggregating the effect of these components, we determine the optimal speculation. Section 4 analyzes the sensitivity of the performance model with respect to the degree of speculation, and validates the proposed analytical model against the experimental performance result. Lastly, section 5 contains our concluding remarks.

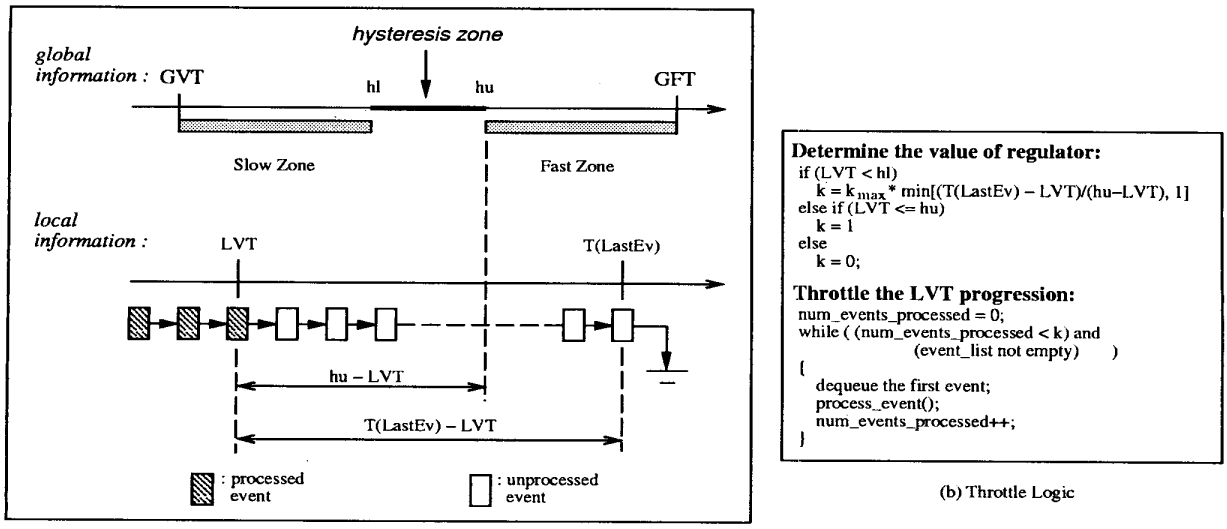
2 Overview of Throttling Mechanism

Our adaptive event-based parallelism throttle mechanism consists of (see figure 1a):

- *global process window* - to characterize the simulation progress of each logical process (LP)
- *hysteresis band* - to smooth the fluctuation of local virtual time (LVT) advancement
- *event regulator* - to regulate event execution, i.e., to accelerate the slow LPs, and suspend the LVT advancement in fast LPs.

The details of the throttling mechanism are described in [7]. Briefly, we let LVT_i denote the local virtual time of LP_i, and LPT_i denote the local progress time of LP_i, i.e., the minimum of the LVT_i and the timestamp of each transitional message sent by LP_i.

0-7695-1092-2/01 $10.00 © 2001 IEEE



(a) Global Progress Window and Parameterization of Event Regulator

(b) Throttle Logic

Figure 1: TW Throttle Design

Let $GVT = \min(LPT_i)\ \forall i$ be the *global virtual time*, and $GFT = \max(LVT_i)\ \forall i$ be the *global furthest time*. We denote a *global progress window* (GPW) by $[GVT, GFT]$. Thus, GPW provides a universal time scale for each LP to calibrate its local simulation advancement. Using the existing GVT acquisition protocol, GPW can be determined without much overhead. For a better classification of LPs, the GPW is divided into three zones: *slow* $[GVT, hl)$, *hysteresis* $[hl, hu]$, *fast* $(hu, GFT]$. The hysteresis zone is an interval in the GPW to smooth out the LVT fluctuation. Let r be a spread ratio of the length of hysteresis zone to the GPW, i.e., $r = \frac{hu-hl}{GFT-GVT}$.

Figure 1b shows the throttling algorithm. A new GPW is computed when the LVTs of a majority (> 50%) of LPs in the slow and hysteresis zones have passed hu, so that the fast LPs are able to resume their event execution.

