A PRACTICAL EFFICIENCY CRITERION FOR THE NULL MESSAGE ALGORITHM

András Varga[†] Y. Ahmet Şekercioğlu[‡] Gregory K. Egan[‡]

[†]Omnest Global Inc., Budapest, Hungary

[‡]Centre for Telecommunication and Information Engineering, Monash University, Melbourne, Australia

KEYWORDS

Parallel simulation, discrete-event simulation, PDES, cluster computing

ABSTRACT

This paper presents a quantitative criterion for efficient execution of the Null Message Protocol, the best-known conservative parallel discrete event simulation (PDES) protocol. By using the criterion, a model designer can use lookahead and communication latency as input and improve the efficiency of parallelization. Earlier works consider lookahead in relation to model properties like timestamp increment and in isolation from the capabilities of the underlying hardware/software simulation environment, and have not been able to provide quantitative criteria for performance prediction.

Our results suggest that the performance impact of lookahead can only be quantified when linked to other performance factors such as communication latency and throughput between partitions of a parallelized simulation model. The latency and throughput issues are becoming of increasing importance as clusters gain popularity as PDES platforms.

The criterion is based on a novel concept of the coupling factor, and allows one to use intuitive and easy-to-measure input parameters. The criterion can be used to assess simulation models' potential for parallel execution as well as the maximum partitioning that may still potentially yield good performance. This paper is also novel in that it uses the Ideal Simulation Protocol as a benchmark.

INTRODUCTION

Telecommunication networks are increasingly becoming more complex as the trend towards integration of telephony and data networks into integrated services networks gains momentum. Discrete event simulation is an important tool for the research and design of these systems. However, simulation of telecommunications networks is generally a computationally intensive task. A single run of a wireless network model with thousands of mobile nodes may easily take several days or even weeks to obtain statistically significant results even on today's computers. Additionally, many simulation studies require several simulation runs to improve statistical reliability of the outcomes (Bagrodia et al., 1998). These conditions have recently led to an increased interest in Parallel Discrete Event Simulation (PDES) techniques. Good overviews of PDES techniques can be found in (Nicol and Fujimoto, 1994) and (Soliman et al., 1995).

Although recent PDES literature focuses more on optimistic algorithms than conservative ones, optimistic algorithms are usually too complex or impractical to implement in practice. At the same time, we can observe a revival of conservative algorithms as they slowly seem to find their way into simulation tools. This is especially true for telecommunication network simulations where the demand for processing power is the strongest: SSFNet (ssfnet), ns2 (PADS Research Group), OMNeT++ (Şekercioğlu et al., 2003). Another trend is that clusters (as opposed to shared memory multiprocessor systems) are becoming an attractive PDES platform (Pham, 1999), mainly because of their excellent price/performance ratio. The best known conservative algorithm is still the classic Chandy-Misra-Bryant (CMB), also known as the null message algorithm (NMA) (Chandy and Misra, 1979) (Bagrodia and Takai, 2000); we will use the latter name throughout the paper.

Despite the intensive research efforts on lookahead and its effect on the performance of the parallel simulation, we still do not have quantitative criteria for predicting the performance of a parallel simulation. This is especially true in a cluster computing environment where finite bandwidth and relatively large communication latencies are present. Also, Ideal Simulation Protocol (ISP) (Bagrodia and Takai, 2000), a significant and powerful tool for the research of performance aspects of PDES algorithms was only discovered in 2000, and since then it has gone relatively unnoticed within the PDES research community.

The first section of this paper describes performance factors of a parallel simulation. The second part focuses on the classic NMA algorithm and derives the criterion that can predict the performance of NMA. The criterion allows the simulation designer to use input data that are easy to produce for any given simulation, and it can also be applied in a cluster environment because it takes into account the communication latency. The third part describes simulation

experiments done on a cluster to validate the performance criterion. Here, we use the ISP as a performance benchmark.

PERFORMANCE FACTORS

Lookahead

Lookahead has crucial importance in conservative algorithms. Lookahead is associated with the ability of a logical process (LP), in other words, a partition of a model, to predict its future behavior. It has many, slightly different definitions in the literature (Preiss and Loucks, 1990); for this paper we settle for the following: *At any simulation time T, if an LP can predict that the earliest event it will cause to occur in another LP is no sooner than $T + L$, it has a lookahead of L towards that LP.* Lookahead may be different for each (ordered) LP pair; moreover, lookahead is, in general, state dependent and may even change during the execution of the lookahead period.

