# Emulating a Paragon XP/S on a Network of Workstations

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**

Networks of workstations are gaining growing importance in high performance computing. Programming environments like MMK/X [Ste93] or PVM [BDG+91] allow for exploiting the aggregate computational power of coupled workstations. A different approach is to use such local area networks as a development platform for parallel applications which should finally run on multicomputer systems. Therefore an environment on the workstations is required which offers the same programming model as the multicomputer system. The work which is presented here describes the NXLib environment which emulates a Paragon system on a network of workstations.

## Motivation

In the field of high performance computing multicomputer systems are currently becoming more popular. Mainly due to their better price performance ratio compared to classical supercomputers they are often preferred if a new high performance system should be bought. A decisive drawback of multicomputer systems is the enormous effort which is often required to port existing applications onto these systems. Applications have to be parallelized which leads to frequent test runs producing much additional workload. To withdraw some of this load an environment is needed which allows the implementation of applications for multicomputers on different hardware platforms.

Today, typical environments in universities and companies consist of several workstations all interconnected via Ethernet. The architectural principle behind multicomputers and coupled workstations is basically the same: independent processing elements with local memory which are interconnected. In difference to the multicomputers' dedicated high

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

performance interconnection network, workstations use a slower interconnect, which has to be shared with other users.

State-of-the-art multicomputer systems like the Paragon currently offer a proprietary message-passing environment. An implementation of that library on coupled workstations would allow for using them as a development platform for applications where the production code should finally run on a multicomputer system.

In the following we will describe the design and implementation of the Paragon communication library for workstations which are interconnected via Ethernet. Therefore we first give a short description of the Paragon and its software environment. Then, we introduce the design of the NXLib package. After that, we provide some performance figures and present a concluding summary in the last chapter.

# Paragon Multicomputer Systems

The nodes of a Paragon system are interconnected in a two-dimensional mesh topology, which is subdivided into three partitions: the I/O partition, the service partition and the compute partition. Usually the largest partition in a configuration is the compute partition. Parallel user applications are executed on the nodes in this partition and use the Paragon NX library. Apart from synchronous, asynchronous and interrupt-driven communication calls, it provides calls for process management. A more detailed description of the Paragon can be found in [LSBL93]. In contrast to that, interactive processes are executed on the nodes in the service partition. Finally, the nodes in the I/O partition are used to connect I/O devices.

# Design of NXLib

In the following sections the design of NXLib is introduced. A more detailed presentation can be found in [SLBL93] or [EK93].

## Virtualization of Paragon nodes

A parallel application on a Paragon system consists of two parts. The application processes on the compute partition and the controlling process on one node of the service partition. In the following discussion the term *Paragon node* (PN) will be referred to as the collection of a hardware node, the operating system kernel and a set of application processes running on top of that.

The basic means to model PNs on coupled workstations is virtualization. Consequently, the term *virtual Paragon node* (VPN) describes a PN on a workstation. The hard- and software properties of a PN which are not available on a workstation are virtualized in the following way. A natural approach to model them is to introduce a daemon process which emulates the node hardware and the operating system. The calls of the application processes to NX communication routines are transformed into requests to the daemon.

In such an implementation every system call would require an interprocess communication. To reduce the amount of interprocess communication parts of the operating system's tasks have been moved into the application processes like illustrated in fig. 1.

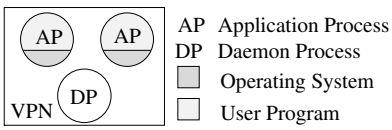


Figure 1: Processes and the distribution of the operating system on a VPN

## Modeling Paragon partitions

In addition to the partitions which were introduced before it is also possible to define sub-partitions of the compute partition. In a workstation environment mapping files can be used to emulate such partitions. Within that file a mapping of virtual node numbers to workstations is provided. Thus, the mapping table defines a *virtual compute partition* on a workstation net.

