Bull

HPC Linux
Installation Guide

ORDER REFERENCE
86 A2 31EG 02

Bull

HPC Linux
Installation Guide

Software

July 2003

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
86 A2 31EG 02

Printed in France

Suggestions and criticisms concerning the form, content, and presentation of this book are invited. A form is provided at the end of this book for this purpose.

To order additional copies of this book or other Bull Technical Publications, you are invited to use the Ordering Form also provided at the end of this book.

Trademarks and Acknowledgements

We acknowledge the right of proprietors of trademarks mentioned in this book.

Intel® and Itanium® are registered trademarks of Intel Corporation.

Windows® and Microsoft® software are registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark in the United States of America and other countries licensed exclusively through the Open Group.

Linux® is a registered trademark of Linus Torvalds.

- Table of Contents

CHAPTER 4. SOFTWARE INSTALLATION AND START-UP

Preface

Purpose of the document

The purpose of this document is to explain how to create a Linux HPC environment with Open Source or proprietary software tested by Bull for Bull hardware with a Itanium-2™ architecture.

Organization of the document

After the introduction,

- Chapter 1 describes the basic notions of HPC in a LINUX environment with any software.
- Chapter 2 describes the software components selected by Bull.
- Chapter 3 describes cluster installation.
- Chapter 4 describes how to install the software components from the CD supplied by Bull or from supplier sites on the NovaScale 4040, NovaScale 5080 and NovaScale 5160.
- Chapter 5 describes how to remove licensed Intel software.
- Chapter 6 provides an FAQ.

Chapitre 1. Linux HPC - General Concepts

1.1 HPC (High Performance Computing)

1.1.1 Definition of HPC

The term HPC describes large scientific applications that require a powerful computation facility with extremely accurate results and that are able to use a very large amount of data.

1.1.2 Use of HPC

HPC is used in different domains:

- Research (molecular dynamics, fluid mechanics, etc.)
- Industrial science (car, nuclear, meteorology, etc.)
- Synthesis imagery (special effects, etc.)
- Data mining (statistical use of large Data Warehouse type databases, etc.).

1.2 LINUX

1.2.1 Definition

Linux is a UNIX type multi-tasking, multi-user operating system that is available on numerous hardware platforms, especially those with ix86 and Itanium™ processors. It integrates most of the latest technology (SMP, clustering, RAID, etc.).

What sets Linux apart is that it is "free" software, developed by thousands of programmers all over the world working together closely and, for many, as unpaid volunteers.

Linux is a kernel. To use it, you require applications, provided by distributions. A distribution is a set of programs and a kernel for installation on a machine. RedHat, Mandrake, Suse, TurboLinux are some of the most common Linux distributions.

1.2.2 A Potted History

Linux is an operating system designed by Linus Torvalds, a Finnish student. He began writing the kernel in 1991. The rule of thumb that he followed through his work was that the license should be open. Each part of his operating system has therefore been rewritten and enhanced over time (today we have reached version 2.4.20).

The Linux kernel is shipped as "distributions" containing the kernel and programs most often together with an "in-house" tool for installation purposes. They are available from FTP sites or on commercial CDs. Linux therefore has at its disposal a large quantity of free software, developed as part of numerous other projects, in particular GNU.

1.2.3 LINUX and HPC

1.2.3.1 The Advantages of Linux

- Multi-platform - the Linux kernel works on PC, Itanium, PowerPC, etc.
- Reliable, robust.
- Much of the software (including the kernel) are Open Source and can therefore be downloaded free-of-charge from the Internet. This also allows a large number of people to work on the software and speeds up development times.
- Widely deployed, Linux has fast growing numbers of adepts, not only in Research Centers and Universities but also in industry. The synergy behind Linux is huge and explains both its quality and its fast emergence.

1.2.3.2 Open Source (Free Software)

Since the days of Linus Torvalds and his Linux system, Open Source has developed extremely quickly.

Free software, as a philosophy was made popular by Richard Stallman in 1984, when he created the Free Software Foundation (FSF) and the GNU project. He set up a number of rights that he believed all users should be able to enjoy, and coded them within a General Public License (GPL) called GNU.
The source code is therefore no longer the private possession of any one individual, group of individuals or company, as had been the case since the birth of computing back in the sixties until the eighties and nineties. Major companies are now close on the heels of the independent developers, themselves now proposing professional quality, Open Source software.

The principles of Open Source and the GPL license are:

- Free use
- Free redistribution
- Source code available to everyone
- License must allow modifications and other work derived from it
- Integrity of author's source code: diffusion of patches
- Redistributor must give same license to recipients

Other licenses also exist such as the BSD license. Unlike GPL, BSD can allow (if stated in the source) the non-diffusion of modifications.

For further information, see:
http://www.gnu.org
http://www.opensource.org
http://www.freebsd.org

1.3 HPC Clusters

1.3.1 What is a cluster?

A cluster is a set of machines (called nodes) ***connected together to fulfill a function.*** Each node can be a monoprocessor or SMP multiprocessor (symmetrical multi-processing), in this case the processors share the memory and the disks.

1.3.2 The Main Types of Clusters

Scientific Cluster (HPC): Computation is divided up into several tasks that will be performed simultaneously on the various machines in the cluster. Such clusters are used above all by the scientific community and graphics world.

High Availability Cluster (HA): This is created to avoid breakdowns in the service, usually due to a hardware or software malfunction on one machine. For example, if one node is no longer able to function, it will automatically be replaced by another machine in the cluster.

Load balancing is a common term used when referring to clustering. Its goal is to ***distribute*** system or network ***processes over the different nodes***. Therefore, when the load balancing node receives a process, it looks at the load and specialization of each of the nodes and allocates the process to the most appropriate one. This principle is used in the network domain and more particularly for services like WEB or FTP servers.

1.3.3 The Architecture of an HPC Cluster

An HPC Cluster system is:

- A set of independent machines that are similar as regards their architecture and speed. One of them (the server node) shares out the tasks among the others (client nodes) that return the result once the computations have been completed.

- The cluster will have a fast dedicated internal communication network for parallel applications (MPI) and another TCP/IP network for its management that will make the cluster appear as a single entity when viewed from the outside. This concept differs from that of a workstation network where each node has its own screen and keyboard.

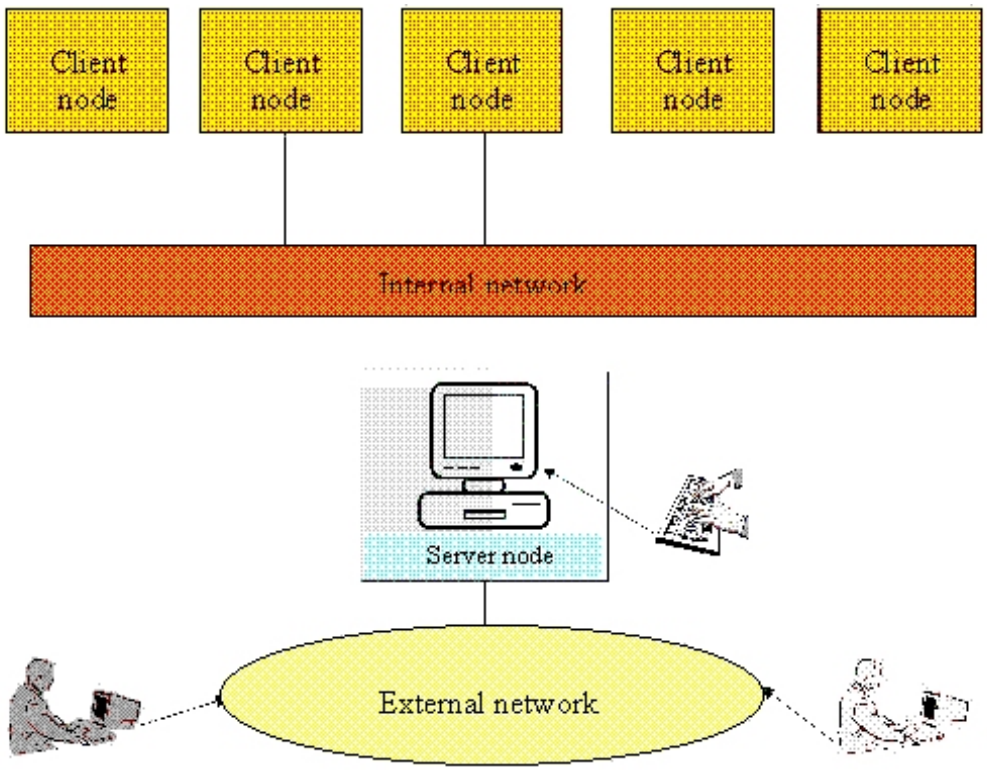


Diagram of a classic cluster

The server node controls the whole cluster. It is also the cluster console and the gateway to the outside world. A large cluster can have more than one server node and client nodes dedicated to specific tasks (e.g. computation, file server, etc.).

1.4 Linux HPC and Bull

Today, Linux is an operating system capable of supporting the needs of HPC environments required by scientific applications in terms of number of CPUs, I/O bandwidth and support for parallel programming. This is why Bull offers it on its SMP Itanium-2 4 channel (NovaScale 4040), 8 channel (NovaScale 5080) and 16 channel (NovaScale 5160) platforms and in cluster configurations interconnecting NovaScale 4040s to increase CPU capacity.

Chapter 2. Description of the Linux HPC Software Environment

2.1 Introduction

This chapter describes a set of software applications making up an HPC software environment that can be used on Bull machines.
Except for software whose name is underlined (software specifically for clusters), all the software mentioned below can be used on either a single machine or a cluster.

2.2 HPC Software Components

			Open Source	Commercial Product
APPLICATION DEVELOPMENT	**Scientific Libraries**		Blas, pblas, blacs, lapack, scalapack, fftw, atlas	Libmkl (Intel), NAG
	Parallel Libraries	Mpi	Mpich, Lam-mpi	ScaMpi
		Others	PVM	OpenMP
	Compilers (C, C++, Fortran)		GNU	Intel, NAG
Tools	**Operation**	Task sharing	OpenPBS, Maui,	LSF, PBSPRO ScaOPBS
		Debuggers	Gdb, dbx	Totalview (Etnus) Idb (Intel)
		Profiling	Cprof/vprof	Vampir (Pallas)
		Performance analysis	Papi, pfmon, perfometer	Vtune (Intel)
	Cluster Administration	Deployment / Installation	Ka-Tools, C3, SystemImager	
		Distributed shell	Ka-Tools, gexec	
		Control & monitoring	Ganglia, top	
OS	**System Administration**		Webmin, Nagios	
	File System		*Distributed:* NFS, PVFS *Local:* EXT2, EXT3, Reiserfs, xfs	
	Operating System		Linux	

The software components of an HPC cluster

2.2.1 Compilers

Compilers play an essential role in exploiting the full potential of Itanium 2 processors. These processors use EPIC (Explicit Parallel Instruction set Computing) that enables several instructions to be executed in parallel. This parallelism must therefore be detected and exploited at compiler level. For these reasons, Bull proposes Intel compilers (C/C++ and Fortran). Intel staff have all the skills and knowledge of the architecture to supply high performance products. A version of GNU compilers is also available for users familiar with this software.

2.2.1.1 Intel C/C++ Compiler

The current version of the Intel C/C++ compiler is version 7.1.
The main features of this compiler are as follows:

- Optimization of throughput of floating point instructions
- Optimization of inter-process calls
- Preloading of data
- Conditional instruction prediction
- Speculative loading
- Optimization of the software pipeline

This compiler complies with ISO standard Ansi C/C++ and ISO standard C/C++. It is also compatible with GNU products. A GNU C object or source code can therefore be compiled with an Intel compiler. emacs and gbd tools can also be used with this compiler.

The compiler supports multithreading functionality:

- OpenMP 2.0 for C/C++ is supported. The compiler accepts OpenMP pragmas and generates a multithreaded application.
- Automatic parallelization: a compiler option detects parallelism (in particular in the computation loops) and generates a multithreaded application.

2.2.1.2 Intel Fortran Compiler

The current version of the Intel Fortran 95 compiler is version 7.1. The main features of this compiler (identical to the C compiler) are as follows:

- Optimization of throughput of floating point instructions
- Optimization of inter-process calls
- Preloading of data
- Conditional instruction prediction
- Speculative loading
- Optimization of the software pipeline

This compiler complies with the Fortran 95 ISO standard. It is also compatible with GNU products. emacs and gbd tools can be used with this compiler. It also supports big endian encoded files. Finally, this compiler allows the development of applications combining programs written in C and Fortran.
The compiler supports multithreading functionality:

- OpenMP 2.0 for Fortran is supported. The compiler accepts OpenMP pragmas and generates a multithreaded application.
- Automatic parallelization: a compiler option detects parallelism (in particular in the computation loops) and generates a multithreaded application.

2.2.1.3 GNU Compilers

Gcc, a collection of free compilers that can compile both C/C++ and Fortran is part of the installed Linux distribution.

2.2.2 Scientific Libraries

These are sets of tested, validated and optimized functions. They enable developers to avoid reinventing such subprograms time and again.

The advantages of these scientific libraries:
- Transportability
- Support for different types of data (real, complex, double precision, etc.)
- Support for different kinds of storage (banded matrix, symmetrical etc.)

Several libraries exist and are recognized by the scientific community. For example BLAS, LAPACK, etc.

BLAS = Basic Linear Algebra Subprograms

This contains linear algebraic operations that include matrixes and vectors. Its functions are separated into three parts:

- Level 1 routines to represent vectors and vector/vector operations.
- Level 2 routines to represent matrixes and matrix/vector operations.
- Level 3 routines mainly for matrix/matrix operations.

BLACS = Basic Linear Algebra Communication Subprograms

Blacs is a specialized communications library (by message passing). After defining a process chart, it exchanges vectors, matrices and blocks… It can be compiled on top of MPI or PVM.

PBLAS = Parallel Basic Linear Algebra Subprograms

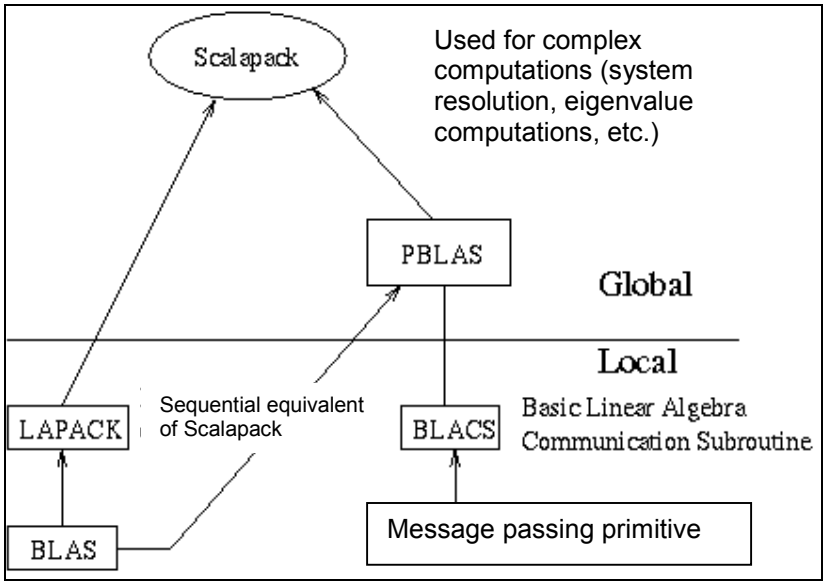
Pblas is the parallelized version of blas for distributed memory machines. It requires cyclic distribution by matrix block that the Blacs library offers.

LAPACK = Linear Algebra PACKage

This is a set of fortran 77 routines used to resolve linear algebra problems such as the resolution of linear systems, eigenvalue computations, matrix computations, etc. However, it is not written for a parallel architecture.

SCALAPACK = SCAlable Linear Algebra PACKage

This library is the scalable version of LAPACK. Both libraries use block partitioning to reduce data exchanges between the different memory levels to a minimum. Scalapack is above all used for eigenvalue problems and factorizations (LU, Cholesky and QR). Matrices are distributed using BLACS.



Interdependence of the different standard mathematical libraries

Local component routines are called by a single process with arguments residing in local memory.
Global component routines are synchronous and parallel. They are called with arguments that are matrices or vectors distributed over all the processes.

Libmkl:

This library, that has been optimized by Intel for its processors, contains the following libraries: blas, lapack and fft.

2.2.3 Parallel Applications

There are 2 types of parallel architectures. The first, called "distributed memory" considers that each processor has its own RAM space that cannot be accessed by the other processors. The second, called "shared memory" enables processors to access the same area of RAM simultaneously and transparently.

For Parallel Distributed Memory Architecture

For this type of architecture, ***message passing*** is the key concept. Message passing is based on an idea that is both simple and logical - a process sends a message to one or more other processes that must receive it.

A message must contain the following information:

- The identifier of the sending process
- The identifier of the receiving process
- The type of data
- Its length
- The data itself

A message must be sent to a specific address and the receiving process must be able to order and interpret the message. This is managed by an environment such as PVM or MPI. A PVM or MPI application is a set of autonomous processes that each execute their own code and that communicate using functions from the library supplied by each of these environments.

MPI = Message Passing Interface

MPI is based on the concept of message exchange. It is an attempt to standardize communication libraries by passing messages.
It is a standard that currently has two versions of its specifications: MPI-1 and MPI-2. The second version adds instructions for parallel input/output (MPI-IO), the C++ interface and parallel debugging. MPI provides development interfaces for C, C++ and Fortran.

Several implementations with very different philosophies exist, the main ones being Lam-MPI and MPICH.

- LAM-MPI:
  www.lam-mpi.org

  The library was developed by the University of Notre Dame (Indiana). This implementation is based on a system of daemons running on each node and ensuring messages are sent and received. It is also possible to launch different processes that communicate between themselves. The typical example is a "master" process with several "slave" processes. They are all different, communicate via MPI and are launched on several different nodes.

- MPICH
  www-unix.mcs.anl.gov/mpi/mpich
  MPI-Chameleon is developed jointly by the University of Mississippi and Argonne National Laboratory. Like LAM-MPI, it is an implementation of MPI.

  MPICH is based on "devices" that condition message exchange. They have a direct impact on performance. Several devices are available, each with its own particularities. The main ones are:

  - ch_p4 - for inter-machine (inter-node) liaison using TCP/IP.
  - ch_shmem - for fast liaison between tasks running on the same machine by using a shared memory for message passing.

PVM = Parallel Virtual Machine

www.csm.ornl.gov/pvm/pvm_home.html

PVM is software developed by Oak Ridge National Laboratory that makes it possible to use a set of Unix workstations linked by a network like a parallel machine.

PVM makes it possible to use a set of Unix workstations linked by a network like a parallel machine. This set of workstations is a virtual parallel machine. The machines may be distributed geographically. They may be heterogeneous, i.e. composed of different processors working under operating systems that may also differ one from another. Once installed, PVM enables different machines to communicate between themselves using messages exchanged by their processes.

PVM is made up of two elements:

1) A programming interface available for C and Fortran. This library contains the primitives required for generating processes running in parallel on several machines.

2) The daemon of the PVM virtual machine is a process running on each node in the cluster. The daemons interexchange messages with instructions supplied by the primitives. These messages are used to load programs on the various nodes of the cluster, to set parameters, collect results, etc.

PVM has the following features:
It is public domain software
It is easy to install
It is very flexible:

- it can run on a great variety of different architectures and machines
- it runs on LANs and other networks (LAN, WAN or a combination of both)
- it is the user's application that decides where and when the components are to be executed and that determines the controls and dependencies
- it allows programming in various languages (essentially C and Fortran)
- the virtual machine is easy to define and to modify

For Parallel Shared Memory Architecture

OpenMP = Open Multi Processing
www.openmp.org
Defined by a group of the world's major computer hardware and software manufacturers, OpenMP is a portable programming interface" that facilitates development of parallel applications for shared memory machines.

The OpenMP parallelization approach, with shared memory, is very different to the one used by MPI with distributed memory. There is just one instance of the program, run in parallel on several processors. Instructions inserted in the program manage the distribution of computation among the processors.

This standard defines directives for Fortran, C and C++ compilers, a "routines" library and environment variables.
The instructions extend the sequential programming possibilities of the compilers by defining the parallel regions of the program, how tasks are to be shared, the synchronization points and managing the sharing (or not) of the data.
The "routines" library and the environment variables manage the program's environment once it has been executed.

