ANL/MCS-TM-199, Rev 1

Users Guide for the ANL IBM SPx

by

William Gropp and Ewing Lusk
Mathematics and Computer Science Division

http://www.mcs.anl.gov/sp1/guide-r2/guide-r2.html

Revised November 30, 1995

MATHEMATICS AND
COMPUTER SCIENCE
DIVISION

Contents

1 Introduction **1**
- 1.1 Getting Started . . . 1
- 1.2 Usage Rules . . . 2
- 1.3 Comments on This Manual . . . 2

2 Machine Configuration **3**
- 2.1 Hardware . . . 3
  - 2.1.1 Details on the High Performance Switch . . . 3
- 2.2 Software . . . 4
- 2.3 What is Not Provided . . . 4

3 Programming **5**
- 3.1 Transport Layers . . . 5
  - 3.1.1 Ethernet/IP . . . 5
  - 3.1.2 Switch/IP . . . 5
  - 3.1.3 Switch/us . . . 6
  - 3.1.4 How do I tell which transport layer I'm using? . . . 6
- 3.2 Parallel Programming Libraries . . . 6
  - 3.2.1 MPL, POE, and PE . . . 6
  - 3.2.2 Chameleon . . . 7
  - 3.2.3 Fortran M . . . 9
  - 3.2.4 MPI . . . 9
  - 3.2.5 p4 . . . 10

4 Using the System **12**
- 4.1 Compiling and Linking Applications . . . 12
- 4.2 Running Applications . . . 13
- 4.3 Parallel Unix Tools . . . 14
- 4.4 Displaying System Information . . . 14
- 4.5 Scheduling Use of the SPx . . . 14
- 4.6 Additional Information . . . 14

5 The File Systems **16**
5.1 Unix filesystems 16
5.2 Parallel I/O 17
5.3 Hierarchical filesystem 17

6 The Fiber Channel **18**

7 Correctness and Performance Debugging **19**
7.1 pdbx and xpdbx 19
7.2 dbx 19
7.3 Profiling with prof and gprof 19
7.4 Chameleon Options 20
7.5 upshot 20
7.6 vt 21

8 Nonportable Programming **22**
8.1 Using MPL 22

9 Benchmarking **23**

10 Known Problems **24**
10.1 IBM Documentation 25

11 In Case of Difficulty **26**

12 Reporting Problems **33**

13 Miscellaneous **34**
13.1 Selecting Interrupt-Driven MPL 34
13.2 Interrupt-driven receives in MPL 34
13.3 Threads 34

Acknowledgments **35**

Bibliography **36**

Chapter 1

Introduction

This guide presents the features of the IBM SPx installed in the Mathematics and Computer Science Division at Argonne National Laboratory. This guide describes the available hardware and software, access policies, and hints for using the system productively.

This document is available online through any World-wide-web (WWW) reader, such as `xmosaic`, `tkwww`, or `www`. Many of the tools mentioned in this document also have on-line documentation; in particular, p4, and Chameleon have info documentation; Chameleon also has an extensive set of man pages.

1.1 Getting Started

This section describes how to get an account and get started on the ANL SPx.

To get an account, send email to `spaccount@mcs.anl.gov` and request an SPx account. Fill out the form that is returned to you. You will be notified when your application is approved or denied.

Publications resulting from your work on the Argonne SP1 should use the following acknowledgment.

> The author(s) gratefully acknowledge use of the Argonne High-Performance Computing Research Facility. The HPCRF is funded principally by the U.S. Department of Energy Office of Scientific Computing.

To get started once you have an account, log into either `bonnie.mcs.anl.gov` or `clyde.mcs.anl.gov`. These are IBM RS/6000s that serve as compile servers and file system servers for the IBM SPx. If you edit your '`.login`' or '`.cshrc`' files, make sure that you understand the consequences of any changes that you make to your `PATH` or `MANPATH`.

To change your default shell, you will have to send mail to `spsupport@mcs.anl.gov`. The only shells supported are `csh`, `ksh`, `sh`, and `tcsh`.

The nodes of the SPx are named `spnode1` through `spnode128`. Names of the form `spnode001` also work. Access to these is through a locally developed scheduler that is described below. If you have received interactive access to some of the nodes, you may log into them by using `rlogin`, `telnet`, or `rsh`.

1.2 Usage Rules

Usage of the SPx is now managed by the `spsubmit` program. Information may be found on the web at http://www.mcs.anl.gov/home/rayl/scheduler/scheduler.html.

1.3 Comments on This Manual

Please send any comments on this manual to `gropp@mcs.anl.gov`. We are particularly interested in new items for the chapters on known problems and "In Case of Difficulty."

Chapter 2

Machine Configuration

This chapter discusses the hardware and software configuration of the Argonne SPx, as well as what is not provided on this system.

2.1 Hardware

The Argonne SPx consists of 128 nodes and two compile servers. Each node is essentially an RS/6000 model 370. This model has a 62.5 MHz clock, a 32-KB data cache, and a 32-KB instruction cache. Key features of this system are:

- 128 MBytes of memory per node
- 1 GByte local disk on each node (400 MBytes available to users, rest for paging and the operating system)
- Full Unix on each node (IBM AIX 3.2.5)
- Each node accessible by Ethernet from the Internet
- High-performance Omega switch (63 μsec latency, 35 MBytes/sec bandwidth when using "user space" option).

In addition, the ANL SPx has a large high-performance file system (220 GBytes of RAID disk and a 6-TByte automated tape library).

The peak performance of each node is 125 MFlops (1 64-bit floating-point add and 1 floating-point multiply in each clock cycle). In practice, each node can achieve between 15 and 70 MFlops on Fortran code. Higher performance can be reached by using the BLAS or ESSL routines.

2.1.1 Details on the High Performance Switch

References for omega networks include [8] and [7].

2.2 Software

As each SPx node is running a full Unix, most of the usual Unix tools are available. Users may log directly into any SPx node using `telnet`, `rlogin`, or `rsh`, as long as they have been granted access by the `spsubmit` program.

- Multiple parallel programming environments (see section 3 [Programming], page 5).
- IBM's ESSL library
- Performance debugging tools

2.3 What is Not Provided

While the nodes support a full AIX, several services are not supported. These include mail, printing, and access to other file systems (with the exception of '`/sphome`'; see section 5 [The File Systems], page 16).

We have made no effort to provide PVM or PVM/e on our system.

The only mathematics software libraries on the SPx are the standard C library '`libm`' (which contains some special functions, such as Bessel functions), IBM's ESSL (includes the BLAS and LAPACK), and various research libraries. Use `-lessl` on your link line to include ESSL.