3 Analytic Performance Model

The modeling assumptions are:

1. simulator contains p homogeneous LPs and p homogeneous processing elements (PEs);
2. the placement of LPs on PEs is one-to-one;
3. each arrival event has a corresponding departure event in the same LP and the execution of each departure event will in turn schedule an arrival event in one of its succeeding LPs;
4. state vector is saved after an event is executed;
5. memory space is sufficient to complete the simulation.

3.1 Characterization of GVT Interval

Let u be the number of events processed when one message is sent across an LP, and d the diameter (or the longest path) of the LP interconnection (refer to figure 2). The number of LPs on the longest path is $d-1$. In our analytical model $u = 2$, i.e., a message will generate an arrival event and departure event on each LP visit. Suppose LP_a is the first LP, and LP_b the last LP on the diameter. As the diameter increases, LP_b event execution will be delayed. Consequently, the deviation of state indices in LP_a and LP_b is increased. Assume c events are executed during each inter-LP transmission, and u events during each LP visit, the total number of events executed for the duration in traversing the longest path is $f(d) = d \times c + (d-1) \times u$.

Let $\widehat{GVT}_{unthrottled}$ and $\widehat{GVT}_{throttled}$ be the GVT interval used in the unthrottled TW and throttled TW respectively. For the unthrottled TW, we assume that GVT computation is performed after a constant number of events is executed and not rolled back. More sophisticated algorithms on deciding when to calculate GVT can be modeled. It follows that

$$\widehat{GVT}_{unthrottled} = \mathcal{W} \qquad (1)$$

where $\mathcal{W}$ is a pre-assigned value.

In throttled TW, LPs falling in the fast zone will be suspended and therefore GVT computation will not be activated. For other LPs, a GVT computation is

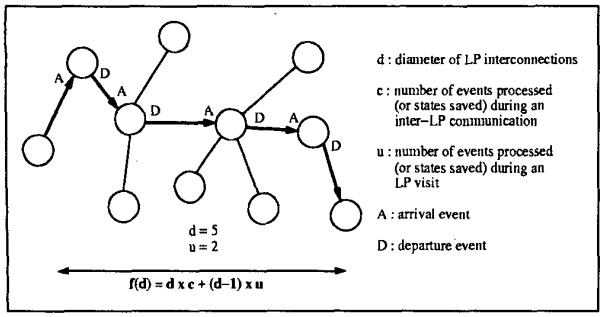


Figure 2: Diameter of LP Interconnections

activated after a majority (> 50%) of LVTs exceeds the hysteresis zone. Consider the slowest logical process LP_0 with local progress time equal to GVT. As the expected number of events executed in the whole global progress window is $f(d)$, the number of events executed by LP_0 before it sends out the request for GVT signal is $f(d) \times (\frac{1}{2} + \frac{r}{2})$. In our analysis, we approximate the throttled GVT interval by

$$\begin{aligned} \widehat{GVT}_{throttled} &\approx f(d) \times (\frac{1}{2} + \frac{r}{2}) \times 50\% \\ &= f(d) \times (\frac{1}{4} + \frac{r}{4}) \end{aligned} \quad (2)$$

In the following derivations, $\widehat{GVT}$ represents $\widehat{GVT}_{unthrottled}$ or $\widehat{GVT}_{throttled}$ interchangeably depending on the type of TW scheme used.

3.2 Characterization of Elapsed Time

We include processor idle time in our formulation so as to predict performance under varying workload condition, thereby to determine the optimal performance. Let E_{Total} be the total number of (true and false) events executed, E_{Arr} the total number of (true and false) arrival events, and E_{Dep} the total number of (true and false) departure events.

3.2.1 Computation Time (T_{Cp})

Let $\overline{D^+}$ be the expected number of undone events caused by a straggler, and $\overline{D_n^-}$ be the expected number of undone events caused by the n-th wave of anti-message, $1 \leq n \leq l$ where l is the last wave of anti-message[1], λ and μ be the arrival rate and service rate respectively, and N_p be the number of events executed by an LP and p be the number of LPs. As the execution of each true event has the expected rollback distances[2] $\overline{D^+}$, and $\overline{D_1^-}$, $\overline{D_2^-}$, ..., $\overline{D_l^-}$, we have

[1] $\overline{D^+}$ and $\overline{D_n^-}$ have been derived in [8].