For specific classes of models, different model components can lead to lookahead. In queueing simulations for example, the fixed processing time of a queue server may provide lookahead; so can a minimum inter-arrival time in a source. In telecommunication networks, propagation delay on links (or propagation delay plus the transmission time of a minimum-length frame) serves as a natural source of lookahead. Also, observing the full picture might provide better lookahead than individual components in the model – simulation techniques like *path lookahead* (Meyer and Bagrodia, 1999) exploit this observation.

It has been observed and is common knowledge that lookahead has a significant effect on the performance (Fujimoto, 1989). Lookahead and its impact on the performance of parallel simulations has been analyzed in (Lin and Lazowska, 1990), (Preiss and Loucks, 1990), (Wand and Abrams, 1995) and other papers, and yet the simple question of *when* lookahead is large enough to provide reasonable performance has remained unanswered.

Earlier works consider lookahead in relation to model properties like timestamp increment, and in isolation from the capabilities of the underlying hardware/software simulation environment. Our results suggest that the performance impact of lookahead can only be quantified when linked to other performance factors such as communication latency and throughput between LPs.

Latency and throughput

Early PDES experiments on clusters have delivered disappointing results compared to shared memory multiprocessor systems (Pham, 1999). This fact is attributable to much higher communication costs on clusters, namely (a) high processing overhead associated with sending and receiving messages, (b) finite bandwidth of the communication channel, and (c) higher latencies.

Several studies have been conducted on the performance of parallel simulations on clusters, e.g. (Lemeire and Dirkx, 2001) and (Xu and Chung, 2001). Most of them focus on aspects (a) and (b) and tend to ignore (c). In particular, no studies have linked the question of communication overheads with proper lookahead in an attempt to predict the performance of parallel simulation algorithms.

The Ideal Simulation Protocol

One cannot expect linear speedup from PDES because, compared to sequential execution, some parts of the model are now separated by LP boundaries, and transmission of model messages across LPs presents an overhead that was not present in the sequential model. Since these messages are part of the model, we can do little to reduce this overhead.

In addition to this *messaging overhead*, we also have *synchronization overhead*, an overhead added by the parallel simulation (synchronization) algorithm. With NMA, this overhead is associated with the transmission of null messages and with time spent blocking on *earliest input times (EITs)* (an NMA overview, including the definitions of terms is provided in the next section. Synchronization overhead may be reduced by tuning the parameters of the PDES algorithm or choosing a different algorithm.

When evaluating the efficiency of a parallel simulation algorithm, it would be useful to know what is the *maximum achievable speedup*, that is, the speedup if synchronization overhead were zero. Until recently, researchers have not been able to directly measure the maximum achievable speedup, and hence, PDES studies have been published without this comparison. The Ideal Simulation Protocol (ISP) introduced by Bagrodia (Bagrodia and Takai, 2000) can provide this missing information.

ISP is not an abstraction as it may sound, but an actual parallel simulation algorithm that can be implemented and models can be run under it. Running a model under ISP does not incur any synchronization overhead (in fact, there *is* some overhead, but it is usually negligible), while all other overheads including messaging overhead remain unchanged. Therefore, ISP presents the upper limit on the performance that any parallel simulation algorithm can achieve under the same circumstances (executing the same model with the same partitioning, on the same hardware, using the same operating system and simulator, with same message transport between LPs, etc.).

When evaluating the efficiency of various PDES algorithms, their performance ratios in relation to ISP are far more useful and relevant numbers than the speedup figures. Comparison to ISP tells us how far we are from the best achievable parallel performance, while speedup also contains the messaging overhead, a factor that cannot be blamed on the synchronization algorithm itself.

PERFORMANCE OF NULL MESSAGE ALGORITHM

The Null Message Algorithm

For the discussion of the Null Message Algorithm (NMA), we use the terminology introduced by Bagrodia in (Bagrodia and Takai, 2000). According to NMA, LPs maintain variables called *earliest input times (EITs)* for each input neighbor, and *earliest output times (EOTs)* for each output neighbor LP. An EIT stores the earliest simulation time the LP may receive an event from another LP. Respectively, an EOT stores the earliest simulation time the LP might send a message to its neighbor. Practically, EOT represents the local simulation time plus the lookahead.

LPs are safe to process events until the minimum of their EITs ("safe" meaning without the danger of receiving events in their past). Once an EIT has been reached by a LP, it has to block until the EIT is updated. EITs are updated by *null messages* arriving from other LPs. A null message carries a timestamp, set by the sender LP to its current EOT. Once the null message arrives, the timestamp will be stored by the receiving LP as a new EIT. For obvious reasons, EITs grow monotonically.

