NXLib — A Parallel Programming Environment for Workstation Clusters

Georg Stellner, Arndt Bode,
Stefan Lamberts and Thomas Ludwig*

Technische Universität München
Institut für Informatik
Lehrstuhl für Rechnertechnik und Rechnerorganisation
D-80290 München
stellner@informatik.tu-muenchen.de

Abstract. Workstations provide an attractive platform to develop software. This paper describes the NXLib environment which enables users to use this platform to develop parallel applications. The source code compatibility of NXLib applications with Intel's Paragon multicomputer system easily allows to migrate these applications to a supercomputer. The performance which can be achieved with either platform is compared using two applications. In addition performance results of NXLib communication calls and comparisons with other widespread message-passing libraries for workstation clusters are presented.

1 Motivation

Upcoming message-passing libraries for coupled workstations provide a facility for programers to exploit the workstations aggregate computational power during times of low user activity [8, 1]. Due to the basic similarity of the architecture of coupled workstations to multicomputer systems[2] a further usage of such networks is to serve as a development platform for multicomputer applications. This approach can withdraw workload from multicomputers to workstations especially in the development phase of an application. Therefore, an environment is needed which allows to run multicomputer applications on workstations. The NXLib environment provides the Paragon NX [2] library calls for different workstations, like Sun Sparc, HP PA, DEC Alpha, IBM RS6000 and SGI. Source code which has been implemented using NXLib simply needs recompilation to bring the application on a Paragon system.

2 Applications and Performance

To evaluate the NXLib environment we have used two coarse grain applications which we have on Paragon systems: NSFLEX [4] and MUMUS [5]. In both cases

* This project was partially funded by a research grant from the Intel Foundation.

[2] In both cases independent computational nodes which are interconnected with a communication network.

only minor changes to the makefiles were necessary to compile and link the source code. After the compilation the applications can be started like on a Paragon system by specifying the name of the executable at a shell prompt. Also Paragon command line switches can be used which for example allows to specify the number of processes which should be created during start up.

The performance comparison is based on the solution of the same problems on both platforms. To achieve comparable results the problems were solved on a four node Paragon partition and on four Sun Sparc 10. These were the most powerful machines and the maximum number at our disposal. Computations on more machines, which included some Sun SLC, made obvious that the performance is driven by the slowest machine in the configuration. On the Paragon its OSF/1 release 1.0.3 was running whereas the Suns executed the SunOS 4.1.1 operating system kernel. With these operating systems a single Paragon node can achieve a floating point performance which is up to three times higher than a single Sparc 10. Table 1 summarizes the execution times.

	Paragon	NXLib
NSFLEX	3040s	8257s
MUMUS	24s	13s

Table 1. MUMUS and NSFLEX execution times on different platforms

For NSFLEX the Paragon system is nearly three times as fast as the workstations. For the computation of the given problem with MUMUS the workstations need about twice the time as the Paragon. Taken into consideration the performance of applications on coupled workstations using NXLib seems very promising. These results have to verified on larger clusters and more powerful machines than the Sparc 10.

Apart from the applications we have used a simple ping-pong measurement to evaluate NXLib. The results of this experiment are summarized in table 2.

	Latency (0)	Latency (2048)	max. Bandwidth
local	0.2s	4.17ms	1761.9KBytes/s
remote	0.1s	6.78ms	656.1KBytes/s
Paragon	102μ s	103μ s	25.7MBytes/s

Table 2. Latency and bandwidth values of NXLib communication calls

Finally in fig. 1 a comparison with P4 and PVM 3.2 is provided. Like above, a simple ping-pong measurement was used to compare the communication speed

of the message-passing libraries. In all cases the communicating processes were executed on Sun Sparc 10 machines with SunOS 4.1.1 and no other active user processes.