[2] Rollback distance refers to the number of undone events.

$$\begin{aligned} E_{Total} &= N_p \times (1 + \overline{D^+} + \sum_{n=1}^{l} \overline{D_n^-}) \\ E_{Arr} &= \begin{cases} \frac{E_{Total}}{2} & \text{if } \lambda < \mu \\ E_{Total} \times \frac{\lambda}{\lambda+\mu} & \text{otherwise} \end{cases} \\ E_{Dep} &= E_{Total} - E_{Arr}. \end{aligned}$$

The cost of processing each event includes the event execution time and state-saving time. We have

$$T_{Cp} = N_p \times (1 + \overline{D^+} + \sum_{n=1}^{l} \overline{D_n^-}) \times (T_e + T_s) \quad (3)$$

3.2.2 Communication Time (T_{Cm})

Let T_b be the time required to construct a message in buffer for each transmission, and T_t be the message transmission time (refer to figure 3). Communication time (T_{Cm}) can be attributed to the transmission and reception of (i) event messages (T_{Cm}^E), and (ii) GVT protocols (T_{Cm}^G).

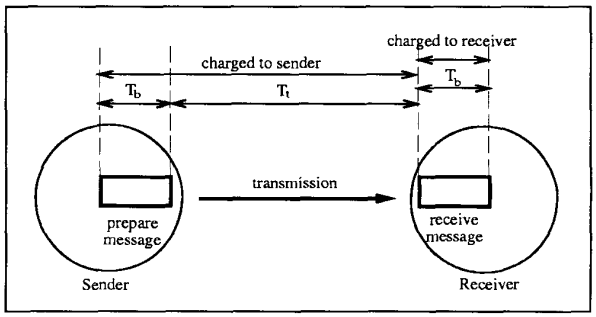


Figure 3: Communication Time Accounting

As E_{Arr} events are received and E_{Dep} events are transmitted, we have

$$T_{Cm}^E = E_{Arr} \times T_b + E_{Dep} \times (T_b + T_t) \quad (4)$$

Let G_{Total} denote the total number of GVT computations performed during the simulation run. As the total number of true and false events executed is E_{Total}, we use the approximation

$$\begin{aligned} G_{Total} &\approx \frac{E_{Total}}{\widehat{GVT}} \\ &= \frac{N_p \times (1 + \overline{D^+} + \sum_{n=1}^{l} \overline{D_n^-})}{\widehat{GVT}} \end{aligned} \quad (5)$$

where $\widehat{GVT}$ is denoted by equations (1) or (2) for the unthrottled TW and throttled TW simulation respectively.

Let LPT denote the local progress time of an LP. We define $LPT = \min(LVT, TS(\mathcal{M}))$, where $TS(\mathcal{M})$

is the smallest timestamp of all the transiting messages sent by the LP. In this performance model, we assume that a coordinator is used in GVT computation, and the GVT protocol consists of four phases:

1. An LP after advancing its LVT beyond hu, will send a request for GVT signal to the coordinator.
2. If the request signals from a majority (> 50%) of LPs in the slow and hysteresis zones have been received, the GVT coordinator broadcasts a request for LPT signal to all LPs. Otherwise the coordinator waits until more than 50% of the request signals have been received.
3. Whenever the request for LPT signal is received in an LP, the LP will report its LPT to the GVT coordinator.
4. After all the LPTs have been received, the GVT coordinator computes the minimum, and broadcasts the new GVT to all LPs.

As for each LP, the GVT computation procedure incurs two transmissions and two receptions, the overhead (refer to figure 3) incurred in communication is

$$2 \times (T_b + T_t) + 2 \times T_b = 4T_b + 2T_t. \quad (6)$$

It follows that the communication overhead due to GVT computation is

$$\begin{aligned} T_{Cm}^G &= G_{Total} \times (4T_b + 2T_t) \\ &= \frac{N_p(1 + \overline{D^+} + \sum_{n=1}^{l} \overline{D_n^-})}{\widehat{GVT}} \times (4T_b + 2T_t) \quad (7) \end{aligned}$$

By adding equations (4) and (7), we have

$$\begin{aligned} T_{Cm} &= E_{Arr} \times T_b + E_{Dep} \times (T_b + T_t) + \\ &\quad \frac{N_p(1 + \overline{D^+} + \sum_{n=1}^{l} \overline{D_n^-})}{\widehat{GVT}} \times (4T_b + 2T_t) \quad (8) \end{aligned}$$

3.2.3 Processor Idle Time (T_{Id})

We abstract the processor idle time by two components: (i) the delay due to GVT computation (T_{Id}^G), and (ii) the time wasted due to insufficient executable events (T_{Id}^E).