LPs must send out null messages often enough to prevent deadlock, i.e. at least before the expiry of the EOT sent out in the in the last null message. In the hope of improving performance, LPs may actually choose to send out null messages more often than necessary, leading to the designation of *eager* and *lazy* algorithms. Laziness is a tunable parameter of NMA. For optimization, null messages may be piggybacked on normal outgoing messages. It has been proven that the nonexistence of zero lookahead cycles in the graph is enough to prevent deadlocks.

Performance of NMA

When the NMA performs poorly compared to ISP (note that NMA can only approximate ISP performance – if parallel performance under ISP is already poor (e.g. because of too much cross-partition messaging), it will inevitably be poor under NMA, too), causes can be traced back to one of the following two reasons:

a. *Too frequent null messages.* If lookahead is poor compared to the simulation time between events, that leads to excessive null message traffic. Resources are consumed by sending, waiting for and receiving null messages, instead of processing events.

b. *Too much time spent on blocking on EITs.* In this case, processors idle too much, waiting for null messages to arrive. This can be caused by too tight coupling of LPs, as a consequence of too little workload on LPs, combined with poor lookahead and/or long communication latencies and/or poor load balancing.

Although not evidenced by measurements, it is strongly felt that additional factors (such as overhead of null message sending in case of many LPs) may only be significant if (a) and (b) are solved. In the following sections we will examine the above two factors and provide quantitative criteria for them. We will use the following parameters as input:

- *P performance* represents the number of events processed per second (ev/sec) (we use the following notation: *ev:* events, *sec:* real seconds, *simsec:* simulated seconds). *P* depends on the performance of the hardware and the amount of computation required for processing an event. *P* is independent of the size of the model. On the authors' computer, simulations using OMNeT++ (Varga, 2001) usually yield *P* values between 20,000 and 120,000 ev/sec, depending on the nature of the model.

- *E event density* is the number of events that occur per simulated second (ev/simsec). *E* depends on the model only, and not where the model is executed. *E* is determined by the size, the detail level and also the nature of the simulated system (e.g. cell-level ATM models produce higher *E* values than call center simulations.)

- *R relative speed* measures the simulation time advancement per second (simsec/sec). *R* strongly depends on both the model and on the software/hardware environment where the model executes. Note that $R = P/E$.

- *L lookahead* is measured in simulated seconds (simsec). When simulating telecommunication networks and using link delays as lookahead, *L* is typically in the msimsec-μsimsec range.

- τ *latency* (sec) characterizes the parallel simulation hardware. τ is the latency of sending a message from one LP to another. τ can be determined using simple benchmark programs. The authors' measurements on a Linux cluster interconnected via a 100Mb Ethernet switch using MPI yielded $\tau = 22\mu$s which conforms to the measurements reported in (Ong and Farrell, 2000); specialized hardware such as Quadrics Interconnect (quadrics) can provide $\tau = 5\mu$s or better.

In large simulation models, *P*, *E* and *R* usually stay relatively constant (that is, display little fluctuations in time). They are also intuitive and easy to measure. For example, the OMNeT++ simulation tool displays these values on the GUI while the simulation is running, see Figure 1.

Too Frequent Null Messages

If lookahead is too small compared to the mean simulation time between events, several rounds of null messages will be needed to advance simulation time over otherwise event-free time periods. As an illustration, consider a queueing network model executed in parallel. Let the processing time of the queue servers be 0.1 seconds. We use the processing time as lookahead, so EIT is increased in 0.1s steps. If jobs arrive at the queues only every 3 seconds, then 30 rounds

Figure 1: Performance bar in OMNeT++ showing P, R and E

of null messages are necessary to advance simulation time to the next event. This phenomenon is widely analyzed in the literature, see (Nicol and Fujimoto, 1994) for example. Each round of null messages takes at least τ time (null messages have to arrive), which can be painful especially on cluster hardware.

In order to achieve good performance, the lookahead needs to be significantly larger than the time between events: $L \gg 1/E$. Rewriting this in a more convenient form:

$$LE \gg 1 \tag{1}$$

The actual threshold for LE depends on the relative cost of sending a null message and processing an event as well as τ and P. It is also worth noting that the number of null messages sent out and thus the null message sending overhead grows as the number of output neighbor LPs grow.

Note that L and E are properties of the model, which means that some models will always perform poorly under NMA, regardless of the software/hardware environment. A remedy to this problem could be to use an alternative synchronization method, e.g. Conditional Event Algorithm (which relies on calculating global virtual time), as suggested by Bagrodia in (Bagrodia and Takai, 2000).

LE is inversely proportional to the *lookahead ratio* (Preiss and Loucks, 1990).

Too Much Blocking on EITs

Null messages must reach a target LP early enough so that the target LP does not need to block on the EIT before the null message arrives. In other words, an LP has to have enough work (events to process) until the next null message arrives.