We do not support the usual MCS environment on the nodes. In particular, if you use a `.cshrc` or `.login` file that needs things like '`/mcs/etc/path`', they will fail and you will need to modify them (if you are using '`/mcs/etc/path`', you should look at the man page on `software`. This is the supported method of getting the basic environment, though it may cause problems for `sh` users.

We also provide no user-support services. We run a "dark" machine room; this means that there is no "operator" to respond to user requests, particularly outside of normal working hours. There are no consultants to help with writing or debugging your programs. We do respond to mail about problems with the machine, and the various research groups often express interest in helping people use their tools and in suggesting new features. See section 12 [Reporting Problems], page 33 for more information.

Please see section 10 [Known Problems], page 24 for a list of known problems. See section 11 [In Case of Difficulty], page 26, for help in identifying problems; please check it before sending mail about a problem with the machine.

Chapter 3

Programming

3.1 Transport Layers

There are a number of *transport layers*, or ways to communicate between nodes, on the SPx. For most uses, programmers will not use these directly; rather, they will use one of the portable programming libraries. However, as the programming libraries use these transport layers to actually accomplish the communication, it is important to understand them so that the proper transport layer can be chosen.

The available transport layers are Ethernet/IP , Switch/IP, and Switch/us. Only the first two support multiple parallel jobs on the same node. Switch/us provides better performance (particularly reduced latency) but can only run as a single process-per-node. Switch/us jobs may co-exist with other processes on the same node as long as only one process is using the Switch/us mode.

3.1.1 Ethernet/IP

All nodes in the SPx are connected by Ethernet, with node names `spnode001` through `spnode128` (leading zeros may be ignored; thus `spnode001` and `spnode1` are the same node). With this choice of transport layer, the SPx looks just like a collection of workstations. This method does suffer from the same drawbacks as any Ethernet-connected system: high latency (about 1 msec), low bandwidth (1 MByte/sec), and low scalability (the 1 MByte/sec is shared among all processors).

3.1.2 Switch/IP

All of the nodes in the SPx are also connected by the high-performance switch (HPS). One easy way to use the HPS is to use the same code but with different node names. Using `swnode...` in place of `spnode...` causes the HPS to be

nodename	Environment	transport layer
spnodex	Unix	Ethernet
swnodex	Unix	Switch/IP
fcnodex	Unix	Fiber Channel
any node	MP_EUILIB=us	Switch/us
any node	MP_EUILIB=ip	Switch/IP

Table 3.1: Relationship between node names and transport layers

used. This may be used with operations such as `rsh` and `rcp`, and by parallel programming libraries such as MPL and MPI. The `swnode...` names apply *only* when using IP over the switch. There is another way to use the switch that does not use IP and is much faster, but has a few limitations. This is the Switch/us mode, described next.

3.1.3 Switch/us

The same high-performance switch that provides the Switch/IP interface also has a higher-performance version that uses a proprietary interface for accessing the HPS. This version is called the "user space" version because it bypasses the AIX operating system (a source of the high latency in the Switch/IP method) and accesses the HPS directly. A drawback to this approach is that only one process per node (or processor) may use this mode to access the HPS.

3.1.4 How do I tell which transport layer I'm using?

The transport layer used depends on the parallel environment chosen and possibly the hostnames. The relationships are shown in Table 3.1.

3.2 Parallel Programming Libraries

We strongly encourage all programmers to use one or more of the systems described in this section when developing parallel programs for the SPx. These have been used on other systems and have made it easy to port significant applications to the SPx very quickly. These systems also offer a spectrum of tradeoffs between functionality and performance; the Chameleon system in particular has no overhead when used in "production" mode.

3.2.1 MPL, POE, and PE

MPL (formerly EUI) is IBM's message-passing interface to the high-performance switch. POE is the parallel operating environment, and controls the running of

MPL programs. PE is the parallel environment, and is IBM's term for all of the parallel tools (such as MPL, POE, pdbx, etc.) PE supports a parallel symbolic debugger (`pdbx`; `xpdbx` is not yet available) and a performance visualization tool (`vt`). MPL is documented in [6]. Any environment that makes use of Switch/us uses POE; all options described in the man page on poe (`man poe`) may be used.

Note that MPL has a large number of options. The options can be set with shell environment variables. The four most important are:

MP_HOSTFILE Gives the file that contains the processors that may be used. The '`SPnodes.xxxxx`' produced by the `spsubmit` command is an appropriate file

MP_PROCS Gives the number of processors to use. Must be no larger than the number of processors given by `MP_HOSTFILE`.

MP_INFOLEVEL Controls the diagnostic output. A value of zero suppresses all but fatal errors.

MP_PULSE Controls a "heartbeat" function in POE; in previous releases, a setting other than zero may significantly degrade performance. As of March 7, 1995, it is no longer necessary to set this parameter on the ANL SPx. It does not hurt to do so, however.

For C-shell users, an example of the use of these is:

```
setenv MP_HOSTFILE SPnodes.124356788
setenv MP_PROCS 8
setenv MP_INFOLEVEL 0
setenv MP_PULSE 0
```

This assumes an interactive session with at least eight nodes, and a job ID of `124356788`.

These options are relavent for all tools that use MPL, including the portable message-passing libraries that are implemented on top of MPL. However, note that the `mpirun` command for the `MPICH` implementation of MPI sets these for you.

3.2.2 Chameleon

Chameleon is a lightweight, portable message-passing system. It provides access to a wide range of communication layers, including MPL, MPI, PVM (not supported on the ANL SPx), and p4. Chameleon provides a common startup model that simplifies choosing a communication layer. (A communication layer is an interface to a transport layer; through the use of p4 and MPL, Chameleon provides access to both the IP and "user-space" transport layers.)

Examples: '`/home/gropp/tools.n/comm/examples`' contains C and Fortran examples as well as makefiles.

Documentation: '`/home/gropp/tools.n/docs/tutorial/parallel.tex`'. The script '`/home/gropp/tools.n/bin/toolman`' will start an `xman` for the manual pages for the Chameleon routines (and others). See also [4].

Supported Communication Layers: MPL, MPI, p4

Special Comments: The Chameleon makefiles provide portability by using names defined on the `make` command line. You should set `ARCH=rs6000` and `BOPT=g` (for debugging) or `BOPT=O` (for production) and `COMM=p4`, `COMM=eui`, or `COMM=mpi` for the interfaces p4, MPL, and MPI respectively. If you choose not to use the Chameleon makefiles, be sure that you get all of the required options (e.g., C programs require `-D_POSIX_SOURCE`).

