PERFORMANCE FACTORS IN PARALLEL DISCRETE EVENT SIMULATION

Jan Lemeire
Erik Dirkx
Free University of Brussels
Faculty of Applied Sciences, INFO
Pleinlaan 2
B-1050 Brussels, Belgium
E-mail: jlemeire@info.vub.ac.be

Published in Proc. of the Int. Multiconference on Simulation and Modeling (ESM 2001), Prague, June, 2001

KEYWORDS
Discrete event simulation, parallel methods, performance analysis.

ABSTRACT

Performance is a key issue in parallel simulation. This paper proposes a standard approach for detailed performance analysis of parallel discrete event simulation. We developed it for our conservative simulation algorithm, but it can serve for parallelisation in general as well. The parallel overhead terms are introduced and model-dependency of the parallel speedup is expressed by the performance factors. Experimental results show that this analysis can serve for speedup bottleneck detection. We propose this approach as a workplan for in-depth investigation of the performance (performance prediction, scalability analysis and algorithm comparison).

INTRODUCTION

This paper deals with *distributed* - in our lab a cluster of PC's - *discrete event simulation* (DES) (Ferscha 1995; Fujimoto 1990) of *logical process* (lp) based models. Simulating a model in parallel is a tricky business. First of all, a parallel simulation algorithm is non-trivial. Furthermore, the speedup is model-dependent and therefore not guaranteed, but it is critical for the parallelisation field (Fujimoto 1993).
This paper proposes a standard approach for detailed performance analysis and develops it for our conservative algorithm (Brissinck 1999; see also parallel.vub.ac.be for more information), which is based on the traditional Chandy/Misra/Bryant (CMB) algorithms (Bryant 1977; Chandy and Misra 1979; Ferscha 1995).
The proposed approach is based on elaborating the performance analysis in steps:
(1) Speedup of any parallelised sequential program can be expressed by (Kumar et al. 1994):

$$Speedup = \frac{p}{1 + \frac{parallel\ overhead}{sequental\ runtime}} \quad (1)$$

The partitioning, the communication, the synchronisation and the processor idle time (due to load imbalances) are the main causes of parallel overhead.
(2) We express the sequential and the parallel simulation time for the domain of DES and more specific for our algorithm.
(3) The model-dependency of the performance is studied by the performance factors.
(4) Finally, we propose a channel-based approach for prediction of the performance factors.
In the formulas we use the following naming conventions: C for model-dependent constants, A for algorithm constants and H for hardware constants. For the indexes we use i for the processor, j for the overhead term, k for the cycle and l for the channel.
The illustrating experiment is the simulation of a detailed IP-switch (Geudens 2000) on our system, which is a cluster of 6 333MHz CPU's connected by a 100Mb non-blocking switch. The measurement of the different constants of the equations is performed by Wei Zhu (2001).

SEQUENTIAL SIMULATION

Sequential simulation is the processing of events in chronological order. The simulation algorithm consists therefore mainly of a sorting of the eventlist (a skewheap sort) and a processing of the events by the processes of the model.
The performance is given by the time to simulate 1 second realtime:

$$SeqSimT = (A_{ev} + C_{proc}).ev^s + H_{sort}.\log_2 ev^p.ev^s \quad (2)$$

- ev^s : number of events simulated in 1s realtime (event density)
- ev^p : average number of events in the eventlist, $\log_2 ev^p$ gives a good approximation for the average number of sort iterations (Nicol 1998)
- C_{proc} : average time for processing 1 event by a logical process
- A_{ev} : algorithm time to prepare 1 event to be processed by the process (1,52 µs/event)

- H_{sort} : time for 1 sort iteration (0,19µs/event)

Equation (2) can be rewritten as

$$SeqSimT = C_{sim}.ev^s \quad (3)$$

where C_{sim} groups all the work to be done for the simulation of one event.

Experimental results of the simulation of the detailed IP-switch (on a 333MHz CPU) are shown in table 1. Our sequential simulator achieves thus a performance of simulating 100.000 events per second.

Table 1: Sequential Simulation

SeqSimT	6839s	100%	$ev^s = 661.259.000$
Event processing	5030s	74%	$C_{proc} = 6{,}08$µs/ev
Event sorting	1809s	26%	$\log_2 ev^p = 14{,}4$

PARALLEL SIMULATION

Parallel simulation divides the simulation work among p processors, but adds extra overhead. Parallel simulation time for 1 second realtime is given by:

$$ParSimT = ParSimT_i = C_{sim}.ev_i^s + overhead^i \quad (4)$$

where:

- i: the index of the processor
- ev_i^s : number of events simulated in 1s realtime on processor i
- $overhead^i$: parallel overhead on processor i.
- *p*: number of processors

