Performance evaluation of two programming models for a cluster of PC biprocessors

Franck Cappello
LRI
Université Paris-Sud
Orsay, France

Olivier Richard
LRI
Université Paris-Sud
Orsay, France

Daniel Etiemble
LRI
Université Paris-Sud
Orsay, France

Abstract *We evaluate performance on NAS benchmarks of a cluster of 2-way SMP PCs connected by a Myrinet network. Two programming models are considered. A Single Memory Model uses the MPI-BIP/CLUMP library and gives the programmer a unified memory view. A Hybrid Memory Model combines shared memory model within nodes and message passing between nodes. We compare 2-way SMP configurations speed-up versus single CPU configurations for each model. Better model depends on the features of the applications. We then show that HMM model gives performance close to performance of scalable high-end multicomputers for 8-node configurations. The overall performance evaluation shows that 2-way SMP nodes are a cost-effective solution for clusters of PCs.*

Keywords: CLUster of MultiProcessors, unifed and hybride memory models, performance evaluation, NAS benchmark

1 Introduction

Most off the shelf CPUs, including x86 microprocessors have multiprocessing support. CPU manufacturers deliver multiprocessor motherboards and 2-way SMP PCs are only slightly more expensive than single CPU PCs. With a high speed network such as Myrinet, CLUsters of PC MultiProcessors (CLUMPs) look more and more a promising solution for low cost and high performance parallel machines.

To evaluate the interest of CLUMPs, two main issues must be addressed: the programming model and the performance compared to specifically designed high-end scalable parallel supercomputers. CLUMPs mix two memory models: each node is an n-way SMP (shared memory) and communication between nodes use the "message-passing" hardware. Several programming models can be used and the trade-off between programming complexity for the user and performance must be carefully considered.

In section 2, we present the programming models for CLUMPs. Section 3 gives more details on the experimental platform and on our implementations of the Single and Hybrid memory models.

Section 4 compares speed-ups for a multiprocessor node versus a single processor node for each memory model. We compares performance of the platform with performance of to day scalable multicomputers with the same number of nodes. Section 5 summarizes the results.

2 CLUMP programming models

There are two basic approaches according to the programmer view of memory in a CLUMP: *A single memory model (SMM)* and *An hybrid memory model (HMM).*

In SMM approach, the programmer only sees one memory model. A mechanism is provided to unify the memory model. Some message-passing libraries have been upgraded to work with CLUMPs [1] [2].

Shared virtual memory environments (DSVM) provide another alternative to unify the memory models, as presented in [4][5]. Performance of the DSVM model mainly relies on the efficiency of the memory consistancy protocol that is used.

Recently, OpenMP [7] has been implemented on a network of workstations on top of the Treadmark DSM system. Again, its performance relies on the shared virtual memory software.

The HMM approach has been the first one to be used with CLUMPs. A 1995 workshop has been dedicated to HMM [8] in the context of Parallel Vector Processors (PVPs). In[9], a hybrid shared memory/distributed memory model is proposed for CLUMPs. Communications between nodes use message-passing and remote memory operations. The model is based on SPMD paradigm. The programmer first distributes the data set between the different nodes. Within each node, each portion of the distributed data set is partitioned among the threads.

3 Methodology

3.1 The experimental platform

The experimental platform consists of a Myrinet network with 8 ports. It uses 400-MHz Pentium II 2-way SMP PCs. Each Myrinet PCI interface has a 1-MB local memory.

The software environment includes Linux 2.2.1, BIP 0.95d version of MPI library, F77 PGI 3.0 programming environment and Linux Pthread library. With BIP on Myrinet connected PCs, we have a latency of 5 us and a bandwidth of 1 Gbit/s. MPI-BIP reaches 20 us (latency) and 1 Gbits/s (bandwidth). All benchmarks have been compiled with the o2, unroll and P6 options.

