High Performance Clusters: A Promising Environment for Parallel Discrete Event Simulation

CongDuc Pham
Université Lyon 1, Claude Bernard Bat 710 LIGIM/RHDAC
43 Blvd du 11 Novembre 1918
69622 Villeurbanne Cedex, France

Abstract *Parallel simulation techniques are promising for reducing the simulation time and for increasing the system size as well. Traditionally executed on state-of-the-art parallel machines, parallel simulation can now benefit from the availability of high-performance clusters (HPC) based on commodity hardware. In this paper, we present our experiments with parallel simulations of large-scale communication networks on a Myrinet-based cluster. The model requires high performance communication as a large number of small messages are sent. The speedup results show that HPC can be a serious alternative to parallel machines even for high-demanding communication applications.*

Keywords: High-speed LANs, Myrinet network, Parallel simulation.

1 Introduction

Simulation is nowadays widely used for system prediction and performance analysis tasks. However as the demand for larger and more complex models increases, sequential simulations, i.e. those performed on a single processor, become very time-consuming and appear less attractive than before. In order to reduce the simulation time, and to increase the system size as well, parallel simulation techniques have been developed during the last 2 decades to distribute a simulation over a set of processors. Radically different from simple replications, parallel simulation consists in partitioning a model into a set of small logical processes (LPs). These LPs are executed on different processors and interact each others by exchanging timestamped messages. The main concern for parallel simulation is to maintain the correct execution of events, i.e. in timestamp order, with a synchronization protocol.

Parallel simulation of small grain applications have been traditionally performed on powerful parallel machines because the communication cost between the processors must be kept small in order to obtain significant speedups. Previous experimentations on networks of workstations have been quite disappointing because of the high latencies of the interconnection network [9]. Even high-throughput environments such as ATM-based clusters show disappointing performances regarding the message latency [7]. The availability of high-performance clusters (HPC), at an affordable price, and adequate communication software is a great opportunity for the parallel simulation community to bring these techniques to a larger audience.

In this paper, we present experiments on parallel simulations of communication network models. A Myrinet-based cluster with the BIP communication software is used to perform the parallel simulations. Comparisons with a traditional state-of-the-art parallel machine (Cray T3E) are provided. The remainder of the paper is organized as follows: Section 2 presents the Myrinet cluster and the software environment for high performance communications. Section 3 presents the parallel simulation application. Section 4 shows the experimental results. Conclusion is given in Section 5.

2 The high performance cluster environment

The HPC is based on Myrinet hardware [1] and the BIP (Basic Interface for Parallelism) communication layer [8]. The Myricom LAN was chosen for its performance over the Gbits/s, its affordable price and its software openness. There are several features that makes this kind of technology much more suitable than a typical commodity network based on Fast or Giga-Ethernet:

- the hardware provides an end-to-end flow control that guarantees a reliable delivery and alleviates the problem of implementing in software the reliability on top on an lossy channel. As message losses are exceptional, it is possible to use algorithms that focus on very low overheads in the normal case,
- the interface card has a general purpose processor to handle most of the communication activity without interrupting the main processor. Efficient overlapping of communication/computation and collective operations can then be performed.
- the interface card has up to one megabyte memory for buffers. As the network being as fast as the PCI bus, this memory isolates transfers on the network from transfers on the PCI bus, so that they will not be limited by the slowest one at all times.

The BIP communication layer provides only a protocol with low-level functionalities. Although specialized parallel applications could interface directly with it, it is intended as the base of other protocol layers like IP, and higher level APIs like the well established MPI [4]. Highly optimized, the raw communication performance for BIP is about $5\mu s$ latency one-way. The low latency allows applications to scale more by allowing the use of a finer grain of computation. This is particularly important for parallel simulation of communication networks as will be explained in the next section. The experimental cluster consists of 12 nodes interconnected by three 8-ports Myrinet switches. Each node is an Intel Pentium Pro 200 MHz with 64MBytes of RAM and a 440FX chipsets. The myrinet board contains a LANai 4.1 and 256Kbytes of memory. The test-bed is provided by the LHPC (Laboratoire pour les Hautes Performances en Calcul), a cooperation between ENS-Lyon and Matra Système Information.

3 Parallel simulation of communication networks

3.1 Parallel simulation protocols

We focus mainly on the *discrete event* approach where changes in the system occur at discrete points in the simulated time with the model stepping from one point to another according to the progress of event processing. The parallel approach usually assumes that the system to be simulated can be spatially partitioned into disjoint sub-systems. Each sub-system is simulated by an associated LP on a dedicated processor. To model the interactions that may exist in the real system messages are exchanged and time stamped by the sending LP according to its *Local Virtual Time* (LVT). At the receiving side, they are put in a local *Future Event List* (FEL) to be processed. Other methods to parallelize a simulation exist but their applications are limited [3].

