Exploiting Clusters of Shared Memory Multiprocessors with BIP-SMP: the Parallel Simulation Application

P. Geoffray C. D. Pham B. Tourancheau

RHDAC, CNRS-INRIA ReMap Project, LHPC
Université Claude Bernard Lyon
69622 Villeurbanne, FRANCE
e-mail: bip-team@rhdac.univ-lyon1.fr

Abstract

Parallel simulation of small grain models have traditionally been performed on powerful parallel machines because the communication cost between the processors must be kept small in order to obtain significant speedups. As we approach the next century, parallel machines are gradually and incrementally being replaced by clusters of commodity workstations. Clusters of SMPs (CLUMPS) machines appear very attractive because of their high performance/price ratio and will certainly be one of the most demanding architecture. To achieve the maximum of performances, the communication layer needs to provide a **multi protocol** support, thereby allowing several processes per node to simultaneously send a message to another process on the same physical node and to send a message via the network. In this paper, we present the BIP-SMP software for exploiting CLUMPS and how parallel simulation applications with challenging communication properties can benefit from BIP-SMP.

1 Introduction

Simulation is a performance prediction tools commonly used for analysis tasks. As models tend to be larger and more complex, sequential simulations, i.e. those performed on a single processor machine, become very time-consuming. In order to reduce the simulation time, and to increase the system size as well, parallel simulation techniques have been developed 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 models 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. As we approach the next century, parallel machines are gradually and incrementally being replaced by clusters of commodity workstations as the result of dramatic improvements in hardware and networks technologies. To build an efficient cluster, it is necessary to mix three ingredients: (*i*) powerful computers, (*ii*) high-performance interconnection networks and, (*iii*) high-performance communication software.

There are several candidates for the interconnection networks and some of them are now well known (Myrinet, Gigabits Ethernet). Regarding the computing nodes, shared-memory multi-processors desktop machines (SMPs) with 2 or 4 processors are also becoming very popular. Therefore, there is a great opportunity to take into account these architectures that presents a high performance/price ratio. The importance of the last ingredient should not be forget: to have the maximum of performances the communication layer needs to provide a **multi protocol** support, thereby allowing several processes per node to simultaneously send a message to another process on the same physical node and to send a message via the network.

In this paper, we present the BIP-SMP software for exploiting CLUMPS and how parallel simulation of models with challenging communication properties can benefit from BIP-SMP. Section 2 presents the target architecture and the BIP software upon which BIP-SMP relies for external communications. Section 3 presents the main features of BIP-SMP and some related works. Section 4 describes the parallel simulation application and Section 5 presents the performance obtained with BIP-SMP on a CLUMPS.

2 The target architecture

2.1 Hardware components

The Myricom LAN[1] target was chosen for its performance over the Gbits/s, its affordable price and its software openness (all the software and specifications are freely available for customers). There are several features that make this kind of technology much more suitable than a traditional commodity network:

- the hardware provides an end-to-end control flow 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 which is powerful enough to handle most of the communica-

To appear in the ACM International Conference on Supercomputing (ICS'99), June 1999, Rhodes, Greek.

tion activity without interrupting the main processor. In particular, it provides an efficient overlapping of communication and computation.

- the interface card has up to one megabyte memory for buffers. As the network being as fast as the computer bus, this memory isolates transfers on the network from transfer on the bus.

The experimental platform is a cluster of 4 commodity SMPs interconnected by a Myrinet network. One Myrinet network interface card (NIC) uses a PCI slot to connect a node to a Myrinet switch. The bandwidth provided by the hardware is approximately 160 MBytes/s, but the PCI bus limits the maximum throughput at 132 MBytes/s. Each node of this cluster is a dual Pentium II at 450 MHz compliant with the Symmetric Multi Processor interface from Intel. Each contains a total of 128 MBytes of main memory. The motherboard is based on the BX chipset which allow us to use a 100MHz memory bus. The operating system is Linux. It is worth to mention that all of the hardware in this cluster is composed of off-the-shelf components.