The main features of this standard are as follows:

- Scalability
- Ease-of-use - incremental parallelism
- Portability to SMP
- High level parallelization
- Flexibility to express different types of parallelization
- Performance orientated

2.2.4 File Systems

Two types of file systems exist - local file systems and distributed file systems. The former are used locally on a machine to give access to the files on the disk. The latter are required for accessing remote files via the network. They also have locking systems to maintain consistency between the various users. For example, if several people are working on the same file, this type of file system will manage this "simultaneous" work.

Above these two types of file systems there are also parallel file systems that share the files over several disks, thus "parallelizing" access to different parts of the file.

Local File Systems

These file systems can be split into two 2 groups - those with journaling and those without it.

Without Journaling

We can say that in this case you work without a "safety net". i.e. if your machine crashes, everything needs to be checked before you can use it again.
This the case for Ext2,for example, a standard Linux file system.

With Journaling

Most of today's file systems use journaling techniques borrowed from the world of databases in order to improve incident recovery. Transactions are written sequentially to a part of the disk called the "log" before being written to the disk in their definitive place in the file system.
If the machines crashes, consistency is maintained and checking can be performed rapidly.

- ReiserFS
  - www.namesys.com
    This is a system with several features: It allows for frequent system stoppages and manages files by b-tree. This is extremely useful when there are a lot of small files. It also optimizes storage of small files.

- XFS - eXtended File System
  - oss.sgi.com/projects/xfs
    XFS, widely recognized as a very high performance 64 bit file system, ensures the system restarts rapidly after a crash and can handle extremely large file systems.

- Ext3
  This file system is ext2 with journaling.

Distributed File Systems

Scientific computation often requires access to very large data files. The time it takes to access such files has a major influence on the speed of the program's execution. The ideal scenario would be to have a program running on a single machine with a high performance hard drive. However, the very principle of clusters is to share computation over all the nodes in a network. Access to remote file systems is therefore unavoidable. It is here that distributed file sharing systems come into play.

NFS - Network File System

A system working in client/server mode that authorizes access to a directory for several clients distributed over a network. It therefore allows data to be shared among the cluster nodes. The limits for the directory and file sizes are those of the server and for the access speeds those of the network.

Parallel File Systems

PVFS - Parallel Virtual File System

PVFS provides a file system shared over a standard file system such as ext2 or ReiserFS. The principle of PVFS is a simple one - instead of storing a file on a single disk, forcing remote machines to perform sequential queries on the network, the file is stored on several disks on the network.

With Linux, "The Parallel Virtual File System" is an implementation of the PFS (Parallel File System) standard.
It provides high performance and a parallel file system for cluster machines with:
A domain of unique names
Transparent access

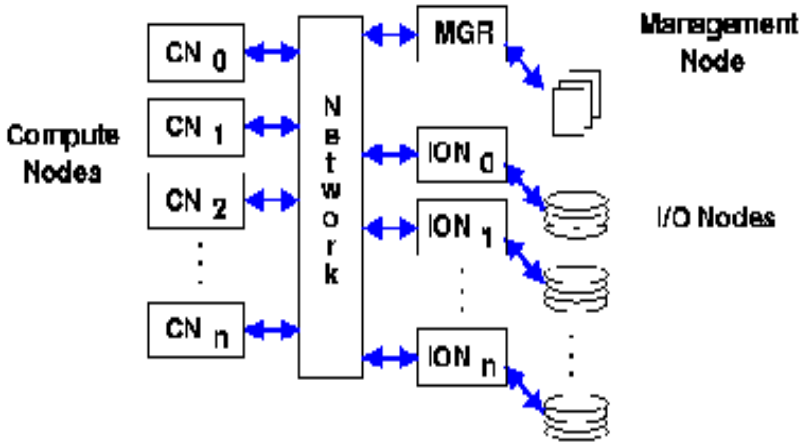
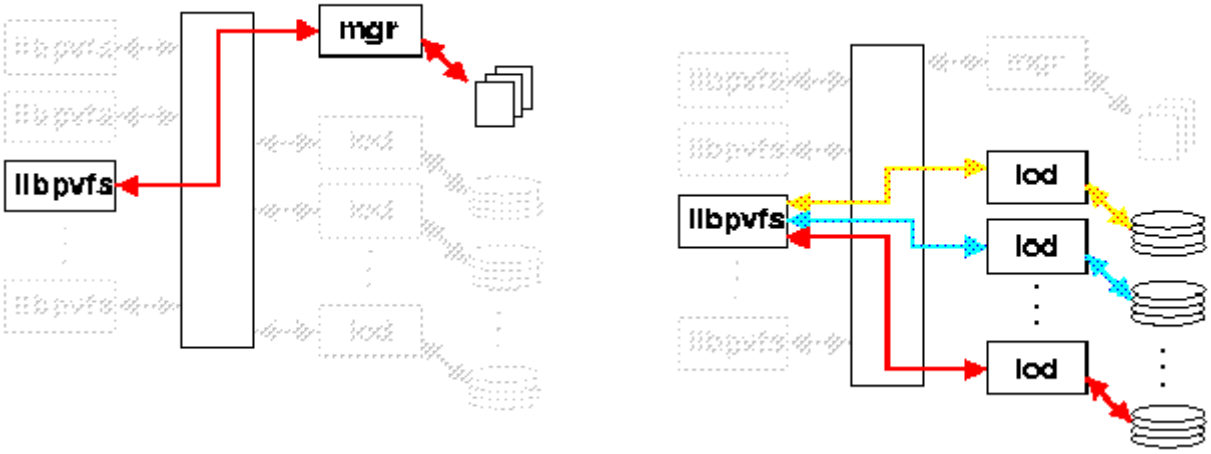


Diagram showing Compute Nodes and Storage Nodes



The diagram shows access to data location meta data and data flows.
It shows the process for initializing transfers, followed by transfers without interfering with the manager. This mechanism gives good performance.

2.2.5 System Administration Tools

These types of tools are required, even for machines that are not in a cluster. They are used, for example, to ensure the machines are running properly, to configure networks and manage users.

2.2.5.1 Webmin

Webmin has a Web interface for the administration of the machine's system and the network. It can be used for such tasks as configuring system parameters, managing users and services, routing, file sharing, etc. on Unix type machines.

The graphic interface of Webmin makes it easy to configure a large number of things. Webmin installs itself on startup or by adding packets. Once the server is installed, you can set up its security and access rights.
You will also be able to install numerous other servers such as:

- The http server: Apache
- The file server: Samba
- Database servers: MySQL and PostgreSQL
- The DNS server: Bind
- The DHCP server

For the network, you will be able to specify the parameters for a "ping", a "traceroute", the DNS (lookup), and a file search (Whois). For hardware, you will have access to the network, printers, disks (for partitioning) and for setting up RAID.

2.2.5.2 Nagios

You can access Nagios® at: www.nagios.com

2.2.5.2.1 Description

Nagios® is a monitoring program for hosts, networks and other services. A monitoring and prevention tool, it sends notifications to administrative contacts if warning thresholds are reached and/or if incidents occur on the network or servers. Nagios therefore assists administrators by informing them of problems so they can help their customers, users and managers to avoid disruption. Nagios was designed for Linux, but is also works well on other Unix systems.

- The Nagios monitoring daemon intermittently tests the hosts (servers) and services as you have defined them using external plugins. These plugins are external programs. They send "status" information to Nagios.
- When a problem occurs, the daemon can also send notifications to "contacts" (generally administrators) via different means: e-mail, SMS, ‘instant message’, etc.

- Nagios stores the information coming from the plugins in files giving the current status of hosts and services, in log files and reports.
- These can be accessed directly but can also be viewed via a Web browser.

2.2.5.2.2 Technical Characteristics

- Monitoring of network services (SMTP, POP3, HTTP, NNTP, PING, etc.).
- Monitoring of server resources (processor load, disk and memory use, smooth running of processes, etc.).
- Simple design of the plugins so you can easily develop your own host and service tests.
- Possibility of defining a network hierarchy to detect and make the distinction between hosts that are down and those that cannot be reached.
- Contact notifications when service or host problems occur and get resolved. Notifications via email, pager, or other user-defined method.
- Optional escalation of host and service notifications to different contact groups.
- Ability to define event handlers to be run during service or host events for proactive problem resolution.
- Support for implementing redundant and distributed monitoring servers.
- External command interface that allows on-the-fly modifications to be made to the monitoring and notification behavior through the use of event handlers, the web interface, and third-party applications.
- Retention of host and service status across program restarts.
- Scheduled downtime for suppressing host and service notifications during periods of planned outages.
- Web interface for accessing Nagios documentation.
- Web interface for viewing current network status: notification and problem history, log file, etc.
- Ability to acknowledge problems via the Web interface.
- Simple authorization scheme that allows you restrict what users can see and do from the Web interface.

2.2.5.2.3 Licence / License

Nagios® is licensed under the terms of the GNU General Public License Version 2 as published by the Free Software Foundation. This gives you legal permission to copy, distribute and/or modify Nagios under certain conditions. Read the 'LICENSE' file in the Nagios distribution or read the online version of the license for more details. Nagios is provided AS IS with NO WARRANTY OF ANY KIND, INCLUDING THE WARRANTY OF DESIGN, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

2.2.5.3 NAGAT

- Nagat is the Nagios administration tool. It is a Web based solution written in php.
- Nagat can be used to configure Nagios – the host and service monitor.

2.2.5.4 NRPE

- NRPE stands for Nagios Remote Plugin Executor.

- NRPE is used to run local plugins (check_disk, check_procs, etc.) on the network's remote hosts monitored by Nagios:
- Nagios calls the check_nrpe plugin. This plugin sends a query to a local plugin installed on the remote host. This local plugin runs and then sends a status that is transmitted to Nagios via check_nrpe.
- For it to work, nrpe must be running on the remote host. It can be enabled as separate daemon or as an inetd or xinetd service.

2.2.5.5 Plugins

2.2.5.5.1 Principle

- Introduction

- Nagios has no internal mechanism for checking the status of a service, host, etc. It therefore uses external programs (called plugins) to run execution tasks. Nagios therefore runs a plugin every time it needs to test a monitored service or host. The plugins "do the necessary" to perform the required check and send the results back to Nagios. Nagios analyzes the result received from the plugin and implements the required measures (e.g. launching event managers, sending notifications, etc.)

- Plugins are separate from the logic of the main Nagios program.

- Advantages

- This architecture enables you to monitor everything you can think of. If you can automate an element's monitoring process, you can supervise it with Nagios. A certain number of plugins have already been created to monitor elementary resources such as: processor load, free disk space and pings. If there are other things you would like to monitor, see the documentation on the Web on how to write plugins.

- Extent of Nagios

- Nagios has no idea what you are monitoring. You can monitor network traffic statistics, error rates, room temperature, CPU voltage, fan rotation speed, processor load, free disk space, etc. Nagios does not therefore trace graphs of value changes for the monitored resources. However, it does track their changes of status.
- It is the plugins that know exactly what they are monitoring and what to do…

- All plugins respecting the minimum development recommendation for this project have internal documentation. This documentation can be displayed by executing the plugin with the "-h" parameter ("--help" if long parameters are enabled).

- For example, if you want to know how the check_http plugin works or which parameters it accepts, try:

    - Either: ./check_httpd --help

    - Or: ./check_httpd --h

- Examples of command definitions for services

- The main plugin distribution includes a configuration file (called checkcommands.cfg) that contains examples of command and monitoring definitions for hosts and services using the latest plugins.
- The services to be monitored are defined in the services.cfg. configuration file. This then relies on checkcommands.cfg, to know which plugin to execute.

```
E.g. For a service
define service{
        use                             generic-service
        host_name                       linux_serv1
        service_description             HTPP-serv1
        ……..
        normal_check_interval  10
        contact_groups          linux_admins
        ….      ……
        check_command                   check_http_com
}

Associated commands and Plugins:
define command{
        command_name                    check_http_com
        command_line            $USER1$/check_http –H $HOSTADDRESS$
}
```

- Where check_http is a plugin shipped with Nagios. It must be compiled when Nagios is installed.

2.2.6 Cluster Administration Tools

Control and Monitoring Tools

- In general terms, the goal of these kinds of tools is to monitor the activity and behavior of the cluster. They do not directly increase the machines' possibilities but they make their management easier. A control and monitoring tool can indicate the following:
  - Whether one of the cluster's nodes is not working
  - Network overload
  - Use of the cluster

    …

- These tools are used by 2 types of people:

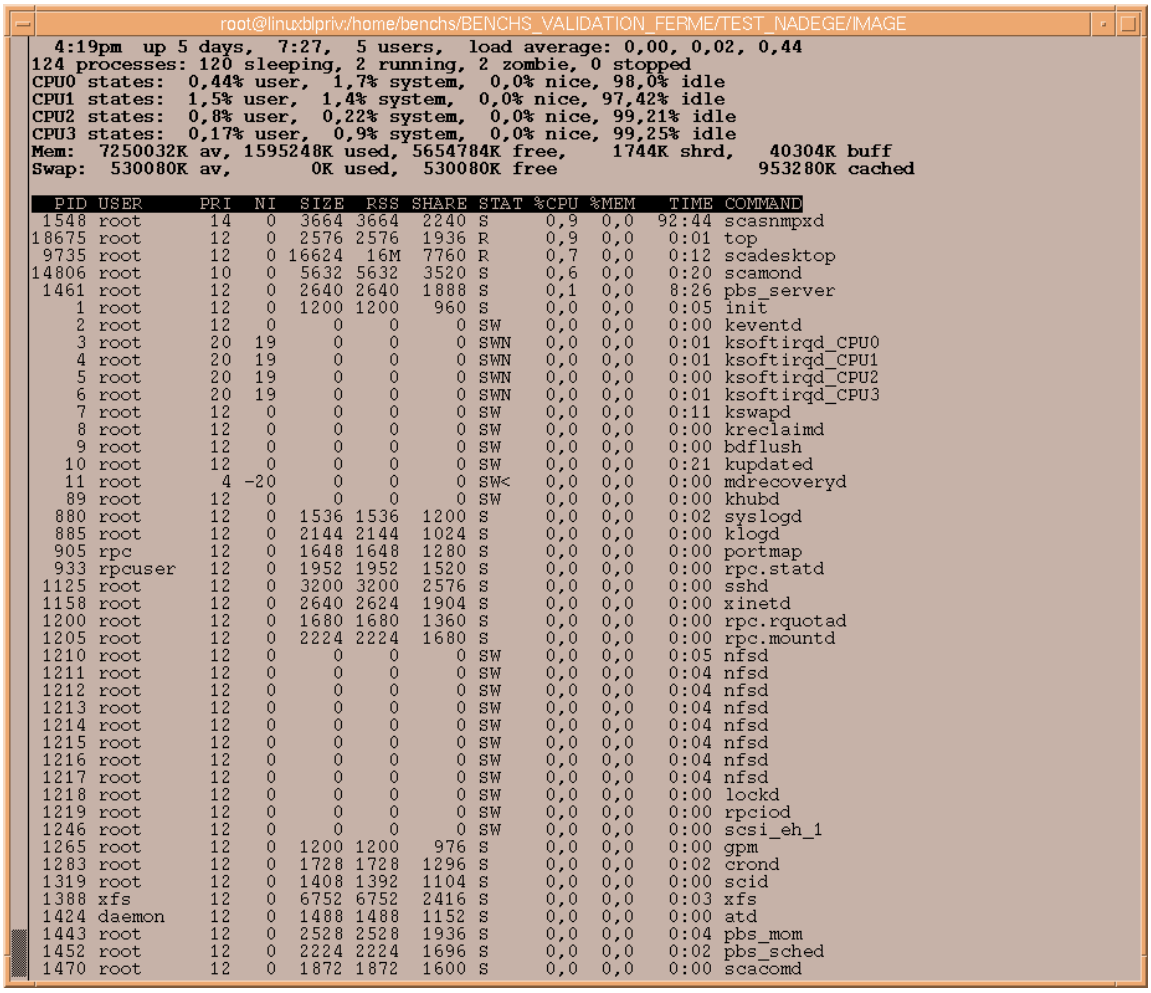
- The system administrator whose aim is to:
  - Maintain the cluster in good working order
  - Diagnose and, if possible, resolve abnormal behavior
- The user:
  - Monitor the impact of executing a program with the aim of optimizing it.
  - See if an application is running smoothly and that all the processes have been launched correctly.

- **Ganglia:**

  - Ganglia is a cluster monitoring tool. It has a graphic interface in Web format that provides graphic views of information concerning cluster use.
  - It is also a monitoring tool since it notifies you if a node has stopped working.

- **Top:**

  - This Unix command provides different pieces of information in real time on the execution of processes and use of a node.
    - The top command can:
    - Provide a list of launched processes
    - Show each process's machine time from the most "greedy" down
    - Determine which processes need a lot of resources
    - Observe the system's life
    - Indicate the amount of memory and percentage of swap used ….
    - …



Example of the top command's output

Tools Launching the Same Command on Several Nodes

A "distributed shell" is essential when using a cluster. There are always instructions you want to carry out on different nodes in the cluster without needing to do it manually on each node in turn. Several tools have this functionality.

Ka-Tools:

This tool was created to make installing and using a Linux cluster easier. All its components are scalable.
The ka-run module of this component is used to run a command on the desired machines.

Gexec:
Gexec has the same functionality.

C3 = Cluster Command and Control:
C3 implements a certain number of commands that assist with cluster administration.

Deployment Tools

This term designates software components used to install a distribution or packages on several machines at once. For large clusters, such a tool is essential to avoid redoing the same installation a large number of times. Today, several tools meet this need and can install a Linux distribution, for example, on several hundred machines in just over ten minutes.

Ka-Tools:
This tool's ka-deploy module is used to clone nodes.

SystemImager:
This is used to automate Linux installation on a cluster of identical machines. It provides for software distribution and the configuration and update of the operating system.

2.2.7 Performance Analysis and Profiling Tools

2.2.7.1 Their Role

To understand a machine's activity and compare its performance with another, the hardware has counters that count event occurrences associated with the processor's functions: Number of Floating Operations, number of wait cycles to access the memory.

The advantage of monitoring these events is to be able correlate the structure of a program and its suitability to the underlying hardware architecture. Manufacturers can therefore evaluate their hardware for a given program, software developers can obtain information useful for explaining poor performance of parts of the code and therefore focus their optimizations depending on the target hardware.

Tracking these events is particularly useful in the case of large programs such as compilers, benchmarks or performance modeling programs.

2.2.7.2 What Tool for What Purpose?

2.2.7.2.1 In the World of Open Source

A whole hierarchy of tools exists.

- If what you want is an overview of all the bottlenecks in your software application, what you need is simple system profiling: **vprof** can meet this need.
- If, however, you want to characterize your hardware architecture or if you want to compare the behavior of two similar programs with respect to hardware performance, then **pfmon** will give you global answers without touching the code.
- If, finally, it is fine details that interest you at specific points of your program that you know perfectly, **PAPI** will enable you to program the instrumentation you need exactly.

To summarize the increasing level of available functionality:

Object	**Nature**	**Availability**	**Functions**
perfmon.c	Source module of kernel: Under arch/ia64/kernel	http://www.kernel.org version >=2.4.18 with patches ia64 valid for kernel configuration by CONFIG_PERFMON=y	Procedures for managing hardware registers
pfmon	Utility for implementing perfmon of the kernel. Composed of a binary: pfmon and a library: libfpm	ftp://ftp.hpl.hp.com/pub/linux-ia64/ pfmon-2.0.ia64.rpm	Exploitation of hardware events (by calling perfmon.c) **Command interface without modifying the code**

PAPI	API: Program interface integrating the pfmon library (libpfm)	http://icl.cs.utk.edu/projects/papi/software/ papi 2.3.4	Management of hardware events by calling specific functions at a higher or lower level as desired. Allows an unlimited number of counters, **places itself in the program code** at the precise points to be measured.
cprof	Profiling tool over papi	http://aros.ca.sandia.gov/~cljanss/perf/vprof/	System profiling or monitoring of hardware event by PAPI. **Simple link editing** applying without modifying the source code. At execution, the application generates a data file exploited by the cprof command.
vprof	Visual version of cprof	http://aros.ca.sandia.gov/~cljanss/perf/vprof/	Same function as cprof but results presented in a graphical window.

