



PGI CDK[®]
Cluster Development Kit[®]
Installation Guide
Release 2011

The Portland Group[®]

While every precaution has been taken in the preparation of this document, The Portland Group® (PGI®), a wholly-owned subsidiary of STMicroelectronics, Inc., makes no warranty for the use of its products and assumes no responsibility for any errors that may appear, or for damages resulting from the use of the information contained herein. The Portland Group retains the right to make changes to this information at any time, without notice. The software described in this document is distributed under license from STMicroelectronics and/or The Portland Group and may be used or copied only in accordance with the terms of the end-user license agreement ("EULA").

PGI Workstation, PGI Server, PGI Accelerator, PGF95, PGF90, PGFORTRAN, and PGI Unified Binary are trademarks; and PGI, PGHPE, PGF77, PGCC, PGC++, PGI Visual Fortran, PVE, PGI CDK, Cluster Development Kit, PGPROF, PGDBG, and The Portland Group are registered trademarks of The Portland Group Incorporated. Other brands and names are property of their respective owners.

No part of this document may be reproduced or transmitted in any form or by any means, for any purpose other than the purchaser's or the end user's personal use without the express written permission of STMicroelectronics and/or The Portland Group.

PGI CDK® 2011 Cluster Development Kit® Installation Guide

Copyright © 2010-2011 STMicroelectronics, Inc.

All rights reserved.

Printed in the United States of America

First Printing: Release 2011, version 11.0, December 2010

Second Printing: Release 2011, version 11.1, January 2011

Technical support: trs@pgroup.com

Sales: sales@pgroup.com

Web: www.pgroup.com

Contents

1. Release 2011 Introduction	1
Product Overview	1
Terms and Definitions	2
Supported Processors	2
Supported Operating Systems	3
Product Support	6
2. PGI CDK 2011 Installation Overview	7
Introduction	7
Open Source Component Overview	9
3. Licensing	11
Licensing Terminology	11
Permanent and Trial License Keys	11
License Keys and System Configurations	12
CDK Licensing	12
The FlexNet License Manager	12
License Support	12
4. PGI CDK Installations on Linux	13
Preparing to Install on Linux	13
Installation Steps for Linux	15
Typical Directory Structure for Linux	23
PGI CDK End-user Environment Settings	24
Set Environment Variables for Licensing	25
Set End-User PATH Variables	25
Common Linux Installation Issues	26
Java Runtime Environment (JRE)	26
5. Using the Open Source Cluster Utilities	27
Running an MPICH, MPICH2, or MVAPICH Program	27
Create “Hello World”	27

Prepare to Run a Program	28
Execute a Program Normally	29
Compile and Execute a Program Summary	29
Invoking PGDBG for MPI Debugging	30
Linking with ScaLAPACK	31
Testing and Benchmarking	32
Limitations	32
Using mpi Scripts	33
6. Contact Information	35

Figures

4.1. CDK Linux Installation Overview	15
--	----

Tables

1.1. Processors Supported by PGI 2011	3
1.2. Operating Systems and Features Supported in PGI 2011	4
4.1. Linux Directory Structure Sample	23

Chapter 1. Release 2011

Introduction

Welcome to Release 2011 of the *PGI CDK[®] Cluster Development Kit[®]*, a set of Fortran, C, and C++ compilers and development tools for 32-bit and 64-bit x64-compatible processor-based workstations and servers running versions of the Linux operating systems.

A *cluster* is a collection of compatible computers connected by a network. The *PGI CDK Cluster Development Kit* supports parallel computation on clusters of 32-bit and 64-bit x86-compatible AMD and Intel processor-based Linux workstations or servers interconnected by a TCP/IP-based network, such as Ethernet.

The *PGI CDK* supports 64-bit x64 (AMD64, Intel 64) processor-based systems, with large array addressing in PGF77, PGF95, PGFORTRAN, PGC++, and PGCC. These systems can utilize a 64-bit address space while retaining the ability to run legacy 32-bit x86 executables at full speed.