GVT Computation Delay

Before a GVT computation is activated, the coordinator has to wait for the request for GVT signals sent by a majority of LPs positioned on the slow and hysteresis zones. This delay corresponds to the processing time used in executing 50% of the events in the interval [GVT, hu), and saving their states. Using equation (2) and considering the expected rollback distances for each processed event, the delay caused by one GVT computation is

$$f(d) \times (\tfrac{1}{4} + \tfrac{r}{4}) \times (1 + \overline{D^+} + \textstyle\sum_{n=1}^{l} \overline{D_n^-}) \times (T_e + T_s) \quad (9)$$

Using equation (5), the total number of GVT computation performed is

$$G_{Total} \approx \frac{N_p \times (1 + \overline{D^+} + \sum_{n=1}^{l} \overline{D_n^-})}{f(d) \times (\frac{1}{4} + \frac{r}{4})} \quad (10)$$

By multiplying the number of GVT computations (equation (10)) and the delay time caused by each GVT computation (equation (9)), we have

$$T_{Id}^G = N_p \times (1 + \overline{D^+} + \sum_{n=1}^{l} \overline{D_n^-})^2 \times (T_e + T_s) \quad (11)$$

Insufficient Executable Events

We model the duration wasted due to insufficient executable events based on the opportunity cost incurred by decreased optimism. As the number of LPs falling in the slow and fast zones of the global progress window is gradually reduced by the parallelism throttle, the number of events available for parallel processing will ultimately converge to $f(d) \times r$. Let π_r^d denote the *degree of throttled parallelism* (or the number of events available for parallel execution) at steady state for an LP interconnection configuration of diameter d and a spread ratio r. We have

$$\pi_r^d = f(d) \times r$$

If $\pi_r^d < p$, some processors will be under-utilized due to insufficient number of executable events[3]. Let $\mathcal{S}$ denote the *total* time required to *complete* the execution of all events and saving their states under the insufficient workload condition, and $\mathcal{A}$ be the *elapsed time amplifier* due to insufficient number of executable events[4]. We have

$$\mathcal{S} = T_{Cp} \times \mathcal{A} \quad (12)$$

$$\text{where } \mathcal{A} = \begin{cases} \frac{p}{f(d) \times r} & \text{if } f(d) \times r < p \\ 1 & \text{otherwise} \end{cases}$$

As $\mathcal{S}$ can also be modeled by the sum of (i) busy time and (ii) idle time[5], we let

$$\mathcal{S} = T_{Cp} + T_{Id}^E \quad (13)$$

By equating equations (12) and (13), we have

[3] π_r^d can be regarded as TW mechanism parallelism, and p (number of processors) as machine parallelism.

[4] Suppose the computation time needed on a uniprocessor is 60 seconds. If the number of PEs used is 6, we have $T_{Cp} = \frac{60}{6} = 10 seconds$. Consider a scenario where *four* of the PEs are idle due to insufficient number of executable events. The total time required to complete the same amount of work becomes $\frac{60}{6-4} = \frac{10 \times 6}{2} = 10 \times 3 = 30$ seconds, i.e., $\mathcal{A} = 3$.

[5] This idle time refers to the time wasted due to insufficient number of executable events.

$$T_{Cp} \times \mathcal{A} = T_{Cp} + T_{Id}$$

It follows that

$$T_{Id}^{E} = T_{Cp} \times (\mathcal{A} - 1)$$
$$= N_p(1 + \overline{D^+} + \sum_{n=1}^{l} \overline{D_n^-})(\mathcal{A} - 1)(T_e + T_s) \quad (14)$$

Adding equations (11) and (14), we have

$$T_{Id} = N_p(1 + \overline{D^+} + \sum_{n=1}^{l} \overline{D_n^-})^2 \times (T_e + T_s) +$$
$$N_p(1 + \overline{D^+} + \sum_{n=1}^{l} \overline{D_n^-})(\mathcal{A} - 1)(T_e + T_s) \quad (15)$$