2.2.7.2.2 In the World of Commerce

VTUNE:

Intel sells a tool for Windows that is capable of sampling events collected on a remote Linux server. The main advantage of this tool is its easy-to-use graphical interface.
Its drawback is the network link required between the two environments and the additional Windows workstation.

VAMPIR:

VAMPIR is a graphical profiling tool marketed by PALLAS. It is used to view and analyze the performance of MPI codes.

2.2.7.3 Examples Comparing Use of Different Profiling Tools

2.2.7.3.1 PFMON

Pfmon is the best tool for obtaining an elementary number that characterizes a program rapidly and, without programming. For example, the number of floating point operations of program test2:

```
pfmon -e FP_OPS_RETIRED test2
2 421 914 081 FP_OPS_RETIRED
```

You can capture 4 events at a time and have a complete output (with header). For example, the command:

```
pfmon --with-header -k -u -e
        L2_MISSES,L2_REFERENCES,L3_MISSES,IA64_INST_RETIRED --
        outfile=result_pfmon/pfmon_miss_16.2081 runspec ……..
```

gives:

```
#
# date: Fri Feb 21 02:16:14 2003
#
# hostname: chircane
#
# kernel version: Linux 2.5.59 #2 SMP Wed Feb 12 06:17:00 PST 2003
#
# pfmon version: 2.0
# kernel perfmon version: 1.3
#
# page size: 16384 bytes
```

```
# CLK_TCK: 1024 ticks/second
# CPU configured: 4
# CPU online: 4
# physical memory: 4189208576
# physical memory available: 2908225536
#
# host CPUs:  4-way 997MHz Itanium 2 (McKinley, B3)
#       PAL_A: 0.7.31
#       PAL_B: 0.7.36
#       Cache levels: 3 Unique caches: 4
#       L1D:    16384 bytes, line  64 bytes, load_lat   1, store_lat   3
#       L1I:    16384 bytes, line  64 bytes, load_lat   1, store_lat   0
#       L2 :   262144 bytes, line 128 bytes, load_lat   5, store_lat   7
#       L3 :  3145728 bytes, line 128 bytes, load_lat  12, store_lat   7
#
# captured events:
#       PMD4: L2_REFERENCES, kernel+user level(s)
#       PMD5: L2_MISSES, kernel+user level(s)
#       PMD6: L3_MISSES, kernel+user level(s)
#       PMD7: IA64_INST_RETIRED, kernel+user level(s)
#
# monitoring mode: per-process
#
#
# instruction sets:
#       PMD4: L2_REFERENCES, ia32/ia64
#       PMD5: L2_MISSES, ia32/ia64
#       PMD6: L3_MISSES, ia32/ia64
#       PMD7: IA64_INST_RETIRED, ia32/ia64
#
#
```

```
# command: pfmon --with-header -k -u -e
            L2_MISSES,L2_REFERENCES,L3_MISSES,IA64_INST_RETIRED --
            outfile=result_pfmon/pfmon_miss_16.2081 runspec -c il420-linux-v7 --iterations=1 --
            tune=base -I --noreportable --rate --users 16 applu
#
#
#
             4234016 L2_MISSES
            69799519 L2_REFERENCES
             527948 L3_MISSES
            772924788 IA64_INST_RETIRED
```

2.2.7.3.2 PAPI

The same thing can be obtained with PAPI but in a much more discriminating manner, by modifying precise sequences of the program:

Three interface levels can be distinguished:

a) The simplest or very high level:

- Use of the PAPI_flops routine, that gives the real time, the CPU time and the number of floating operations per second.

In a Fortran program:

```
#include "fpapi.h"

real real_time,cpu_time, mflops
 integer*8 fp_ins

call PAPIf_flops( real_time, cpu_time, fp_ins, mflops, ierr )
```

....processing....

```
call PAPIf_flops(real_time,cpu_time, fp_ins, mflops, ierr)
```

...results:

```
      write (6,120) real_time, cpu_time, fp_ins,mflops
120   format (//'PAPI result'/'       real time (secs) :', f15.3,
     $  /'       CPU time (secs) :',f15.3,
     $  /'floating point instructions:', i15,
     $  /'                    MFLOPS:', f15.3)
```

b) The high-level interface (non thread-safe):

- Include PAPI definitions:

```
      #include "fpapi.h"
```

- Declare the events to count and error retrieval:

```
      integer events(2),numevents, ierr
      character*PAPI_MAX_STR_LEN errorstring
```

- Declare the variables to retrieve the counters

```
      integer*8 values(2)
```

- Set each event to the type wanted (cf fpapi.h)

```
 numevents= 2
events(1)=PAPI_FP_INS
 events(2)=PAPI_TOT_CYC
```

- Start the counters and test the report

```
call PAPIf_start_counters(events, numevents, ierr)
  if ( ierr .NE. PAPI_OK ) then
    call PAPIF_perror(ierr,errorstring,PAPI_MAX_STR_LEN)
    print *, errorstring
  end if
```

- Start processing, then read and reinitialize the counters without stopping them

```
   call PAPIf_read_counters(values,numevents,ierr)
if ( ierr .NE. PAPI_OK ) then
      call PAPIF_perror(ierr,errorstring,PAPI_MAX_STR_LEN)
      print *, errorstring
end if
```

- You can also accumulate the counters by using the routine

```
   PAPIf_accum_counters(values,numevents,ierr)
```

- Continue processing then stop the counters:

```
   call PAPIf_stop_counters(values, numevents, ierr)
if ( ierr .NE. PAPI_OK ) then
      call PAPIF_perror(ierr,errorstring,PAPI_MAX_STR_LEN)
      print *,errorstring
end if
```

c) The detailed low-level interface (thread safe)

This is used to handle predefined events (there are 100 of them in PAPI!) as well as totally native events, defined by their coding in the registers.

- Start by initializing the library with:

```
   call PAPIf_library_init(ierr)
```

- Then manage sets of events that will be used together with:

```
call PAPIf_create_eventset(es,ierr)
```

- Next, introduce events into this set with:

```
call PAPIf_add_event( es,  PAPI_TOT_CYC,, ierr )
```

- Start, read and stop a set

```
call PAPIf_start(es)
call PAPIf_read(es)
call PAPIf_accum(es)
call PAPIf_stop(es)
```

See the PAPI user's guide for the exact interface.

2.2.7.3.3 VPROF

If you want to be able to view the distribution of the floating point rate per second on the test2 program, proceed as follows:
First of all, compile the program (test2.cc) en –g and link-edit a module to collect: vmonauto.o (to avoid inserting a call to vprof at the start and finish);
next do a link edit with the vmon and PAPI libraries:

```
make test2
c++ -g -O2 -Wl,-static -o test2 test2.cc ../lib/vmonauto.o ../lib/libvmon.a –L
/opt/envhpc/papi-linux-ia64/lib
```

When you run the program, specify:

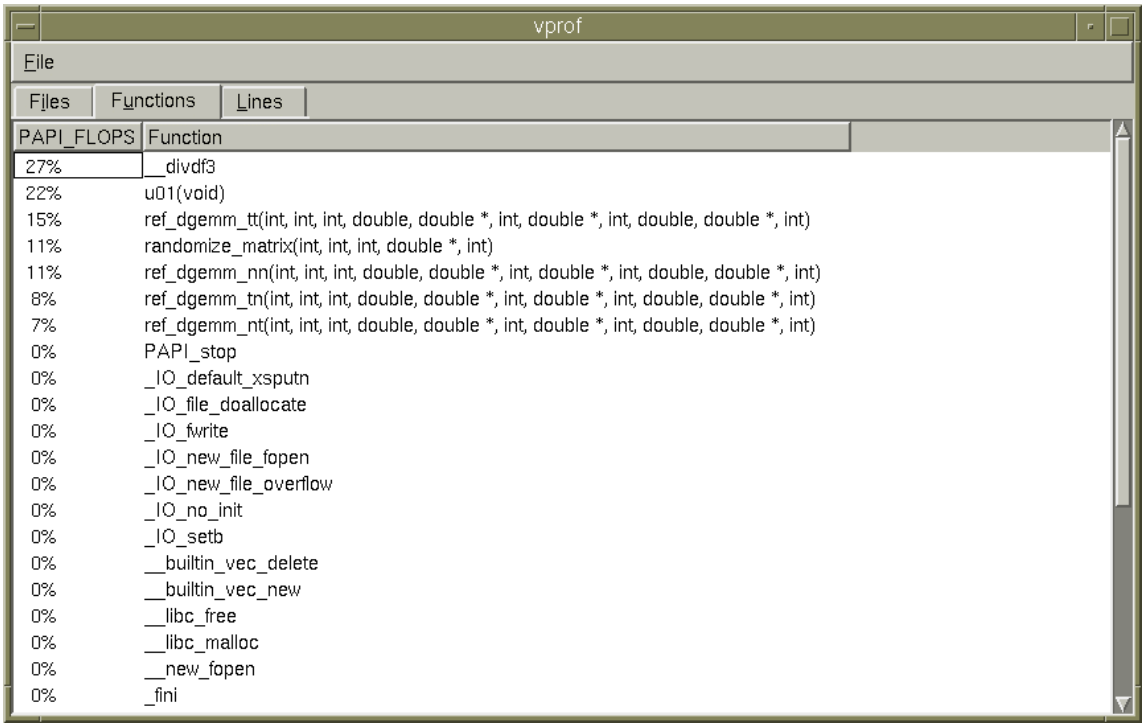
- Which metric is being studied: VMON=, (by default the system profile)
- The name of the file that will collect the information :VMON_FILE=, (by default vmon.out)

```
VMON=PAPI_FLOPS test2
```

Next, you can call vprof after having defined the Display, and specify par –d the source directory if it is not the current directory.

```
export DISPLAY=.... :0.0
../bin/vprof –d. test2
```

This is the window you get, that declines the various functions with the percentage of floating point operations per second.



The cprof command gives the same information but in text form.

The following example is taken from a different test run on the same program:

```
 ../bin/cprof -r test2
Columns correspond to the following events:
  PAPI_FLOPS - Floating Point instructions per second (1145301 events)
Function Summary:
  20.6% __divdf3
  15.3% ref_dgemm_tn(int, int, int, double, double *, int, double *, int,
double, double *, int)
  14.8% ref_dgemm_nn(int, int, int, double, double *, int, double *, int,
double, double *, int)
  13.9% u01(void)
  12.2% ref_dgemm_tt(int, int, int, double, double *, int, double *, int,
double, double *, int)
  11.9% randomize_matrix(int, int, int, double *, int)
```

```
  11.3% ref_dgemm_nt(int, int, int, double, double *, int, double *, int,
double, double *, int)
```

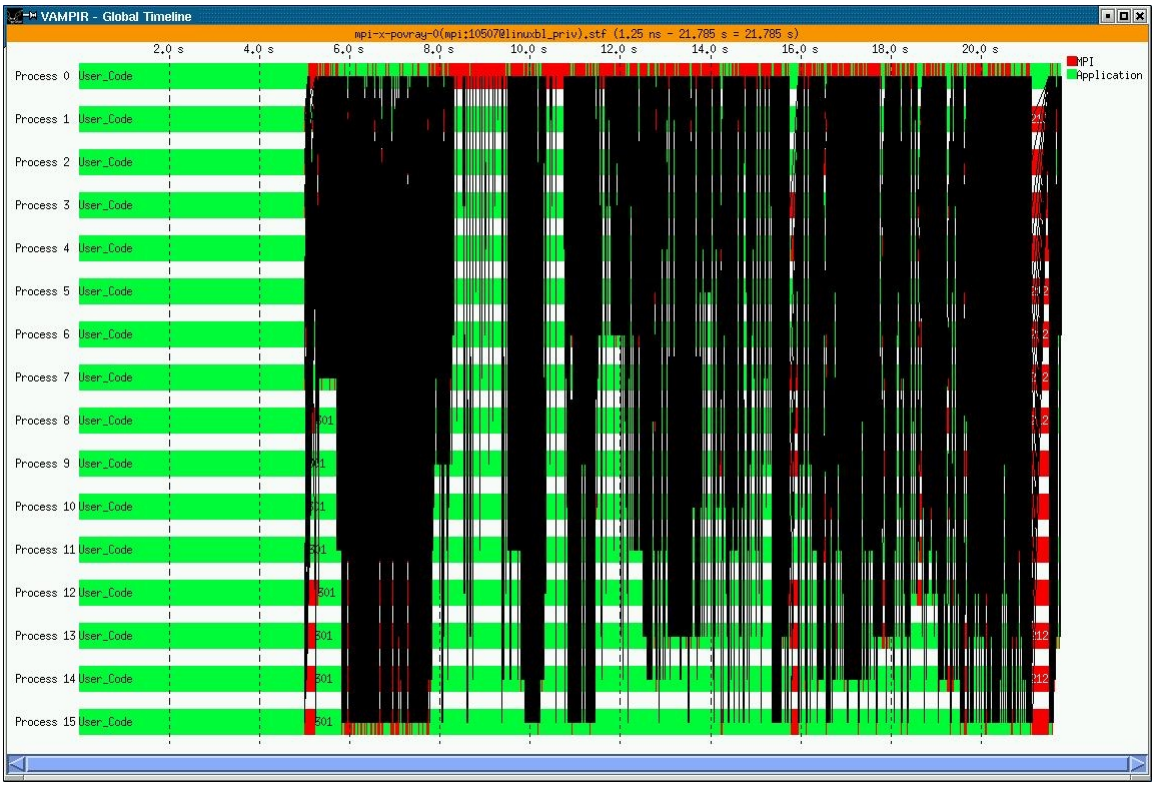
Note:
In some cases, you will notice segmentation errors with PAPI in the form of a shared library; in this case try a static link-edit.

2.2.7.4 VAMPIR

VAMPIR is a graphical profiling tool marketed by PALLAS. It is used to view and analyze the performance of MPI codes.

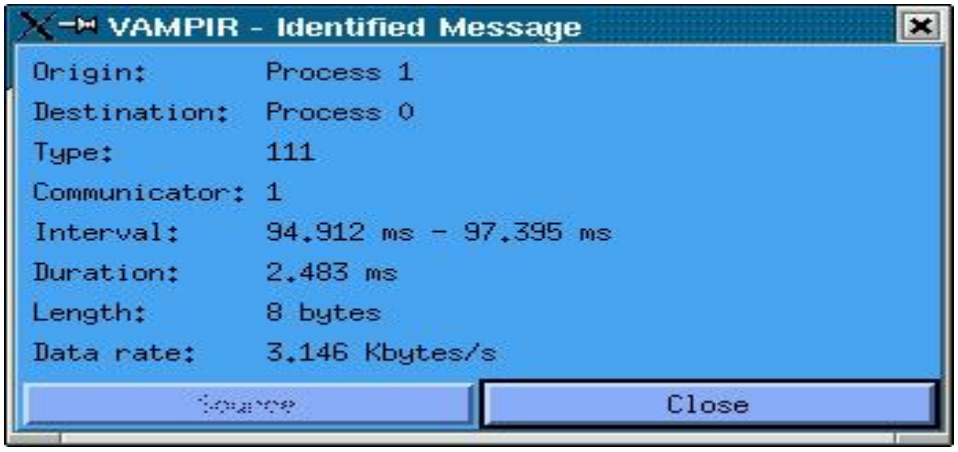
After creating trace files using VampirTrace, simply start Vampir to interpret them graphically.

Here are some examples of the graphs available:



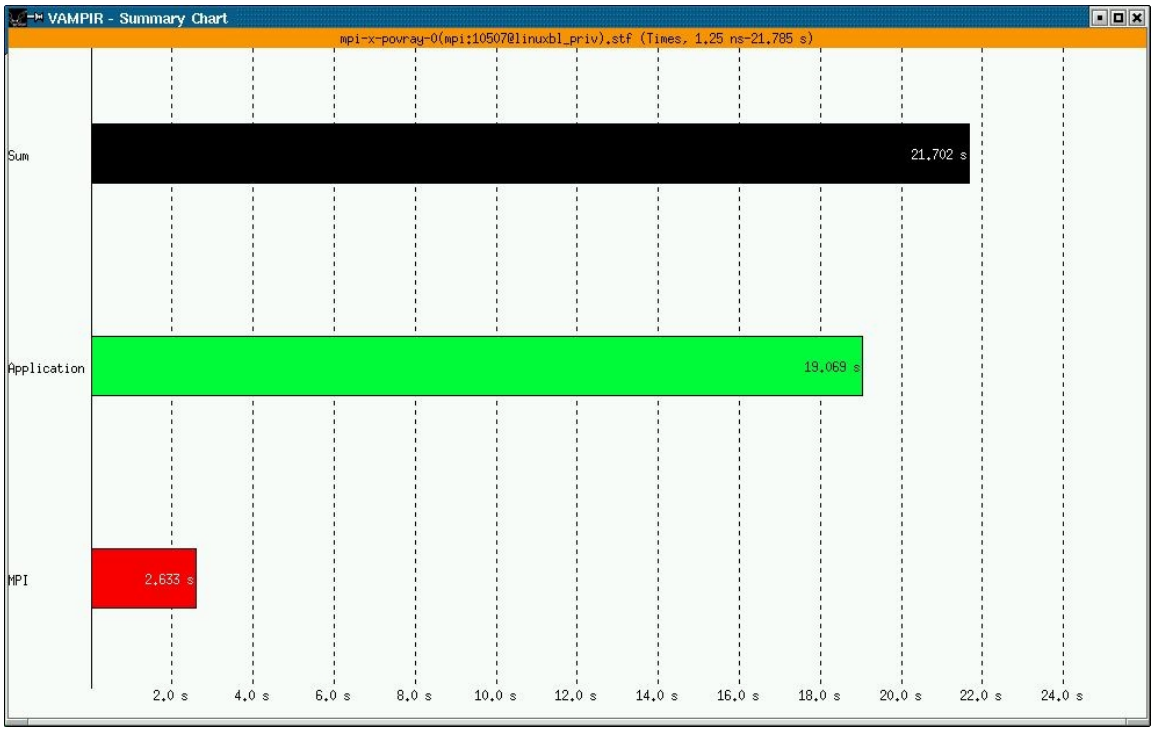
Overview of an application's MPI messages

This graphic shows all the MPI messages occurring during the application. You can zoom to different parts and have more precise information on a particular message by clicking it.