We use a simple simulated scenario to examine and quantify this criterion. We model two networks that are connected by an 1000km fiber optic cable, with a propagation delay of about 5ms (Figure 2). The two networks are executed in separate LPs, using the Null Message Algorithm. We use the cable delay as lookahead, that is, $L = 5$ms. We run the simulation on a cluster computer that provides us $\tau = 10\mu$s latency. Also, assume the following ideal conditions: idle link (no modeled traffic), lazy null message sending, and virtual times proceed in sync in the two LPs. Then, LPs will periodically exchange null messages with each other (every $L = 5$ms simulation time; see Figure 3). Null messages sent out from one LP take τ (real) time to arrive. They have to arrive at the other LP before that LP reaches the EIT received in previous null message, that is, in less time than it takes the target LP to advance L simulation time. This results in the following criterion:

$$\tau < \frac{L}{R} \tag{2}$$

Using $R = P/E$ and rearranging the equation we get that

$$\tau P < LE \tag{3}$$

That is, if (3) holds and the described ideal conditions are present, LPs will never block on EITs. But what if conditions are different?

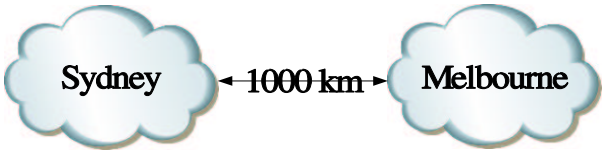

Figure 2: The simulated scenario

If null message sending is not lazy, there will be more frequent null messages, which further reduce the probability of blocking. Actually, more frequent null messages help "smooth out" fluctuations in latency and virtual time progression, at the cost of more messaging overhead.

If there is traffic on the link, it also means that more frequent null messages (piggybacked on simulated packets) are present, which reduces the possibility of blocking.

What if simulation times are not in sync? In real life, simulation time does not pass completely evenly (R relative speed fluctuates in both LPs), and this causes simulation times to be skewed. The NMA algorithm places an upper bound equal to the lookahead on the skew, because LPs have to block whenever an EIT has been reached. We will show that in order to reduce the amount of blocking, the inequality (3) has to hold "stronger" (that is, $\tau P \ll LE$).

If there are more partitions, inequality (3) should hold for all ordered LP pairs. This is covered in greater detail in the section entitled "Applying To Several LPs".

Inequality (3) displays the following property: the left hand side, τP, depends mainly on properties of the hardware (and much less on the model) and the right hand side, LE only depends on the model (and not at all on the hardware). τP expresses how many events are processed during τ time. Its value characterizes the hardware: a small value indicates fast communication compared to processing speed, in other words, shared-memory-type hardware, while large values indicate strong processing power and slower communication, that is, cluster-type hardware. The actual boundary seems to be around $\tau P = 1$. LE shows the model's potential for efficient PDES under NMA. It expresses "how many events the lookahead cover". Small values mean poor lookahead, and large values mean good lookahead. The larger the value of LE, the more potential the model has to perform well on cluster-type hardware.

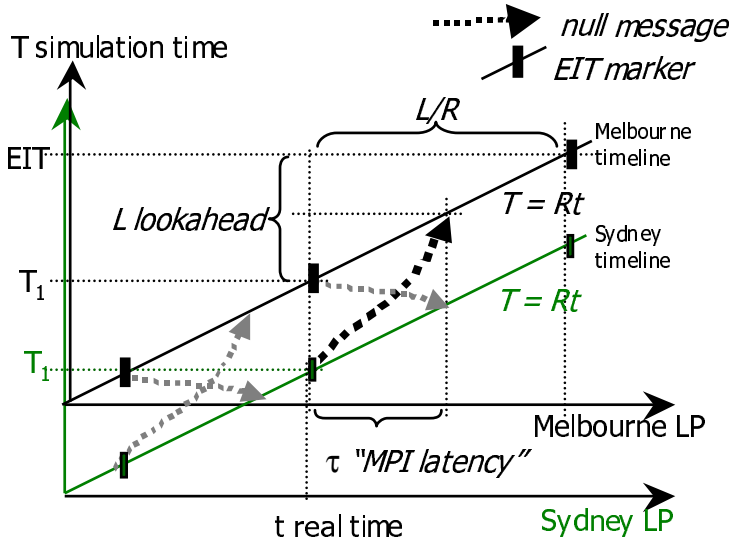

Figure 3: Periodic exchange of null messages in the simulated scenario

$\tau P < LE$ only guarantees that there's no blocking if simulation time passes completely evenly, that is, $R = P/E$ relative speed is constant. In practice, there are fluctuations in P and E, so $\tau P = LE$ will lead to frequent blocking because the LPs are too tightly coupled. $\tau P < LE$ allows for fluctuations in P and E to occur, reducing the chance of blocking on EITs. Let us introduce λ *coupling factor*, the ratio of LE and τP:

$$\lambda = \frac{LE}{\tau P} \tag{4}$$

It follows from its definition that if $\lambda < 1$, frequent blocking on EITs is guaranteed and one cannot expect good performance from the simulation; if $\lambda \geq 1$ but small, more or less blocking will occur depending on how much P and E fluctuate, and as experimental results show, this heavily affects performance. If $\lambda > 1$ and large enough, then blocking on EITs is no longer a major performance factor. Our experimental results indicate that λ values below 10 should be regarded as "small" and values above 100 as "large", the exact performance characteristics being dependent on the simulation model.

The question of "when is lookahead big enough" can then be answered: when λ is greater than a λ_0 threshold chosen in the range $10 \ldots 100$, that is:

$$L > \frac{\lambda_0 \tau P}{E} \tag{5}$$

One can think of several intuitive interpretations of λ: "how many times τ it takes to progress L simulation time," or "how big is lookahead L, in units of simulation time that can be covered during τ time".

Another interpretation can be derived from the skew of the simulation times in the LPs. As one can deduce from Figure 3, the simulation times can be skewed at most $L - R\tau$ if we want to avoid blocking. (Intuitively: if the null message took zero time to arrive ($\tau = 0$), simulation times could be skewed by L amount, but this is decreased by the simulation time that is covered during τ time.) If expressed in terms of λ, the maximum skew is $R\tau(\lambda - 1)$. This explains why λ is called coupling factor: $\lambda = 1$ means tight coupling (because it forces zero skew and simulation times to pass in sync in the two LPs, possibly at the cost of frequent blocking), and a large λ means loose coupling (allows for more skew and therefore more fluctuations in P and E, without heavily affecting performance).

Applying To Several LPs

We need to apply the criteria for all (ordered) pairs of LPs, because the slowest LP pair determines simulation speed. Note that when virtual time progresses slowly in an LP, it also "pulls back" other LPs. Let E_i and P_i be the event density and performance in LP_i, and let $L_{i,j}$ and $\tau_{i,j}$ be the lookahead and latency *toward* LP_i from its LP_j input neighbor (Figure 4). Let us define $\lambda_{i,j}$ as follows:

$$\lambda_{i,j} = \frac{L_{i,j} E_i}{\tau_{i,j} P_i} \tag{6}$$

To achieve good performance, all $L_{i,j}E_i$ (1) and $\lambda_{i,j}$ (4) values have to be sufficiently high, thus we can define an *effective coupling factor* as

$$\lambda = \min_{i,j} \lambda_{i,j} \tag{7}$$

NMA Scalability and Partitioning

How does NMA scale with the number of processors used? We can expect P_i and $\tau_{i,j}$ values to stay relatively constant as

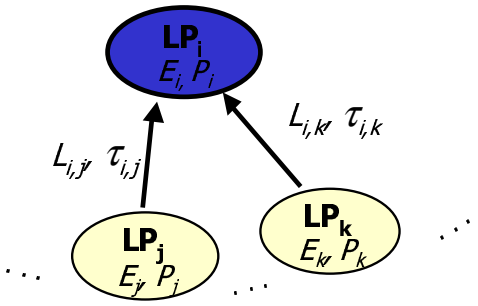

Figure 4: Multiple LPs

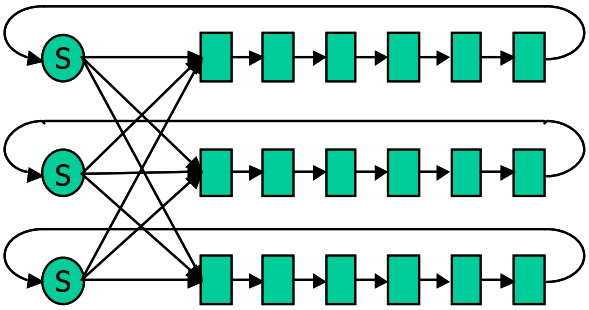

Figure 5: The Closed Queueing Network (CQN) model, which consists of N tandem queues where each tandem consists of a switch and k single-server queues

the number of LPs grows (n). $L_{i,j}$ lookahead might or might not decrease as n grows, depending on the model topology. E_i, however, will decrease. Events do not change as we partition the model, so the event densities, E_i, have to sum up to E_{seq}, the event density of the sequential execution. Thus, for the event density in LP$_i$ with n LPs, $E_{n,i}$:

$$\min_i E_{n,i} \leq \frac{E_{seq}}{n} \tag{8}$$

Consequently, the effective coupling factor for n LPs, λ_n, will also diminish as n grows:

$$\lambda_n \leq \frac{\lambda_0}{n} \tag{9}$$

where λ_0 can be derived from the two-processor simulation as $\lambda_0 = 2\lambda_{n=2}$. As λ_n falls below a critical value, performance degrades. The intuitive explanation is that with heavy partitioning, some processors might not get enough work to do between receiving null messages and thus they are often forced to block on EITs.