Care must be taken when distributing virtual clocks and event lists among physical processors to ensure that the parallel simulation is still *correct* and produce the same output when compared to a sequential one. In particular, the *causality constraints* that represent the logical evolution of time must be maintained. Unfortunately since the different LPs may advance at different rates synchronization problems are likely to occur. A *causality error* happens when a message arrives at a receiving LP and is outdated, or old, according to its LVT. These problems arise the need of explicit synchronization mechanisms that are traditionally classified in two main categories: *conservative* and *optimistic*.

The conservative approach [2] only processes events that can definitely be considered safe, i.e. that processing the event would not result in further causality violations. Safety is guaranteed by forcing the LP to block if bad decisions could be taken. The advantage is to obtain a correct simulation at any time. Unfortunately, this scheme introduces deadlocks that must be avoided by sending *null-messages*. These messages rely on a *lookahead* ability which allows for an artificial propagation of the simulation time. On the other side, optimistic approaches [5] do not search for safety but provisions are made to *roll back* to an earlier coherent state when they occur. Periodic checkpointing and *anti-messages* to cancel bad computations are then needed as a counterpart of more freedom. In addition, a *Global Virtual Time* (GVT) is required to monitor the simulation progress and to reclaim memory used by obsolete information. In general, an optimistic approach is much more complex than a conservative one but can transparently exploit the parallelism in the system without the explicit determination of a lookahead property.

3.2 The parallel simulation application

The application we consider is a communication network model based on the Asynchronous Transfer Mode (ATM) technology. The network model we consider consists of 78 ATM switches and traffic sources. The sources periodically open and close connections to random destinations. Cells are sent by traffic sources and are transfered to the destination by the ATM switches. The system under study includes a routing algorithm with the introduction of link cost and dynamic routing functions that provide load-balancing routing features. The simulation of a routing algorithm implies to simulate (*i*) the mechanism that consists in constructing and updating the routing tables and (*ii*) the flow of cells that is handled by the network [6]. The parallelization process of the network model consists in spatially partitioning the entire network in n regions and to assign each region to an LP on a processor. Cell transfers from one region to another are represented by timestamped messages exchanged between the LPs. The message synchronization is performed conservatively, because the application exhibits good lookahead with the link propagation delay.

The life cycle of the simulated objects in the parallel simulator basically consists in processing internal messages and generating messages. Internal messages are taken from a global event list and processed in timestamped order. Generated messages that have to go off the processor are sent to the destination using the communication system, otherwise they are directly inserted as new internal events. The destination is identified by a couple (*node*, *ic*) where *ic* is the logical input channel for the receiver. External messages are received in the communication layer buffers and periodically inserted in the local global event list to become internal events. The *ic* value of each message is used to compute a safe window for event processing, according to the conservative rules. When an LP has no more eligible internal messages to processed, i.e. no more messages or it has reached the upper bound of the time window, it would send out a null-message and would get all external event from the communication buffers.

From the parallel simulation perspective, this application presents a very small granularity (processing time of an event) and requires a lot of small messages to be exchanged between processors (as opposed to more traditional high-performance computing applications where the computation part is larger). It is not unusual to send several millions of messages per minute of wall clock time! These messages are either model messages, e.g. cell exchanges between ATM switches, or synchronization messages, e.g. null-messages from the conservative kernel. Therefore the application typically requires very low latencies from the communication system.

4 Experimental results

The network model is simulated for 500000 time slots that represent 0.31s of the real system. More than 50 millions of events are simulated. In order to compare with parallel computers, previous results obtained on a Cray T3E are presented.

4.1 Previous experiments on Cray T3E

The Cray was provided by the *Institut du Développement et des Ressources en Informatique Scientifique* (IDRIS) and consists of 256 processing nodes interconnected by a very low latency 3D torus. Each node on the Cray is a DEC Alpha EV5 with 128 MBytes of memory running at 300Mhz. The communication libraries are the native SHMEM (SHared MEMory) and MPI. The first one provides a latency of approximately $7\mu s$ while MPI shows a latency of $13\mu s$. Table 1 summarizes the results.