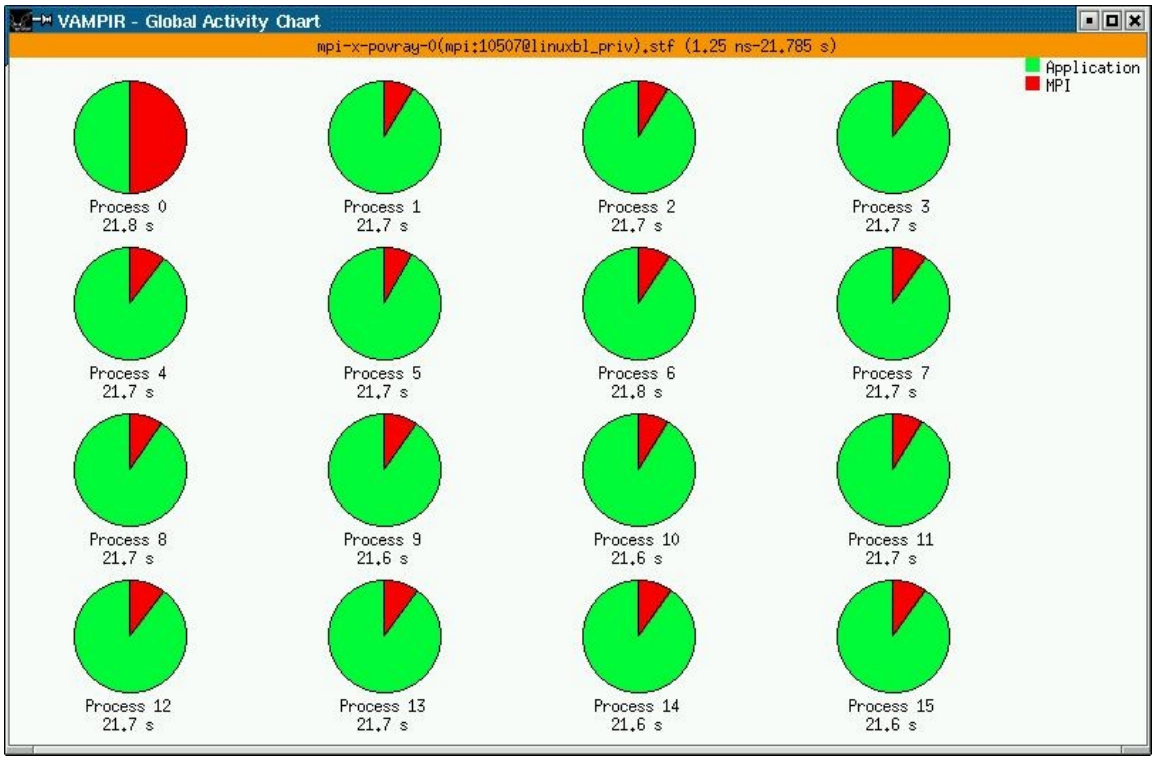
Information about an MPI message

Information about an MPI communication



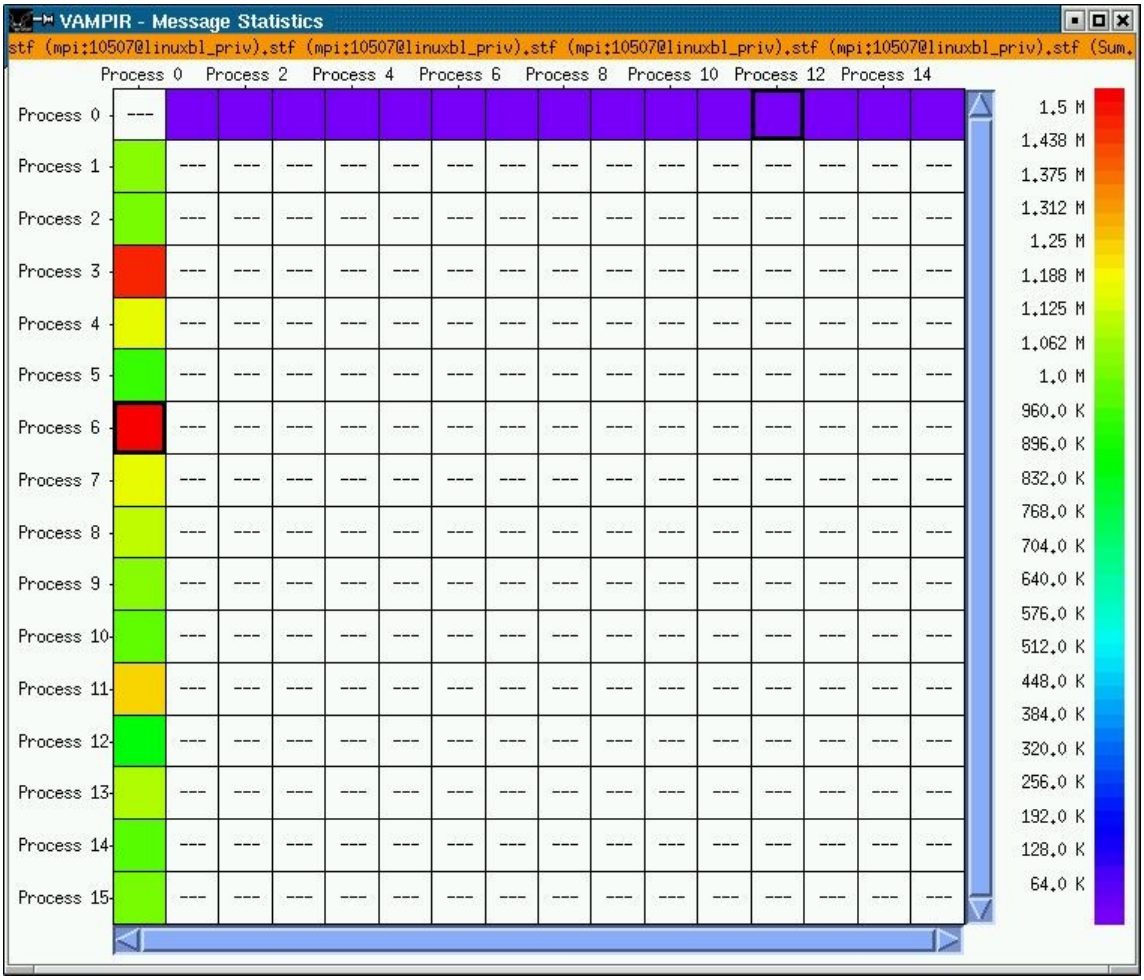
Time spent on different activities

The darker color is the total time spent on application activities and the lighter, the time spend on MPI communications. In this example, we can see that the program spends more time in application activities than in MPI communications.



Time spent by each process in application versus MPI activities

This graph can be used to make comparisons between the various processes and to determine those that make the most communications.



Statistics for the exchange of messages between the processes

This last example shows the quantity of messages exchanged between each of the processes. You will note that in this case the processes only communicate via process 0.

All these graphs give an insight into the possibilities of Vampir and the way it can be used. The information can then be used to optimize the application, to find the best number of processes for a given problem, etc.

2.2.8 Debuggers

There are two types of debuggers: symbolic ones and non symbolic ones. A symbolic debugger gives access to a program's symbols.

- You can access the lines of the source file.
- You can access the program's variables by name, whereas with a non-symbolic debugger, you only have access to the lines of the machine program and top physical addresses.

2.2.8.1 GDB

GDB is the acronym for Gnu DeBugger. It is a powerful debugger with a fully command line interface.
GDB is so well liked that it is also found in graphic interfaces like XXGDB or DDD. GDB is published as part of the GNU license. It supports parallel applications and threads.

2.2.8.2 DBX

dbx is a symbolic textual debugger. It works in optimal fashion on code compiled with option -g (option -g3 recommended) and without optimization (option -O0 recommended). The effect of this is to add debugging information to the object code.

2.2.8.3 TOTALVIEW

http://www.etnus.com/
TotalView™ is a symbolic debugger working with windows, for C, C++ and Fortran(77, 90 and HPF). It can debug PVM or MPI applications. TotalView™ fully supports multi process programs and works just as well on monoprocessor, SMP, clustered, distributed and MPP systems.
TotalView™ accepts new processes and threads exactly as they were generated by the application, whatever the processor they execute on. You can also connect to a process started up outside TotalView™. Data tables can be filtered, displayed and viewed in order to monitor program behavior. Finally, you can descend ("call the components and details of…") into the objects and structures of the program.
By simply clicking, the context-sensitive help provides you with basic information from the manual on screen. The full documentation is available on the etnus site: http://www.etnus.com/

2.2.9 Task Sharing Tools

Merely grouping several machines on a network is not enough to constitute a real *cluster*. Software is required to manage this and therefore make the best possible use of the computation power of these independent nodes. The goal of a task management system is to manage this computation power in order to give the

best performance possible and to enable several users to run tasks simultaneously.

Job management systems generally have the following functionality:

- A user interface
- A scheduler
- A resource manager
- Log and configuration files stored on the same machine
- A secure environment

The user interface is used to execute both local and remote tasks. These are launched via queues. Users can:

- Specify the resources required for the tasks they want to run.
- Delete the execution of a task.
- Suspend or resume the execution of a task.
- Find out the status of a task in progress.

To do this, users have either a command line or graphical interface or both.

A *scheduler* is used to define a *scheduling policy* for submitting tasks. These can be put in an order of priority. Priorities are generally established according to the following criteria:

- Queuing time before execution
- Required resources (number of nodes, execution time, memory, disk, etc.)
- Type of task (interactive, parallel, batch, etc.)
- User's identity

The resource manager is used to allocate resources, to find out the status of resources, to collect task execution information and from it to apply the scheduling policy.

OpenPBS:

http://www.openpbs.org
OpenPBS is a job management system to share machine resources so they best suit the needs of the users (CPU time, number of processors, memory space, etc.). It provides tools to build, submit, process and monitor sequential and parallel jobs.

LSF(Platform):

http://www.lerc.nasa.gov/WWW/LSF/lsf_homepage.html
This is a proprietary batch manager. It is a set of tools intended for sharing the CPU load over the machines of a cluster in a way that is totally transparent to the user.
Features of LSF include:

- Use of queues
- Choice of least "loaded" machine
- Concept of priority jobs
- …

Maui Scheduler:

http://mauischeduler.sourceforge.net/
Maui is a scheduler that can plan when and where tasks are to be started. It can be used with a load sharer (or resource manager) like PBS.

Chapter 3. Installing a Cluster

3.1 Hardware Administration

3.1.1 Implementing the Administration Node

The administration node representing the part of the cluster visible from the outside, its startup and hardware administration are normally handled by the console directly connected to it.
The operation of the rest of the cluster is therefore dependent on the availability of this administration node.
It must therefore be started up first, installed with its operating system and configured.
The hardware administration of the other elements follows and uses its console or any other console connected to it from the external network.

3.1.2 Starting up and Checking Other Nodes

The hardware administration of the other cluster nodes is done via the serial interconnection network.
On the administration node, a minicom terminal emulation program (or xminicom in an X window) must be used to gain access to the console of the other nodes.

This program is started by:

$ minicom n0x0y &
(n0x0y identifies the name of the node you want to access. This name is used as the suffix of file /etc/minirc.n0x0y created on the node to be administrated as shown below).
To exit mimicom, hit "CTRL A" then "X".

Configuring minicom:
minicom uses a separate configuration file for each node, located in directory "/etc/" and named "minirc.<node name >".
Each of these files sets the parameter values of each serial line.

Example of a four node cluster (admin, n0101, n0102, n0103):

```
$ cat /etc/minirc.n0101
# minicom default configuration file for node N0101
pr port             /dev/ttyD000
pu baudrate         115200
pu bits             8
pu parity           N
pu stopbits         1
pu mautobaud        Yes
pu statusline       disabled
pu hasdcd           No
pu minit
pu mreset
$

$ cat  /etc/minirc.n0102
# minicom default configuration file for node N0102
pr port             /dev/ttyD001
pu baudrate         115200
pu bits             8
pu parity           N
pu stopbits         1
pu mautobaud        Yes
pu statusline       disabled
pu hasdcd           No
pu minit
pu mreset
$

$ cat /etc/minirc.n0103
# minicom default configuration file for node N0103
pr port             /dev/ttyD002
pu baudrate         115200
pu bits             8
pu parity           N
pu stopbits         1
pu mautobaud        Yes
pu statusline       disabled
pu hasdcd           No
pu minit
pu mreset
$
```

3.2 SCALI SSP 3.0.1

Introduction:
This paragraph describes how to install SCALI SSP 3.0.1.
Check http://www.scali.com for the release of new updates.

Prerequisites:

- Linux already installed on each node.
- Cables connected as per the SCALI system guide".
- SSP_3_0_1 is supplied on a CD ROM, however you can also download it from http://www.scali.com/Download/ssp.shtml.
  If you want to download the software:

```
mkdir /home/scali
cd /home/scali
### Download SSP_3_0_1_Linux2_ia64.tar.gz  ###
tar xvzf SSP_3_0_1_Linux2_ia64.tar.gz
```

- The name of the front-end node must be "admin": Verify command "uname –n" displays "admin".
- Configure the Operating System as shown in the OS file (in subdirectory SSP_3_0_1/doc/SSP). **On all the nodes of the cluster:**
  - While logged in as root, insert all the node names in file */root/.rhosts:*

```
admin root
n0101 root
n0102 root
n0103 root
etc …
```

  - Insert all the node names in file */etc/hosts.equiv:*

```
admin
n0101
n0102
n0103
etc …
```

  - Add *rsh*, *rlogin* and *login* to file */etc/securetty*
  - Run the following commands:

```
chkconfig rsh on
chkconfig rlogin on
```

- Check you have at least 200MB available on /opt/scali on all nodes.
- Make sure you have received a temporary license by e-mail and have copied it to the front-end (fichier /opt/license.dat)

Installation:

Installation of the SCALI SSP interconnection software is done in several passes:

1. An initial pass to install the license, define the nodes (admin, n0101, n0102, n0103 etc …) and the functionality to be installed (OpenPBS) or not (Console Server, Power Switch Server, NIS/YP).
2. An intermediate phase to update the rpm and the SCI driver.
3. A second pass to finish and check the installation.

1) Installation: 1st Pass

- From the CD ROM (on the admin node), enter command:

```
mount –t iso9660 /dev/cdrom /mnt/cdrom
cd /mnt/cdrom
./install
```

- Or, from the files you have downloaded, enter command:

```
/home/scali/install
```

- Follow the installation procedure by answering the questions. Some steps make take a long time.
- For step "**Expert mode" select the default choice** ("**n**").
- For step "**Specifying frontend**", validate "admin" ("**y**").
- For the step "**Specifying node names**", list the nodes (order: admin, n0101, n0102, n0103 for 4 nodes).
- For step "**Setup license(s)"** select "**2) License file"** and give the full path "/opt/license.dat".
- For step "**Determine node categories**", you can exclude certain nodes that are not computation nodes (e.g. storage nodes). To do this:
  - When asked "Do you accept the above configuration ?", reply "n".
  - Select the nodes to be modified.
  - Only select category "eth_node" for the nodes to be modified

- For step "**Configuring SCI network**", for a cluster of 4 nodes, choose "2D torus", then give the number of nodes (2) connected to the rings of the L0 interface , then enter the number of nodes (2) connected to the rings of interface L1. This gives 2 x 2 = 4 (total number of nodes). Then choose, by default, the automatic method (choice "**1**").
- For step "**install the console server**" reply "**n**".
- For step "**install the power switch server**" reply "**n**".
- For step "**install OpenPBS queue system"** reply "**y"**, then enter the username "**linux**".
- The next three steps issue "WARNINGS". Ignore these by replying "i" (ignore). They are:
  - "Checking kernel versions"
  - "Checking HW support"
  - "Checking OS support"
- For step "**Are you using NIS/YP** .. » reply "**n**"
- Some of the next steps issue "WARNINGS". These are not abnormal the first time the installation software is run, so reply "i" (ignore). They are:
  - ScaSCI installation problems, "Checking SCI driver": These problems occur because the SCI driver does not load, which is not unusual at this stage of the installation.
  - "Checking SCI links"
  - "Testing SCI communication"

2) Intermediate Phase: Updating "rpm"s and the SCI module:

- Scali makes available updates for part of its software. You are recommended to install them and to use the "rpm update" procedure below:

- Scali installs everything on the front-end and on each of the nodes of the cluster and tests if everything is working (communication between nodes). However, the SCI driver supplied by default may not be suited to the installed kernel (see the "WARNINGS" displayed during the first phase of installation) and these tests will not work. Update the driver using the "SCI module update" procedure below.

Updating rpms:

- Download into /opt/scali/repository/Linux2.ia64 the updates available on site www.scali.com (menu: «Download->Software->SSP3.0 updates).
  - ✓ ScaMPI
  - ✓ ScaSCI
  - ✓ ScaSCIddk
  - ✓ ScaSCIadap
  - ✓ ScaSISCI
  - ✓ Please note that the list of rpms to be updated may change.

```
cd /home/scali/repository/Linux2.ia64
### Download the rpms (packages) ###
```

- Install the packages on the nodes of the cluster:

```
/opt/scali/sbin/scapkg –p ``<Package>''
```

(for example: "/opt/scali/sbin/scapkg –p ScaMPI", without the "Linux2.ia64-<version>.rpm" extension).

Updating the SCI module:

Scali offers a tool (ScaSCIadap) to adapt the SCI driver to a different kernel to the one supported by default.

To use this tool, read /opt/scali/doc/ScaSCIadap/README.
However, please note when reading this documentation:

In the part including the definition of environment variables, if the kernel sources are in /usr/src/linux, proceed as follows:

```
export KERNEL_HEADER_BASE=/usr/src
export KERNEL_DIR=linux
mkdir –p /opt/scali/scascibuild/kernel
cd /opt/scali/scascibuild/kernel
/opt/scali/kernel/rebuild/configure
```

For a version.h file located in: /usr/src/linux/include/linux

The next part does not differ from the original documentation:

```
make
make install
/opt/scali/sbin/scireload  (on each node)
/opt/scali/sbin/SSPinstall
```

3) Installation: 2nd Pass

For the second pass, start the installation as follows:

```
/opt/scali/sbin/SSPinstall
```

- For step "**Do you want to upgrade"**, reply "**y**"
- Then accept the default responses, except for the installation of OpenPBS (reply "**y**", then supply "linux" as the username).
- For the step "**Checking kernel versions**" a "WARNING**"** is displayed, reply "**i**" to ignore it.
- For the step "**Checking HW support**" a "WARNING**"** is displayed, reply "**i**" to ignore it.
- For the step "**Checking OS support**" a "WARNING**"** is displayed, reply "**i**" to ignore it.

In the second phase of the installation, only the three "WARNINGS" above should be displayed. Ignore them as they do not affect the smooth running of the system.

Checking the Installation:

The installation can be tested very easily by the following commands (making sure no error is displayed):

```
export PATH=/opt/scali/bin:/opt/scali/sbin:$PATH
SSPinstall -v
SSPinstall -t
```

Performance can be evaluated using:

```
export MPI_HOME=/opt/scali
export PATH=/opt/scali/bin:/opt/scali/sbin:$PATH
mpirun –np 2 /opt/scali/examples/bin/bandwidth
```

- File latency should run at about 5 us for a 0 byte message and bandwidth should be above 320 MB/s for large messages.

IMPORTANT: At this stage you can create a file defining the environment variables for using the following tools:

- Intel v7.0 compilers
- Intel math libraries
- SCALI ScaMpi

```
. /opt/envhpc/intel/compiler70/ia64/bin/efcvars.sh
. /opt/envhpc/intel/compiler70/ia64/bin/eccvars.sh
export MPI_HOME=/opt/scali
export PATH=/opt/scali/bin:/opt/scali/sbin:$PATH
export LD_LIBRARY_PATH=/opt/envhpc/intel/mkl/lib/64:/opt/scali/lib:$LD_LIBRARY_PATH
export MANPATH=/opt/scali/man:$MANPATH
```

Proposed name: ENV_VAR_SCALI
An example of this file is proposed on the "Bull ENH Open Source" CD in the ENVIRONMENT_VARIABLES directory (see the "Software Installation" section below)
Copy it to /opt/envhpc

Uninstall Procedure:

- On the front-end, enter the following command:

```
/opt/scali/sbin/SSPuninstall
```

- On all the nodes: Delete the /opt/scali directory (on all nodes) and some files under /tmp:

```
rm -rf /opt/scali
rm /tmp/SSP_3_0_1*
```

3.3 SCALI SSP 3.1.0

Introduction:
This paragraph describes how to install SCALI SSP 3.1.0.
Check http://www.scali.com for the release of new updates.

Prerequisites:

- RedHat Linux and the kernel must already be installed on each node.
- Cables connected as per the "SCALI system guide".
- SSP_3_1_0 is supplied on a CD ROM, however you can also download it from http://www.scali.com/Download/ssp.shtml.
  If you want to download the software:

```
mkdir /home/scali
cd /home/scali
### Download SSP_3_1_0_Linux2_ia64.tar.gz  ###
tar xvzf SSP_3_1_0_Linux2_ia64.tar.gz
```

The name of the front-end node must be "admin": Check command "uname –n" diplays "admin".

Configure the Operating System as shown in the OS file (in subdirectory SSP_3_1_0/doc/SSP). On all the nodes of the cluster:

- Insert the names of the nodes with their LAN IP addresses in file */etc/hosts:*

```
<IP address> admin
<IP address> n0101
<IP address> n0102
<IP address> n0103
etc …
```

- While logged in as root, insert all the node names in file */root/.rhosts:*

```
admin root
n0101 root
n0102 root
n0103 root
etc …
```

- Insert all the node names in file */etc/hosts.equiv:*

```
admin
n0101
n0102
n0103
etc …
```

- Add *rsh*, *rlogin* and *login* to file */etc/securetty*
- Run the following commands:

```
chkconfig rsh on
chkconfig rlogin on
```

- Check you have at least 200MB available on /opt/scali on all nodes.
- Make sure you have received a temporary license by e-mail and have copied it to the front-end (file /opt/license.dat)

Installation:

Installation of the SCALI SSP interconnection software is done in several steps:

1. An initial step to install the license, define the nodes (admin, n0101, n0102, n0103 etc …) and the functionality to be installed (OpenPBS) or not (Console Server, Power Switch Server, NIS/YP).
2. An intermediate phase to update the SCI driver.
3. A second pass to finish and check the installation.