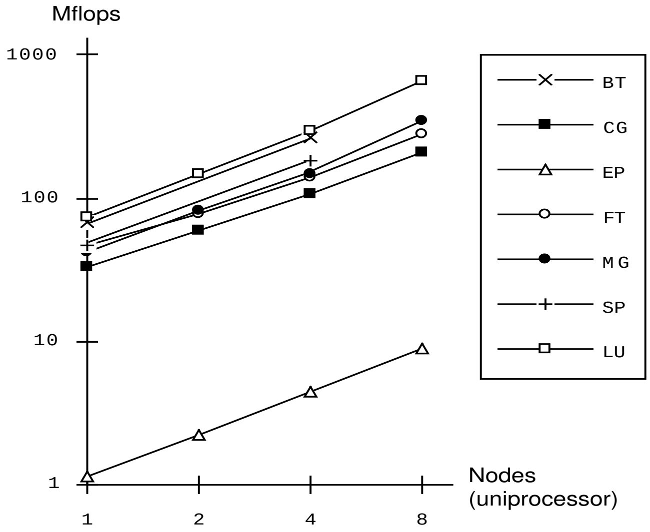

Figure 1: *Uniprocessor performance scaling of a network of Pentium II 400 Mhz PC for the NAS benchmark.*

Figure 1 presents how the performance scales in the platform according to the number of nodes for the NAS benchmarks and using only one processor per node. Preliminary measurements on a larger platform show linear speed-ups up to 32 nodes.

3.2 HMM implementation

Our HMM implementation uses MPI for message passing between nodes and OpenMP for shared memory inside the nodes. Parallelizing an application for the message passing following the SPMD paradigm often produces a program with the typical layout presented in figure 2. The program starts initializing the communication system. Then it performs some local computations and calls some

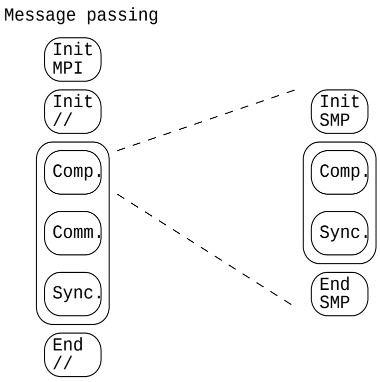

Figure 2: *Parallelizing the MPI code.*

communication subroutines to split the data sets among the nodes participating to the application. The program continues executing a main block typically containing a loop nest. The main block is designed for parallel execution. The body of the loop nest can be described as a succession of three sections: local computations, communications, and synchronization. The final part of the program gathers individual partial results and computes the final result.

Within each node, the program is executed inside a process. For most of the NAS programs, we have parallelized the computational section of the main block. The intra-node parallel execution is performed by a group of threads within the same process. The parallelization directives are simply applied on parallelizable loop nests of the original MPI code. A 2-way SMP node with multi-threaded execution of a fraction of the MPI code just looks like a uniprocessor node for the other nodes.

3.3 SMM implementation

Our SMM implementation uses the MPI-BIP library [10]. In MPI-BIP, many features have been suppressed to get a high-speed communication library.

Specifically, the OS no longer manages the network interface. Only one process can access and use this interface. For multiprocessors, the consequences are important as several processors are sending and receiving communications. We implement the mt-BIP solution in which several execution threads of a process share the same network interface. With this approach, the interface sharing and message multiplexing and demultiplexing can be implementing without significant modifications in the library. figure 3 summarizes the method. Communications between different nodes use the standard BIP mechanisms and a specific BIP library (local-BIP) is used for SMP communications. This library has the BIP semantics and implement local communication through shared memory.

We outline the four specific features of this implementation:

1) To keep the MPI program code, the API should remain similar for local and remote transfers. Another layer is introduced between MPI and BIP that specify local or remote transfer for each communication, according to the target address. Figure 3 shows the new communication stack for MPI-BIP/CLUMP.

2) Two techniques are used for the local copy between emitter buffer and receptor buffer. Up to 256 byte message, an optimized memcpy() function is used with the REP MOVS x86 instructions. For larger message, the Myrinet DMA is used.

3) Third point deals with multiplexing and demultiplexing of messages between nodes for the execution threads of a process. Lower layer of MPI-BIP uses three receptor queues (short, long and control message) for each process. To multiplex and demultiplex, we multiply the number of queues by the number of execution threads (each thread has the three different types of queue).