2.2 Software architecture

A very efficient software architecture is needed to fully exploit the potential of the hardware. Previous communication layers were usually designed for low performance unreliable networks. Using a generic software over a very efficient cluster is a common error and gives an incorrect impression of the cluster performance. We present in this section the BIP software and a port of MPICH over BIP. BIP is used by BIP-SMP for all the external traffic that should go off the SMP on the Myrinet network.

2.2.1 BIP

BIP [8] stands for Basic Interface for Parallelism and is available on Linux systems. The idea was to build it with a library interface accessible from applications that will implement a high speed protocol on the Myrinet network. BIP 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 and PVM. BIP provides both blocking and non-blocking communication. Communications are as reliable as the network, errors are detected and in-order delivery is guaranteed. BIP is composed of a user library, a Linux kernel module and a Network Interface Card (NIC) program.

The communication latency of BIP is 6 μs, and the maximal bandwidth is 126 MB/s (95% of the theoretical hardware maximum, actually represented by the PCI bottleneck in our case). Half of the maximum bandwidth is reached with a message size of 4 KB ($= N_{1/2}$).

2.2.2 MPI-BIP

MPI-BIP [11] is an implementation of MPICH [4] over BIP. The implementation uses the BIP's API at an intermediate level, above the Channel Interface and below the Abstract Device Interface. The cost of MPI-BIP is roughly a constant overhead of 2 μs for the latency. The latency of MPI-BIP is about 8 μs and the bandwidth is limited to 70 MBytes/s because of the internal fragmentation inside MPICH.

3 The multi-protocol communication layer: BIP-SMP

BIP provides almost all of the hardware performance at the user level but the constraints are: single user, no fault tolerance, basic services, mono process, etc. These constraints can be accepted if one wants high-performance computing on a dedicated cluster. However, in the context of SMPs, BIP is not able to exploit all of the hardware performance as only one process per node is able to gain access to the Myrinet board while the other processors must remain idle for communication.

A first solution is to use the other processors with BIP threads. Communications between threads use simple memory copies and communications between nodes are performed with the classical BIP strategy [3]. However, this solution forces the use of the multi-thread paradigm for communications so we chose to keep the message-passing semantic. Providing the support of several processes per node with the same performance as multi-threaded programming presents the following difficulties: manage the concurrent access to the hardware and the Myrinet board, and provide local communication with the same level of performance as BIP over Myrinet.

3.1 Handling the concurrent accesses

When a BIP process wants to send a message to another BIP process, it starts by asking for an entry in a queue of mapped send requests. A send request is a structure containing all of the information required to process the sending. This queue maps send requests on the Myrinet memory and when a structure is filled, the NIC completes the communication. For a small message the processor writes the data directly in the send request, otherwise it writes the physical addresses of memory pages used by the data. A DMA engine on the Myrinet board then gathers the data to be sent on the network. Once a send request is filled, all of the rest of the communication is asynchronously managed by the LANai (the embedded processor on the Myrinet board). The best solution is to protect the concurrent access to this send request's queue by a lock, and of course, the lock needs to be very efficient. With BIP-SMP, two processes can then overlap the filling of a send request and the only operation that is locked is the entry in the send request's queue. This strategy is similar to the mechanism for the concurrent access implemented in Active Messages [5].

Althought several Myrinet interfaces per node can be supported by BIP-SMP, the bottleneck is the PCI bus that is usually unique in the regular SMP machines.

3.2 BIP-SMP main features

There are several ways to move data from one process to another process. One solution is to use shared memory to implement mailboxes. One can also move data directly from user space to user space. For efficiency reasons, we will use both mechanisms: the shared memory to move small buffers, with two memory copies but small latency; and the direct copy for large messages, with a kernel overhead but a high sustained bandwidth.

An innovation in this communication layer is the direct memory copy. For instance, moving data between threads can just be a simple memory copy but this is impossible between processes. Unix systems protect different user spaces and forbid access to foreign virtual memory addresses. In order to enable direct memory copy, we have implemented a Linux kernel module to move data from a user space to another user space. This module needs to know the virtual address and the length of the source buffer, and the virtual address of the receive buffer in the other process.