The simulation time for all processors is equal (they have to wait for each other) and that $\sum_i ev_i^s = ev^s$. Equation (4) can than be rewritten as:

$$ParSimT = \frac{\sum_i (C_{sim}.ev_i^s + overhead^i)}{p} = \frac{C_{sim}.ev^s + \sum_i overhead^i}{p} \quad (5)$$

The parallel overhead can be written as a sum of the several overhead causes (indicated with index j), which we call the *parallel overhead terms* OT_j: $overhead^i = \sum_j OT_j^i$ and $OT_j = \sum_i OT_j^i$ (the sum of the overhead over all processors). Parallel performance, measured by the speedup S, can than be written as (using (3) and (5)):

$$S = \frac{SeqSimT}{ParSimT} = \frac{C_{sim}.ev^s}{\frac{1}{p}.(\sum_i C_{sim}.ev_i^s + \sum_i \sum_j OT_j^i)} = \frac{p}{1 + \sum_j \frac{OT_j}{C_{sim}.ev^s}}$$

We define $ST_j = \frac{OT_j}{C_{sim}.ev^s}$ as the *slowdown terms*, they give the ratio of the overhead versus the simulation work. They are expressed in percentages, giving the impact of the overhead on the parallel performance. This results in:

$$S = \frac{p}{1 + \sum_j ST_j} \quad (6)$$

Parallel discrete event simulation starts with **partitioning** the model, this is the initial overhead term (OT_{part}).

Conservative simulation happens in cycles of length *CycleT* consisting of 4 phases:

1. **Computation**: simulation of the events advances the realtime with *CycleT*.
2. **Communication**: send outgoing events to the other processors.
3. **Blocking**: wait until all processors have finished with the last cycle.
4. **Synchronisation**: determine next safe simulation cycle (CycleT).

Phases 2, 3 and 4 cause the parallel overhead, respectively OT_{comm}, OT_{block} and OT_{sync}.

OVERHEAD TERMS

The next step is calculating the overhead terms for specific algorithms. Here we neglect the partitioning overhead because our research focusses on symmetric models, which are straightforward to partition.

For our algorithm, each phase of the cycle results in 2 overhead terms.

The communication overhead (OT_{comm}) is the per event overhead (OT_1), which is proportional to the number of communicated events between the processors, and second the constant communication overhead (for setting up a communication link) (OT_2), proportional to the number of links and the cycle frequency.

The synchronisation overhead (OT_{sync}) is the processing in each cycle of the synchronisation information (OT_3). For our conservative algorithm this is the null event processing. Moreover, our algorithm induces extra overhead by the conditional queue (OT_4).

The blocking overhead (OT_{block}) is the processor idle time, the waiting of all processors for the slowest processor (with the largest computation phase), summed over all cycles. We identified 2 types of load imbalances: the *global load imbalance* (OT_5), caused by inequal total work (number of events to be simulated) of the processors and the *temporal load imbalance* (OT_6), caused by the fluctuations of the

processor load during simulation. This splitting up of the load imbalance is important, but couldn't be found in any literature. OT_5 is caused by bad partitioning, but OT_6 happens on any model for any partitioning. Moreover OT_6 increases with more processors and is difficult to counter. In our experiments OT_5 stays relatively small, but OT_6 goes up to 25% on 8 processors and causes the main slowdown.

The overhead terms can thus be written as:

$$OT_{part} = OT_0 \approx 0$$

$$OT_{comm} = OT_1 + OT_2 = H_{comm}.ev^{comm} + \frac{C_{link}.H_{setup}}{CycleT}$$

$$OT_{sync} = OT_3 + OT_4 = C_{sync}.ev^{null} + A_{cond}.ev^{cond}$$

$$OT_{block} = OT_5 + OT_6 = p.\sum_{k}^{cycles}(CompT^k_{\max} - CompT^k_{avg})$$

$$OT_5 = p.(CompT_{\max} - CompT_{avg})$$

$$OT_6 = OT_{block} - OT_5$$

where:

- ev^{comm} : density of communicated events
- ev^{null} : density of null events
- ev^{cond} : density of conditional events
- C_{link} : number of links between the processors
- *CycleT*: average cycle time
- C_{sync} : average time to process a null event
- $CompT_i = C_{sim}.ev_i^s$: time to process the events on processor i. *Max* and *avg* denote respectively the maximum and the average over all processors. Index *k* denotes the cycle.
- A_{cond} : time to process a conditional event (0,48µs/ev)
- H_{comm} : time for communicating 1 event (10,3µs/ev)
- H_{setup} : time to setup a communication link between two processors (285µs/link)

PERFORMANCE FACTORS

The impact of the model on the slowdown terms ST_j translates in the *parallel performance factors* (Table 2).