A practical use of (9) is to assess the available potential for parallelism in the model, that is, the maximum number of LPs where NMA can still be expected to produce good results.

Partitioning algorithms aware of coupling factor λ should consider only partitions where all $\lambda_{i,j}$ values are above a threshold.

EXPERIMENTAL VERIFICATION OF THE CRITERION

Experiments have been performed to verify the criterion. We have used the closed queuing network (CQN) model described in (Bagrodia and Takai, 2000) (Figure 5). The model consists of N tandem queues where each tandem consists of a switch and k single-server queues with exponential service times. The output of the last queue is looped back to the switch. Each switch randomly chooses the first queue of one of the tandems as its destination, using uniform distribution. The queues and switches are connected with links that have nonzero propagation delays. At the beginning of the simulation, a fixed number of jobs are injected in the system; no jobs are created or destroyed during simulation.

We used a model consisting of $N = 16$ tandems. We performed experiments with 2-way and 4-way partitioning (that is, with 8 and 4 tandem queues per LP). The propagation delay between switches and first queues was used as the lookahead L, and we conducted a series of experiments with various lookahead values. The other variable parameter in the model is k, the number of queues per tandem, which we used to control the event density, E. Initially we inserted 2 jobs in each queue, propagation delays between queues were set to 1, and we used an exponential distribution with a mean of 10 as the queue service time.

The simulation environment was an in-house development version of OMNeT++ framework (Varga, 2001) (Şekercioğlu et al., 2003). The hardware environment was a Linux cluster (kernel 2.4.9) of dual 1 Ghz Pentium III PCs, interconnected using a 100Mb Ethernet switch. The communication library was LAM-MPI (lam-mpi). The MPI latency τ was measured to be 22μs. Sequential simulation of the CQN model achieved $P_{seq} = 120,000$ ev/sec performance, a value that was independent of k.

We performed the simulation under NMA and (for comparison) under ISP. We recorded the average per-LP P_i performances, and also the E_i event densities so that we could calculate λ. We performed experiments with the following $k = 1,2,5,10,20,50,100,200$ and $L = 1,2,5,10,20,50,100,500$ values on 2 and 4 processors.

An excerpt of the results is shown in Table 1. S_{ISP}, S_{NMA} are the speedups achieved under ISP and NMA, respectively (measured as P/P_{seq}); NMA/ISP is the ratio of the NMA and ISP speedups. Graph 6 shows NMA/ISP versus λ. $\lambda > 50$ values all yielded $NMA/ISP > 0.78$ and are not shown in the graph. $\lambda > 200$ yielded $NMA/ISP > 0.95$. We should note that NMA/ISP values slightly bigger than 1 have been observed for very large λ values.

The performance shows strong correlation to λ. Small λ values (under 5 or so) result in poor performance, and large values (over 100) result in a performance near ISP (values larger than 100 do not have significant impact any more). One can observe that the threshold for λ is somewhere in the range $10 \ldots 50$.

LPs	k	L	E_i	λ	S_{ISP}	S_{NMA}	NMA/ISP
2	5	1	5.81	2.20	0.94	0.41	0.43
2	5	2	5.80	4.40	0.99	0.62	0.62
2	5	5	5.76	10.91	1.01	0.92	0.92
2	20	1	21.66	8.20	1.34	0.84	0.63
2	20	2	21.62	16.38	1.40	1.10	0.79
2	20	5	21.59	40.90	1.41	1.33	0.94
4	5	1	2.90	1.10	2.03	0.21	0.10
4	5	5	2.88	5.46	2.01	0.77	0.38
4	5	20	2.78	21.08	2.12	1.67	0.78
4	5	50	2.57	48.72	2.16	2.08	0.97
4	20	1	10.83	4.10	2.94	0.68	0.23
4	20	2	10.81	8.19	2.90	1.10	0.38
4	20	5	10.80	20.46	2.92	1.84	0.63
4	20	10	10.77	40.79	2.99	2.35	0.79

Table 1: Experimental results on 2- and 4-processor configurations with various k and L values