4) MPI programs are executed through processes. With our approach,

several instances of the same program are executed in a single process by several threads. As the original MPI program can have global variables and/or non-reentrant functions, the program should be modified. We have developed an automatic tool to modify the object code of MPI executables such as they could execute properly with execution threads sharing the same protection domain. The tools mainly duplicate symbols and functions, except for the functions of the standard library, which are reentrants.

Normally, the techniques that have been used for designing MPI-BIP/CLUMPs from MPI-BIP could be used for any high speed communication library that has been designed for uniprocessor nodes (no sharing of the network interface between processes).

4 Performance

In this section, we compare a parallel platform based on single CPU nodes with a parallel platform based on 2-way SMP nodes. Several configurations are compared: 1, 2, 4 and 8 nodes. All the measurements are made with 400-MHz Pentium II nodes. Then we compare the performance of the platform with the performance of typical scalable multicomputers with the same number of nodes.

4.1 SMM versus HMM models

Figure 4 compares speed-ups with the two different memory models for a subset of the programs of NAS 2.3 benchmark.

Neither HMM nor SMM is better for all the programs of the NAS benchmark. SMM speed-up is greater when the impact of non-parallelizable loops (for the HMM approach) is significant in MPI program codes. This occurs for LU and FFT. HMM model is more efficient for CG. Technical report [11] shows the significant part of execution time for communications in the CG overall execution time, opposed to LU. Conflicts for access to the network interface (2 CPUs for 1 network board) and frequent access to local bus that correspond to local communications are limiting factors of speed-up in SMM mode. So, the better programming model depends on the features of the program to execute. According to our observations on LU and CG, analyzing computation/communication ratio and parallel computation loops can give useful insights on the most efficient programming model.

HMM provides a subtential benefit by allowing to speed-up BT and SP. BT ad SP require a square number of processors. SMM can be used for such programs only when the number of processors in the CLUMP matches a square number (4 processes on 2 nodes, 16 processes on 8 nodes, 36 processes on 18 nodes, etc.for 2 way SMP nodes). In the other hand, HMM requires the programmer to parallelize the original MPI codes. Parallelizing the NAS to get high speed-ups has required a long and carefull examination of each benchmark. Even with high efforts, some benchmarks still show low speed-up (LU).

4.2 COW versus CLUMP for a constant number of processors

With the best of HMM and SMM models, Figure 5 presents the performance of the 2-way SMP based CLUMP against the uniprocessor based platform having the same number of processors.