Product Overview

Release 2011 of the *PGI CDK* includes the following components:

- PGFORTRAN[™] native OpenMP and auto-parallelizing Fortran 2003 compiler.
- PGF77[®] native OpenMP and auto-parallelizing FORTRAN 77 compiler.
- PGHPF[®] data parallel High Performance Fortran compiler.
- PGCC[®] native OpenMP and auto-parallelizing ANSI C99 and K&R C compiler.
- PGC⁺⁺[®] native OpenMP and auto-parallelizing ANSI C⁺⁺ compiler.
- *PGPROF*[®] MPI, OpenMP, and multi-thread graphical profiler.
- *PGDBG*[®] MPI, OpenMP, and multi-thread graphical debugger.
- MPICH MPI libraries, version 1.2.7, for both 32-bit and 64-bit development environments (Linux only).
- MPICH2 MPI libraries, version 1.0.5p3, for both 32-bit and 64-bit development environments.
- MVAPICH MPI libraries, version 1.1, for both 32-bit and 64-bit development environments

- ScaLAPACK linear algebra math library for distributed-memory systems, including BLACS version 1.1- the Basic Linear Algebra Communication Subroutines) and ScaLAPACK version 1.7 for use with MPICH or MPICH2 and the PGI compilers on Linux systems with a kernel revision of 2.4.20 or higher. This is provided in both linux86 and linux86-64 versions for AMD64 or Intel 64 CPU-based installations, though linux86-64 versions are limited.
- FlexNet license utilities.
- A UNIX-like shell environment for 32-bit and 64-bit Windows platforms.

Depending on the product configuration you purchased, you may not have licenses to all of the above components.

The release contains the following documentation and tutorial materials:

- Online documentation in PDF, HTML, and man page formats.
- Online HPF tutorials that provide insight into cluster programming considerations.

Note

Compilers and libraries can be installed on other platforms not in the user cluster, including another cluster, as long as all platforms use a common floating license server.

Terms and Definitions

This Installation Guide contains a number of terms and definitions with which you may or may not be familiar. If you encounter an unfamiliar term in these notes, please refer to the online glossary at

www.pgroup.com/support/definitions.htm

These two terms are used throughout the documentation to reflect groups of processors:

- **AMD64** – a 64-bit processor from AMD designed to be binary compatible with 32-bit x86 processors, and incorporating new features such as additional registers and 64-bit addressing support for improved performance and greatly increased memory range. This term includes the AMD Athlon64, AMD Opteron, AMD Turion, AMD Barcelona, AMD Shanghai, AMD Istanbul, and AMD Bulldozer processors.
- **Intel 64** – a 64-bit IA32 processor with Extended Memory 64-bit Technology extensions designed to be binary compatible with AMD64 processors. This includes Intel Pentium 4, Intel Xeon, Intel Core 2, Intel Core 2 Duo (Penryn), Intel Core (i3, i5, i7) both first generation (Nehalem) and second generation (Sandy Bridge) processors.

Supported Processors

[Table 1.1](#) lists the processors on which Release 2011 of the PGI compilers and tools is supported. The table also includes the CPUs available and supported in multi-core versions.

The `-tp <target>` command-line option generates executables that utilize features and optimizations specific to a given CPU and operating system environment. Compilers included in a 64-bit/32-bit PGI

installation can produce executables targeted to any 64-bit or 32-bit target, including cross-targeting for AMD64 and Intel 64-bit compatible CPUs.

Table 1.1. Processors Supported by PGI 2011

Brand	CPU Micro Architecture	Target Processor -tp <target> [,target...]	
		32-bit	64-bit
AMD			
	Opteron Six-core Istanbul	istanbul-32	istanbul-64
	Opteron Quad-core Shanghai	shanghai-32	shanghai-64
	Opteron Quad-core Barcelona	barcelona-32	barcelona-64
	Opteron Quad-core	k8-32	k8-64
	Opteron Rev E7F	k8-32	k8-64e
	Turion / Turion 64	k8-32	k8-64e
	Athlon64	NA	k8-64e
	Athlon	athlon	NA
	Athlon XP/MP	athlonxp	NA
Intel			
	Nehalem	nehalem-32	nehalem-64
	Penryn	penryn-32	penryn-64
	Core	core2-32	core2-64
	P4/Xeon EM64T	p7-32	p7-64
	Xeon Pentium4	p7-32	NA
	Pentium III	piii	NA
	Pentium II	p6	NA
Generic			
	Generic x86	p5 or px-32	NA