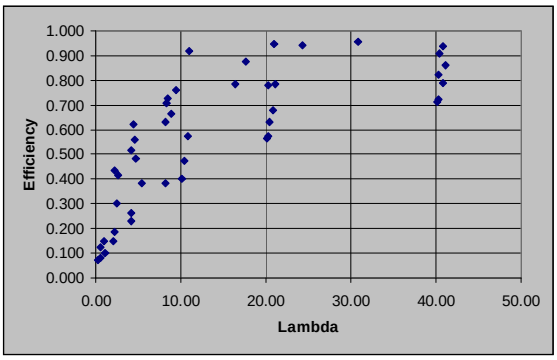

Figure 6: NMA/ISP performance ratio vs λ

APPLYING THE CRITERION IN PRACTICE

The criterion can easily be applied in practice by the simulation designer to find out if a model has the potential for parallel execution with NMA. The P and E_{seq} values can be collected from a sequential run of the model, with the help of the simulation tool. For example, OMNeT++ displays P and E in both its command-line and GUI user interface. τ can be measured by simple test programs, or its approximate value can be guessed from the hardware and software components (as mentioned earlier, a commonly available Linux cluster with MPI yielded $\tau = 22\mu$s in our measurements). Lookaheads, L, can be determined from the model itself.

With L, E_{seq}, τ and P present, it is then possible to assess LE and λ for different numbers of LPs (n), using the $E = E_{seq}/n$ approximation. If satisfactory LE and λ values are present, the model could be deemed to have good potential for NMA.

An example: assume that we intend to run Internet simulations in parallel on the above mentioned Linux cluster ($\tau = 22\mu$s). Experiments show that IP simulations produce around $P = 80,000$ ev/sec on a single CPU. We use link delays as lookahead, typically around $L = 1$ms. What is the smallest network that has a chance to produce good speedup on $n = 4$ CPUs (we would like at least $\lambda = 20$)?

Using Equation (4) and inequality (8)

$$E_{seq} \geq \frac{n\lambda\tau P}{L} = 140,800 \text{ ev/simsec}$$

Here, $LE_{seq}/n = 35.20$, thus inequality (1) also holds. Further simulation experiments can cast light on what event densities are expected to be generated for host, router models etc, under a given simulated network traffic. It is thus possible to deduce the size of the network.

CONCLUSION

We have derived a quantitative criterion to predict in which settings the null message algorithm (NMA) has a potential to perform well. The criterion accounts for the cases when the protocol would send too many null messages or it would block too frequently, and it is based on the newly introduced concept of the coupling factor. We have experimentally verified the criterion, using the performance ratio to the Ideal Simulation Protocol as a benchmark. Although the criterion was introduced in the context of network simulation and cluster computing, it is applicable to any simulation model and shared memory architectures as well.

The criterion provides a quick and practical way to predict whether a simulation model has potential to perform well under NMA in a given simulation environment, and may also help to determine the maximum degree of partitioning where the model can still be expected to produce good performance under NMA.

It is left to further studies to experimentally verify the criterion on different types of models and on a larger number of processors. Further studies are needed to analytically explain why the coupling factor has to be larger than

10 . . . 100. For this, probably stochastic tools and a quantification of fluctuations in performance P and event density E values will be needed.

Acknowledgment

Authors would like to express their gratitude to Brett Pentland for his help on improving the text.

References

R. L. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, and H. Y. Song. Parsec: A parallel simulation environment for complex systems. *IEEE Computer*, pages 77–85, October 1998.

R. L. Bagrodia and M. Takai. Performance evaluation of conservative algorithms in parallel simulation languages. *IEEE Transactions on Parallel and Distributed Systems*, 11(4):395–414, 2000. URL citeseer.nj.nec.com/bagrodia98performance.html.

M. Chandy and J. Misra. Distributed simulation: A case study in design and verication of distributed programs. *IEEE Transactions on Software Engineering SE-5*, (5): 440–452, 1979. URL http://citeseer.nj.nec.com/context/58222/0.

Y. A. Şekercioğlu, A. Varga, and G. K. Egan. Parallel simulation made easy with OMNeT++. In *Proceedings of European Simulation Symposium (ESS2003)*, Delft, The Netherlands, October 2003. Society for Computer Simulation.

R. M. Fujimoto. Performance measurements of distributed simulation strategies. *Trans. of the SCS*, 6(2):89–132, apr 1989.

lam-mpi. LAM-MPI home page. URL http://www.lam-mpi.org/.

J. Lemeire and E. Dirkx. Performance factors in parallel discrete event simulation. In *Proc. of the Int. Multiconference on Simulation and Modeling (ESM 2001), Prague, June, 2001*. Society for Computer Simulation, 2001.