Contact: `gropp@mcs.anl.gov` for more information.

How to compile and link: Examples may be found in '`/usr/local/tools.core/comm/examples`'. Chameleon uses fairly complicated makefiles to achive portability to a wide variety of systems; you should look at the Chameleon manual [4] for more details. The value of `ARCH` for the SPx is `rs6000`. If you are using the usual Chameleon makefile, an appropriate make line for MPL is

```
make ARCH=rs6000 COMM=eui BOPT=O
```

How to run: Chameleon provides a nearly consistent interface for all transport layers. The special cases are detailed here by transport layer:

COMM=eui Run as an MPL program.

COMM=p4 To use `p4` with Ethernet, you need to do

```
setenv TOOLSHOSTS /sphome/gropp/hosts
```

To use `p4` with the High-Performance switch, you need to use

```
setenv TOOLSHOSTS /sphome/gropp/hosts.hps
```

and the command-line argument `-p4 altnet`.

COMM=pvm Not yet supported.

COMM=pvm3 Not yet supported.

Note that when using COMM=eui, you must use the MPL syntax to run a parallel program. In particular, you need to set the environment variables `MP_PROCS`, `MP_EUILIB`, etc. You can not use the `-np` command-line argument to choose the number of processes. See the section on running MPL jobs in the `spsubmit` documentation (http://www.mcs.anl.gov/home/rayl/scheduler/scheduler.html).

3.2.3 Fortran M

Fortran M is a small set of extensions to Fortran that supports a *modular* approach to the construction of sequential and parallel programs. Fortran M programs use *channels* to plug together *processes* which may be written in Fortran M or Fortran 77. Processes communicate by sending and receiving messages on channels. Channels and processes can be created dynamically, but programs remain deterministic unless specialized nondeterministic constructs are used.

Examples: '`/usr/local/fm/examples`' contains Fortran M examples as well as makefiles.

Documentation: Bound hardcopies (the ANL Technical Report) may be obtained from the contact listed below.

Supported Transport Layers: Ethernet/IP and Switch/IP

Special Comments: See the "Network Specifics" section of the manual for details on running Fortran M on the SPx. The hostnames of the switch interface (i.e., swnode1) should be used. The latest version of the Fortran M compiler is installed in '`/usr/local/fm`'.

Contact: `fortran-m@mcs.anl.gov` for more information.

3.2.4 MPI

MPI (Message-Passing Interface) is a new message-passing system "standard" that has recently been defined by a broadly based group of parallel computing vendors, library writers (including us), and users. The current draft is now in the public-comment stage [3]. It was completed in the spring of 1994. The standard draft is available by anonymous ftp from '`info.mcs.anl.gov`' in '`pub/mpi/mpi-report.ps.Z`', or on the WWW in http://www.mcs.anl.gov/mpi/mpi-report/mpi-report.html.

There are two implementations: one being done by us and a group from Mississippi State University, and the other being done by IBM. Both are relatively complete and run now on the SPx.

3.2.4.1 The IBM Version

An experimental IBM implementation of MPI, from IBM Yorktown, is available. This version works with POE/MPL system. It is located in '`/usr/lpp/mpif`'.

Examples: The directory '`/usr/lpp/mpif/samples`' contains examples in both Fortran and C as well as a makefile.

Documentation: There is documentation in '`/usr/lpp/mpif/docu/mpif.ps`' and '`/usr/lpp/mpif/docu/errata.tex`'. Documentation on MPI is available in the MPI WWW page. (http://www.mcs.anl.gov/mpi)

Supported Transport Layers: Switch/IP and Switch/us.

Contact: `gropp@mcs.anl.gov` or `lusk@mcs.anl.gov` for more information.

How to compile and link: No special options are needed when compiling. The include file '`mpi.h`' is located in '`/usr/lpp/mpif/include`' and is needed by an C program. The include file '`mpif.h`' should be included by any Fortran routine that uses an MPI call. However, you need to use `mpicc` instead of `mpcc` and `mpixlf` instead of `mpxlf`.

How to run: MPI-F programs are run using much the same approach as MPL programs. To run a program

```
poe a.out ...
```

Use the same environment variables (e.g., `MP_EUILIB`) to control the transport layer, choice of nodes, and level of messages.

3.2.4.2 The MPICH Version

Examples: '`/usr/local/mpi/examples`' contains examples in Fortran and C, and a makefile.

Documentation: No implementation-specific documentation is yet available. Our implementation uses Chameleon, so the usual Chameleon methods of preparing programs and starting jobs apply. There are man pages for all of the MPI routines in '`/usr/local/mpi/man`'.

Supported Transport Layers: Switch/IP, Switch/US, Ethernet/IP.

Contact: `gropp@mcs.anl.gov` or `lusk@mcs.anl.gov` for more information.

3.2.5 p4

p4 [2, 1] is a portable message-passing system that runs on a very wide variety of parallel systems and workstations. It is in use at approximately 200 sites around the world. Existing p4 programs will run unchanged on the SPx. The current version is version 1.4, but experimental versions (such as 1.4a) and later releases (1.5) may appear from time to time.

Examples: '`/usr/local/p4-1.4a/SP2/examples`' on `bonnie` and `clyde` contains C and Fortran examples, example makefiles, and a makefile that can be used to construct your makefiles with the appropriate options for the SPx's various transport layers.

Documentation: '`/home/lusk/ibm/p4-1.4b/doc`' contains the latexinfo source for the manual [1], an ASCII version, and the postscript for a reference card. '`/home/lusk/p4.manual`' contains the postscript for the manual itself in ('`p4.ps`'). The manual is also online via `info` and any of the inside-Emacs or stand-alone `info` readers (e.g., `gnu-info`).

Supported Transport Layers: Ethernet/IP, Switch/IP, Switch/us.

Special Comments: See the section of the p4 manual entitled "Running on Specific Machines" for how to specify each of the transport layers.

Contact: `lusk@mcs.anl.gov` for more information.

How to compile and link: For MPL, you must use a `p4` made with `P4ARCH=SP1_EUIH`. At Argonne, these libraries are in '`/usr/local/p4-1.4a/SP2`' where `x.y` are currently `1.3c`.