Table 1: Cray T3E with SHMEM and MPI.

np.	*time SHMEM (s)*	*speedup SHMEM*
1	488	-
4	131	3.72
8	72	6.77
16	46	10.60
32	33	14.78
64	31	15.74
np.	*time MPI (s)*	*speedup MPI*
1	488	-
4	253	2.07
8	133	3.66
16	82	5.95
32	59	8.27
64	55	8.87

On the Cray T3E, the SHMEM library yields excellent speedups that are almost linear on 4 and 8 processors. The speedups obtained with MPI are smaller because of the larger latency introduced by the additional software layer. A maximum speedup of 15.74 with SHMEM can be achieved with 64 processors . Using so much processors is an extreme case and no significant improvements have been obtained with more than 32 processors. With so many processors, the proportion of messages that go off the local processor dramatically increases and synchronization overhead limits the speedup. The fact that the speedup does not decrease either must be related to the very low cost of the Cray's communication network. In a slower communication architecture, using so many processors would certainly decrease the speedups dramatically. The most difficult problem we had to face was the load-balancing issue. Practically, as systems with a large number of processors and very low communication cost are not accessible to the majority of researchers, parallel simulations are likely to be executed on a smaller number of processors with higher communication costs. In this case, it is certainly better to use a small number of processors with optimal partitioning and load-balancing features.

4.2 Experiments on the Myrinet cluster

The Myrinet cluster is the one described previously. The latency for sending a 70-bytes message (the size of a message of in our application) is about $10\mu s$ with BIP and $17\mu s$ with MPI/BIP. Table 2 summarizes the speedup results with MPI/BIP and shows the number of remote messages sent for a given run. Only 8 PCs have been used so far.

Table 2: Myrinet with MPI/BIP.

np.	*time*	*speedup*	*RM*
1	333	-	0
2	228	1.46	710,905
4	141	2.36	2,304,835
6	98	3.39	2,845,867
8	104	3.20	3,418,919

If we compare the results obtained on the Myrinet test-bed to those obtained on the Cray, we can see that the sequential version run faster on a Pentium Pro 200MHz than on a DEC Alpha EV5 300MHz. The main advantage of using standard products is the availability of the most recent processors. As we stated before, parallel computers were interesting because they typically use customized high performance interconnection networks. The low latency on the Cray has a direct impact on the performances of the simulator. However the results on the Myrinet cluster are very encouraging. Using MPI/BIP shows very interesting speedups (close to those obtained on the Cray T3E with MPI) compared to the cost of the hardware used. We haven't been able to test the parallel simulation on more than 8 nodes so far (4 nodes on the 12 available were reserved for special use).

Regarding scalability when the problem size increases, preliminary experiments with a network model of 156 switches have shown that the communication sub-layer must be scaled accordingly (the sequential version also requires a machine with 120 MBytes of memory whereas the parallel version could use 64 MBytes nodes). For instance, if we keep the same number of processors for a larger problem size, e.g. more ATM switches, the number of messages sent from one processors to another can increase dramatically. At the receiving side, the MPI reception buffers could be rapidly saturated resulting in a tight flow control and additional buffering at the sender side that can dramatically increase the simulation time. This observation on a 156 switches model lead us to think that the same problem may have happen in the 78 switches model as well without being noticed. Therefore we started looking at the MPI layer a little bit closer. The results presented in table 2 have been obtained with 32 reception buffers per processor instead of 6 by default. Figure 1 illustrates the evolution of the simulation time as the number of reception buffers is varied. We could not increase this number to more than 32 buffers when a maximum of 8 processors are used and to more than 64 when a maximum of 4 processors are used because of memory limitations. We can see that the execution time on a small number of processors (2 and 4) can decrease significantly as the number of reception buffers increases. The blocking effect of the flow control explains why the 4 nodes version runs faster on the cluster than on the Cray (we could not tune the MPI implementation on the Cray) although the overall latency is larger on the former architecture. However, for a network model with 78 nodes, using 16 or 32 reception buffers shows little improvements. For 6 and 8 processors, the number of reception buffers has a small impact on the simulation time as the number of messages sent per processor is much smaller. This allows for a comparison between the cluster and the Cray, even if we could not tune the MPI layer on the Cray.