Y.B. Lin and E.D. Lazowska. Exploiting lookahead in parallel simulation. *IEEE Transactions on Parallel and Distributed Systems*, (4):457–469, oct 1990. URL http://citeseer.nj.nec.com/context/58222/0.

R. A. Meyer and R. Bagrodia. Path lookahead: a data flow view of PDES models. In *Proceedings of the 13th Workshop on Parallel and Distributed Simulation (PADS'99), pages 12-9, 1999*, 1999.

D. M. Nicol and R. M. Fujimoto. Parallel simulation today. *Annals of Operations Research*, (53):249–285, 1994. URL http://citeseer.nj.nec.com/nicol94parallel.html.

H. Ong and P. A. Farrell. Performance comparison of LAM/MPI, MPICH and MVICH on a Linux cluster connected by a Gigabit Ethernet network. In *Proceedings of the 4th Annual Linux Showcase & Conference, Atlanta, October 10-14, 2000*. The USENIX Association, 2000.

Atlanta PADS Research Group, Georgia Institute of Technology. PDNS - Parallel/Distributed NS home page. URL http://www.cc.gatech.edu/computing/compass/pdns.

C. D. Pham. High performance clusters: A promising environment for parallel discrete event simulation. In *Proceedings of the PDPTA'99, June 28-July 1, 1999, Las Vegas, USA*, 1999.

B. R. Preiss and W. M. Loucks. The impact of lookahead on the performance of conservative distributed simulation. In *Proc. 1990 European Multiconference–Simulation Methodologies, Languages and Architectures, pages 204-209, Nuremberg, FRG, June 1990*. Society for Computer Simulation, 1990.

quadrics. Quadrics home page. URL http://www.quadrics.com/.

H. M. Soliman, A. S. Elmaghraby, and M. A. El-Sharkawy. Parallel and distributed simulation: An overview. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC'95), June 27-29, 1995, Alexandria, Egypt*, 1995.

ssfnet. SSFNet home page. URL http://www.ssfnet.org.

A. Varga. The OMNeT++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM'2001). June 6-9, 2001. Prague, Czech Republic*, 2001.

J. J. Wand and M. Abrams. The impact of lookahead on conservative simulation. Technical Report ncstrl.vatech_cs//TR-95-03, Computer Science, Virginia Polytechnic Institute and State University, 1995. URL http://eprints.cs.vt.edu:8000/archive/00000418/.

J. Xu and M. J. Chung. Predicting the performance of synchronous discrete event simulation systems. In *Proceedings of ACM/IEEE International Conference on Computer Aided Design, San Jose 2001, pp. 18-12.*, 2001.

AUTHOR BIOGRAPHIES

András Varga received his M.Sc. in computer science with honors from the Technical University of Budapest, Hungary in 1994. He worked for several years as software architect for Encorus (formerly Brokat Technologies), which has provided distributed application server technologies for financial institutions in Europe and Asia, and now focusing on Internet and mobile payment solutions.

He is the author of the OMNeT++ open-source network simulation tool currently widely used in academic and industrial settings, and founder of Omnest Global, Inc. which provides commercial licenses and services for OMNeT++ worldwide. He is currently working towards PhD, his research topic being large-scale simulation of communication networks. Between February and September 2003 he visited CTIE at Monash University (Melbourne, Australia) to participate in the parallel simulation research project.

Y. Ahmet Şekercioğlu is a researcher at the Centre for Telecommunications and Information Engineering (CTIE) and a Senior Lecturer at Electrical and Computer Systems Engineering Department of Monash University, Melbourne, Australia. He also holds the position of Program Leader for the Applications Program of Australian Telecommunications Cooperative Research Centre (ATcrc, http://www.atcrc.com). He completed his PhD degree at Swinburne University of Technology, Melbourne, Australia (2000), MSc (1985) and BSc (1982) degrees at Middle East Technical University, Ankara, Turkey (all in Electrical Engineering). He has lectured at Swinburne University of Technology for 8 years, and has had numerous positions as a research engineer in private industry.

His recent work focuses on development of tools for simulation of large-scale telecommunication networks. He is also interested in application of intelligent control techniques for multiservice networks as complex, distributed systems.

His e-mail address is : `ASekerci@ieee.org` and his Web-page can be found at http://titania.ctie.monash.edu.au.

Gregory K. Egan's principal research interests are the design, programming and the application of high-performance parallel distributed computer architectures.

He is currently Professor of Telecommunications and Information Engineering, Director of the Centre for Telecommunications and Information Engineering and Head of the Department of Electrical and Computer Systems Engineering at Monash University in Australia.