Finally, the elapsed time of TW simulator is computed as follows:

$$T_{TW} = T_{Cp} + T_{Cm} + T_{Id}$$
$$= N_p(1 + \overline{D^+} + \sum_{n=1}^{l} \overline{D_n^-}) \times (T_e + T_s) +$$
$$E_{Arr} \times T_b + E_{Dep} \times (T_b + T_t) +$$
$$\frac{N_p(1 + \overline{D^+} + \sum_{n=1}^{l} \overline{D_n^-})}{\widehat{GVT}} \times (4T_b + 2T_t) +$$
$$N_p(1 + \overline{D^+} + \sum_{n=1}^{l} \overline{D_n^-})^2 \times (T_e + T_s) +$$
$$N_p(1 + \overline{D^+} + \sum_{n=1}^{l} \overline{D_n^-})(\mathcal{A} - 1)(T_e + T_s) \quad (16)$$

3.3 Optimized Degree of Parallelism

The *knee performance* can be determined by the trade-off between the number of processors and their lost opportunity cost. Our heuristic begins by first minimizing the lost opportunity cost in equation (16), i.e., we ensure that the number of executable events is at least equal to the number of processors, through increasing the optimism. By equating $T_{Id}^{E} = 0$ in equation (14), we have $\mathcal{A} = 1$, i.e.,

$$r \geq \frac{p}{f(d)} \quad (17)$$

Next, we reduce the rollback cost by constraining the optimism (reducing the spread ratio r). It follows from equation (17) that the *time-optimized* spread ratio is

$$r_{optimized}^{time} = \frac{p}{f(d)} \quad (18)$$

Equation (18) shows that the best performance of TW does not happen when the LVT advancement is *strictly* in-pace ($r = 0$) or when there is no rollback. This characteristic conforms to the empirical results reported in [6].

4 Model Validation and Performance Analysis

We use heterogeneous examples to highlight the improved performance due to the parallelism throttle, and use homogeneous examples to validate the performance model. Parallel simulation experiments were conducted on a Fujitsu AP3000 distributed-memory parallel computer. Other TW schemes such as using a static window size is analyzed in [9]. Four parameter values used in the analytical model are obtained by taking measurements (see table 1) on the implementation platform. The values for computation costs (T_e and T_s) in table 1 are obtained by timing the execution time of the respective code segments in the simulation program over 1000 iterations and taking their average. The buffer access time (T_b) is obtained by clocking the elapsed time of PVM code segment for packing and unpacking the message and taking their average over 1000 iterations. As additional protocols are required by the PVM to allocate memory space for send and receive buffers, T_b has a higher value as compared to the other measurements. Transmission time (T_t) is obtained using a Ping-Pong program (sending null messages between 2 LPs) over 1000 iterations. In cases where the parallel simulator has not been implemented and the computer is not available, we can estimate the parameters from the existing sequential or similar parallel simulators and machine specifications. Performance figures presented below have been averaged over five replicated simulation runs.

parameter	mean time (μsec)
T_e	1200
T_s	990
T_b	2750
T_t	1290

Table 1: Performance Parameter Measurements

4.1 Application Examples

Exponential distribution is assumed for arrival time and service time. Homogeneous examples consists of (i) MIN and (ii) Torus. As for the 8×8 Omega MIN the diameter of the LP interconnection is equal to 3, the cascading rollback stops at the second wave. The mean interarrival time used in the packet generator and the mean service time used in each switching element are $\frac{1}{100}$ second. The 4×4 torus consists of 16 nodes each with the same mean service time of $\frac{1}{60}$ second. For the $n \times n$ torus network, the diameter is $d = n$ due to the feedback connection. The routing on the homogeneous torus network is uniformly distributed on the four directions.