- The first overhead term (OT_1), the per event communication overhead, leads to PF_1: the ratio communication versus simulation or ev^{comm}/ev^s .
- OT_2 leads to evSim/cycle (PF_2), the number of simulated events per cycle. This ratio is called *granularity* or *grain size* (Choi 1995) (or *event simultaneity* in Peterson 1993).
- OT_3 is the synchronisation overhead per cycle. It is proportional to ev^{null}/ev^s (PF_3).
- OT_4 leads to ev^{cond}/ev^s (PF_4).
- OT_5 leads to PF_5: the relative deviation of the workload of the processors.
- Finally, OT_6 is the sum of the per cycle ev^s fluctuation (PF_6).

Note that PF_2 must be as big as possible for better speedup, whereas the other 5 *PF*'s as small as possible.

Table 2: Parallel Performance Factors

Causes				Performance Factors
Communication	OT_1	ST_1	PF_1	ev^{comm} / ev^s
	OT_2	ST_2	PF_2	ev^s / cycle
Synchronisation	OT_3	ST_3	PF_3	ev^{null} / ev^s
	OT_4	ST_4	PF_4	ev^{cond} / ev^s
Computation	OT_5	ST_5	PF_5	($ev^s_{\max}$ - ev^s_{avg}) / ev^s
	OT_6	ST_6	PF_6	(temporal Δev^s) / ev^s

EXPERIMENTAL RESULTS

The parallel simulation time (with 6 processors) of 1s realtime was 1683s, resulting in a speedup of 4,06. The total slowdown of 45,6% (equation (6)) was caused by:

Table 3: Experimental Performance Results

ST_{comm}		ST_{sync}		ST_{block}	
14,4 %		4,3 %		26,9 %	
ST_1	ST_2	ST_3	ST_4	ST_5	ST_6
3,8 %	10,6 %	4,3 %	0 %	7,0 %	19,9 %
PF_1	PF_2	PF_3	PF_4	PF_5	PF_6
4,1%	7601	1,4%	0	5,5%	15,9%

Our parallel simulator achieves thus a performance of simulating 400.000 events per second with 'good' parallelisable models .

Another experiment done at the lab, the simulation of Field Programmable Gate Arrays (FPGAs) (Bousis 2000) revealed a low speedup of 0,7. This is due to a low PF_2 (70 events per cycle), leading to an OT_{comm} up to 25%, and a high PF_3 (470%), giving a very high OT_{sync}. The very low granularity (PF_2) causes the communication and synchronisation overhead to overwhelm the parallelisation. This indicates the need for a more specific algorithm with less synchronisation overhead.

This performance analysis is integrated in our simulation environment. It measures and calculates automatically the OT's, ST's and PF's.

PERFORMANCE PREDICTION

For performance prediction, we should estimate the performance factors. We propose a novel channel-based approach. The model-characteristics are calculated by summation over the channels:

$$ev^s = \sum_{ch} ev_l^s$$
$$ev^p = \sum_{ch} ev_l^p$$
$$ev^c = \sum_{ch^c} ev_l^s$$

- *ch*: the total number of channels. (= degree of parallelism)
- ch^c : the communication channels

Then, the event densities (ev_l^s and ev_p^s) of each channel must be estimated. They are determined by the type of the channel and the channel parameters.
For example, for a 'bandwidth channel', characterised by a rate of events (the bandwidth BW), a certain delay and a load, the channel characteristics are:

$$ev_l^s = BW.load \quad (7)$$
$$ev_l^p = BW.delay.load$$

In a traffic model (Aerts 2000), a street is identified by the velocity v of the cars, its length and a load. The delay is the length divided by the velocity. The bandwidth of the street is then (assume a street width of 1):

$$BW = v = \frac{length_{street}}{length_{event}} . \frac{1}{delay} \quad (8)$$

By combining equations (7) and (8), the channel event densities can be expressed in general as:

$$ev_l^s = \frac{1}{delay} . \frac{length_{channel}}{lenght_{event}} .load$$

where the *delay* is the *time scale*, $length_{channel} / length_{event}$ is the *space scale* and the *load* is the *scale of the experiment*. These densities must be summed over all channels (the number of channels can be called the *scale of parallelism*).
In the proposed approach, one must first identify all channel types, elaborate the density equations, calculate them with the channel parameters and sum them (eg. in a spreadsheet).

FUTURE WORK

(1) Elaborate and test above prediction approach.

(2) For scalability analysis, the overhead terms should be expressed in function of p and model scale W. For example, for a full interconnect model (all partitions have to communicate) this results in (Jian Nan 2001):

$$SeqSimT \sim W$$
$$OT_1 \sim W.p \quad (9)$$
$$OT_2 \sim C_{link} \sim P_p^2 \sim p^2 \quad (10)$$

This results in:

$$ST_1 \sim p \quad (11)$$
$$ST_2 \sim p^2 / W \quad (12)$$

In this model the communication grows with the scale (9) and the number of links grow exponentially (10). When the communication doesn't grow with the scale (the model 'scales in depth') and the number of links is proportional to p, (11) and (12) become

$$ST_1 \sim p / W$$
$$ST_2 \sim p / W$$

and this model is better scalable.
This analysis has to be further developed for the other overhead terms.