How to run: To use p4 over MPL on the SPx, invoke your program from a node with:

```
<progname> -procs <numprocs> <user args>
```

Specific nodes can be chosen by the same method as for other POE jobs. The p4 procgroup file should contain the single line:

```
local <numprocs-1> <pathname of program>
```

The appropriate environment variables can be set with

```
set JID = `/usr/local/bin/getjid`
setenv MP_EUILIB us
setenv MP_PROCS `cat /sphome/$LOGNAME/SPnodes.$JID | wc -l`
setenv MP_HOSTFILE "/sphome/$LOGNAME/SPnodes.$JID"
```

To use Switch/IP with MPL, use `setenv MP_EUILIB ip` instead, with '`/sphome/$LOGNAME/SWnodes.$JID`'.

To use Switch/IP without using MPL, you need a procgroup file of the form

```
swnode1 0 executable
swnode2 1 executable
swnode3 1 executable
```

The line `swnode 0 executable` replaces the usual `local 0`. You can use `awk` to create this from the file '`/sphome/$LOGNAME/SWnodes.$JID`'.

Chapter 4

Using the System

This section describes how to run parallel programs and how to monitor the state of the system (IBM provides a powerful system monitoring utility, but it is only available to `root`. The monitoring tools described here were locally developed).

4.1 Compiling and Linking Applications

All compilation and linking should be done on the compile servers, `bonnie.mcs.anl.gov` and `clyde.mcs.anl.gov`). The compile servers should not be used to run any other programs.

The compilers are named `xlc` (for C) and `xlf` (for Fortran). (For MPL programs you must use `mpcc` and `mpxlf` instead.) These support most of the usual options but sometimes with unusual (for Unix systems) names. Useful options include

-c Compile to a `.o` file

-O3 Produce optimized code

-qstrict Ensure that optimizations do not alter program semantics

-qsource -qxref Produce a source listing with cross-references

-qlist Produce an object listing. This is equivalent to `-S` on other Unix systems

-qwait=-1 Wait indefinately for a network license token. You may need this if too many people are using the compilers at once, where too many is defined by the license server.

One warning: the Fortran compiler takes around 50 seconds to compile a trivial program; this is caused by the license server. This is considered a feature

by IBM. You may wish to consider using `f2c` to convert Fortran programs to C.

The GNU `C++` compiler is available in '`/usr/local/gcc/bin/c++`'. It is not supported for the SPx but should work.

We also have the IBM `C++` compiler `xlC`; however, we do not have any information on it (no man pages and no info information). We have been able to gleen that to link Fortran subroutines to a program whose main program is in C++, add the library `-lxlf` to your link line.

The following arguments are available only to Fortran programmers:

-qautodbl=dblpad Promote `REAL` declarations to `DOUBLE PRECISION`. This can be useful with Cray codes that require more than 32-bit precision (the default precision of `REAL` on RS/6000's). However, any double precision or `REAL*8` statements will be promoted to `REAL*16` (128 bits) which is significantly slower and probably not what is wanted.

-qdpc=e Treat floating-point constants as double precision.

-qextname Suffix an underscore to all external names. This makes Fortran routines available to C programs that expect Fortran to add an underscore suffix; this is common (but not standard) behavior of many Unix Fortran compilers. Note that some software packages do not expect Fortran users to use this option; their Fortran interfaces adjust to the default behavior of the Fortran compiler. Using `-qextname` with these libraries can lead to unresolved references.

When linking, this option can be useful:

-qloadmap:filename Produce a load map. This is handy for identifying from which library an object file was loaded, or for seeing the total memory requirements of your module.

-o name Provide a name for the module. The usual Unix default of `a.out` is used if no name is specified.

4.2 Running Applications

Each parallel programming system has its own requirements for running an application. In this section, we describe some issues that are common to all systems.

All access is through the scheduler http://www.mcs.anl.gov/home/rayl/scheduler/scheduler.html. That document contains information on running programs on the SPx.

4.3 Parallel Unix Tools

We have developed parallel analogs of the common Unix tools `cp`, `ls`, `ps`, and others. These are scalable and relatively fast. They simplify the task of executing commands on all or part of the SPx. However, they are currently unavailable on the SPx. We expect to provide them again in the near future.

4.4 Displaying System Information

Currently, there are four X Windows tools for displaying the state of the system. These are `wish` scripts that require `tk-3.3`. This code is available in the MCS Division. As `tk` is in the public domain, other sites should have no problem acquiring it. In addition, there is a program provided by IBM that gives some information on POE.

Currently, the only supported tool is `xspusage`. This shows which nodes are in use and which are available.

The program '`spusage`' will display the users of the the SPx nodes in ASCII format. This is useful on a text terminal or if running X over a slow connection.

4.5 Scheduling Use of the SPx

All access to the SPx is made through the use of the `spsubmit` program. This program maintains queues of pending jobs. The program `spq` lists the contents of the queues. `sphelp` lists the programs that are available for submitting jobs and manageing them. On-line documentation is provided through WWW.

4.6 Additional Information

There are three IBM manuals on the Fortran (and C) compilers:

The Language Reference In addition to describing the Fortran language it explains a large number of IBM-provided system-interface type of subroutines.

The User's Guide describes compiling, linking, use of the preprocessors, interaction of Fortran with files, etc.

The Optimization and Tuning Guide for the XL Fortran and XL C Compilers a very useful (and readable) description of optimization considerations for the RS/6000 architecture.

If you are using an X-windows system to access the compile server, you may view these manuals on line using IBM's `info` system. Before invoking `info`, define your display to the compile server:

```
setenv DISPLAY myid.mcs.anl.gov:0.0    (csh)
export DISPLAY=myid.mcs.anl.gov:0.0    (ksh)
info
```

where `myid.mcs.anl.gov` is the TCP/IP address of your computer or X-station. When `info` finally comes up (it can take several minutes), select the "List of Books" panel, find the desired manual, and double-click on it.

When connecting to Sun workstations, `info` (and most IBM X Windows tools) will generate several warning messages. These may be ignored.

Chapter 5

The File Systems

There are currently four file systems accessible to the nodes: two Unix (sequential) filesystems, '`/sphome`' (shared) and '`/tmp`' (local), a parallel file system (PIOFS), and a hierarchical file system (the 220 GByte RAID arrays).

5.1 Unix filesystems

The file system '`/sphome`' is shared between the nodes and the compile servers. This file system is NFS mounted.

Warning:

There is a problem where files that are copied over an existing file do not always seem to be noticed. You should always `rm` a file before you copy over it.

Be frugal with your use of '`/sphome`'. There is only 1 GByte of space in '`/sphome`', and this must be shared by all users, including those with large input and output files. Try not to keep any large files in '`/sphome`' except when you need them for a run (don't move files back and forth during the day but also don't leave a large output file on '`/sphome`' once it is generated). The program `xsponu` may be used to display the usage of '`/sphome`'.