1) Installation: 1st Pass

- From the CD ROM (on the admin node), enter command:

```
mount –t iso9660 /dev/cdrom /mnt/cdrom
cd /mnt/cdrom
./install
```

- Or, from the files you have downloaded, enter command:

```
/home/scali/SSP_3_1_0/install
```

- Follow the installation procedure by answering the questions. Some steps make take a long time.
- For step "**Expert mode" select the default choice** ("**n**").
- For the step "**Specifying node names**", list the nodes (order: admin, n0101, n0102, n0103 for 4 nodes).
- For step "**Setup license(s)"** select "**2**) License file" and give the full path "/opt/license.dat".
- For the step "**Determine node categories**", if all the nodes are computation nodes, make the default choice ("**y**"). Otherwise, you can exclude certain nodes that are not computation nodes (e.g. storage nodes). To do this:
  - When asked "Do you accept the above configuration ?", reply "n".
  - Select the nodes to be modified.
  - Only select category "eth_node" for the nodes to be modified
- For step "**install the console server**" reply "**n**".
- For step "**install the power switch server**" reply "**n**".
- For step "**install OpenPBS queue system"** reply "**y"**, then enter the username "**linux**".
- The next three steps may display "WARNINGS". Ignore these by replying "i" (ignore). They are:
  - "Checking kernel versions"
  - "Checking HW support"
  - "Checking OS support"
- For step "**Are you using NIS/YP** .. » reply "**n**"

- For step "**Configuring SCI network**", for a cluster of 4 nodes, choose "2D torus", then give the number of nodes (2) connected to the rings of the L0 interface , then enter the number of nodes (2) connected to the rings of interface L1. This gives 2 x 2 = 4 (total number of nodes). Then choose the default automatic method (choice "**1**").
- Some of the next steps issue "WARNINGS". These are not abnormal the first time the installation software is run, so reply "i" (ignore). These problems occur because the SCI driver does not load, which is not unusual at this stage of the installation. They are:
  - "Checking SCI driver"
  - "Testing MPI communication"
  - "Testing OpenPBS"

2) Intermediate Phase: Updating the SCI module:

Scali installs everything on the front-end and on each of the nodes of the cluster and tests if everything is working (communication between nodes). However, the SCI driver supplied by default may not be suited to the installed kernel (see the "WARNINGS" displayed during the first phase of installation) and these tests will not work. Update the driver using the "SCI module update" procedure below.

Updating the SCI module:

Scali offers a tool (ScaSCIadap) to adapt the SCI driver to a different kernel to the one supported by default.

To use this tool, read /opt/scali/doc/ScaSCIadap/README.
However, please note when reading this documentation:

In the part including the definition of environment variables, if the kernel sources are in /usr/src/linux, proceed as follows:

```
export KERNEL_HEADER_BASE=/usr/src
export KERNEL_DIR=linux
mkdir –p /opt/scali/scascibuild/kernel
cd /opt/scali/scascibuild/kernel
/opt/scali/kernel/rebuild/configure
```

For a version.h file located in: /usr/src/linux/include/linux

The next part does not differ from the original documentation:

```
make
make install
/opt/scali/sbin/scireload  (on each node)
```

3) **Installation: 2nd Pass**

The installation can be tested very easily by the following commands (making sure no error is displayed):

```
export PATH=/opt/scali/bin:/opt/scali/sbin:$PATH
SSPinstall -v
SSPinstall -t
```

Performance can be evaluated using:

```
export MPI_HOME=/opt/scali
export PATH=/opt/scali/bin:/opt/scali/sbin:$PATH
cd /tmp
mpirun –np 2 /opt/scali/examples/bin/bandwidth
```

File latency should run at about 5 us for a 0 byte message and bandwidth should be above 320 MB/s for large messages.

The checks and tests can be enabled at any time to ensure the interconnection is working properly.

IMPORTANT: At this stage you can create a file defining the environment variables for using the following tools:

- Intel v7.0 compilers
- Intel math libraries
- SCALI ScaMpi

```
. /opt/envhpc/intel/compiler70/ia64/bin/efcvars.sh
. /opt/envhpc/intel/compiler70/ia64/bin/eccvars.sh
export MPI_HOME=/opt/scali
export PATH=/opt/scali/bin:/opt/scali/sbin:$PATH
```

```
export LD_LIBRARY_PATH=/opt/envhpc/intel/mkl/lib/64:/opt/scali/lib:$LD_LIBRARY_PATH
export MANPATH=/opt/scali/man:$MANPATH
```

Proposed name: ENV_VAR_SCALI
An example of this file is proposed on the "Bull Extension Pack for HPC Linux" CD in the ENVIRONMENT_VARIABLES directory (see the "Software Installation" section below).
Copy it to /opt/envhpc

Uninstall Procedure:

- On the front-end: enter the following command:

```
/opt/scali/sbin/SSPuninstall
```

- On all the nodes: Delete the /opt/scali directory (on all nodes) and some files under /tmp:

```
rm -rf /opt/scali
rm /tmp/SSP_3_1_0*
```

3.4 Configuring the 3COM Gigabit Switch

So that Ganglia works correctly, the switch used must support IGMP multicast otherwise only the first multicast packets will pass. This gives the initial impression that GANGLIA is working correctly but then each node loses all visibility of the other nodes.

This problem is seen with the 3COM Gigabit switch for which IGMP multicast has to be reconfigured - first invalidating this function and then revalidating it so it works correctly.

3.4.1 The Operating Mode Taken from the Switch Documentation

To configure the switch, you need to connect to the serial port on the front of the Gigabit appliance. One of the "minicom" serial lines (e.g. n0101) is used for this connection.

Enter the "minicom n0101" command, the connection is configured in 9600 bauds. Hit "Return" to access the login ("monitor", password "monitor").

In the available menus, find the option that authorizes IGMP multicast, invalidate it then revalidate it.

You can then also configure the IP address of the switch (e.g. 172.16.12.254, mask 255.255.255.0) to access the switch configuration later by telnet or via a Web interface on port 80.

Chapter 4. Software Installation and Start-up

What follows is a guide to installing Open Source software from the " **Bull Extension Pack for HPC Linux V2.0 June 2003** CD **(ref : 76 741 190-001)"** or directly from **CDs** or **supplier sites**.
The installation of this software is identical on the NovaScale 4040, NovaScale 5080, NovaScale 5160 in mono-node or cluster mode.
Software whose name is underlined is only available in cluster mode.

			Open Source	Commercial Product
APPLICATION DEVELOPMENT	**Scientific Libraries**		Fftw, Petsc	Libmkl (Intel)
	Parallel Libraries	Mpi	Mpich, Lam-mpi	
		Others	PVM	
	Compilers (C, C++, Fortran)		GNU	Intel
TOOLS	**Operation**	Task sharing	OpenPBS, Maui,	
		Debuggers	gdb	Intel idb TotalView(Etnus)
		Profiling	Cprof/vprof	Vampir (Pallas)
		Performance Analysis	Papi, pfmon, perfometer	
	Cluster Administration	Distributed shell	gexec	
		Control & monitoring	Ganglia	
OS	**System Administration**		Webmin, Nagios	
	Operating System		Linux	

The compilation environment used is that of Intel compilers.

4.1 Installing the Operating System

See the document "**Linux for HPC: Installation Guide (ref: 86 F2 37EG 01)"**

4.2 Installing GNU Components

The basic Linux distribution installs GNU components (gcc, g77, gdb, pthreads…)

4.3 Automated Installation of Software Components on the CD Bull Extension Pack For HPC Linux

First, mount the CD-ROM:

```
mount -t iso9660 /dev/cdrom /mnt/cdrom

cd /mnt/cdrom
```

Start the automatic installation script for Open Source components:

```
./Bull_installer
```

This is the list of components installed automatically by the Bull_installer script:

- **MPICH / HPL**
- **FFTW**
- **LAM_MPI**
- **MAUI**
- **NAGIOS**
- **PETSC**
- **PROFILING**
- **PVM**

Please note that this automated installation is for a standalone machine. For a cluster installation, please see section "*Manual Installation of Software Components on the Bull CD"*

For other components, please see section "*Manual Installation of Software Components on the Bull CD"*

Use:

1. Give the root installation path of the Intel C and Fortran compilers.
   By default the script considers they are installed in /opt/envhpc/intel.

   Note: Installation of these Open Source packages cannot take place if the Intel C and Fortran compilers have not been installed.

2. Give the root installation path of the Intel math library.
   By default the script considers it is installed in /opt/envhpc/intel/mkl.

   Note: If the math library is not installed, some tests will not be performed. For example, an application like Linpack (HPL), or a library like PETSC will not be installed or tested.

3. Give the path where the HPC Open Source packages are to be installed.
   By default the script considers they are installed in /opt/envhpc.

   Note: Most of the packages will be installed in this directory.

4. Next, the script asks in turn whether each of the Open Source packages are to be installed:
   reply "y" or "n" for each of them

   Do you want to install MPICH ? y/n
   Do you want to install FFTW ? y/n
   Do you want to install LAM_MPI ? y/n
   Do you want to install MAUI ? y/n
   Do you want to install NAGIOS ? y/n
   Do you want to install PETSC ? y/n
   Do you want to install PROFILING ? y/n
   Do you want to install PVM ? y/n

   Special Cases:

   - If you want to install MAUI, you will be asked some additional questions:

     A release of PBS has to be installed in order to install MAUI
     Did you install PBS on your node ? y/n

   If PBS has not been installed, you will not be able to install MAUI.

   Where did you install your PBS release? [/usr/local]

   If PBS has already been installed, give the root installation path.
   By default the script considers it is installed in /usr/local.

- For the installation of FFTW and PETSC:

Version 1.2.5 of mpich is required to install these packages. Consequently, if you do not select MPICH as one of the components to be installed, you will be asked some additional questions:

A release of MPICH has to be installed in order to install FFTW
Did you install MPICH on your node ? y/n

If Mpich version 1.2.5 has not been installed, FFTW and PETSC cannot be installed.

Where did you install your MPICH release ? [/opt/envhpc/mpich-1.2.5]

If Mpich version 1.2.5 has already been installed, give the root installation path.
By default the script considers it is installed in </opt/envhpc/mpich-1.2.5>.

5. Next, hit "ENTER" to install each of the selected packages.

6. For each package, the procedure is as follows:

   Example: PVM package

   - A message tells you that installation is starting:

     pvm3 Installation is beginning ... It can take a while ...

   - The script then displays some instructions that should be followed to test that the installation has run successfully:

   **** IN ORDER TO TEST pvm3:*
   Launch the following command:
   . /opt/envhpc/ENV_VAR_PVM
   Then, Launch : pvm
   If there is no error message, you can exit by:
   quit
   You should have the following message:
   Console: exit handler called
   pvmd still running.

   - A message tells you that installation has finished:

   ... pvm3 Installation is ending

   - A last message tells you the log file to check to see how the installation went:

   You can read /tmp/bull/pvm3.log for more information

- When all the packages have been installed, a message tells you that they are complete:

Installation complete in /opt/envhpc !!

4.4 Preparation of the HPC Development Environment for Manual Installation

For a cluster, you must have completed the installation of your cluster, in this case SCALI, before starting to prepare the HPC environment. From then on, the installation and launch operations are to be performed on the node chosen for cluster administration. In this document, examples are given for a 4 node configuration whose administrator node is called "admin" and the other nodes n0101 n0102 n0103.

For the rest of this chapter, unless specifically stated, there is no difference between an installation on a mono-machine or on the administrator node of a cluster.

The first step consists in uninstalling the rpm package LAM-MPI if it was installed by the distribution:

```
rpm –e lam-6.5.4-1
```

In order to centralize development software (compilers, libraries, tools) we are going to create the directory /opt/envhpc on the machine or sever for a cluster (by default the administration node).

Configuration for a Single Machine:

Create directory:

```
mkdir /opt/envhpc
```

Configuration for a Cluster:

Access to directory /opt/envhpc by the other nodes (clients) is done by nfs mounting. You are also recommended to perform an nfs mount of /home/packages_sources to be able to carry out the various installation tests.

To do this:

On the server (admin):

Create directory:

```
mkdir /opt/envhpc
```

Put the following in /etc/exports:

```
/opt/envhpc n0101(rw,no_root_squash) n0102(rw,no_root_squash)
n0103(rw,no_root_squash)
/opt/envhpc n0101(rw,no_root_squash) n0102(rw,no_root_squash) n0103(rw,no_root_squash)
```

Restart the NFS server.

```
service nfs stop
service nfs start
```

In order to restart the nfs server when the machine starts

```
chkconfig --level 35 nfs on
```

On the client machines (n0101….):

Create directory:

```
mkdir /opt/envhpc
```

Put the following in /etc/fstab:

```
mkdir /opt/envhpc      /opt/envhpc     nfs      rsize=8192,wsize=8192,intr 0
0
admin:/home/packages_sources                /home/packages_sources
        nfs      rsize=8192,wsize=8192,intr 0 0
```

Do the corresponding mount

```
mount -a
```

4.5 Manual Installation of the Software Components on the Bull CD

These software components are shipped via the **"Bull Extension Pack For HPC Linux"** CD-ROM.

- You must therefore first mount the CD-ROM:

```
mount –t iso9660 /dev/cdrom /mnt/cdrom
```

- Next, create a directory to contain the source software:

```
mkdir /home/packages_sources
```

- Copy the source software into the directory just created:

```
cp -arf /mnt/cdrom/* /home/packages_sources
```

- Go to this directory to install the various packages:

```
cd /home/packages_sources
```

SOME SOFTWARE IS PRESENT AS A SOURCE AND MUST THEREFORE BE RECOMPILED; OPTIONS MAY BE MODIFIED BY THE INSTALLER. YOU MUST THEREFORE INSTALL FIRST:

- THE COMPILERS
- THE NUMERIC LIBRARIES
- THE PARALLEL COMMUNICATIONS LIBRARIES

4.6 Intel Compilers

Fortran Compiler:

Example of how to install the FORTRAN 7.0.0.64 compiler with a l_for_03387691.lic license

- Create a "licenses", directory if one does not already exist, at the location where you are going to install the Intel compiler. For example, if you are going to install the Intel compiler in /opt/envhpc/intel, create a /opt/envhpc/intel/licenses directory:

```
mkdir -p /opt/envhpc/intel/licenses
```

- Install the licenses in this license directory. e.g.

```
cp l_for_03387691.lic /opt/envhpc/intel/licenses/
```

- Dearchive the .tar file via the command. e.g.

```
tar xvf l_fc_p_7.0.064.tar
```

- Install the Fortran compiler on the installed Linux distribution via the install command:

```
export INTEL_LICENSE_FILE=.:/opt/envhpc/intel/licenses
./install
```

- Choose "2" to select installation on an "Itanium®-based system" platform.
- Choose "1" to install the fortran itanium compiler.
- After the license agreement, type "accept".
- Choose the installation path (by default: /opt/intel), for example, /opt/envhpc/intel
- Accept the installation options
- A compiler70 directory is created in this path
- X to exit
- Check that <***installation path***>/compiler70/ia64/bin/efcvars.sh points to the right paths then execute it:

```
. <installation path>/compiler70/ia64/bin/efcvars.sh
```

- The fortran compiler is now available under the name of efc

Testing the Installation:

- Run command:

```
efc
```

- The following result should be displayed:

```
/usr/lib/crt1.0 : In function `_start' :
/usr/lib/crt1.0(.text+0x41) : undefined reference to `main'
```

C/C++ Compiler:

Example of how to install the C/C++ 7.0.0.65 compiler with a l_cpp_40599793.lic license

- Create a "licenses", directory if one does not already exist, at the location where you are going to install the Intel compiler. For example, if you are going to install the Intel compiler in /opt/envhpc/intel, create a /opt/envhpc/intel/licenses directory:

```
mkdir –p /opt/envhpc/intel/licenses
```

- Install the licenses in this license directory. e.g.

```
cp l_cpp_40599793.lic /opt/envhpc/intel/licenses/
```

- Dearchive the .tar file via the command. e.g.

```
tar xvf l_cc_p_7.0.065.tar
```

- Install the C/C++ compiler on the installed Linux distribution via the install command:

```
./install
```

- Choose "2" to select installation on an "Itanium®-based system" platform.
- Choose "1" to install the C/C++ itanium compiler.
- After the license agreement, type "accept".
- Choose the installation path (by default: /opt/intel), for example, /opt/envhpc/intel
- Accept the installation options
- A compiler70 directory is created in this path
- X to exit
- Check that <***installation path***>/compiler70/ia64/bin/eccvars.sh points to the right paths then execute it:

```
. <installation path>/compiler70/ia64/bin/eccvars.sh
```

- The C/C++ compiler is now available under the name of ecc

Testing the Installation:

- Run command:

```
efc
```

- The following result should be displayed:

```
/usr/lib/crt1.0 : In function `_start' :
/usr/lib/crt1.0(.text+0x41) : undefined reference to `main'
```

Intel Debugger:

First step:

The package used to install this debugger is located in either the Fortran or C tar archive.

Second step: Installation

After installing the C compiler, for example:

- Install the Intel debugger on the installed Linux distribution, via the install command:

```
./install
```

- Choose "2" to select installation on an "Itanium®-based system" platform.
- Choose " 2 " to install the Intel debugger
- After the license agreement, type "accept".
- Choose the installation path (by default: /opt/intel), for example, /opt/envhpc/intel
- Accept the installation options
- A compiler70 directory is created in this path
- X to exit

- Check that **<installation path>compiler70/ia64/bin/efcvars.sh** points to the right paths then execute it to have the Intel debugger idb in the path:

```
. <installation path>/compiler70/ia64/bin/efcvars.sh
```

The Intel debugger is now available under the name of idb

Testing the Installation:

Run command:

```
idb
```

The following result should be displayed:

```
(idb)
```

To exit:

```
quit
```

4.7 MPICH 1.2.5

- Download mpich.tar from the following site:

http://www-unix.mcs.anl.gov/mpi/mpich/download.html

or:

```
cd /home/packages_sources/MPICH
```

- Dearchive the following file:

```
tar xvfz mpich.tar.gz
cd mpich-1.2.5
```

For the installation itself, follow the instructions given in the README and the files in the doc directory located in the mpich-1.2.5 directory.

What follows is a quick guide so you can compile your version of Mpich rapidly:

First step:

Note: The **efc** and **ecc** compilers must be in your PATH environment variable (cf. section "Intel V7.0 Compilers")

To use mpich on a single SMP machine, compile mpich using the ch_shmem device:

```
./configure --prefix=<installation path> --with-device=ch_shmem
-cc=ecc -clinker=ecc -fc=efc -flinker=efc -f90=efc -f90linker=efc
```

where <Installation path> could be, for example, /opt/envhpc/mpich-1.2.5

However, if you want to compile mpich for an SMP machine cluster, choose the ch_p4 device:

```
./configure --prefix=<installation path> --with-device=ch_p4
-comm=shared -cc=ecc -clinker=ecc -fc=efc -flinker=efc -f90=efc
-f90linker=efc
```

where <Installation path> could be, for example, /opt/envhpc/mpich-1.2.5

To find out more about these flags, please see the documentation supplied.

However, to summarize, we can say that it will enable you to construct mpich tools (mpirun, mpicc, mpif90, etc.) using the shared memory and including the Intel ecc and efc compilers.

Second step:

```
make
```

(Construction of tools using the previous configuration)

Last step:

```
make install
```

In our case, this will install mpich binaries in <installation path>/mpich-1.2.5, for example /opt/envhpc/mpich-1.2.5

- To then use the mpich tools, you will first need to perform the following operations:

```
export PATH=<installation path>/mpich-1.2.5/bin:$PATH
```

Note:
When installation is done with the ch_p4 device, modify file <*installation path*>/mpich-1.2.5/share/machines.LINUX and enter the names of the machines and the number of processors on each of them.

e.g.

```
admin:4
n0101:4
n0102:4
n0103:4
```

Tests:

We can then run a basic test on our configuration:
1.

```
cd <installation path>/mpich-1.2.5/examples
make cpi
```

to create the cpi binary that you can execute via the following command:

- For a four processor machine

```
mpirun -np 4 ./cpi
```

- For a 16 processor machine (or a 4x4 cluster)

```
mpirun -np 16 ./cpi
```

2.

```
make pi3f90
```

to create the pi3f90 binary that you can execute via the following command:

- For a four processor machine

```
mpirun -np 4 ./pi3f90
```

- For a 16 processor machine (or a 4x4 cluster)

```
mpirun -np 16 ./pi3f90
```

Enter the number of intervals, for example 10000, or 0 to exit the application.