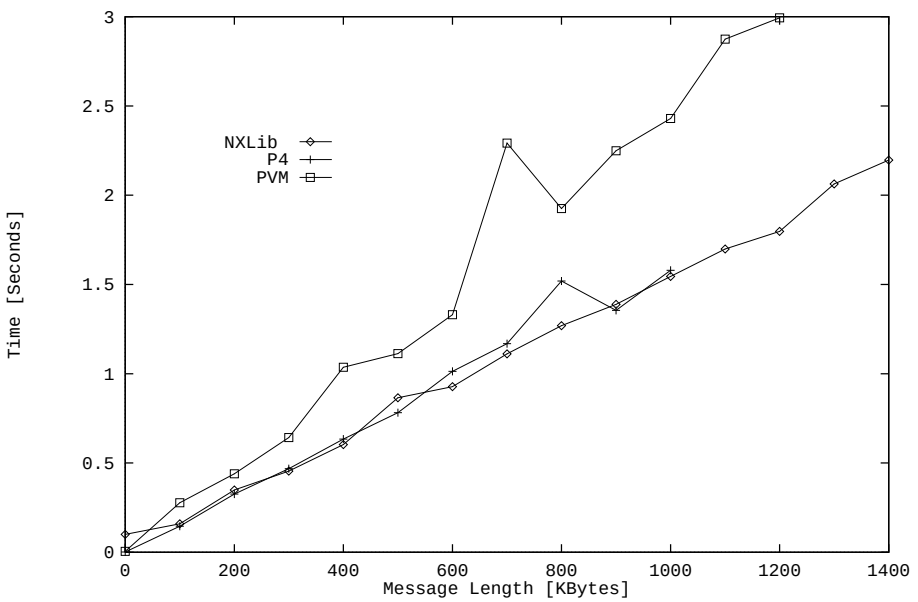

Fig. 1. Comparison of P4, PVM 3.2 and NXLib

It turned out that PVM achieves the slowest communication speed whereas P4 and NXLib nearly achieve the same values. This is due to the underlying communication protocol which is used for the implementation of the communication calls. In case of PVM an indirect communication scheme based on UDP is used. In contrast to that both P4 and NXLib[3] use a direct communication protocol based on TCP which offers better performance than UDP.

Compared to the Paragon performance values which are also given in table 2 the communication performance of NXLib is rather slow. Hence, applications which are intended to use the computational power of idling workstations should be medium or even better coarse grain. But this is not a restriction of NXLib itself but applies also to all other message-passing libraries for coupled workstations. A further conclusion is that due to the different performance values of NXLib and the Paragon the performance tuning has to be done separately for both architectures.

[3] More details about the communication protocol in NXLib can be found in [3]

3 Future Work

An important issue for scientific and commercial applications is the support of parallel I/O. Due to the restricted network bandwidth of bus coupled workstations it is not feasible to use a single disk as I/O facility. A more interesting approach would be to use the local disks of the workstations and to set up a virtual Paragon file system on these disks. Concepts for disk and file striping in such an environment must be examined therefore.

Up to now, there is no support for the programmer during the implementation phase of an application. Efficient coding is an important issue for software projects. Thus, the support of a tool environment which assists the programmer during all steps in the software life cycle is very desirable. Tools which can be used to visualize or debug parallel applications require the possibility to gather run-time information. This can either be done on-line with a monitoring system or off-line with trace files. In both cases an instrumentation of NXLib is necessary to produce the data.

References

1. Ralph Butler and Ewing Lusk. *User' s Guide to the P4 Programming System.* Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois 60439-4801, October 1992.
2. Rüdiger Esser and Renate Knecht. Intel Paragon XP/S — Architecture and Software Environment. In *Proceedings of the Supercomputer '93, Mannheim*, Postfach 1913, W-5170 Jülich, June 1993. Forschungszentrum Jülich, Zentralinstitut für Angewandte Mathematik.
3. Stefan Lamberts, Georg Stellner, Arndt Bode, and Thomas Ludwig. Paragon parallel programming environment on sun workstations. In *Sun User Group Proceedings* [7], pages 87–98.
4. T. Michl, S. Maier, S. Wagner, M. Lenke, and A. Bode. Dataparallel Navier-Stokes Solutions on Different Multiprocessors. In ASE'93, editor, *Applications of Supercomputers in Engineering*, September 1993.
5. M. Schumann, M. Kiehl, and R. Mehlhorn. Performance Evaluation of NXLib Using Parallel Multiple Shooting. In *Developing Multicomputer Applications on Networks of Workstations Using NXLib* [6], pages 58–64.
6. G. Stellner, M. Schumann, S. Lamberts, T. Ludwig, A. Bode, M. Kiehl, and R. Mehlhorn. Developing Multicomputer Applications on Networks of Workstations Using NXLib. SFB-Bericht 342/17/93 A, Technische Universität München, 80290 München, December 1993.
7. Sun User Group. *Sun User Group Proceedings*, December 1993.
8. V. S. Sunderam. Pvm: A Framework for Parallel Distributed Computing. Technical report, Emory University, Atlanta, GA 30322, 1992.

This article was processed using the LaTeX macro package with LLNCS style