Each local disk holds 400 KBytes and may also be used for temporary data files and for executables, but again, you should not leave large files on them. The program `xsp1df` (see section 4.3 [Parallel Unix Tools], page 14) can be used to display the amount of space available on each node's '`/tmp`' partition. (The rest of the local disk holds the local '`/var`' partition and the swap area.)

The SPx nodes do *not* mount the MCS file systems. Thus, the only way to move files is either by first moving them to '`/sphome`' or by using `rcp`.

For additional information on other file systems being considered or developed for the ANL SPx, see '`/home/nickless/Documents/io.ps`'. Contact `nickless@mcs.anl.gov` for more information.

Important Note:
Neither '`/sphome`' nor '`/tmp`' are backed up. In addition, the '`/tmp`' areas will be cleaned of old files on a regular basis; large files may be removed at any time.

5.2 Parallel I/O

The parallel I/O systems remain experimental. We are beta-testing IBM's parallel file system, PIOFS. Send mail to `pierce@mcs.anl.gov` to ask for access to PIOFS.

The PIOFS users guide is in '`/usr/local/ibmdoc/piofs/UsersGuide.ps`'. Header files are in '`/usr/lpp/piofs/include`'; the libraries are in '`/usr/lpp/piofs/lib`'.

This file sysetm is not yet considered stable; data loss is possible. On the up side, performance is much better than using Ethernet to the regular Unix filesystem, so PIOFS may be a good place to store data generated by a parallel program that could be re-generated in an emergency.

5.3 Hierarchical filesystem

The large hierarchical file system is managed by Unitree and is documented separately.

Chapter 6

The Fiber Channel

NOTE: THE FIBER CHANNEL IS NOT AVAILABLE YET; ANY INFORMATION IN THIS SECTION IS SUBJECT TO CHANGE WITHOUT NOTICE

Nodes $2 + 4i$ have Fiber Channel connections for $i = 0, \ldots, 31$.

Fiber Channel supports both TCP/IP and direct (`ioctl`) interfaces. The node names for Fiber Channel are `fcnode`i.

Performance is about 2.5 MB/sec for IP and 17 MB/sec for direct.

p4 provides access to Fiber Channel. Here is a sample p4 procgroup file:

```
fcnode2 1 /sphome/lusk/p4test/systest
fcnode6 1 /sphome/lusk/p4test/systest
```

Chapter 7

Correctness and Performance Debugging

Several tools may be used to do correctness and performance debugging of parallel programs. This section describes these tools and how to use them.

7.1 pdbx and xpdbx

IBM's POE provides parallel versions of `dbx` and `xdbx` called `pdbx` and `xpdbx` respectively. These work with POE/MPL programs, and allow you to run and debug parallel applications.

7.2 dbx

It is also possible to use `dbx` and `xdbx` with parallel programs by loging into each node and using the command

`dbx` *process'id*

after the parallel programs have started. You can get the *process_id* with

```
ps -ef | grep $LOGNAME
```

7.3 Profiling with prof and gprof

It is now possible to use both `prof` and `gprof` to profile the performance of your codes. Just compile with the `-p` or `-pg` option and execute

```
setenv MP_LIBPATH /usr/lib/profiled
```

before running your program. The `-p` option will create a file called `mon.out.<taskid>`. The `-pg` option will create a file called `gmon.out.<taskid>`. These can be used as input to `prof` and `gprof` respectively.

(Thanks to Tim Pierce for this information.)

Then just use prof and gprof with these output files.

7.4 Chameleon Options

Chameleon provides a number of features that support both correctness and performance debugging. These apply only to programs compiled with `BOPT=g` or `BOPT=Opg`. These featuers may be selected by command-line switch when the Chameleon application is started:

-event Generate an event log that may be viewed with upshot (see section 7.5 [upshot], page 20).

-summary Generate a summary of message passing activity, giving the amount of time spent in each process sending and receiving messages

-trace Produce a line of output for each message-passing operation (including the begining and ending of receives). This is useful for finding the cause of deadlock in parallel applications.

7.5 upshot

Upshot [5] is a portable X Windows program for visualizing the behavior of a parallel program. Both Chameleon and p4 can generate the event logs that `upshot` reads. To generate an event log from Chameleon, use the `-event` flag

```
a.out -np 4 -event
```

This generates a file '`bl`'. To view this file with `upshot`, use

```
upshot -l bl
```

In order for a Chameleon program to be able to generate this event file, the `BOPT=g` or `BOPT=Opg` version must be used.

Currently, you should have '`/home/gropp/bin/sun4`' in your path to use `upshot` on the Sun workstations. The version in '`/usr/local/bin`' will be updated in the near future.

Sample output from `upshot` is shown in Figure 7.1.

Chameleon can generate event logs with any transport layer.

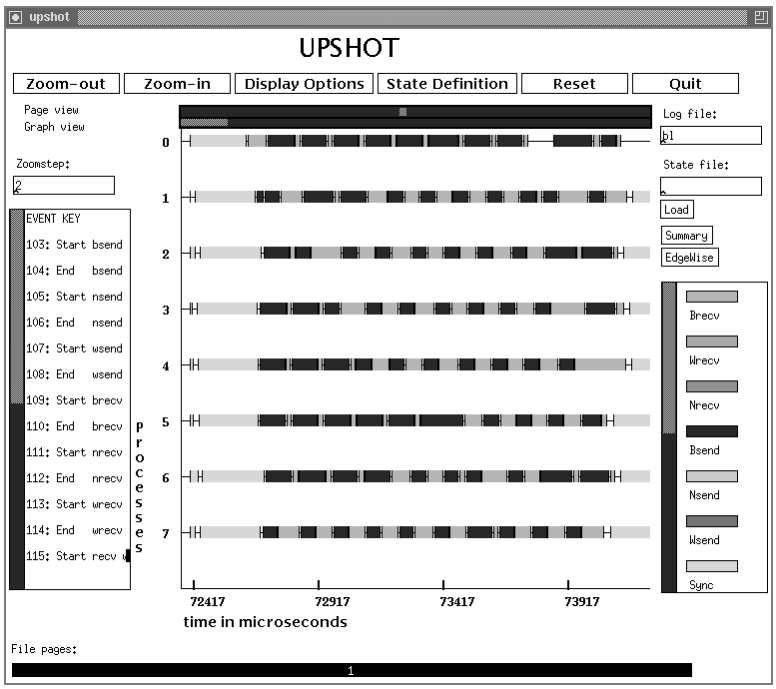

Figure 7.1: Sample `upshot` output

7.6 vt

The POE (Parallel Operating Environment) provided by IBM includes a performance visualization tool called `vt`. This tool, like `upshot`, can analyze a tracefile that is generated by a parallel program.