IMPORTANT: At this stage you can create a file defining the environment variables for using the following tools:

- Intel v7.0 compilers
- Intel math libraries
- mpich-1.2.5

Example:

```
. /opt/envhpc/intel/compiler70/ia64/bin/efcvars.sh
. /opt/envhpc/intel/compiler70/ia64/bin/eccvars.sh
export PATH=/opt/envhpc/mpich-1.2.5/bin:$PATH
export LD_LIBRARY_PATH=/opt/envhpc/intel/mkl/lib/64:$LD_LIBRARY_PATH
export MANPATH=/opt/envhpc/mpich-1.2.5/man:$MANPATH
export P4_GLOBMEMSIZE=260000000
```

Proposed name: ENV_VAR_MPICH
An example of this file is given in
/home/packages_sources/VARIABLES_ENVIRONNEMENT

Copy it to /opt/envhpc

You can now check the first HPC packages installed with the HPL application (cf. next section)

4.8 Math Libraries

4.8.1 Libmkl

Installing mkl 6.0

The installation should be started as root.

Copy the l_mkl_xxxxxxxx.lic license file to the /opt/envhpc/intel/licenses directory.

Decompress and dearchive the archive file:

```
tar xvf l_mkl_p_6.0.011.tar
```

Start installation by starting:

```
cd  l_mkl_p_6.0.011
```

```
./install.sh
```

The access path to the license is /opt/envhpc/intel/licenses.
The access path to specify is /opt/envhpc/intel.

In order to remain independent of the library version number, you are recommended to make the following symbolic link:

```
ln –f –s /opt/envphc/intel/mkl60 /opt/envhpc/intel/mkl
```

Note: To be able to use the dynamic libraries, remember to include the directory in the environment variable LD_LIBRARY_PATH.

```
export LD_LIBRARY_PATH=/opt/envhpc/intel/mkl/lib/64:$LD_LIBRARY_PATH
```

Note: For a cluster, it is essential to have the LD_LIBRARY_PATH variable in the login environment of all the nodes. If you do not, the execution of a program using libmkl on several nodes will stop with the following message:

error while loading shared libraries: libmkl_itp.so: cannot open shared object file: no such file or directory

You can, for example, insert the initialization of LD_LIBRARY_PATH in the /etc/bashrc file.

Testing the Installation:
Running the HPL benchmark checks the installation.

4.8.2 FFTW

Download the archive from the following site:

http://www.fftw.org/

or:

```
cd /home/packages_sources/FFTW
```

Dearchive the file:

```
tar xvfz fftw-2.1.3.tar.gz
cd fftw-2.1.3
```

First step:

Note: The **efc** and **ecc** compilers must be in your PATH environment variable (cf. section "Intel V7.0 Compilers")

To use it with SCALI:

```
export MPICC='ecc -D_REENTRANT -I/opt/scali/include -L/opt/scali/lib'
export CC=ecc
export CFLAGS=-O2
export F77=efc
./configure --prefix==<installation path> -host=i786 --enable-mpi.
```

where <Installation path> could be, for example, /opt/envhpc/fftw-2.1.3

However, if you want to compile fftw-2.1.3 with mpich:

Load the MPICH environment via the /opt/envhpc/ENV_VAR_MPICH file.

```
export CC=ecc
export CFLAGS=-O2
export F77=efc
./configure --prefix=<installation path>  -host=i786 --enable-mpi
```

where <Installation path> could be, for example, /opt/envhpc/fftw-2.1.3

Second step:

```
make
make install
```

(Construction of tools using the previous configuration)

We can then run a basic test on our monoprocessor configuration:

```
cd  tests
./fftw_test -s 100
```

We can then run a basic test on our configuration with mpi

```
cd ../mpi
mpirun -np 4 ./fftw_mpi_test -s 100
```

4.8.3 PETSC

Download the archive from the following site:

http://www-fp.mcs.anl.gov/petsc/index.html

or:

```
cd /home/packages_sources/PETSC
```

Dearchive the file:

```
tar xvfz petsc.tar.gz
cd petsc-2.1.5
```

First step:

Initialize the following variables:

```
export PETSC_ARCH=linux64_intel
export PETSC_DIR = <path containing sources>/petsc-2.1.5
export LD_LIBRARY_PATH=/opt/envhpc/intel/mkl/lib/64:$LD_LIBRARY_PATH
```

Modify the file

```
$PETSC_DIR/bmake/linux64_intel/packages
```

To use it with MPICH:
Modify the *$PETSC_DIR/bmake/linux64_intel/packages* file

```
BLASLAPACK_LIB = -L/opt/envhpc/intel/mkl/lib/64 -lmkl_itp
            -lmkl_lapack /opt/envhpc/intel/mkl/lib/64/libguide.so
#
# Location of MPI (Message Passing Interface) software
#
#MPI_HOME      = /usr/local/vmi/mpich
#MPI_LIB       = -L${MPI_HOME}/lib/ecc -lfmpich -lmpich -lpmpich
#-lvmiquiet -lpthread -ldl
#MPI_INCLUDE    = -I${MPI_HOME}/include
#MPIRUN        =  /home/balay/bin/mpirun
#
MPI_HOME      = /opt/envhpc/mpich-1.2.5
MPI_LIB       = ${MPI_HOME}/lib/libmpich.a
MPI_INCLUDE    = -I${MPI_HOME}/include
MPIRUN        =  ${MPI_HOME}/bin/mpirun
#
# ----------------------------------------------------------------
#  Locations of OPTIONAL packages. Comment out those you do not have.
# ----------------------------------------------------------------
#
# Location of X-windows software
#
X11_INCLUDE    =
X11_LIB       = -L/usr/X11R6/lib -lX11
PETSC_HAVE_X11 = -DPETSC_HAVE_X11
```

To use it with SCAMPI:
Modify the *$PETSC_DIR/bmake/linux64_intel/packages* file

```
BLASLAPACK_LIB = -L/opt/envhpc/intel/mkl/lib/64 -lmkl_itp
```

```
            -lmkl_lapack /opt/envhpc/intel/mkl/lib/64/libguide.so
#
# Location of MPI (Message Passing Interface) software
#
#MPI_HOME       = /usr/local/vmi/mpich
#MPI_LIB        = -L${MPI_HOME}/lib/ecc -lfmpich -lmpich -lpmpich
#-lvmiquiet -lpthread -ldl
#MPI_INCLUDE    = -I${MPI_HOME}/include
#MPIRUN         =  /home/balay/bin/mpirun
#
MPI_HOME       = /opt/scali
MPI_LIB        = ${MPI_HOME}/lib/libmpi.a
MPI_INCLUDE    = -I${MPI_HOME}/include
MPIRUN         =  ${MPI_HOME}/bin/mpirun


#
# ----------------------------------------------------------------
#  Locations of OPTIONAL packages. Comment out those you do not have.
# ----------------------------------------------------------------
#
# Location of X-windows software
#
X11_INCLUDE    =
X11_LIB        = -L/usr/X11R6/lib -lX11
PETSC_HAVE_X11 = -DPETSC_HAVE_X11
```

Modify the *$PETSC_DIR/bmake/linux64_intel/variables* file

```
C_CC            = ecc -D_REENTRANT -I/opt/scasli/include
C_CLINKER       = ecc  -L/opt/scali/lib         -lmpi -Wl,-R/opt/scali/lib
C_FLINKER       = efc   -L/opt/scali/lib -lfmpi  -lmpi -Wl,-R/opt/scali/lib
```

Second step:

```
make BOPT=O allclean
make BOPT=O all | tee  make_log
```

(Construction of tools using the previous configuration)

For the installation, this is done manually by transferring the files located in $PETSC_DIR/lib/libO/linux64_intel to the <Installation path> directory where <Installation path> could be, for example, /opt/envhpc/petsc-2.1.5/lib

We can then run a basic test on our configuration:

```
make BOPT=O testexamples |tee  examples_log
```

NOTE: Each result is compared to the expected result by a diff. With SCAMPI, mpirun runs mpimon and displays "mpimon started". Therefore all the tests include differences with the expected result.
e.g.

######
> /opt/scali/bin/mpimon -stdin all ex22 -da_grid_x 10 -nox -- n0101 1
Possible problem with ex22_1, diffs above
######

4.9 HPL

HPL is the HPC version of Linpack.

This standard benchmark can be used to measure the scalability and performance of a cluster.
Its aim is to solve a linear system of N equations to N unknowns (Ax=B). The matrices used are dense matrices and the code is composed of double precision floating point computations and synchronous communications. Linpack is written in Fortran and uses the functions of Blas and MPI libraries for communications.

As results it returns:

- An execution time in seconds
- The maximum number of floating operations performed in one second (Gflops)

For more information, see:

http://www.netlib.org/benchmark/hpl

And particularly:

http://www.netlib.org/benchmark/hpl/results.html
http://www.netlib.org/benchmark/hpl/tuning.html
http://www.netlib.org/benchmark/hpl/faqs.html

HPL requires:
- MPI implementation: Here we are testing it with mpich-1.2.5 then Scali ScaMPI
- The BLAS math library: Here we are using Intel libmkl

To retrieve the package:

```
cd /opt/envhpc
```

Decompress and dearchive the file by:

```
tar xvfz /home/packages_sources/HPL/hpl.tgz
cd hpl
```

In Makefile make the following change:

```
arch = Itanium
```

Retrieve an example of Make.Arch by entering, for example:

```
cp setup/Make.Linux_PII_CBLAS Make.Itanium
```

First case: You want to compile HPL with mpich and libmkl

Make the following changes to Make.Itanium:

```
ARCH = Itanium
Add HOME = /opt/envhpc

CC = mpicc
CCFLAGS =

LINKER = mpif90
LINKFLAGS=$(CCFLAGS) /opt/envhpc/intel/mkl/lib/64/libguide.so

MPdir = /opt/envhpc/mpich-1.2.5
MPinc = $(MPdir)/include
MPlib = -L$(MPdir)/lib –lmpich –lpmpich

LAdir =
LAlib = -L/opt/envhpc/intel/mkl/lib/64 –lmkl_itp

HPL_OPTS =
```

To prepare the compilation, run:

```
. /opt/envhpc/ENV_VAR_MPICH
```

Second case: You want to compile HPL with SCALI ScaMPI and libmkl (in a cluster configuration)

Make the following changes to Make.Itanium:

```
ARCH = Itanium
```

```
Add HOME = /opt/envhpc

CC = ecc –D_REENTRANT –I$(MPI_HOME)/include
CCFLAGS =

LINKER = efc
LINKFLAGS=$(CCFLAGS) /opt/envhpc/intel/mkl/lib/64/libguide.so

MPdir = /opt/scali
MPinc = $(MPdir)/include
MPlib = -L$(MPdir)/lib –lmpi

LAdir =
LAlib = -L/opt/envhpc/intel/mkl/lib/64 –lmkl_itp

HPL_OPTS =
```

To prepare the compilation, run:

```
. /opt/envhpc/ENV_VAR_SCALI
```

In both cases:

```
make clean_arch_all
make : creation of the xhpl executable in ./bin/Itanium
cd bin/Itanium/
Modification of HPL.dat :
```

Look for the parameter file that gives the best global performance of the cluster: HPL.dat

Further details:

N = size of problem to be solved – find the biggest problem filling the physical memory without swapping.

NB = block size – numerous attempts are required to find the best value

P*Q: matrix to solve the problem – divides up the problem into P*Q linpack processes. To be as efficient as possible, the number of linpack processes must correspond to the number of nodes.

Note:

Since the mkl math library is parallelized, for maximum efficiency only one MPI process should be run per SMP node. On a quadri-processor SMP machine, when you run an MPI process, the parallelization of the math library will create 4 instances of the hpl program.

Parallelization of the libmkl can be inhibited via the following command:

```
export OMP_NUM_THREADS=1
```

However, the following examples were carried out with OMP_NUM_THREADS=4 (default value on a quadri-processor machine)

Examples:

MPICH: HPL.dat to execute HPL on a 4 processor SMP machine:

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
10000  Ns
1            # of NBs
120      NBs
1            # of process grids (P x Q)
1        Ps
1        Qs
16.0         threshold
1            # of panel fact
2        PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
8          NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
2        RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
0            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
1            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

Execution:

```
mpirun -np 1 xhpl
```

Result:

```
“1 tests completed and passed residual checks”
```

MPICH: HPL.dat to execute HPL on four 4 processor SMP machines:

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
10000  Ns
1            # of NBs
120      NBs
             1           # of process grids (P x Q)
4        Ps
1        Qs
16.0         threshold
1            # of panel fact
2        PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
8          NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
2        RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
0            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
1            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

Configuration:

Modify the .bashrc of the user in question on the remote nodes by adding the following line:

```
export LD_LIBRARY_PATH=/opt/envhpc/intel/mkl/lib/64:$LD_LIBRARY_PATH
```

On the administration node, modify the /opt/envhpc/mpich-1.2.5/share/machines.LINUX file **temporarily** to have one MPI process per SMP node, for example:

```
admin
n0101
n0102
n0103
```

Execution:

```
mpirun -np 4 xhpl
```

Result:

```
"1 tests completed and passed residual checks"
```

Put the /opt/envhpc/mpich-1.2.5/share/machines.LINUX file back into the state it was in before modification.

SCALI: HPL.dat to execute HPL on four 4 processor SMP machines:

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
10000  Ns
1            # of NBs
120      NBs
1            # of process grids (P x Q)
4       Ps
1       Qs
16.0         threshold
1            # of panel fact
2       PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
8         NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
2       RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
```

```
0       BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1       # of lookahead depth
1       DEPTHs (>=0)
2       SWAP (0=bin-exch,1=long,2=mix)
64       swapping threshold
0       L1 in (0=transposed,1=no-transposed) form
0       U  in (0=transposed,1=no-transposed) form
1       Equilibration (0=no,1=yes)
8       memory alignment in double (> 0)
```

Execution:

```
mpimon xhpl – admin 1 n0101 1 n0102 1 n0103 1
```

Result:

```
“1 tests completed and passed residual checks”
```

4.10 LAM_MPI 6.5.9

- Download the lam-6.5.9.tar.gz archive from the following site:

http://www.lam-mpi.org/download

or:

```
cd /home/packages_sources/LAM_MPI
```

- Dearchive the file:

```
tar xvfz lam-6.5.9.tar.gz
cd lam-6.5.9
```

- For the installation itself, follow the instructions given in the README and the INSTALL file in the lam-6.5.9 directory.

- What follows is a quick guide so you can compile your version of lam quickly:

First step:

Note: The **efc** and **ecc** compilers must be in your PATH environment variable (cf. section "Intel V7.0 Compilers")

To use LAM on a single SMP machine, compile LAM using the "usysv" option:

```
./configure --prefix=<installation path> --with-cc=ecc –-with-cflags=-O2 --with-cxx=ecc --with-cxxflags=-O2 --with-fc=efc –-with-fflags=-O2  --with-rpi=usysv –-with-select-yield
```

where <Installation path> could be, for example, /opt/envhpc/lam-6.5.9

For a cluster of SMP machines, compile with the "tcp" option.

```
./configure --prefix=<installation path> --with-cc=ecc –-with-cflags=-O2 --with-cxx=ecc --with-cxxflags=-O2 --with-fc=efc  --with-fflags=-O2 --with-rpi=tcp
```

where <Installation path> could be, for example, /opt/envhpc/lam-6.5.9

To find out more about these flags, please see the documentation supplied in the lam-6.5.9.tar.gz archive (INSTALL file).

However, to summarize, we can say that it will enable you construct LAM tools (mpirun, mpicc, mpif77, etc.) using the shared memory and including the Intel ecc and efc compilers.

Second step:

```
make
```

Third step:

```
make install
```

In our case, this will install lam-mpi binaries in <installation path>/lam-6.5.9, for example /opt/envhpc/lam-6.5.9

To then use the LAM-MPI tools, you will first need to perform the following operations:

```
export PATH=<installation path>/lam-6.5.9/bin:$PATH
```

Last step:

You can then compile examples by adding the Vaxlib option to the FFLAGS flag in the Makefile file in lam-6.5.9 and by modifying the */home/packages_sources/LAM_MPI/lam-6.5.9/mpi2c++/contrib/test_suite/signal.cc* file:

```
Replace line:
        n.sa_mask= 0;
With line:
        sigemptyset(&n.sa_mask);
```

Finally:

```
make examples
```

IMPORTANT: At this stage you can create a file defining the environment variables for using the following tools:

- Intel v7.0 compilers
- Intel math libraries
- lam-6.5.9

Example:

```
. /opt/envhpc/intel/compiler70/ia64/bin/efcvars.sh
. /opt/envhpc/intel/compiler70/ia64/bin/eccvars.sh
export PATH=/opt/envhpc/lam-6.5.9/bin:$PATH
export LD_LIBRARY_PATH=/opt/envhpc/intel/mkl/lib/64:$LD_LIBRARY_PATH
export MANPATH=/opt/envhpc/lam-6.5.9/man:$MANPATH
```

Proposed name: ENV_VAR_LAM

An example of this file is given in
/home/packages_sources/VARIABLES_ENVIRONNEMENT

Copy it to /opt/envhpc

To run the examples, you must first and foremost be a "non root" user, then run the ENV_VAR_LAM file.