Having inter-node and intra-node communication primitives is not enough. Another part of this work is to enable BIP-SMP to simultaneously use both and to hide this new feature in BIP-SMP's API. We have chosen to use two receive queue pools per node: one for the internal communication and the other one for Myrinet. One reason for this choice is to allow the receipt of a message from the Myrinet network and the receipt of a message from another process in the shared memory at the same time, without a synchronization between them.

The use of BIP-SMP is completely transparent, as each process receives a different logical number. When a message is sent from a logical node to another logical node, BIP-SMP automatically determines whether to use the shared memory or the Myrinet network. For message reception, BIP-SMP polls both internal and external receive queues. Flags that indicate a new message arrival are in the same cache line thus minimizing the overheads. This transparency is extended to MPI-BIP to provide a uniform high performance message-passing interface. Of course, if users want to exploit the asymmetric performance between shared memory and Myrinet, BIP-SMP variables are available to provide the information about the localization of the other logical nodes, the number of processes on the same physical node, etc.

3.3 Related works on SMP support

Efficient management of CLUMPs is a very active research topic and several project have been lunched to address this issue. Most of them have investigated issues related to a multi-protocol message passing interface using both shared memory and the network within a CLUMP. Multi-Protocol Active Messages (MPAM) [6] is a very efficient implementation of Active Messages [10] using Myrinet with SMP Sun Enterprise 5000. The main restriction is the use of the Sun Gigaplane memory system [9] instead of a common PC memory bus. The polling is also a problem in MPAM, as polling for external messages is more expensive than for internal messages. However, MPAM is the first message passing interface to efficiently manage CLUMPs. Another project, closely related to our solution, comes from MPICH-PM/CLUMPs [7] and can be used on commodity SMP nodes. Finally, one of the first message-passing interfaces to manage CLUMPs as a platform is the well-known device P4 [2] with MPICH. P4 provides mechanisms to start multiple threads on machines and uses either message passing or shared memory copies to communicate between these threads. However, the programmer must explicitly select the appropriate library call.

4 The parallel simulation application

4.1 Backgrounds

Parallel discrete event simulation usually assumes that the system to be simulated can be spatially partitioned into disjoint sub-systems. Each sub-system is simulated by an associated *Logical Process* (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.

Care must be taken when distributing virtual clocks and event lists among physical processors to ensure that the distributed 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 [12] 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 [13] do not search for safety but provisions are made to *roll back* to an earlier coherent state when they occur. Periodic check-pointing 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.

4.2 The communication network model

As the demand for broadband communication is becoming larger and larger it is very urging to offer high-speed backbones to provide sufficient bandwidth for a various number of multimedia applications such as video-conferencing and TV distribution. Although simulations can be carried for small or specific parts of the network (e.g. switch element for switch architecture design), the challenge is now to simulate it in one piece.

We started our project 4 years ago and the goal was to study complex ATM network models. One application is the simulation of the routing algorithm described in [14] with connection management and dynamic routing facilities. Tests are performed to study the recovery on failure properties of the routing algorithm under extreme conditions, e.g. when failures occur more frequently than normally assumed. Performing at the cell-level gives the possibility to inspect the smallest interactions between network components but is very time-consuming.

Nowadays, ATM can be found in Local Area Networks (LANs) and starts to appear in Metropolitan Area Networks. The next step is to develop it for large scale configurations. The application 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 [15]. 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.

4.3 Properties of the models

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 local 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. When an LP has no more eligible internal messages to process, 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 events from the communication buffers.

From the distributed simulation perspective, this model has several challenging properties. First, the simulation of a shortest-path routing algorithm on a general topology creates disparities between links and many of them may not be often used. Therefore the conservative simulator has to generate a lot of null-messages to propagate the simulation time on these channels. We think that this situation is a highly realistic one. The second property is a consequence of the first: bottlenecks can appear in the simulation by concentrating a large part of the messages through a given processor. The difficulty resides on the very dynamic nature of load repartition making optimal data distribution almost impossible. Third, the synchronous behavior of the application regarding message retrieval can trigger a tight flow control from the communication layer. Finally, simulating high-speed networks at the cell-level produces a very low granularity and requires a large number 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! Depending on the number of processors used, these events must be sent over the interconnection network with high communication overheads.