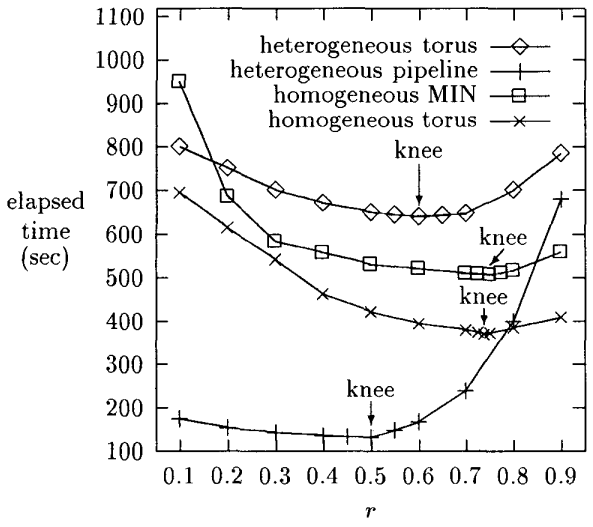


Figure 4: Determining the Parameter r by Empirical Approach

Heterogeneous examples are also used to compare the measured elapsed time of the two TW schemes but the predicted timings are left as future work. The first heterogeneous example is the linear pipeline consisting of 6 service stations. The mean interarrival time used is 20 seconds, and the service time of the ith server has a mean of $100 \times 5^{i-1}$ seconds. Since the LVT is advanced after an arrival or departure event is executed, LVT advancement in each LP is faster than its predecessor on the pipeline. As a result, this set of service times produces excessive rollbacks during the simulation run. The second heterogeneous example is the 4×4 torus network but with the mean service time at each node set to $\frac{1}{60} \times (4 \times i + j)$ second, where $i \geq 1$ is the row index and $j \geq 1$ is the column index. As the event routing is uniformly distributed on the *east* and *south* directions where the mean service times are increasing, the number of rollbacks increases.

We first estimate the optimal value of r for each application example by empirical approach. Later, we will compare the optimal values obtained by experiments with the predicted values for homogeneous systems. For each example we run the TW simulator with step increment in r, and use the divide-and-conquer technique to improve the accuracy of r on the prospective interval. Figure 4 indicates knee performance at $r \approx 0.75$ for homogeneous MIN, $r \approx 0.7375$ for homogeneous torus, $r \approx 0.5$ for heterogeneous linear pipeline, and $r \approx 0.6$ for heterogeneous torus. The performance of throttled TW is more sensitive to the value of r, which is the optimism allowed, for heterogeneous applications. The upper bound (k_{max}) of event execution acceleration (refer to figure 1b) is adapted empirically based on the number of unprocessed events in each LP so that the number of events executed in each throttle cycle does not exceed the number of events available for processing. Based on experimental results, we have $k_{max} = 11$ for the homogeneous 8×8 MIN, $k_{max} = 14$ for the homogeneous 4×4 torus network, $k_{max} = 8$ for the heterogeneous pipeline, and $k_{max} = 12$ for the heterogeneous 4×4 torus network.

4.2 Effectiveness of Parallelism Throttle on Heterogeneous Examples

Tables 2 and 3 show that unthrottled TW exhibits a larger number of rollbacks if the lookahead, which is the service time, along the network is of increasing order. However, this can be controlled when the parallelism throttle is used to level the LVT advancement. On the average, the rollback count of the experiments is reduced by 66.8% for the heterogeneous pipeline, and 34.8% for the heterogeneous torus network.

simulation duration (sec)	number of rollbacks	
	unthrottled TW	throttled TW
2000000	37068	12326
4000000	60261	22652
6000000	90053	32578
8000000	144937	42904
10000000	174341	55630
12000000	216712	65945

Table 2: Comparison of Measured Rollback Counts (Heterogeneous Linear Pipeline)

simulation duration (sec)	number of rollbacks	
	unthrottled TW	throttled TW
500000	144195	120162
1000000	313698	241302
1500000	508389	362421
2000000	728465	485641
2500000	964502	602814
3000000	1232452	724972

Table 3: Comparison of Measured Rollback Counts (Heterogeneous Torus Network)

Figures 5 and 6 show the reduction in measured elapsed time when the parallelism throttle is used in TW simulation. Similar to the trend observed in rollback counts, unthrottled TW produces an abrupt growth in elapsed time when the duration of simulation is increased, but the growth is almost linear when

the throttling scheme is used.