```
. /opt/envhpc/ENV_VAR_LAM
```

For a cluster, the ENV_VAR_LAM file must also be loaded when a remote machine user logs on. To do this, on all nodes modify the .bashrc of the user in question by adding the following line:

```
. /opt/envhpc/ENV_VAR_LAM
```

Next run the LAM daemon via the following command:

For a single machine:

```
lamboot
```

A simple test is program fpi located in /home/packages_sources/LAM_MPI/lam-6.5.9/examples/pi on the admin machine

```
cd /home/packages_sources/LAM_MPI/lam-6.5.9/examples/pi
mpirun –np 4 fpi
```

For a cluster:

For example, create a "hostfile" file in the /home/packages_sources/LAM_MPI/lam-6.5.9/examples/pi directory:

```
cd /home/packages_sources/LAM_MPI/lam-6.5.9/examples/pi
#create the hostfile file and insert:
admin cpu=4
n0101 cpu=4
n0102 cpu=4
n0103 cpu=4
```

```
lamboot hostfile
```

Make program fpi accessible on all 4 nodes and run:

```
cp /home/packages_sources/LAM_MPI/lam-6.5.9/examples/pi/fpi /opt/envhpc/lam-6.5.9/bin
mpirun –np 16 /opt/envhpc/lam-6.5.9/bin/fpi
```

At the end of the tests, stop the LAM daemon with:

```
lamhalt
```

4.11 PVM 3.4.4

- Go to site http://www.netlib.org/pvm3/index.html and download pvm3.4.4.tgz or retrieve it from the Bull CD.
- Decompress and dearchive the file by:

```
cd /opt/envhpc
tar xvfz /home/packages_sources/PVM/pvm3.4.4.tgz
```

- PVM is now installed in the pvm3 directory.

- Creation of environment variables.
  To do this, create script ENV_VAR_PVM (in /opt/envhpc for example):

```
. /opt/envhpc/intel/compiler70/ia64/bin/efcvars.sh
. /opt/envhpc/intel/compiler70/ia64/bin/eccvars.sh
export LD_LIBRARY_PATH=/opt/envhpc/intel/mkl/lib/64:$LD_LIBRARY_PATH

export PVM_ROOT=/opt/envhpc/pvm3
export PVM_ARCH=LINUX64
export PATH=$PVM_ROOT/lib/$PVM_ARCH:$PVM_ROOT/bin/$PVM_ARCH:$PATH
```

An example of this file is given in
/home/packages_sources/VARIABLES_ENVIRONNEMENT

- Run environment variables:

```
. /opt/envhpc/ENV_VAR_PVM
```

- Go to the /opt/envhpc/pvm3 directory:

```
cd /opt/envhpc/pvm3
```

For information: The various architectures proposed are in the conf subdirectory. The one corresponding to our machine is LINUX64.

- Make the following changes to Makefile.aimk:

```
CC = ecc
F77 = efc
```

- Next, simply type:

```
make
```

- To run PVM and its daemon, simply run the following command:

```
pvm
```

Testing the Installation:

- Run

```
pvm
```

- The following prompt should be displayed:

```
pvm>
```

- To exit, type:

```
quit
```

- The following is displayed:

```
Console : exit handler called
pvmd still running
```

4.12 Performance Analysis and Profiling Tools

There are two types of distribution:
- Official packages in rpm format
- Sets with source code in zipped tar format.

The following packages:

libpfm-2.0-1.ia64.rpm
pfmon-2.0-1.ia64.rpm

are retrieved in the following order, with root user permissions:

```
cd /home/packages_sources/PROFILING
rpm –U libpfm-2.0-1.ia64.rpm
rpm –U pfmon-2.0-1.ia64.rpm
```

To retrieve the tar files:

```
cd /opt/envhpc
```

Restore the .tar.gz archives by:

```
tar xvfz  /home/packages_sources/PROFILING/papi-linux-ia64.tar.gz
tar xvfz  /home/packages_sources/PROFILING/vprof-linux-ia64.tar.gz
```

4.12.1 Pfmon

The documentation for this tool can be found on the following site:
http://www.hpl.hp.com/research/linux/perfmon/pfmon.php4

Run, for example, on the ls command:

```
pfmon /bin/ls
```

and you get:

```
cycles.sh  pfdbg  pfmon

            1 559 994 CPU_CYCLES
```

You can capture 4 events at once:

There is also an option that runs pfmon system-wide to be able to observe the entire machine.

4.12.2 PAPI

```
cd /opt/envhpc/ papi-linux-ia64
```

This contains libraries, "includes", a series of man files and tests.

To read the man files associated with PAPI functions:

```
cd man
export MANPATH=.:$MANPATH
man PAPI_read
```

Otherwise the documentation can be found on the following site:

http://icl.cs.utk.edu/projects/papi/documents

To perform the first PAPI operability test:

```
cd ../ctests
 ./zero
Test case 0: start, stop.
------------------------------------------------
Default domain is: 1 (PAPI_DOM_USER)
Default granularity is: 1 (PAPI_GRN_THR)
Using 10000000 iterations of c += a*b
-------------------------------------------------------------------------
Test type    :            1
PAPI_FP_INS  :     20000000
PAPI_TOT_CYC :    450008301
Real usec    :       452815
Real cycles  :    450550583
-------------------------------------------------------------------------
Verification: none
zero.c                          PASSED
```

If you want to have a debug trace, set the PAPI_DEBUG variable:

```
export PAPI_DEBUG= y
```

However, if this trace pollutes:

```
unset PAPI_DEBUG
```

Once you have done these checks, you are advised to read a source file in the tests directory which will help you to understand the modifications to be made in the application source. Use it as a base to carry out modifications to your program.

To use the PAPI library in shared mode, do not forget to initialize the LD_LIBRARY_PATH environment variable:

```
export LD_LIBRARY_PATH=/opt/envhpc / papi-linux-ia64/lib:$LD_LIBRARY_PATH
```

Note:
Take care when using threads that are still not fully compatible with certain PAPI functions.

4.12.3 VPROF/CPROF

```
cd /opt/envhpc/vprof-linux-ia64
```

The documentation for these tools is contained in a simple README file:

vprof-linux-ia64.README

Read it before starting to understand the tools.

Next, there are the binaries (commands) and the libraries (to edit the linkage of the programme to be profiled) and a few examples to practise on.

The sources of these products are available at:

http://aros.ca.sandia.gov/~cljanss/perf/vprof/

The Makefile that was used to create the binaries is supplied for information purposes.

It is possible, especially if you are only interested in system profiling, to produce a version of cprof/vprof that is independent from PAPI. If this is the case, start again from the site's sources and run the configuration:

```
configure
```

With the desired options.

This shipped version requires PAPI.

Therefore, before using the cprof/vprof commands, specify:

```
export LD_LIBRARY_PATH=/opt/envhpc / papi-linux-ia64/lib:$LD_LIBRARY_PATH
export PATH= /opt/envhpc/vprof-linux-ia64/bin:$PATH
```

See the Use chapter for how to implement a profiling scenario.

4.12.4 VAMPIR

If VAMPIR is supplied, it is available in /home/packages_sources/VAMPIR. Install it as follows:

```
mkdir /opt/envhpc/VAMPIR
```

```
cd /opt/envhpc/VAMPIR
tar xvfz /home/packages_sources/VAMPIR/VA-IA64-LIN-PRODUCT.3.0.0.6.tar.gz
tar xvfz /home/packages_sources/VAMPIR/VT-IA64-LIN-72-PRODUCT.3.0.0.5.tar.gz
```

- Run:

```
./install-Vampir
```

- Answer the questions with "y"
- Save the license in the etc subdirectory:

```
cp /home/packages_sources/VAMPIR/license.dat ./etc/
```

- Run:

```
./install-Vampirtrace
```

- Answer the questions with "y"
- Copy the environment file ENV_VAR_VAMPIR into /opt/envhpc:

```
cp /home/packages_sources/VARIABLES_ENVIRONNEMENT/ENV_VAR_VAMPIR /opt/envhpc
```

This file defines the following environment variables:

```
export PAL_ROOT=/opt/envhpc/VAMPIR
export PAL_LICENSEFILE=$PAL_ROOT/etc/license.dat
export VAMPIR_ROOT=$PAL_ROOT/bin
export PATH=$PATH:$VAMPIR_ROOT
export VAMPIR_LIB_DIR=$PAL_ROOT/lib
```

Obtaining a Program Trace

Note: Before running VAMPIR remember to update the environment: ". /opt/envhpc/ENV_VAR_VAMPIR".

With MPICH:
In the Makefile of the program to execute, make the following change to the linkage settings:

```
-L$(VAMPIR_LIB_DIR) -lVT
```

Compile then run the program. This creates a file with a bvt extension: "Vampirtrace INFO: Writing tracefile <programme>.bvt"

Example:

```
cd /opt/envhpc/mpich-1.2.5/examples
mpicc –o cpi cpi.c -L$VAMPIR_LIB_DIR -lVT
#Run the cpi program
mpirun -np 4 ./cpi
```

Using VAMPIR

To run VAMPIR , execute the following command:

```
vampir
```

Once VAMPIR is open, simply open the <program>.bvt file.

For details on use, see the User Guide in $PAL_ROOT/doc

4.13 System Administration Tools

4.13.1 Installing Webmin and Nagios

Webmin is installed with the HPC system. It can be used immediately.

Webmin is a system administration tool. It has a Web interface.

The Nagios software module is available on the Bull CD-ROM.

To install and use it, you must run the installation script provided by Bull.

Nagios is tracking and monitoring software for the servers and services (e.g. http, pop3, cpu load, etc.) of an information system, via the network. Focusing on prevention, it enables administrators to set critical thresholds and warns them when the thresholds are reached or an incident occurs. The administrator is also informed when the situation returns to normal.

Nagios is shipped non-compiled. The installation script provided by Bull does the required compilation and prepares the config. files.

This script is available on the Bull CD-ROM in the NAGIOS directory.

To run it:

Copy the script to your hard drive (optional).
Go to the directory containing the script.
Then run:

sh installBullnagios.sh *param1 param2* 2>&1 | tee param2/installBullnagios.log

where:
param1= tree structure containing the NAGIOS directory (and associated software)
(e.g. /mnt/cdrom/bull_packages_sources)

param2= directory to which log files will be written
(e.g. /tmp/bull)

Then the installation is complete, the user has an environment that is ready to configure according to the architecture of the information system.

The use, and therefore the configuration, of Nagios are optional.

Throughout the document, the server on which Nagios is installed will be called the 'Nagios server'.

If you decide to use Nagios, you must first:
Determine the other servers (remote hosts) that you want to monitor with Nagios (make sure you have their names and IP addresses).

- Determine which services you want to monitor.
- These services will require plugins (Nagios terminology).
- (cf the /usr/local/nagios/etc/services.cfg and /usr/local/nagios/etc/checkcommands.cfg, config files to see examples of available services and plugins).

Next you can configure Nagios on the Nagios server.
Then install and configure NRPE on the other servers.

In this document we give you some advice about configuring Nagios and NRPE.

4.13.2 Starting and Using Webmin and Nagios

4.13.2.1 Starting Webmin

- Open a Linux shell window and run:

```
# service webmin start      (#: unix prompt)

# service httpd start

# netscape &
(ou mozilla&, …)

------------------------
```

- In your browser, connect to:
  - **https://localhost :10000**
  - **or: http://localhost:10000**

- Note: When you make changes, you may need to run the following commands:

```
service webmin start /restart  (but also stop, reload, status)
service httpd start / restart  (and also stop, …).
```

4.13.2.2 Example of Use - Configuring the Network with Webmin

If you have not yet configured the network, you can do it with Webmin:

Select the 'Networking' folder then:
'network configuration'
then each of the available modules
network interface, routing & gateways, DNS client, Host Addresses

To ensure your declarations are accepted, run the following command in a Linux shell window.

```
# service network restart
```

4.13.2.3 Starting NAGIOS

During the Nagios installation phase, the script compiles and installs the following software:
Nagios, nagios-plugins, nagat, and nrpe
During this phase, the **'nagios'** and 'nagat' users are also created.

Nagat is the administration interface used to **configure Nagios**. After you have installed Nagios, this interface is immediately available.
To access it, type in the following url:
..http://localhost/nagat/

The **Nagios Web interface** (CGI scripts) is also available. To access it, type in the following url:

..http://localhost/nagios/

This interface will give you access to the **Nagios documentation**.

If Nagios has been started, this interface will also enable you to **view the status of the servers and services monitored with Nagios.**

Note: If you have installed Nagios, with the Bull installation script you should be able to run Nagios as soon as the installation is complete and view a fictitious configuration, (cf menu: status map and service detail).

To start / stop nagios:

```
→ etc/rc.d/init.d/nagios start  / restart / reload / status / stop
```

- Note:
  If you encounter difficulties calling these urls, try restarting httpd and your browser.

```
# httpd service start / restart …
# netscape & (or mozilla&, …)
```

4.13.3 Configuring Nagios (and nrpe)

4.13.3.1 Configuring Nagios on the Nagios Server

Nagios requires 10 config files (.cfg) to work.
These are located in /usr/local/nagios/etc.

In each of these files, examples are supplied as standard with Nagios.
We recommend you model your own configuration on these examples. Your configuration will depend in particular on:

- The servers to be monitored (including the nagios server).
- The way you choose to group these servers together.
- The services and monitoring commands that you implement.
- The administrator(s) to be informed if a warning or change of status occurs.

You must therefore at least configure the data described in the following files:
hosts.cfg
hostgroups.cfg
services.cfg
checkcommands.cfg.
contacts.cfg
contactgroups.cfg

To test the consistency of your configuration, run the following command:
service nagios reload

Note:
Restart and/or reload Nagios if any changes are made to the config files.
→ service nagios restart, or reload

Once the configuration has been defined, view the status of the services and servers via the Web interface. http://localhost/nagios

4.13.3.2 (Installing and) Configuring NRPE and check_nrpe for a Remote Host

Preliminary comment: With Nagios, you can monitor a remote host via some of the available nagios services (plugins): ping, http, …

However, for other operations, such as monitoring the cpu load on a remote host, you need to install:
- nrpe and nrpe commands on this host
- the corresponding nrpe services on the nagios server

Recommendation: Run Nagios on the nagios server, with at least one server querying the remote host, before adding nrpe and nrpe services.

You can obviously define and manage several remote hosts individually.

4.13.3.2.1 Operations to be Performed on a Remote Server (Host)

The information given below is valid if your remote host has the same environment as the nagios server, i.e. if it has the same operating system, compiler and C and graphics libraries.
If this is not the case, you will need to install 'downloader' and compile nrpe on the remote host.

1) Install the nrpe files on the remote server.

Check if the following directories exist:
/usr/local/nagios/etc
/usr/local/nagios/libexec
(These directories should only exist if Nagios has been pre-installed on this server).

If these directories do not exist, create them and copy to them all the files in the same directories on the nagios server.
Note: During this operation, you should in particular copy a file called nrpe.cfg and a nrpe executable.

In the same way, copy the /etc/xinetd.d/nrpe file from the nagios server to the same tree structure on the remote server(s).

Create the nagios user:
adduser nagios

Edit file /etc/services

Remove the comment character ('#' at the start of the line) from the following line:
nrpe 5666/tcp NRPE
to validate it,
otherwise create the line

Edit /etc/xinetd.d/nrpe

remove all the comment characters from the lines between:
'service nrpe' '}' inclusive.
Note: These last two actions have enabled you to define the nrpe daemon

Validate (enable) the nrpe daemon
Run the following command:
/etc/rc.d/init.d/xinetd.d restart

View file /usr/local/nagios/nrpe.cfg
Check that several commands have been defined, including check-host.

This completes the installation and implementation of nrpe on the remote Linux server(s) to be monitored for the check-host service.
This service is defined by way of an example.

It is also possible to define (configure) other services. To do this, see the examples given in the config file: /usr/local/nagios/etc/nrpe.cfg on the remote host

It is then necessary to complete these installations:

- By installing and validating check_nrpe on the nagios server.
- By creating a service that will require the execution of the appropriate service on the remote host, via chek_nrpe.

4.13.3.2.2 Operations to be Performed on the Nagios Server

1) Edit file /usr/local/nagios/etc/checkcommands.cfg
Find the example of the check_nrpe command.
Validate this command, by removing the comments and replacing them by the address of a remote server or creating one or more according to the same principle.

Edit file /usr/local/nagios/etc/cservices.cfg
Find the example of the service with nrpe supplied: service c-nrpe-dist1
Validate this service, by removing the comments and replacing them with the name of the remote server.
Or by creating one or more along the same lines.

Note: In this example, the c-nrpe-dist1 service calls the check-host command defined on the remote server(s). See the previous paragraph and the nrpe.cfg file.

4.13.3.2.3 Enabling nrpe and Associated Services

So that nrpe and its associated elements are included:

- Reload and restart nagios on the nagios server:
  - service nagios reload
  - service network restart
- On the remote host side: automatic (performed by the daemon), if you have installed and configured nrpe as described earlier and if you have restarted xinetd.d:
  - /etc/rc.d/init.d/xinetd.d restart.

4.14 Cluster Administration Tools

4.14.1 Ganglia

Prerequisites

- The server node must be a Web server.

Installing RRDTOOL 1.0.41

This component is required to use Ganglia (It can be downloaded from: http://rrdtool.com/download.html).

- Install RDDTOOL:

```
cd /home/packages_sources/GANGLIA
rrdtool-1.0.41-1.ia64.rpm
```

Installing Ganglia

Ganglia documentation is available in PDF format in the /home/packages_sources/GANGLIA directory.

Note: Files in rpm format are from http://ganglia.sourceforge.net/ (and are also available from http://prdownloads.sourceforge.net/ganglia/).

To have a graphical display of the node information, you will need to install *ganglia-monitor-core-gmond-2.5.3-1.ia64.rpm* on all nodes:

On all the nodes:

```
cd /home/packages_sources/GANGLIA
rpm –ivh ganglia-monitor-core-gmond-2.5.3-1.ia64.rpm
```

For the graphical display, install *ganglia-monitor-core-gmetad-2.5.3-3.ia64.rpm* on the server node:

```
cd /home/packages_sources/GANGLIA
rpm –ivh ganglia-monitor-core-gmetad-2.5.3-3.ia64.rpm
```

To find out if the files have installed correctly, use the following command:

```
rpm –qa | grep ganglia
```

If everything has worked correctly, it should return:
ganglia-monitor-core-gmetad-2.5.3-1 (on the server node only)
ganglia-monitor-core-gmond-2.5.3-3

To find out where the various ganglia-monitor-core-gmetad-2.5.3-1 and ganglia-monitor-core-gmond-2.5.3-3 files have been placed, do the following commands:

```
rpm –ql ganglia-monitor-core-gmetad-2.5.3-1
rpm –ql ganglia-monitor-core-gmond-2.5.3-3
```

on the server, modify /etc/gmetad.conf and restart the gmetad daemon.

```
vi /etc/gmetad.conf
### change the "data_source" line
data_source "my_cluster" <admin>
service gmetad restart
```

Modify the /etc/gmond.conf config file on all modes and then restart the daemon.

```
vi /etc/gmond.conf
### Modify the name line
name "my_cluster"
service gmond restart
```

Installing *ganglia-web-frontend*

Install *ganglia-webfrontend-2.5.3-1.ia64.rpm* on the server node:

```
rpm –ivh /home/packages_sources/GANGLIA/ganglia-webfrontend-2.5.3-1.ia64.rpm
```

Open a Web page at the following address:
http:// <***IP address of server node***>/ganglia-webfrontend-2.5.3

If the page does not open, it may be that the Apache server is not running. Simply enter the following command:

```
service httpd start
```

If the page is displayed but the information about the nodes is clearly wrong and the switch being used is a 3COM Gigabit switch, check the configuration in the following section: "Configuring the 3COM Gigabit Switch"

Using Ganglia

List of metrics and their meanings:

Metric	Meaning
boottime	Date of last machine reboot
cpu_idle	% of CPU not being used
cpu_nice	% of CPU used to handle priorities
cpu_num	Number of processors
cpu_speed	Processor speed
cpu_System	% of CPU used by the kernel
cpu_user	% of CPU used by the processors
gexec	
load_fifteen	Load on the system over a 15 minute interval
load_five	Load on the system over a 5 minute interval
load_one	Load on the system over a 1 minute interval

machine_type	= ia64
mem_buffers	Amount of memory allocated to the buffer
mem_cached	
mem_free	Amount of free memory
mem_shared	Segment of shared memory
mem_total	Total memory
os_name	= Linux
os_release	=2.4.9-18smp
proc_run	Number of processors running (should be checked as the number = 0 if no major processor is running)
proc_total	Total number of processes
swap_free	Amount of free swap
swap_total	Total amount of swap
sys_clock	Date when gmond was started

4.14.2 Gexec

Prerequisites:

To use **gexec,** you first need to install the authentication by key tool - **authd**, having created the set of private/public keys to be distributed on all the nodes of the cluster.

1) Generating keys:

```
openssl genrsa -out /etc/auth_priv.pem
```

- If you get a message telling you to use the "-rand" option, proceed as follows:

```
openssl genrsa –rand /etc/mime-magic –out /etc/auth_priv.pem
```

- Then, for the public key:

```
chmod 600 /etc/auth_priv.pem
openssl rsa –in /etc/auth_priv.pem –pubout –out /etc/auth_pub.pem
```

-

2) Distributing keys:

- Copy the public key to all the nodes of the cluster (those on which the **authd** demon is running):

```
scp /etc/auth_pub.pem node(i):/etc
```

- Copy the private key to all the nodes on which **gexec** will be used (usually the administration node):

```
scp /etc/auth_priv.pem node(i):/etc
```

-

3) Installing **authd**:

- *Install on all the nodes of the cluster from the RPM archive*

```
cd /home/packages_sources/GEXEC
rpm –ivh authd-0.2.1-1.ia64.rpm
service authd start
```

Installation:

- *Install on all the nodes of the cluster from the RPM archive*

```
cd /home/packages_sources/GEXEC
rpm –ivh gexec-0.3.4-1.ia64.rpm
service xinetd restart
chkconfig --add authd
```

Use:

Define an environment variable describing the list of nodes in the cluster (IP or DNS):

```
export GEXEC_SVRS=“node1 node2 node3 node4 node5"
```

Run a command on the first i nodes on the list (0 for all):

```
export PATH=/opt/envhpc/gexec-0.3.4/bin:$PATH
gexec –n i command
```

A description of **gexec** is available in the **ganglia** documentation version pre-2.5.0, available in /home/packages_sources/GEXEC.

A document describing the specific installation procedures for an adherence between **ganglia** and **gexec** is available in /home/packages_sources/GEXEC

4.15 Task Sharing Tools

4.15.1 OpenPBS

Prerequisites:

- **Register your contact details and get download authorization.** To obtain authorization to use OpenPBS "open source" software free-of-charge, you will need to register and accept the conditions listed on the site. www.openmp.org Your user name and password will by sent to you by e-mail within a few hours.

Generating Binaries:

- **Download the sources from the site given in the mail**

- Download «OpenPBS Version 2.3.16" by right-clicking. The file you retrieve is OpenPBS_2_3_16.tar.gz. Put it in the /home/packages_sources/OPENPBS directory.
- The documentation can also be downloaded from here (especially file v2.3_admin.pdf)
- Decompress the file you retrieved.