A problem occurs for the service partition. It is not part of the Paragon partition management which is available for the user. Consequently a different means has to be provided to establish a *virtual service partition*. This is simply done by defining the machine where the application has been started as the *virtual service partition*.

## NX message passing calls on workstations

An important issue for message passing libraries is the performance of the communication calls. Both local and remote communication use TCP sockets because this protocol achieves high throughput rates. To reduce the latency it is desirable to use direct paths between communication partners. Every stage in an indirect scheme increases the latency as additional calls have to be performed. On the other hand, on most UNIX systems the number of socket descriptors is limited. A full interconnection of all application processes would therefore drastically reduce the number of participating processes. Establishing and terminating a communication link between two processes for every communication call is not reasonable either as this would introduce much additional effort for every communication.

The basic assumption of our implementation is that typical parallel applications have a regular communication structure in the sense that certain processes regularly communicate with each other. Thus, two processes are either connected and use this link frequently during the computation or they do not communicate at all. Consequently, links need only to be created for those processes that wish to communicate. As the communication structure of an application can not be determined at start time, the interconnection of the processes can certainly not be done during the initialization of the application. So the links between processes are set up on demand during run time. Once established, a connection between two processes is kept until the application terminates. Building up the

connections on demand has the advantage that all interacting processes are fully interconnected. So communication latencies can be kept minimal for established links. Finally, as only those processes are interconnected which need to communicate more processes can participate in an application. The only drawback is that the first communication between two processes is more time consuming than the following because the connection has to be set up.

# Performance Measurements

All results presented in this section were produced either on a Paragon system running OSF/1, version 1.0.3 or on Sun Sparc 10/30 with SunOS 4.1.1, which were the most powerful machines at our disposal. As we only had access to four Sparc 10 we used a four node partition on the Paragon to achieve comparable results. Computations on more machines including some slower Sun SLCs made obvious that the performance is driven by the slowest machine in the configuration. Basically a single PN can achieve a floating point performance which is up to three times better than a single Sparc 10.

## Communication speed, latencies and bandwidth

In order to get a first overview on the communication speed, latency and bandwidth that can be achieved by NXLib calls a simple pingpong measurement has been done (fig. 2).

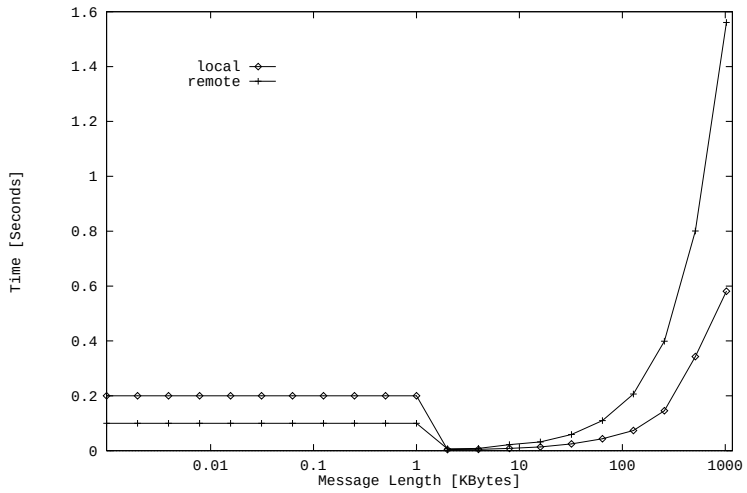


Figure 2: Local and remote transmission times of messages

For messages larger than 1416 Bytes a linear increase of transmission times can be observed with linear increasing message size and remote transfers are slower than local communications. For smaller messages the results are surprising for two reasons. First of

all, local transfers need twice the time of remote transfers and second, the transmission times are nearly constant. An explanation for both effects is the buffering mechanism which is used in the TCP protocol implementation for the Sun machines. IP packets which are smaller than the Ethernet packet size of 1514 Bytes are buffered and sent after a timeout of 0.1s. Thus, for messages smaller than the Ethernet packet size the transmission time is nearly constant. For local message exchanges this timeout obviously applies twice: once for the send and once for the receive operation.