Chapter 8

Nonportable Programming

It is possible to use MPL without using the portability tools. This section describes how to run these programs. There are man pages for the individual routines (see '`/usr/man/cat1`' on `bonnie` or `clyde`).

8.1 Using MPL

MPL programs must be compiled with `mpxlf`, `mpcc`, and `mpCC` (in fact, only the main program must be compiled with this routines, but it is safer to use them for all routines). The switch `-us` selects Switch/us; the switch `-ip` selects Switch/IP.

Chapter 9

Benchmarking

Benchmarking on the SPx is more difficult than on most other MPPs because the SPx provides a friendly multiprocessing environment on each node. Avoiding the effects of contention with other users requires that the machine be scheduled as single user. To schedule single-user time, see section 4.5 [Scheduling Use of the SPx], page 14.

It is important to distinguish between elapsed (also called wall-clock) and CPU time when running a parallel program. Depending on the transport layer that is used, CPU time may not include the time in which processes were idle while waiting for messages to arrive. It is best when benchmarking to compute both the elapsed and CPU time; for most purposes, elapsed time is the better indicator of true performance.

The Chameleon package (see section 3.2.2 [Chameleon], page 7) provides portable routines for measuring both elapsed and CPU time.

```
#include "system/system.h"
double start_cpu, cpu_time, start_elapsed, elapsed_time;
...
start_cpu     = SYGetCPUTime();
start_elapsed = SYGetElapsedTime();
... work to time ...
elapsed_time  = SYGetElapsedTime() - start_elapsed;
cpu_time      = SYGetCPUTime() - start_cpu;
```

Fortran versions of these are also available.

Chapter 10

Known Problems

1. NFS file updates. We are seeing a problem where changing a file on '`/sphome`' does not seem to actually change the file. Until this is fixed, the best work-around is to always `rm` the file rather than `cp`'ing over it. If you change a program, rebuild it, and then run it and don't see the effect of your change, you are probably seeing this problem.

2. Linking takes *forever*. This is a feature. However, there are some things that you can do to (sometimes) speed things.

   (a) If you are relinking an application, use your old executable as *input* to the linker, along with *only* the modules that you have changed. For example, if the executable is *a.out* and the file '`changed.f`' contains the *only* changes, then

   ```
   xlf changed.f a.out
   ```

   will generate a new '`a.out`' executable. You will also get an error message about "xlf: 1501-218 file a.out contains an incorrect file suffix": you can ignore this message. If you use an executable name with no suffix, you will not get this message.

   This also works with `xlc` (including the error message).

   (b) If you are using one or more object libraries, consider moving them to a local disk. On `bonnie`, this means to '`/sphome`'.

   (c) Consider prebinding any libraries. For example,

   ```
   ld -r foo.a bar.a -o foobar.o
   ```

   will produce a file, '`foobar.o`', that is "prebound"; that is, all of the references between functions and symbols within these files are resolved. This can speed up the time to link, particularly if you make sure that this file is local (on `bonnie` or `clyde`, on '`/sphome`').

(d) **(Experts only):** Consider building a *shared* library. Information on this may be found by using `info` and looking for "How to Create a Shared Library."

(e) Check that only four `biod` processes are running on the compile server. In experiments, using 16 `biod` processes caused a significant increase in linking time compared to 4 `biod` processes.

3. `info` generates "X Error of failed request: BadValue" when it starts. This is a feature. You can ignore it (it has to do with the fonts used by the X11 server).

10.1 IBM Documentation

This note lists various SP-related documents. Those documents followed by "[== some date, 1993]" are orderable, bound, IBM documents. There is at least one copy of these documents in the lab and Tim Lehmann has unbound equivalent copies (with the noted date). All documents not followed by "[== some date, 1993]" are available unbound (from Tim Lehmann; some are not orderable and some came with the software). There's a day or two turn around to have copies made for the big documents.

The following SP documents exist for Release 1.0

- IBM 9076 Scalable POWERparallel Systems: Administration Guide, SH26-7221-00, September, 1993 [== _, 1993]
- IBM 9076 Scalable POWERparallel Systems: Maintenance Information, SY66-0299-00, September, 1993 [== _, 1993]
- IBM 9076 Scalable POWERparallel Systems: Planning and Installation Guide, SA23-2481-0, September, 1993 [== _, 1993]
- IBM 9076 System Support Programs, AIX 3.2.4, Release Notes, Release 01, Mod level 00, no date

The following PFS document exists

- Parallel File System External User Interface, Proposal, Version 1.0, September 17, 1993

The following Vesta documents exist

- Overview of the Vesta Parallel File System, no date
- Satisfying the I/O Requirements of Massively Parallel Supercomputers, no date
- Interfacing Vesta to an External File System, Version 0.3, August 23, 1993
- Vesta File System Programmer's Reference, Version 0.82, August 10, 1993

Chapter 11

In Case of Difficulty

This chapter describes some common problems and possible solutions or workarounds. You can also send mail to `spsupport@mcs.anl.gov` if you have questions.

1. Q: You run your program and it seems to be running the "wrong" program (e.g., print statements that you added do not seem to work).

   A: You may be running afoul of the NFS bug (see section 10 [Known Problems], page 24). Remove the executable and object files and rebuild the program.

2. Q: When running your MPL program, the application never seems to start.

   A: Unknown at present.

3. Q: I specify Switch/us but get error messages like this:

   ```
   setenv MP_EUILIB  us
   setenv MP_HOSTFILE   /sphome/<myhostlist>
   setenv MP_INFOLEVEL  0
   setenv MP_PROCS  4
   poe a.out -procs 4
   ERROR: 0031-619  mp_euilib specifies us, adapter not us
   ERROR: 0031-619  mp_euilib specifies us, adapter not us
   ERROR: 0031-619  mp_euilib specifies us, adapter not us
   ERROR: 0031-619  mp_euilib specifies us, adapter not us
   ```

   A: This may mean that you linked the executable with `-bnso`. This specifies no shared libraries and caused `mpxlf` or `mpcc` to link in only the Switch/ip libraries.