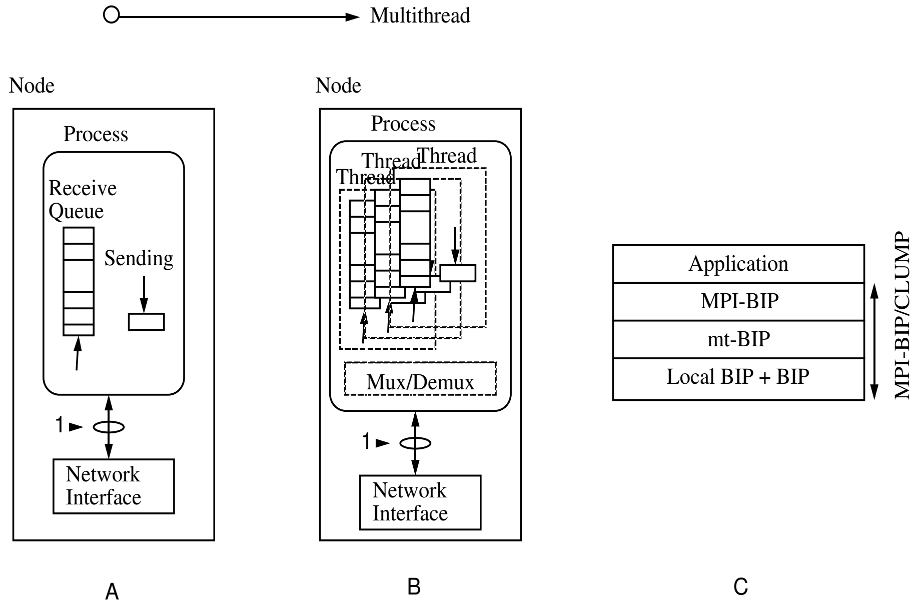

Figure 3: *MPI-BIP/CLUMP. Scheme A shows a high speed communication library with a single access to the network interface. With B scheme ((MPI-BIP/CLUMP), several execution threads can share the network interface. Scheme C describes the protocol stack between an application using MPI and BIP library (original + local BIP).*

4.3 Performance of a PC CLUMP versus scalable multicomputers

Speed-ups worth considering only if the overall performance of the CLUMP are significant when compared with typical high-end multicomputers. Figure 6 compares our platform using the best of HMM and SMM models with some scalable multicomputers. The benchmark is NAS NPB 2.3 (class A). For HMM model, we added directives for parallel execution of parallel loop nests. Only 5% of the code lines have been modified. EP has been improved significantly by avoiding falsesharing.

Comparisons assume 8 nodes for each machine. For our CLUMP, we present two performance measurements: COW (only one processor is used on each node) and CLUMP (two processors are used on each node). Performance of parallel machines listed in the figure has been collected in the NAS web site.

The overall performance of a PC COW is close to the performance of small size version of scalable parallel supercomputers. By using two processors per node, 8 nodes PC CLUMP outperformes most of 8 nodes parallel supercomputers.

5 Conclusion

We have presented two programming models that are currently used on our Myrinet network of 2-way PC SMPs. The Single Memory Model relies on the MPI-BIP/CLUMP library. The Hybrid Memory Model combines shared memory within SMP nodes (using OpenMP for the parallelization) and message passing between nodes (MPI).

We have compared the speed-up for a 2-way SMP node versus the 1-CPU node for the same benchmark according to memory models.

For both models, NAS speed-ups are less than the optimum speed-up. They depend on the program and on the number of nodes that are used. According to