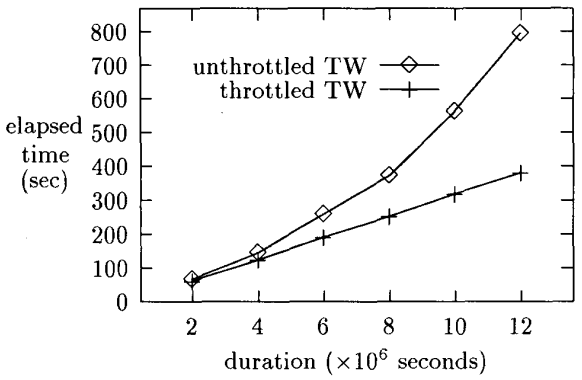


Figure 5: Measured Elapsed Time of Heterogeneous Linear Pipeline Simulation

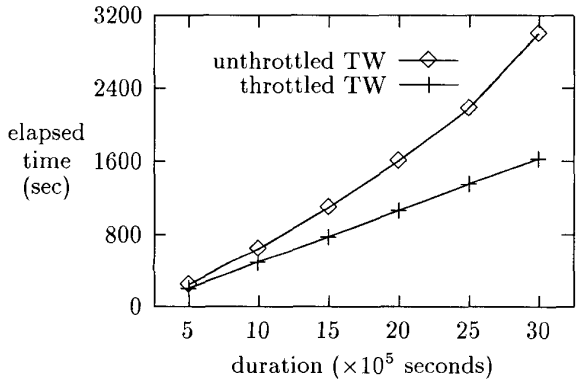


Figure 6: Measured Elapsed Time of Heterogeneous Torus Simulation

4.2.1 Sensitivity Analysis of Throttling Scheme on Homogeneous Examples

Using the 8 × 8 MIN as an example, we analyze how the elapsed time of TW changes with respect to the degree of optimism. Figure 7 shows that an increase in spread ratio causes an increase in the predicted total workload in event execution, communication and GVT computation ($T_{Cp} + T_{Cm} + T_{Id}^G$), but a decrease in the predicted processor idle time caused by insufficient executable events (T_{Id}^E). Based on our formulation in equation (18), the *knee* is predicted at $r_{optimized}^{time} = \frac{p}{f(d)} = \frac{12}{3\times4+(3-1)\times2} = 0.75$, which coincides with the knee determined by empirical approach (refer to figure 4). Figure 8 shows the elapsed time of throttled TW scheme using $r = 0.75$. On the average, the predicted elapsed time deviates 6% from the measured value for throttled TW.

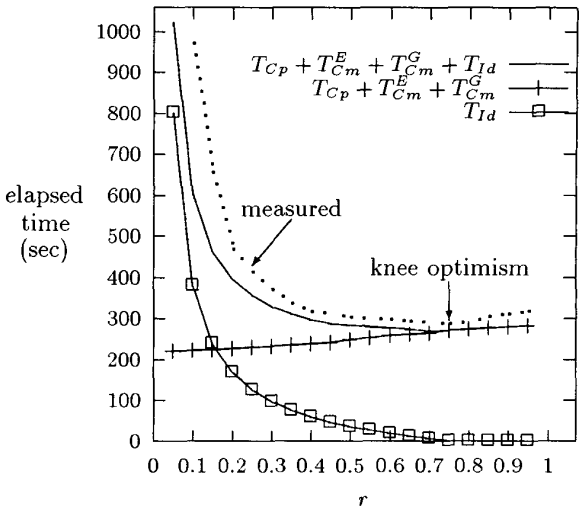


Figure 7: Elapsed Time for Different Spread Ratio used in Homogeneous MIN Simulation

For the homogeneous torus the *knee* is predicted at $r_{optimized}^{time} = \frac{p}{f(d)} = \frac{16}{4\times4+(4-1)\times2} \approx 0.7273$ using equation (18). This value shows that the knee prediction is close to the value (0.7375) determined by the empirical approach (refer to figure 4). As empirical observations indicate that most of the cascading rollbacks in the 4 × 4 torus simulation stops at the third wave, we approximate the total number of events executed in each LP by

$$E_{Total} \approx N_p \times (1 + \overline{D^+} + \sum_{n=1}^{3} \overline{D_n^-})$$

Figure 9 shows the elapsed time of throttled TW scheme using $r = 0.7273$ for homogeneous torus with 8% deviation from the measured values. We observe that the deviation of prediction from the measured elapsed time is larger in the torus example as compared to the MIN example. This larger deviation is caused by the feedback loops where some cascading rollbacks go beyond the third wave in torus network. A more accurate characterization of number of cascading rollback waves for feedback systems is left as further work.