```
cd /home/packages_sources/OPENPBS
### Download OpenPBS_2_3_16.tar.gz and the documentation ###
tar xvzf OpenPBS_2_3_16.tar.gz
```

- ***Compilation***
  - Modify "config.sub" to support ia64

```
vi OpenPBS_2_3_16/buildutils/config.sub
```

After lines:

```
        pentiumpro | p6)
            basic_machine=i686-intel
            ;;
```

Add:

```
        ia64-pc)
            basic_machine=i686-intel
            ;;
```

- Compile the OpenPBS sources

```
cd OpenPBS_2_3_16
./configure
make
```

Installation:

```
make install
```

- Set the suid bit on the program pbs_iff

```
chmod +s /usr/local/sbin/pbs_iff
```

Configuration:

- Run the daemons

```
pbs_mom
pbs_server –t create
### Warning – the –t create option initializes the database and must only be used the first time.
qmgr -c "set server scheduling=true"
pbs_sched
```

- Declare the various nodes of the cluster

```
vi /usr/spool/PBS/server_priv/nodes
### Enter the names of the nodes, for example:
node1 np=4
node2 np=4
node3 np=4
node4 np=4
vi /usr/spool/PBS/mom_priv/config
### insert the lines:
$logevent 0x1ff
$clienthost admin
```

- Declare the various queues of the cluster. e.g.

```
qmgr
c q dque
s q dque queue_type=Execution
s q dque enabled=true
s q dque started=true
```

- Configure the pbs_mom, pbs_server and pbs_sched services

```
cp /home/packages_sources/OPENPBS/pbs_mom /etc/init.d
chkconfig --add pbs_mom
chkconfig --level 35 pbs_mom on
cp /home/packages_sources/OPENPBS/pbs_server /etc/init.d
chkconfig --add pbs_server
chkconfig --level 35 pbs_server on
cp /home/packages_sources/OPENPBS/pbs_sched /etc/init.d
chkconfig --add pbs_sched
chkconfig --level 35 pbs_sched on
```

- For computation nodes, if these exist
  - Create the following directories on all the computation nodes

```
mkdir /usr/spool
mkdir /usr/spool/PBS
mkdir /usr/spool/PBS/aux
mkdir /usr/spool/PBS/checkpoint
mkdir /usr/spool/PBS/mom_logs
mkdir /usr/spool/PBS/mom_priv
mkdir /usr/spool/PBS/mom_priv/jobs
mkdir /usr/spool/PBS/spool
mkdir /usr/spool/PBS/undelivered
```

- Copy the following files to all the computation nodes

```
/usr/init.d/pbs_mom
/usr/init.d/pbs_server
/usr/init.d/pbs_sched
/usr/local/bin/chk_tree
/usr/local/bin/hostn
/usr/local/bin/nqs2pbs
/usr/local/bin/pbs_tclsh
/usr/local/bin/pbs_wish
/usr/local/bin/pbsdsh
/usr/local/bin/pbsnodes
/usr/local/bin/printjob
/usr/local/bin/qalter
/usr/local/bin/qdel
/usr/local/bin/qdisable
/usr/local/bin/qenable
/usr/local/bin/qhold
/usr/local/bin/qmgr
/usr/local/bin/qmove
/usr/local/bin/qmsg
/usr/local/bin/qorder
/usr/local/bin/qrerun
/usr/local/bin/qrls
/usr/local/bin/qrun
/usr/local/bin/qselect
/usr/local/bin/qsig
/usr/local/bin/qstart
/usr/local/bin/qstat
/usr/local/bin/qstop
/usr/local/bin/qsub
/usr/local/bin/qterm
```

```
/usr/local/bin/tracejob
/usr/local/include/pbs_error.h
/usr/local/include/pbs_ifl.h
/usr/local/lib/libattr.a
/usr/local/lib/libcmds.a
/usr/local/lib/liblog.a
/usr/local/lib/libnet.a
/usr/local/lib/libpbs.a
/usr/local/lib/libsite.a
/usr/local/lib/pbs_sched.a
/usr/local/sbin/pbs_demux
/usr/local/sbin/pbs_iff
/usr/local/sbin/pbs_mom
/usr/local/sbin/pbs_rcp
/usr/spool/PBS/pbs_environment
/usr/spool/PBS/server_name
/usr/spool/PBS/mom_priv/config
```

- Configure the pbs_mom service on each computation node and start it.

```
chkconfig --add pbs_mom
chkconfig --level 35 pbs_mom on
service pbs_mom start
```

- Restart pbs_mom, pbs_server and pbs_sched on the administration node.

```
service pbs_mom restart
service pbs_server restart
service pbs_sched restart
```

First test:

- **Define at least one login ("linux" for example**) present on all nodes with the same uid/gid. These logins, different from "root", will be used for job submission.
- Create a shell, start it with qsub and ensure it is in the queue

With a login other than root (e.g. linux) create an executable file named SLEEP:

```
vi SLEEP
#!/bin/sh
hostname
sleep 30
date
```

***Then type the following commands**:*

```
chmod +x SLEEP
qsub –q dque SLEEP
qstat -a
```

Use:

- Using OpenPBS implies you accept the license terms (see file PBS_License.text in the home/packages_sources/OpenPBS/OpenPBS_2_3_16 directory).

- The OpenPBS documentation (v2.3_admin.pdf) is available on the PBS Web site (see "Generating binaries" above). It explains in more detail how to generate, configure and use OpenPBS.

4.15.2 SCALI OpenPBS: ScaOPBS

Prerequisites:

- ***Define at least one login ("linux" for example***) present on all nodes with the same uid/gid. These logins, different from "root", will be used for job submission.

Installation:

- The installation procedure for SCALI distribution of OpenPBS is described in detail in Chapter 8 of the "Scali System Guide".

Testing the Installation:

- The installation procedure automatically runs a test that can also be called by the following command:

```
/opt/scali/libexec/scaopbs.config -t linux
```

Using SCALI OpenPBS implies acceptance of the license terms (see /opt/scali/contrib/pbs/doc/PBS_License.text).

The OpenPBS documentation (pbs_admin_guide.pdf) can be found in the /opt/scali/contrib/doc directory. It explains in more detail how to generate, configure and use OpenPBS.

4.15.3 MAUI

Prerequisites

- OpenPBS is already installed.
- If MAUI is installed on SCALI OpenPBS, you must also install the "ScaOPBSdk" rpm.

```
cd /opt/scali/repository/Linux2.ia64
rpm    -ivh ScaOPBSdk.Linux2.ia64-2.3.15-3.rpm
```

Installing Sources

- The "maui-3.0.7.tar.gz" file containing the MAUI sources is located in the /home/packages_sources/MAUI directory (this file has been downloaded from: http://supercluster.org/downloads/maui)
- Decompress the file

```
cd /home/packages_sources/MAUI
tar xvzf maui-3.0.7.tar.gz
```

Compilation and Installation

- Create directory /opt/envhpc/MAUI
- Configure the makefiles (use the gcc compiler)
- Generate and install the binaries
- Run maui

```
mkdir /opt/envhpc/MAUI
cd maui-3.0.7
./configure
### specify:
### - installation directory: /opt/envhpc/MAUI
### - Home directory for MAUI: /opt/envhpc/MAUI
### - Compiler              : gcc
### - Random number         : 10
### - Validate the configuration: Y
### - Use of PBS       : Y
### - PBS directory       : /opt/scali/contrib/pbs (SCALI)
###                       or /usr/local if OpenPBS
make all
make install
# In the /etc/init.d/maui file, check the value of variable
            processname= "/opt/envhpc/MAUI/bin/$prog"
# Configure the maui service:
cp /home/packages_sources/MAUI/maui /etc/init.d
chkconfig --add maui
chkconfig --level 35 maui on
```

```
# Start maui (this command also stops the pbs_sched service)
service maui start
/opt/envhpc/MAUI/bin/maui
```

Configuration and Use

- All the MAUI documentation can be found at http://supercluster.org/documentation. It explains how to configure and use MAUI.
    - Directory /home/packages_sources/MAUI/samples provides examples of config files.

Chapter 5. Uninstalling Intel Products

The use of Intel software, C/C++ and Fortran compilers and the mkl library is subject to license. The end user must accept the terms of the license. In order to respect this process, Intel software used during the HPC software installation phase must be uninstalled before delivery to the customer.

The uninstall is done as root. Run the following script:

```
opt/envhpc/intel/compiler70/ia64/bin/uninstall
```

A list of installed Intel software is displayed. Uninstall one by one the C/C++ compiler, the Fortran compiler, the debugger and the mkl library.

Delete the directory containing the Intel licenses.

```
rm –rf /opt/envhpc/intel/licenses
```

Chapter 6. FAQ

Problems when compiling and executing

- What do I do when I get the error message: "**error while loading shared libraries**" when executing a program?
- My parallel program cannot find the program on the other machines.
- How do I optimize compilation with the **Intel fortran compiler**?

Problems when compiling and executing with MPICH

- I have a problem with **memory allocations** when I use MPICH.

OpenMP

- To run a program parallelized with OpenMP, how do I **define the number of threads** (processors) used?

▪ *What do I do when I get the error message: "error while loading shared libraries" when executing a program?*

Add the path for this library to the LD_LIBRARY_PATH environment variable.

▪ *My parallel program cannot find the program on the other machines.*

You must have the binaries on all machines running the benchmarks and respect the tree structure of the machine from which the benchmark is started, or use NFS.

▪ *How do I optimize compilation and debugging with the Intel fortran compiler?*

For optimization, add the following compilation options:

- **implicitnone**: forces the declaration of variables: If a variable is used without being declared, this triggers errors on compilation.

- **w95**: removes warnings for non standard f95 instructions.
- **mp**: Respects IEEE standard double precision.
- **unroll2**: To unroll a loop: This favors vectorization and the instructions pipeline.
- **ip,ipo**: Optimizes calls to a subprogram (parameter management).
- **auto**: Allocates the variables dynamically to the stack rather than in static storage in the memory.
- **zero**: Implicitly initializes variables to 0.
- **ftz:** flush-to-zero.
- **i_dynamic:** Avoids loading static libraries and therefore reduces the size of the executable.
- **parallel**: Parallelizes certain sequences (supplied by the par_report option).
- **par_report3**: Provides information about how successful the compilation has been (e.g. parallelized loops).
- **openmp:** Takes into account OpenMP directives.
- **restrict:** This is a default option.

For debugging, add the following compilation options:

- **g**: debugging
- **fpp**: pre-processing

▪ *How do I optimize compilation and debugging with the Intel C / C++ compiler?*

- **O3**: Highest code optimization possible.
- **mp**: Respects IEEE standard double precision.
- **ip,ipo**: Optimizes calls to a subprogram (parameter management).
- **unroll**: To unroll a loop: This favors vectorization and the instructions pipeline.
- **prof_gen and prof_use**: Instrumentation.

- ***I have a problem with memory allocations when I use MPICH.***

Error message displayed during execution:

> p3_1858: (18446744073792.328125) xx_shmalloc: returning NULL; requested 65584 bytes
>
> p3_1858: (18446744073792.328125) p4_shmalloc returning NULL; request = 65584 bytes
>
> You can increase the amount of memory by setting the environment variable
>
> P4_GLOBMEMSIZE (in bytes)

The memory the communication requires cannot be allocated correctly. To do this:

- export P4_GLOBMEMSIZE=260000000

- ***To run a program parallelized with OpenMP, how do I define the number of threads (processors) used?***

export OMP_NUM_THREADS=2 to run the program on 2 processors
export OMP_NUM_THREADS=4 to run the program on 4 processors

Glossary

BANDWIDTH
The bandwidth is the frequency interval (the "band") transmitted without major distortion on a well-defined transmission medium. It is measured in Hz. For networks and by extension for other media, it represents the quantity of data transmitted by unit of time, i.e. the throughput. In this case it is measured in bit/s.

BEOWULF
Supercomputer comprising numerous client stations executing the same code in parallel. The clients, as 'thin" as possible are called "nodes" and are controlled via a network by one or more servers. Beowulf requires no special hardware, just common network machines and equipment. The software is also common, for example the Linux operating system, PVM and MPI. The definition is sometimes restricted to machines built in line with the first Beowulf model, designed by NASA. Beowulf systems are divided into two classes, depending whether they are built from elements found in the local hardware store (Class I), or whether they include specially designed elements (Class II).

CLUSTER
Architecture of a group of computers used to form large servers. Each machine is a node of the cluster, the whole is considered as a single machine. Used for scientific computation, decision-support, transactional computing and data warehousing.

THROUGHPUT
Amount of information using a communication channel during a certain time period. Measured in Mbit/s or MB/s.

FSF
The Free Software Foundation is an American Organization dedicated to abolishing restrictions concerning copying, redistribution, understanding and modification of computer programs. It promotes the development and use of Free Software and initiated the GNU project.

GNU
A recursive acronym "GNU's Not Unix".

GNU/LINUX
The pairing formed by the Linux kernel and the GNU system.

GPL
General Public License. The legal status of software distributed "freely". First used for the FSF GNU project.

HPC
High Performance Computing .

Itanium™ Architecture
The 64 bit architecture of the latest Intel chips, designed to replace the x86. This is a complete break with the x86 series, the instruction set and the architectural elements of the processor being totally different. The result is something far more simple and therefore much faster, giving the software more precise control (in particular the compilers) over the hardware. The first version is called Itanium (previously called Merced). It has taken 6 years of development for it to appear on the market. The second generation, Itanium-2, has now been launched.

ITANIUM™ Processor
64 bit Intel processor, designed to replace the x86s (including the Pentiums), while remaining compatible. Itanium is the first model of the Itanium architecture and Itanium-2™ is the second.

LATENCY
Minimum time for propagating a signal. By extension, minimum time for transmitting a set of data through a network.

LINUX
Linux is a UNIX type multi-tasking, multi-user operating system that is available on numerous hardware platforms, especially those with x86 and Itanium processors. It integrates most of the latest technology (SMP, clustering, RAID, etc.).
Linux is a kernel. To use it, you require applications, provided by distributions. A distribution is a set of programs and a kernel for installation on a machine. RedHat, Mandrake, Suse and TurboLinux are some of the most common Linux distributions.

MONITORING
Step by step control, i.e. that leaves no freedom to the system being monitored. Monitoring controls and checks a process in real time.

MPI
Message Passing Interface. Portable library used for parallel applications.

NFS
Network File System. A system for managing network files. It was developed by Sun in 1985 for its disk-free workstations. The versions that are mainly used today are version 2 (using UDP) and, since 1993, 3 (that can use UDP or TCP).

NODE
A computer connected to a network. In the clustering world, each machine in a cluster is called a node. Their functions may differ: management node, computation node, storage node.

OPEN SOURCE
Particular definition of free software, created in 1998 by Eric Raymond, who was trying to adapt the principle to companies. For the moment it has 9 points: Free redistribution, availability of source code, derivative work can be distributed, original source code is respected, no one is excluded, no applications are excluded, the license is distributed without being product-specific, sublicensees have the same rights as the licensee and it does not infringe on the work of others.

PARALLELIZATION
Transforming a program so it can be executed effectively on several processors.

PROCESS
Program executing, with its environment. Term originally used essentially in the world of Unix.

PVFS
Parallel Virtual File System. Project defined as an exploration of the "design, implementation and potential uses of parallel input/output". Developed by the University of Clemson and NASA, PVFS is designed for workstation clusters and Beowulf machines.

PVM
Parallel Virtual Machine API managing communication between the nodes of a cluster of machines.

RAID
Installation of a set of hard disks to increase storage and reliability.

RISC
Processor architecture.

SCALABILITY
Ability of a system to handle an increase in its limitations in a particular domain or in all domains. For example, a system is said to be scalable in terms of the number of users if its performance is just as good with 100 connected users as with 10.

FILE SERVER
A server that only places files at the network's disposal and not its other resources (e.g. computation power, connections, etc.). In general its strong point is its hard drive.

SMP
Symmetric MultiProcessing. Multiprocessor system that distributes tasks symmetrically between different processors sharing a common memory and that ensures that they will not all start writing to the same address at the same time.

SWAP
The technique of using a part of the hard drive as RAM.

FMS (FILE MANAGEMENT SYSTEM)
This defines, for example, the internal structure of a tree structure, backup procedures, disk partitioning, file metadata, etc.
An FMS is made up of a management service (to organize files) and a file system (for secure operations within files).

TRACE
A program's Trace is the successive statuses of its environment as it executes.

References

General Information about Clusters

www.beowulf.org
www.tldp.org/HOWTO/Parallel-Processing-HOWTO.html
www.top500.org
www.phy.duke.edu/brahma/beowulf_online_book/
www.Linux-Consulting.com/Cluster/

Parallel Architectures

www.lri.fr/~fci/support95.html
www.idris.fr

Interconnects

www.scali.com
www.dolphin.com
www.essi.fr/~riveill/rapport01-these-cecchet.pdf
www.ens-lyon.fr/~rewestrel/thses.ps

HPC

www.epcc.ed.ac.uk/HPCinfo
"High Performance Computing" by Dowd & Severance published by O'Reilly.

Scientific Libraries

www.lifl.fr/west/courses/cshp/bibsp.pdf
www.irisa.fr/orap/Publications/Forum6/petitet.ps
www.netlib.org

Vos remarques sur ce document / Technical publication remark form

Titre / Title : Bull HPC Linux Installation Guide

Nº Reférence / Reference Nº : 86 A2 31EG 02

Daté / Dated : July 2003

ERREURS DETECTEES / ERRORS IN PUBLICATION

AMELIORATIONS SUGGEREES / SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Vos remarques et suggestions seront examinées attentivement.
Si vous désirez une réponse écrite, veuillez indiquer ci-après votre adresse postale complète.

Your comments will be promptly investigated by qualified technical personnel and action will be taken as required.
If you require a written reply, please furnish your complete mailing address below.

NOM / NAME : _____ Date : _____

SOCIETE / COMPANY : _____

ADRESSE / ADDRESS : _____

_____

Remettez cet imprimé à un responsable BULL ou envoyez-le directement à :

Please give this technical publication remark form to your BULL representative or mail to:

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

Technical Publications Ordering Form

Bon de Commande de Documents Techniques

To order additional publications, please fill up a copy of this form and send it via mail to:
Pour commander des documents techniques, remplissez une copie de ce formulaire et envoyez-la à :

BULL CEDOC
ATTN / Mr. L. CHERUBIN
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

Phone / Téléphone : +33 (0) 2 41 73 63 96
FAX / Télécopie +33 (0) 2 41 73 60 19
E–Mail / Courrier Electronique : srv.Cedoc@franp.bull.fr

Or visit our web sites at: / Ou visitez nos sites web à:
http://www.logistics.bull.net/cedoc
`http://www-frec.bull.com` `http://www.bull.com`

CEDOC Reference # Nº Référence CEDOC	**Qty** Qté	**CEDOC Reference #** Nº Référence CEDOC	**Qty** Qté	**CEDOC Reference #** Nº Référence CEDOC	**Qty** Qté
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]		_ _ _ _ _ _ _ _ _ [_ _]	
[_ _] : **no revision number means latest revision** / pas de numéro de révision signifie révision la plus récente					

NOM / NAME : ______________________________ Date : __________

SOCIETE / COMPANY : ______________________________

ADRESSE / ADDRESS : ______________________________

____

PHONE / TELEPHONE : ____________________ FAX : ____________________

E–MAIL : ______________________________

For Bull Subsidiaries / Pour les Filiales Bull :
Identification: ______________________________

For Bull Affiliated Customers / Pour les Clients Affiliés Bull :
Customer Code / Code Client : ______________________________

For Bull Internal Customers / Pour les Clients Internes Bull :
Budgetary Section / Section Budgétaire : ______________________________

For Others / Pour les Autres :
Please ask your Bull representative. / Merci de demander à votre contact Bull.

PLACE BAR CODE IN LOWER LEFT CORNER

BULL CEDOC
357 AVENUE PATTON
B.P.20845
49008 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
86 A2 31EG 02

Bull

Utiliser les marques de découpe pour obtenir les étiquettes.
Use the cut marks to get the labels.

HPC Linux
Installation Guide

86 A2 31EG 02

HPC Linux
Installation Guide

86 A2 31EG 02

HPC Linux
Installation Guide

86 A2 31EG 02