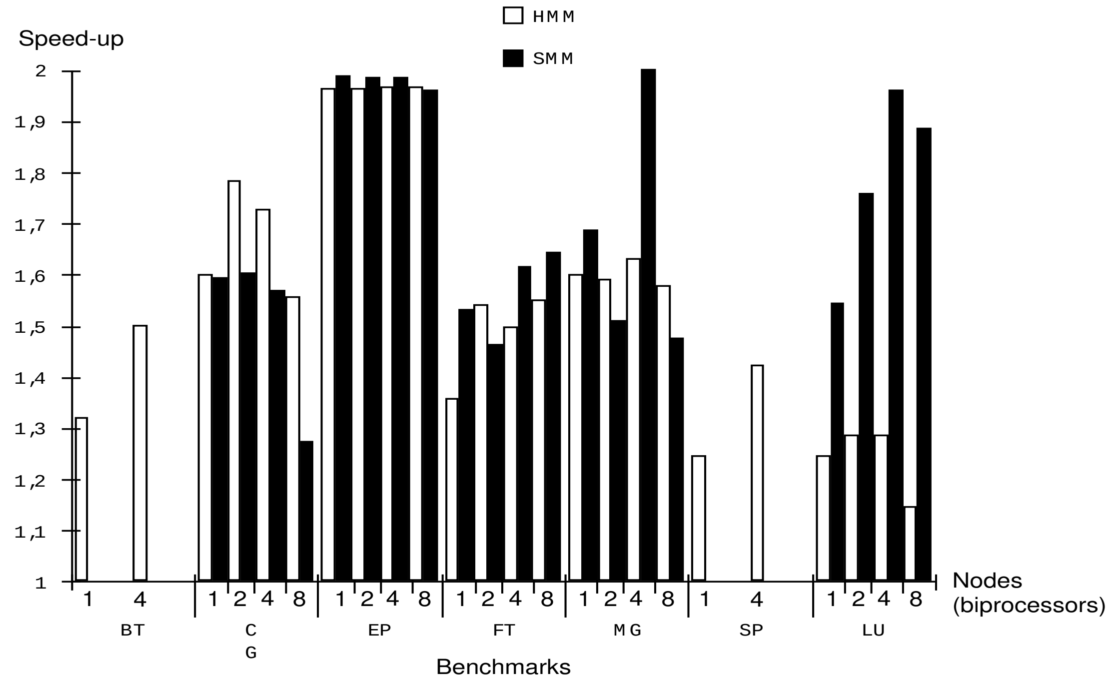

Figure 4: *Speed-up comparison for HMM (MPI + OpenMP) and SMM (MPI-BIP/CLUMP) approaches using the NAS NPB 2.3. benchmark*

the basic features of program (communication, parallel parts), one or the other model is better.

We have shown that the overall performance of a 8-nodes PC CLUMP (Pentium II 400 Mhz) is close or even higher than the performance of the typical to day scalable multicomputers with the same number of nodes on NAS NPB 2.3 benchmark.

References

[1] M. Bernaschi. Efficient message passing on shared memory multiprocessors. *Lecture Notes in Computer Science*, 1156:221, 1996.

[2] Steven S. Lumetta et al. Multi-protocol active messages on a cluster of SMPs. In *SC'97: High Performance Networking and Computing: Proceedings of the 1997 ACM/IEEE SC97 Conference: November 15–21, 1997, San Jose, California, USA.*, 1997.

[3] D. J. Scales et al. Fine-grain software distributed shared memory on SMP clusters. In *Proc. of the 4th IEEE Symp. on High-Performance Computer Architecture (HPCA-4)*, February 1998.

[4] R. Samanta et al. Home-based SVM protocols for SMP clusters: Design and performance. In *Proc. of the 4th IEEE Symp. on High-Performance Computer Architecture (HPCA-4)*, February 1998.

[5] Andrew Erlichson et al. SoftFLASH: Analyzing the performance of clustered distributed virtual shared memory. In *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 210–220, 1996.

[6] David E. Culler et al. Parallel programming in Split-C. In IEEE, editor, *Proceedings, Supercomputing '93: Portland, Oregon, November 15–19, 1993*, pages 262–273. IEEE Computer Society Press, 1993.

[7] Charlie Hu Honghui Lu et al. OpenMP on networks of workstations. In *Proc. of Super Computing 98*, Orlando, 1998.

[8] Mike Norman et al. University of Minnesota, Minneapolis, 1995.

[9] M. Ando et al. Compas: A pentium pro PC-based SMP Cluster and its experience. In *IPPS Workshop on Personal Computer Based Networks of Workstations*, pages 486–497. LNCS, 1998.

[10] L. Prylli et al. Bip: a new protocol designed for high performance networking on myrinet. In *Workshop on Personal Computers based Networks Of Workstations*, 1998.

[11] Cappello F. et al. Intra-node parallelization of MPI programs with OpenMP. Report TR-CAP-9901.