Parallel simulation of communication networks is therefore an area where high-performance communication is essential to obtain interesting speedups. In addition, it is desirable to simulate a large number of nodes making scalability an important issues. Our direction towards cluster of SMPs is motivated by these 2 factors, with also by the desire to achieve high-performance on more accessible parallel architectures. BIP-SMP offers several features that make it attractive for us: (i) it completely hides the underlying architecture and no modifications of the simulation code is requires, (ii) it offers very low latencies and contention overheads between the local processors and, (iii) the number of physical processors is increased very simply without any compromise on the performances.

5 Experimentations

Experiments are performed on the test-bed described previously (dual Pentium Pro, 450MHz). The parallel simulations are run for 500,000 time slots that represents 0.31s of the real system. More than 53 millions of events are simulated. MPI/BIP-SMP is used in all the simulation runs. The tests are performed by incrementing the number of processors. Also, message aggregation is added to increase the computation grain. Asymmetric message aggregation between external and internal nodes is also investigated. We will indicate message aggregation with *Aggr.* followed by 2 values: the first one is the maximum aggregate size in bytes for external communication; the second one is for internal communication. One exception is for the 2 processors case where only one value will be given, for external or internal aggregates. For experiments with only external nodes, only the first value is relevant. To compare the performance of the parallel simulator, the sequential version shows an execution time of 121s.

5.1 2 external nodes vs 2 internal nodes

The following test consists in comparing the simulation time between 2 external nodes and 2 internal nodes on an SMPs machine. With 2 external nodes, only one processor on the two available is used. Table 5.1 shows the execution time of the simulation and the corresponding speedups.

	2 ext.		2 int.	
strategy	time	speedup	time	speedup
No Aggr.	129	0.93	104	1.16
Aggr. 156	92	1.31	78	1.55
Aggr. 256	85	1.42	77	1.57
Aggr. 512	76	1.59	76	1.59
Aggr. 1024	76	1.59	76	1.59

Table 1: 2 external nodes vs 2 internal nodes

Without aggregation, using 2 external processors gives a slow-down because of the high communication overheads without significant data distribution improvements. With 2 internal nodes, the latency goes down from about $7.5\mu s$ to about $2.5\mu s$ making speedups achievable. Previous experiments with 2 external nodes on slower CPU had given speedups because of the higher time taken by the sequential simulation. With a Pentium II running at 450MHz, this is not the case anymore. As the aggregation size increases, the simulation time can decrease until a threshold is reached. At this point, the application behavior limits the benefit of the aggregation. This threshold appears since 512 bytes for 2 external nodes and since 156 bytes for 2 internal nodes. The difference in the threshold is explained by the fact that message aggregation on external nodes is still beneficial until 512 bytes.

5.2 4 external nodes vs 2x2 internal nodes

The following test consists in comparing the simulation time between 4 external nodes and 2 SMPs with the 2 internal processors active. With 4 external nodes, only one processor on the two available per machine is used. Table 5.2 shows the execution time of the simulation and the corresponding speedups.

4 external nodes now show little speedup as data distribution is improved. Without aggregation better speedup can be obtained using SMPs features. Then aggregation makes the two versions equivalent. We think that the application behavior limits the speedups to the slowest link.

By using asymmetric aggregation, we can reduce the simulation time. Using the optimal threshold obtained for

	4 ext.		2x2 int.	
strategy	time	speedup	time	speedup
No Aggr.	92	1.31	73	1.65
Aggr. 156-156	71	1.70	64	1.86
Aggr. 256-256	63	1.92	63	1.92
Aggr. 512-512	62	1.95	62	1.95
Aggr. 1024-1024	62	1.95	63	1.92
Aggr. 256-156	-	-	59	2.05
Aggr. 512-156	-	-	**56**	**2.16**
Aggr. 1024-156	-	-	59	2.05
Aggr. 512-256	-	-	61	1.98

Table 2: 4 external nodes vs 2x2 internal nodes

each case gives the optimal asymmetric aggregation strategy. This is explained by the fact that waiting overheads are minimizing.