In addition to the capability to generate binaries optimized for specific AMD or Intel processors, the PGI 2011 compilers can produce PGI Unified Binary object or executable files containing code streams fully optimized and supported for both AMD and Intel x64 CPUs. To produce unified binary files, you use one of the following -tp command-line options: -tp x64 or -tp <target1>, <target2>, <target3>..., where <target> is any of the valid values in [Table 1.1](#).

Supported Operating Systems

[Table 1.2](#) lists the operating systems, and their equivalents, on which PGI 2011 compilers and tools are supported. To determine if Release 2011 will install and run under a Linux equivalent version, such as Mandrake, Debian, CentOS, and so on, check the table for a supported system with the same `glibc` and `gcc`

versions. Version differences in other operating system components can cause difficulties, but often these can be overcome with minor adjustments to the PGI software installation or operating system environment.

- Linux operating systems with support for x64 compatible processors are designated 64-bit in the table. These are the only distributions on which the 64-bit versions of the PGI compilers and tools will fully install.
- If you attempt to install the 64-bit/32-bit Linux version on a system running a 32-bit Linux distribution, only the 32-bit PGI compilers and tools are installed.

Most modern operating systems, with the notable exception of MacOS, include support for Intel Hyper-threading (HT)

Most modern Linux distributions support the *Native Posix Threads Library (NPTL)*. Distributions that include NPTL are designated in the table. Parallel executables generated using the *OpenMP* and auto-parallelization features of the PGI compilers will automatically make use of NPTL on distributions when it is available. In addition, the *PGDBG* debugger is capable of debugging executables built using either NPTL or earlier `pthread` implementations.

Multi-socket AMD Opteron processor-based servers use a *NUMA* (Non-Uniform Memory Access) architecture in which the memory latency from a given processor to a given portion of memory can vary. Newer Linux distributions, including SuSE 9/10 and SLES 9/10, include NUMA libraries that can be leveraged by a compiler and associated runtime libraries to optimize placement of data in memory.

In the table headings:

HT = hyper-threading

NPTL = Native POSIX Threads Library

NUMA = Non-Uniform Memory Access

Table 1.2. Operating Systems and Features Supported in PGI 2011

Distribution	Type	64-bit	HT	NPTL	NUMA	gliobc	GCC
RHEL 5.6	Linux	Yes	Yes	Yes	Yes	2.12	4.4.4
RHEL 5.5	Linux	Yes	Yes	Yes	No	2.5	4.1.2
RHEL 5.4	Linux	Yes	Yes	Yes	No	2.5	4.1.2
RHEL 5.3	Linux	Yes	Yes	Yes	No	2.5	4.1.2
RHEL 5.0	Linux	Yes	Yes	Yes	No	2.5	4.1.2
RHEL 4.0	Linux	Yes	Yes	Yes	No	2.3.4	3.4.3
RHEL 3.0	Linux	Yes	Yes	Yes	No	2.3.2	3.2.3
Fedora 14	Linux	Yes	Yes	Yes	Yes	2.12	4.4.5
Fedora 13	Linux	Yes	Yes	Yes	Yes	2.12	4.4.4
Fedora 12	Linux	Yes	Yes	Yes	Yes	2.11	4.4.2
Fedora 11	Linux	Yes	Yes	Yes	Yes	2.9	4.3.3
Fedora 10	Linux	Yes	Yes	Yes	Yes	2.9	4.3.2
Fedora 9	Linux	Yes	Yes	Yes	Yes	2.8	4.3.0