| | Latency (0) | Latency (2048) | avg. Bandwidth | max. Bandwidth |
|---|---|---|---|---|
| local | 0.2s | 4.17ms | 590.2KBytes/s | 1761.9KBytes/s |
| remote | 0.1s | 6.78ms | 242.3KBytes/s | 656.1KBytes/s |

Table 1: Latency and bandwidth values of NXLib communication calls

The latency and bandwidth values which can be derived from the pingpong measurement are summarized in tab. 1. The low average bandwidth values can be explained by the exponential increasing messages sizes which were used for the pingpong measurement. A lot of small messages were used to show the effect of buffering. Small messages on the other hand achieve bad bandwidth values which also leads to bad average values.

## Applications using NXLib

To evaluate the NXLib environment we have used two applications which we have running on Paragon systems: NSFLEX [MMW+93] and MUMUS [SKM93]. In both cases 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 by specifying the name of the executable at a shell prompt. To select a virtual partition the same command line switch like on a Paragon can be used. Instead of the partition name the mapping file has to be specified. In a similar way the number of processes which should be created during the start up can be defined with the appropriate Paragon command line switch.

For each application the same problem was solved on the two different hardware platforms. The computation times which were necessary to solve the given problems on each architecture are summarized in tab. 2.

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

Table 2: Execution times of MUMUS and NSFLEX on different platforms

For NSFLEX the Paragon 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 be verified on larger clusters and more powerful machines than the Sparc 10.

## Conclusion and Future Work

The NXLib environment supports using a network of workstations for two main purposes. First, the network of workstations can be used to develop software which should finally run on a Paragon system. Workload can therefore be withdrawn from the multicomputer system. The CPU time which is gained by shifting the development of applications to workstations can be used for production runs of computational intensive problems. Second, instead of using the workstations merely as a development platform they can also be used as a production environment for certain applications. Especially coarse grain applications can achieve good speed-ups on a workstation environment.

NXLib basically offers the same software environment as a Paragon system. Virtualization is the basic means to achieve this. Therefore, source code which has been implemented using NXLib can be ported to a Paragon without any changes.

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 stripping in such an environment must be examined therefore.

Up to now there is no support for the programmer during the implementation process 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 through trace files. In both cases an instrumentation of NXLib is necessary to produce the data.

## References


[BDG+91] A. Beguelin, J. Dongarra, A. Geist, R. Manchek, and V. Sunderam. Opening the Door to Heterogeneous Network Supercomputing. *Supercomputing Review*, pages 44–45, September 1991.

[EK93] R. Esser and R. Knecht. Intel Paragon XP/S - Architecture and Software Environment. Technical Report KFA-ZAM-IB-9305, Forschungszentrum Jülich GmbH, Zentralinstitut für An gewandte Mathematik, D-52425 Jülich, April 1993.

[LSBL93] Stefan Lamberts, Georg Stellner, Arndt Bode, and Thomas Ludwig. Paragon parallel programming environment on sun workstations. In *Sun User Group Proceedings* [Sun93], pages 87–98.

[MMW+93] 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.

[SKM93] 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* [SSL+93], pages 58–64.

[SLBL93] G. Stellner, S. Lamberts, A. Bode, and T. Ludwig. Design and Implementation of NXLib. In *Developing Multicomputer Applications on Networks of Workstations Using NXLib* [SSL+93], pages 6–17.

[Spi93] Peter Paul Spies, editor. *Euro-ARCH '93*, Informatik aktuell, Berlin, October 1993. Gesellschaft für Informatik e.V., Springer-Verlag.

[SSL+93] 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.

[Ste93] Georg Stellner. MMK/X — Using a Network of Workstations as a Supercomputer. In Spies [Spi93], chapter 1, pages 5–21.

[Sun93] Sun User Group. *Sun User Group Proceedings*, December 1993.