4. Q: Whenever I run a Switch/us job, I get these error messages:

```
spnode002.mcs.anl.gov:/sphome/gropp(4) poe ring
WARNING: 0031-402  Using css0 as euidevice for User Space job
WARNING: 0031-403  Forcing dedicated adapter for User Space job
WARNING: 0031-403  Forcing dedicated adapter for User Space job
...
WARNING: 0031-403  Forcing dedicated adapter for User Space job
```

   with $p + 1$ lines of output if I am running on p processors.

   A: This is a feature. To disable this, you can do

```
setenv MP_INFOLEVEL 0
```

   but this will disable all (almost all?) informational and warning messages. There is no known way to select specific messages, or to reduce the output to a single line for all processors. The default value of `MP_INFOLEVEL` is `1`; there appears to be no way to change the default.

5. Q: My job fails with these messages:

```
jm: 023-005 Failed to connect() spnode001.mcs.anl.gov, errno = 79.
jm: 023-005 Failed to connect() spnode001.mcs.anl.gov, errno = 79.
jm: 023-005 Failed to connect() spnode001.mcs.anl.gov, errno = 79.
jm: 023-005 Failed to connect() spnode001.mcs.anl.gov, errno = 79.
jm: 023-005 Failed to connect() spnode001.mcs.anl.gov, errno = 79.
jm: 023-005 Failed to connect() spnode001.mcs.anl.gov, errno = 79.
jm: 023-005 Failed to connect() spnode001.mcs.anl.gov, errno = 79.
jm: 023-005 Failed to connect() spnode001.mcs.anl.gov, errno = 79.
jm: 023-005 Failed to connect() spnode001.mcs.anl.gov, errno = 79.
jm: 023-005 Failed to connect() spnode001.mcs.anl.gov, errno = 79.
jm: 0023-161 connect jmd exceeded max times.
jm: 0023-023 Quitting...
ERROR: 0031-117  Unable to contact Resource Manager
ERROR: 0031-635  Non-zero status -1 returned from pm_mgr_init
```

   A: This is caused when the MPL job manager dies (for reasons unknown at this time).

6. Q: I get error messages like this when I run an MPL job (using Switch/ip):

```
ERROR: 0031-161  EOF on socket connection with task 28
ERROR: 0031-160  I/O error on socket connection with task 29
ERROR: 0031-619  : Bad file number
ERROR: 0031-161  EOF on socket connection with task 29
```

```
ERROR: 0031-160  I/O error on socket connection with task 31
ERROR: 0031-619  : Bad file number
ERROR: 0031-161  EOF on socket connection with task 31
...
```

A: Unknown at this time.

7. Q: I get error messages like this when I run an MPL job

```
ERROR: 0031-630  pm_contact: read timeout occurred; nprocs = 32
ERROR: 0031-618  The following nodes were not contacted:
ERROR: 0031-623  task   5: hostname spnode009.mcs.anl.gov
ERROR: 0031-635  Non-zero status -1 returned from pm_mgr_init
```

A: Unknown at this time.

8. Q: I try to compile my program and I get an error message:

```
        xlf  -c utility.f
NETLS:  StartNetLS: communications failure (network computing sys-
tem/RPC runtime)
The error code from the last failed command is -1.
```

A: Welcome to the world of licensed software. Our compilers (and possibly other products) are controlled by a network license server. This error message means that the license server could not be contacted; please notify `spsupport`. Another error that you might encounter is some message about there being no licenses left; you can for the compiler to wait for a license with the option `-qwait=-1`.

9. Q: It takes forever to compile my small Fortran program.

   A: The Fortran compiler takes around 50 seconds to compile a trivial program; this is caused by the license server. This is considered a feature by IBM. You may wish to consider using `f2c` to convert Fortran programs to C.

10. Q: My program is in '`/sphome/$LOGNAME/myname`', but when I try to execute it, I get

```
myprog: Command not found.
```

    A: This means that your `PATH` does not contain the current directory (usually a single "." in the PATH. Either modify your `PATH` or prefix the program name with `./`:

```
./myprog ...
```

Also see section 1.1 [Getting Started], page 1.

11. Q: My program runs correctly when I compile it `-g` but not when I compile it with optimizations (`-O`).

    A: The compiler may be generating incorrect code. Try using bisection on your source files: compile half with `-g` and half with `-O` and test again. Continue replacing files compiled with `-O` with ones compiled with `-g` until the code works. You may be able to simply use a few modules compiled with `-g` in your application.

    If you can produce a simple example (a few dozen lines of code) that you are sure is incorrectly compiled, send it to `spsupport`.

12. Q: I tried to login but got messages like

    ```
    cat: cannot open /mcs/etc/path
    arch: Command not found.
    hostname: Command not found.
    biff: Command not found.
    tset: Command not found.
    hostname: Command not found.
    ```

    A: Your '`.login`' or '`.cshrc`' file contains references to the MCS filesystem (e.g., '`/mcs/etc/path`'). These are unavailable on the nodes. Modify your files accordingly. See '`/sphome/INIT/initcsh`' for a script to create initial '`.login`' and '`.cshrc`' files.

13. Q: I tried to use `xspusage` but I got the error message

    ```
    bonnie % Xlib:  connection to "neptoon:0.0" refused by server
    Xlib:  Client is not authorized to connect to Server
    couldn't connect to display "neptoon:0"
    ```

    A: You need to do `xhost + bonnie` on the X11 server you are using (in this example, `neptoon`). Some other tools will require that you have allowed connections from the individual nodes; you may want to add `xhost + spnode$i` for `i` from 1 to 128.

14. Q: My parallel program runs on other parallel machines but seems to deadlock on the SPx when using MPL, MPI, or Chameleon.

    A: The following parallel program can deadlock on *any* system when the size of the message being sent is large enough:

    ```
    send( to=partner, data, len, tag )
    recv( from=partner, data, maxlen, tag )
    ```

where these are blocking send's and receives (`mp_bsend` in MPL and `PIbsend` in Chameleon). For many systems, deadlock does not occur until the message is very long (often 128 KBytes or more). The limit for MPL is smaller.

To fix this you have several choices:

- Reorder your send and receive calls so that they are pair up. For example, if there are always an even number of processors, you could use

```
if (myid is even) {
   send( to=partner, data, len, tag )
   recv( from=partner, data, maxlen, tag )
   }
else {
   recv( from=partner, data, maxlen, tag )
   send( to=partner, data, len, tag )
   }
```

  Another possible source of problems is trying to receive messages in a different order than they were sent. For example, if processor 1 does

```
send( to=0, data, len, tag1 )
send( to=0, data, len, tag2 )
```

  and processor 0 does

```
recv( from=1, data, len, tag2 )
recv( from=1, data, len, tag1 )
```

  and these are blocking receives, then when using MPL, this program may deadlock. To fix this, reorder either the sends or receives.