5.3 4x2 internal nodes

The following test uses 4 SMPs machines with the two internal processors active (total 8 processors). Table 5.3 shows the execution time of the simulation and the corresponding speedups.

	4x2 int.		**156 sw.** 4x2 int.	
strategy	time	speedup	time	speedup
No Aggr.	48	2.52	150	2.14
Aggr. 156-156	39	3.10	85	3.77
Aggr. 256-256	33	3.66	78	4.11
Aggr. 512-512	32	3.78	72	4.45
Aggr. 1024-1024	32	3.78	**71**	**4.52**
Aggr. 256-156	34	3.55	73	4.39
Aggr. 512-156	32	3.78	73	4.39
Aggr. 1024-156	32	3.78	74	4.33
Aggr. 512-256	32	3.78	73	4.39

Table 3: 4x2 internal nodes

Using asymmetric aggregation does not show any improvements in this case. We obtain the same result with a larger network model containing 156 switches. However in this case, the speedup increases to 4.52 as the sequential time is now 321s.

6 Conclusions

We have experimented a parallel simulation application with intensive high communication needs with BIP-SMP on a cluster of multiprocessors machines. The results show that BIP-SMP can efficiently exploit the SMP machines by providing transparency and high performance for the application. The main advantage for parallel simulations to use BIP-SMP is to benefit directly from the available dual processor machines without any significant contention on the network interface while taking advantage of the very low latency between internal nodes.

Preliminary experiments for exploiting the asymmetric performance of BIP-SMP with the parallel simulation application show encouraging results on some specific cases. It is still difficult to say whether it would be possible to fully exploit the asymmetric latency from the application. Future works should focus on scalability issues of BIP-SMP.

References

[1] Nanette J. Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawik, charles L. Seitz, Jakov N. Seizovic, and Wen-King Su. Myrinet - a gigabit-per-second local-area network. In *IEEE Micro*, volume 15, February 1995.

[2] Ralph M. Butler and Ewing L. Lusk. Monitors, messages, and clusters : the p4 parallel programming system. Technical report, University of North Florida and Argonne National Laboratory, 1993.

[3] Franck Capello and Olivier Richard. Performance characteristics of a network of commodity multiprocessors for the nas benchmarks using a hybrid memory model. Technical Report 1197, LRI, Université Paris-Sud, 91405 Orsay cedex, FRANCE, 1999.

[4] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–828, September 1996.

[5] Steven S. Lumetta and David E. Culler. Managing concurrent access for shared memory active messages. In *IPPS*, Orlando, Florida, April 1998.

[6] Steven S. Lumetta, Alan M. Mainwaring, and David E. Culler. Multi-protocol active messages on a cluster of smp's. In *SuperComputing (SC'97)*, University of California at Berkeley, August 1997.

[7] Parallel and Japan Distributed System Software Laboratory,
Real World Computing Partnership. Mpich-pm/clump : an mpi librairy based on mpich 1.0 implemented on top of score, 1998. http://pdsw3.trc.rwcp.or.jp/mpich-pm/.

[8] Loïc Prylli and Bernard Tourancheau. BIP: a new protocol designed for high performance networking on myrinet. In *Workshop PC-NOW, IPPS/SPDP98*, Orlando, USA, 1998.

[9] A. Singhal, D. Broniarczyk, F. Cerauskis, J. Price, L. Yuan, C. Cheng, D. Doblar, S. Fosth, N. Agarwal, K. Harvey, E. Hagersten, and B. Liencres. Gigaplane : A high performance bus for large smps. In *Hot Interconnects IV*, pages 41–52, Standford, California, August 1996.

[10] T. von Eicken. *Active Messages: an Efficient Communication Architecture for Multiprocessors.* PhD thesis, University of California at Berkeley, November 1993.

[11] Roland Westrelin. Réseaux haut débit et calcul parallèle: étude de myrinet. Master's thesis, LHPC, CPE-Lyon, 1997.

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

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

[14] Jaffe, J. M., Moss, F. H. : A Responsive Distributed Routing Algorithm for Computer Networks. IEEE Trans. on Comm., **30**(7), 1758–1762.

[15] 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.