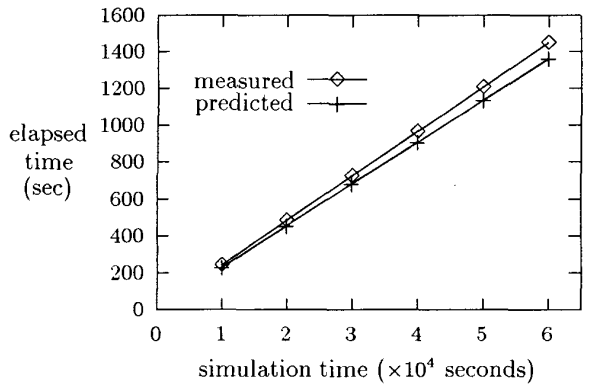


Figure 8: Elapsed Time for Homogeneous MIN Simulation (Throttled), where $r = 0.75$

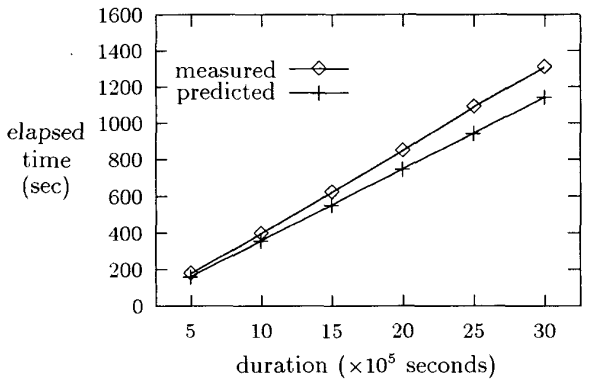


Figure 9: Elapsed Time for Homogeneous Torus Simulation (Throttled), where $r = 0.7273$

5 Conclusions

The performance of TW scheme is greatly affected by the amount of speculation allowed in the simulator. In the worst case, the unbounded optimism can adversely degrade performance as CPU cycles are wasted in executing events which are out of sequence, and in performing rollback recovering. Throttling schemes reported in the literature have suggested regulation of speculation and advancement of LVTs. In this paper, we proposed an analytical model for analyzing the performance of throttled TW. The proposed model considers a homogeneous system and calculates computation, communication and processor idle times. The model provides (i) a practical framework for predicting the performance of throttled TW simulators and (ii) a tool to determine the optimal degree of speculation in throttle-based TW simulations. As for heterogeneous systems, numerical methods can be used to approximate the performance measure.

Our analysis using the analytical framework show that (i) Throttling is an efficient method to reduce the elapsed time in certain TW-based parallel simulations. (ii) The best elapsed time is neither obtained by a completely in-pace LVT advancement (no causality error), nor by a completely uncontrolled TW. Instead, a controlled degree of causality error is desirable in achieving optimal performance. Hence, the importance of adaptive throttling becomes apparent.

References

[1] R. Fujimoto, *"Parallel and Distributed Simulation Systems"*, 1st Edition, NewYork : John Wiley and Sons, 2000.

[2] D. R. Jefferson, *"Virtual Time,"* ACM Transactions on Programming Languages and Systems, Vol. 7, No. 3, pp. 404-425, July 1985.

[3] D. Nicol and X. Liu, *"The Dark Side of Risk"*, Proc. of 11th Workshop on Parallel and Distributed Simulation (PADS'97), Lockenhaus, Austria, IEEE Computer Society Press, pp. 188-195, June 10-13, 1997.

[4] B. L. Noble and R. D. Chamberlain, *"Analytic Performance Model for Speculative, Synchronous, Discrete-Event Simulation"*, Proc. of 14th Workshop on Parallel and Distributed Simulation (PADS'2000), Bologna Italy, pp. 35-44, IEEE Computer Society Press, 2000.

[5] B. D. Plateau and S. K. Tripathi, *"Performance Analysis of Synchronization for Two Communicating Processes","* Performance Evaluation Journal, Vol. 8, pp. 305-320, 1988.

[6] Sudhir Srinivasan and Paul F. Reynolds, Jr., *"NPSI Adaptive Synchronization Algorithm for PDES,"* Proc. of the 1995 Winter Simulation Conference, pp. 658-665, 1995.

[7] S.C. Tay Y.M. Teo and S.T. Kong, *"Speculative Parallel Simulation with an Adaptive Throttle Scheme"*, Proc. of 11th Workshop on Parallel and Distributed Simulation (PADS'97), Lockenhaus, Austria, pp. 116-123, IEEE Computer Society Press, 1997.

[8] S.C. Tay Y.M. Teo and R. Ayani, *"Performance Analysis of Time Warp Simulation with Cascading Rollbacks"*, Proc. of 12th Workshop on Parallel and Distributed Simulation (PADS'98), pp.30-37, IEEE Computer Society Press, Canada, May 1998.

[9] S.C. Tay, *"Parallel Simulation Algorithms and Performance Analysis"*, Department of Computer Science, School of Computer, National University of Singapore, May 1998.