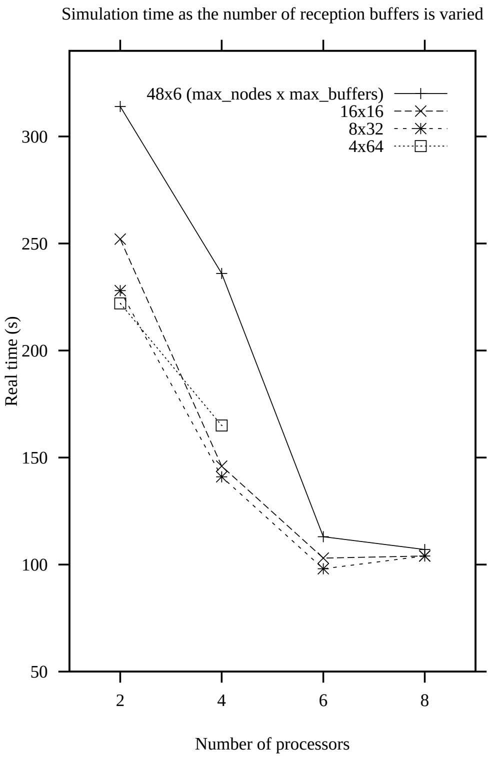

Figure 1: Simulation time as the number of reception buffers is varied.

Once the number of reception buffers has been increased to 32, we can see that using 8 processors is a little bit slower than using 6 processors (cf. table 2). The reason is that using 8 processors adds more communication overheads. We said previously that the speedup progression on the Cray could not be reproduced on a slower communication architecture. These figures show that the 8 processors case with the latencies of MPI/BIP adds too much communication overheads. Of course, if the problem size is increased the optimal number of processors should be greater. With 64 reception buffers, we could only test with 2 and 4 nodes. On 2 nodes, figure 1, the 64 buffers case in a little bit faster than the 32 buffers case (222s vs. 228s). However, for 4 nodes, using the 64 buffers case is much slower that the 32 buffers case (165s vs. 141s). This is a phenomenon that we could not explain for the moment.

Since the MPI functions we used are relatively simple (basic send and receive, simple synchronization barrier), we are currently porting the communication layer to use native BIP instead of MPI on top of BIP. This would in the best case scale down the latency by about 1.6. Early tests with BIP have shown that it is not trivial for the application to get a benefit from the theoretical smaller latency. This is because we had to add a custom flow control for BIP and packing/unpacking primitives. Even though the scale in latency is about 1.6, we could only get for the moment 1.15 for the application. Future works will focus on optimizing these components and on the scalability of cluster computing (both in term of problem size and cluster size) but we have no doubt that this kind of architecture is already very promising because of their excellent performance/price ratio.

5 Conclusions

HPCs represent a serious alternative to expensive parallel computers. In this paper, we mainly focus on a Myrinet-based cluster with the BIP software suite for optimized low-level communications. Although the technology has been ready for some time, it is still used only in a confidential manner outside of the high-speed network research community. One of the principal aims of this paper is to show through a parallel simulation application the maturity of the more recent technologies. Generalizing the use of HPCs will provide to the end-user a way to reach a higher level of scalability, and to make parallel solutions usable for a wider range of applications, including those that exhibit fine grain computation.

References

[1] Boden, Cohen, Feldermann, Kulwik, Seitz, Seizovic, and Su. MYRINET: A Gigabit per second Local Area Network. *IEEE-Micro*, 15:29–36, February 1995.

[2] Chandy, K., Misra, J.: Distribution Simulation: A Case Study in Design and Verification of Distributed Programs. *Trans. on Soft. Eng.*, **5**(5) 440–452.

[3] Ferscha, A., Tripathi, S. K.: Parallel and Distributed Simulation of Discrete Event Systems. CR-TR-3366, Dept. of Comp. Science, Univ. of Maryland.

[4] Gropp, W., Lusk, E., Skjellum, A.: Using MPI: Portable Parallel Programming with the Message-Passing Interface. *MIT Press*, Cambridge, MA, 1994.

[5] Jefferson, D. R. : Virtual Time. *ACM Trans. on Prog. Lang. and Sys.*, **7**(3) (July 1985) 405–425.

[6] Pham, C. D., Brunst, H., Fdida, S.: Conservative Simulation of Load-Balanced Routing in a Large ATM Network Model. *In Proceedings of PADS'98*, Banff, Canada, pp142-149.

[7] Prylli, L., Tourancheau, B.: Parallel computing on an ATM LAN. *In IFIP ATM97*, Bradford, UK, 1997.

[8] Prylli, L., Tourancheau, B.: BIP: a new protocol designed for high performance networking on myrinet. *In Parallel and Distributed Processing, IPPS/SPDP'98*. **1388** of *LNCS*, pages 472–485. Springer-Verlag.

[9] Tallieu, F., Verboven, F.: Using Time Warp for Computer Network Simulations on Transputers. *In Proc. of the 24th Annual Simulation Symposium*, New Orleans, 1991, pp112-117.