(3) For algorithm comparison, one must elaborate the equations for the overhead terms and determine the performance factors. The choice of the optimal algorithm is model-dependent, this must be done by measuring or estimating the PF's. At our lab we developed an alternative sort algorithm for sequential simulation (see equation (2)), which performs better for dense eventlists (high ev^p). A decision function determines which type of sorting is used.

(4) Integrate other research in this proposed framework.

CONCLUSION

To study the performance of parallel simulation we should understand the parallel overhead in detail. In this paper we propose a standard approach that separates the domain (here DES), the algorithm and the model aspects. Therefore the Overhead Terms and the corresponding model-dependent Performance Factors are defined. The analysis is developed for our CMB-based conservative algorithm, but the same approach can be used for any algorithm or parallelisation in general.

Other performance studies (Jha and Bagrodia 1996; Ferscha et al. 1996; Nicol 98; Liu et al. 1999; Lim et al. 1999) contain the same elements, but indicate less clear boundaries between algorithm, model, hardware and other aspects. Also the lack of standards makes synchronisation of researches difficult at the moment. For, as we know, a general solution for parallel DES does not exist, due to the model-dependency of parallel performance. Therefore, structuring and combination of the different researches is indispensable for mastering parallel performance, the sole justification of the existence of parallelisation.

REFERENCES

Aerts J. 2000. "Dynamische Simulatie van Verkeersstromen". *Thesis*, Free University of Brussels.

Bousis L. 2000. "Study and Implementation of a Scalable Simulator for Complex Digital Systems." *Thesis*, Free University of Brussels.

Brissinck W. 1999. "Tuneable Granularity Parallel Discrete Simulation." *PhD thesis*, Free University of Brussels (May).

Bryant, R.E. 1977. "Simulation of Packet Communications Architecture Computer Systems." MIT-LCS-TR-188, Massachusetts Institute of Technology.

Chandy, K.M., and Misra, J. 1979. "Distributed Simulation: A Case Study in Design and Verification of Distributed Programs." *IEEE Trans. on Softw. Eng.* SE-5, 5 , 440-452, (Sep).

Choi E., Chung M. J. 1995. "An important factor for optimistic protocol on distributed systems: granularity." *In 1995 Winter Simulation Conferences Proceedings*, pp 642-649.

Ferscha A. 1995. "Parallel and Distributed Simulation of Discrete Event Systems." *Handbook of Parallel and Distributed Computing,* McGraw-Hill.

Ferscha A., Johnson J. 1996. "A testbed for parallel simulation performance prediction." *Proceedings of the 1996 Winter Simulation Conference,* pp 637-644.

Fujimoto R.M. 1990. "Parallel Discrete Event Simulation." *Communications of the ACM*, 33, pp 29-53 (Oct).

Fujimoto, R. M. 1993. "Parallel Discrete Event Simulation: Will the Field Survive?" *ORSA Journal of Computing*, 5(3):218:230.

Geudens S. 2000. "Quantitative Study of a Highly Formant Network Switch with Distributed Simulation." *Thesis*, Free University of Brussels.

Jha V., Bagrodia R. 1996. "A performance evaluation methodology for parallel simulation protocols." *Proceedings of the 10th Workshop on Parallel and Distributed Simulation (PADS*).

Jian Nan G. 2001. "Scalability and Performance Study of Parallel Discrete Event Simulation." *Thesis*, Free University of Brussels.

Kumar V., Grama A., Gupta A. and Karypsis G. 1994. *Introduction to Parallel Computing. Design and Analysis of Algorithms*. Benjamin Cummings, California, chapter 4.

Lim C-C., Low Y-H., Gan B-P. and Jain S. 1999. "Performance Prediction Tools for Parallel Discrete-Event Simulation". *Proceedings of the 13th Workshop on Parallel and Distributed Simulation (PADS)*.

Liu J., Nicol D.M., Premore B. and Poplawski A. 1999. "Performance Prediction of a Parallel Simulator*". Proceedings of the 13th Workshop on Parallel and Distributed Simulation (PADS)*.

Nicol D.M. 1998. "Scalability, Locality, Partitioning and Synchronization in PDES". *Proceedings of the 12th Workshop on Parallel and Distributed Simulation (PADS*).

Peterson G.D., Chamberlain R.D. 1993. "Exploiting lookahead in synchronous parallel simulation". *In 1993 Winter Simulation Conferences Proceedings*, pp 706-712.

Zhu W. 2001. "Experimental Study of Influence of Model Characteristics on the Performance of Parallel Discrete Event Simulation". *Thesis*, Free University of Brussels.