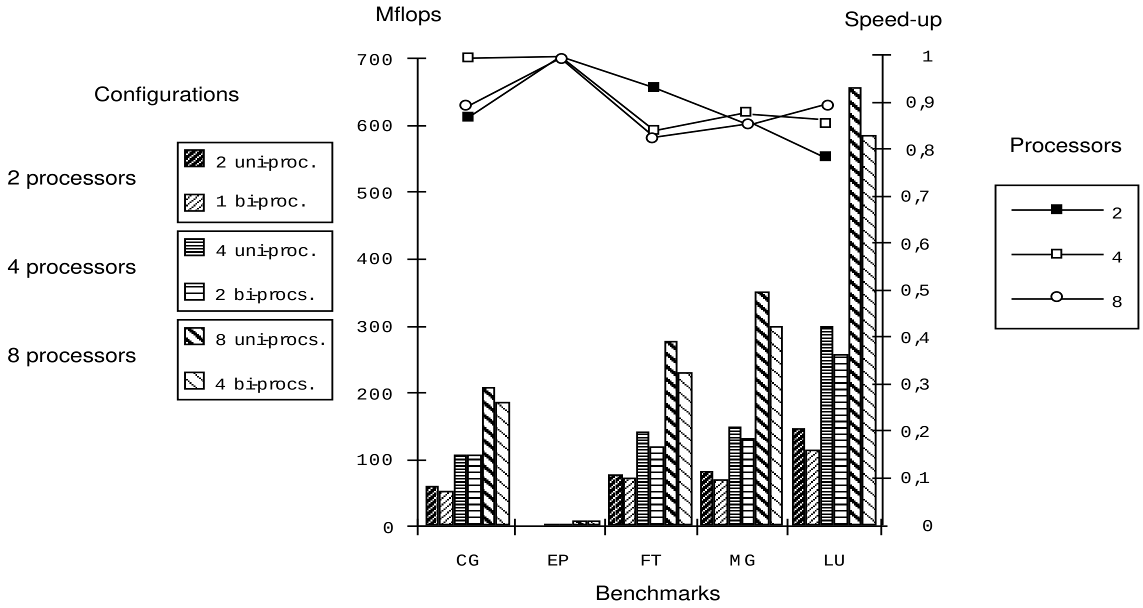

Figure 5: *Best of HMM and SMM Models - Global speed-up of the 2-way SMP nodes over 1-CPU node for the NPB 2.3 Benchmarks and for the same number of processors*

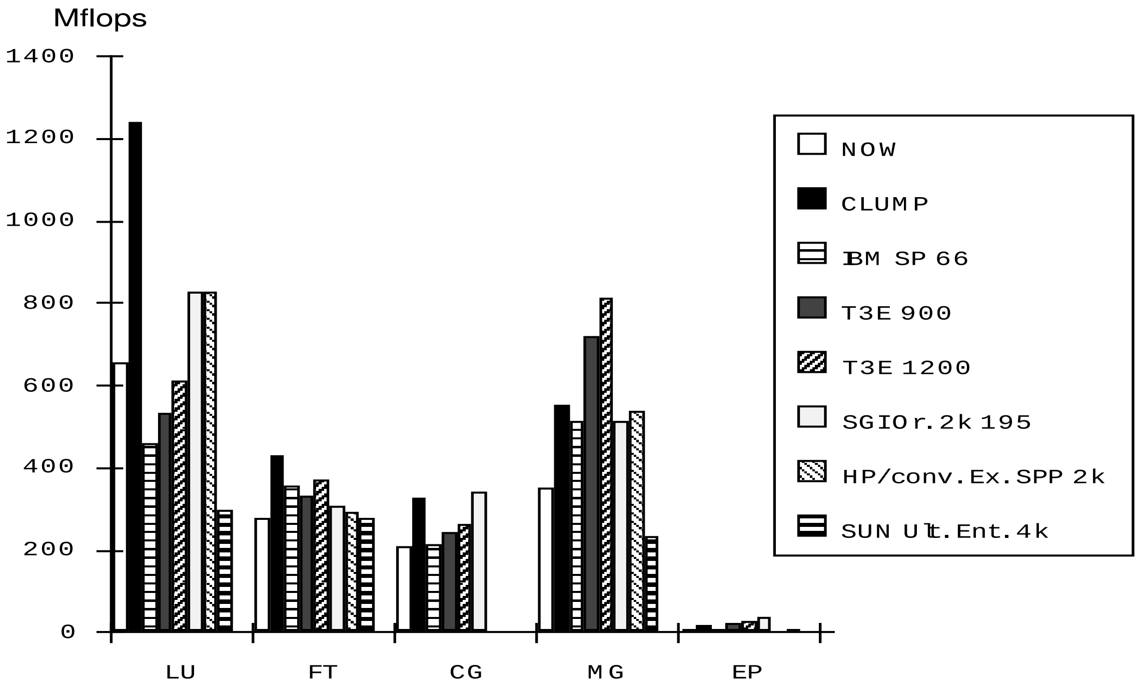

Figure 6: *Performance of several 8 nodes supercomputers for the NAS NPB 2.3 Class A benchmark*