- Use non-blocking sends and receives instead
- If you are using Chameleon, you can use the command line argument `-eui nsend=0` to cause Chameleon to provide buffering from user-space for all blocking sends. This works with the `COMM=eui`.

15. Q: My C program include `fcntl.h` but does not find symbols like `O_RDONLY` (needed for `open`.

    A: You need to add `-D_POSIX_SOURCE` to your compile flags. This is usually required if you use `xlc` instead of `cc`.

16. Q: I can't seem to run MPL jobs in the background or with `nohup job`; the just seems to die when I log off. (May only happen to `ksh` users)

    A: (quick answer): Don't log off; just leave an `xterm` running the program)

(long answer; not recently tested): If one is using ksh, and one starts a normal job, or a p4-socket job on one of the nodes, using

```
nohup job.....
```

then when one logs off, the job continues, providing one used telnet to get to the node (I think rlogin won't work).

If makes a script for the job and puts

```
#! /bin/csh
```

at the beginning of the script and then runs

```
nohup script.name &
```

it still dies.

However if one enters csh first:

```
csh
nohup script.name &
```

then all is ok.

17. Q: My Fortran program uses `NAMELIST` and no longer works.

    A: There are reports that the behavior of `NAMELIST` has changed. `xlf` now conforms to Fortran 90, and the Fortran 90 standard specifies the form and interpretation of `NAMELIST`. The earlier verision of the Fortran standard, Fortran 77, did *not* include `NAMELIST`. However, many Fortran compilers implemented some form of `NAMELIST`. The problem comes from incompatibilities between these previous non-standard definitions of `NAMELIST` and the Fortran 90 specification.

18. Q: How do I use `info` with ASCII text (e.g., over a modem)?

    A: To access the InfoExplorer Help pages via the info ASCII interface:

```
 info -a
  CTRL-O to access the menu bar
   D to select the "Display" menu option
    B to select the "Books" sub-menu option
     CTRL-F several times to the "Getting Started" topic
      ENTER to select this topic
       CTRL-N, CTRL-F, and CTRL-B as needed to reach "Chapter 9.  Us-
ing the
              InfoExplorer ASCII Interface"
        ENTER to select this topic
         Use the following keys as needed to move around this chapter
```

```
         and learn more details about the info ASCII interface:
         CTRL-O to toggle to/from the menu bar (E in the menu bar ex-
its info)
         CTRL-N to scroll to the next page
         CTRL-P to scroll to the previous page
         CTRL-F to move forward within a topic list
         CTRL-B to move backward within a topic list
```

To access the Parallel Environment documentation, execute

```
info -a -l pe
```

and use the same movement keys as noted above.

Chapter 12

Reporting Problems

If you believe that the SPx has a hardware or software problem, please send mail to `spsupport`. Please be as specific as you can; if relevant, include samples of code or output.

The mail alias `spusers` should be used for sending mail to fellow users of the SPx. This list is appropriate for mail about new tools, user-group meetings, pleas for reducing disk space usage, and the like.

If the '`/sphome`' file system fills up, you should do the following

1. Use the command `df /sphome` to verify that the file system really is full.
2. Remove whatever you can. The command

   ```
   du -k /sphome/$LOGNAME
   ```

   will show (in kilobytes) the disk usage of all of your subdirectories.
3. Send a message to the other SPx users alerting them to the fact that '`/sphome`' is full. This can be done by sending a message to `spusers@mcs.anl.gov`. You might want to include (some) of the output of

   ```
   du  -k -s /sphome/* | sort -r -n | head
   ```

   which will show the disk usage of each user (note that '`/sphome/pass`' is a mounted file system and hence does not take space away from the '`/sphome`' file system).

Chapter 13

Miscellaneous

13.1 Selecting Interrupt-Driven MPL

By default, MPL uses a polling method to check for incoming messages. This has the advantage of reducing the number of interrupts that must be handled, but can in some cases introduce long delays as processes wait for messages to be delivered. Programs containing large amounts of nonblocking communications can benefit from setting the environment variable `MP_CSS_INTERRUPT` to `yes` before running. This works with Chameleon and p4 as well as for MPL jobs started with `poe`.

13.2 Interrupt-driven receives in MPL

As an *experimental* feature, our version of MPL includes a way to specify that a function should be called when a message of a particular type is received. Please see Mike Minkoff for more information.

13.3 Threads

DCE threads are available on all nodes and compile servers.

Acknowledgments

We thank all the users who have contributed to this guide. Particular thanks go to Steve Pieper of the ANL Physics Division, who has been an aggressive, pioneering, and always helpful user, and has uncovered many problems for the first time. His initial efforts to document his findings on the use of the SP provided the initial text for the first edition of this guide, and the authors of this edition gratefully acknowledge his contributions.

The work described in this report has benefited from conversations with and use by a large number of people. Steven Tuecke provided the descriptions for Fortran M and a number of valuable comments. Thanks to the many people who provided comments on the early draft of this document. In particular, Bill Nickless and Paul Plassmann made a number of useful suggestions. Tim Lehmann read the document with particular care, identifying a number of places where files or commands refered to had moved.

Bibliography

[1] Ralph Butler and Ewing Lusk. User's guide to the p4 parallel programming system. Technical Report ANL-92/17, Argonne National Laboratory, October 1992.

[2] Ralph Butler and Ewing Lusk. Monitors, messages, and clusters: The p4 parallel programming system. *Journal of Parallel Computing*, 1993. To appear (Also Argonne National Laboratory Mathematics and Computer Science Division preprint P362-0493).

[3] Message Passing Interface Forum. Document for a standard message-passing interface. Technical Report CS-93-214, University of Tennessee, November 1993.

[4] William D. Gropp and Barry F. Smith. Chameleon parallel programming tools users manual. Technical Report ANL-93/23, Argonne National Laboratory, March 1993.

[5] Virginia Herrarte and Ewing Lusk. Studying parallel program behavior with Upshot. Technical Report ANL-91/15, Argonne National Laboratory, August 1991.

[6] IBM. *IBM AIX Parallel Environment Parallel Programming Subroutine Reference Release 2.0*, June 1994.

[7] D. H. Lawrie. Access and alignment of data in an array processor. *IEEE Transactions on Computers*, C-24(12):1145–1155, December 1975.

[8] H. J. Siegel. *Interconnection Networks for Large Scale Parallel Processing.* Lexington Books, 1985.