Distribution	Type	64-bit	HT	NPTL	NUMA	glibc	GCC
Fedora 8	Linux	Yes	Yes	Yes	Yes	2.7	4.1.2
Fedora 7	Linux	Yes	Yes	Yes	Yes	2.6	4.1.2
Fedora 6	Linux	Yes	Yes	Yes	Yes	2.5	4.1.1
Fedora 5	Linux	Yes	Yes	Yes	Yes	2.4	4.1.0
Fedora 4	Linux	Yes	Yes	Yes	No	2.3.5	4.0.0
Fedora 3	Linux	Yes	Yes	Yes	No	2.3.3	3.4.2
Fedora 2	Linux	Yes	Yes	Yes	No	2.3.3	3.3.3
SuSE 11.3	Linux	Yes	Yes	Yes	Yes	2.11.2	4.5
SuSE 11.2	Linux	Yes	Yes	Yes	Yes	2.10.1	4.4.1
SuSE 11.1	Linux	Yes	Yes	Yes	Yes	2.9	4.3.3
SuSE 11.0	Linux	Yes	Yes	Yes	Yes	2.8	4.3.1
SuSE 10.3	Linux	Yes	Yes	Yes	Yes	2.6.1	4.2.1
SuSE 10.2	Linux	Yes	Yes	Yes	Yes	2.5	4.1.0
SuSE 10.1	Linux	Yes	Yes	Yes	Yes	2.4	4.1.0
SuSE 10.0	Linux	Yes	Yes	Yes	Yes	2.3.5	4.0.2
SuSE 9.3	Linux	Yes	Yes	Yes	Yes	2.3.4	3.3.4
SuSE 9.2	Linux	Yes	Yes	Yes	Yes	2.3.3	3.3.4
SuSE 9.1	Linux	Yes	Yes	Yes	No	2.3.3	3.3.3
SuSE 9.0	Linux	Yes	Yes	No	No	2.3.2	3.3.1
SLES 11	Linux	Yes	Yes	Yes	Yes	2.9	4.3.3
SLES 10	Linux	Yes	Yes	Yes	Yes	2.4	4.1.0
SLES 9	Linux	Yes	Yes	No	Yes	2.3.3	3.3.3
RedHat 9.0	Linux	No	No	Yes	No	2.3.2	3.2.2
Ubuntu 10.10	Linux	Yes	Yes	Yes	Yes	2.12.1	4.4.5
Ubuntu 10.04	Linux	Yes	Yes	Yes	Yes	2.11.1	4.4.3
Ubuntu 9.10	Linux	Yes	Yes	Yes	Yes	2.10.1	4.4.1
Ubuntu 9.04	Linux	Yes	Yes	Yes	Yes	2.9	4.3.3
Ubuntu 8.10	Linux	Yes	Yes	Yes	Yes	2.8	4.3.2
Ubuntu 8.04	Linux	Yes	Yes	Yes	Yes	2.7	4.2.1

Note

www.pgroup.com/support/install.htm lists any new operating system distributions that may be explicitly supported by the PGI compilers. If your operating system is newer than any of those listed in [Table 1.2](#), the installation may still be successful.

Product Support

All new PGI licenses include 60 days of PGI Subscription Service.

The PGI Subscription Service provides support and other benefits, including:

- Ongoing technical support.

Support requests may be sent in a number of ways:

- By electronic mail to trs@pgroup.com
 - Faxed to +1-503-682-2637
 - By using the online support request form available at www.pgroup.com/support/support_request.php
 - Phone support is not currently available.
- Notification by email when maintenance releases occur and are available for electronic download and installation.
 - Release upgrades for licensed Product(s) at no additional cost, except for any administrative fee that may apply.
 - Full license fee credits on Product upgrades, except for any administrative fee that may apply. "Product upgrades" refer to exchanging one Product license for a more expensive Product license, and is not the same as a version or Release upgrade previously referenced.
 - Full license fee credits on user-count upgrades, except for any administrative fee that may apply.

Important

To continue receiving these benefits after 60 days, you can purchase an extension to your PGI Subscription Service. Extensions are available in yearly increments.

Contact sales@pgroup.com if you would like information regarding the subscription service for the PGI products you have purchased.

Chapter 2. PGI CDK 2011

Installation Overview

The following sections contain the information needed for you to successfully install the *PGI CDK* software.

As stated earlier, a cluster is a collection of compatible computers connected by a network. The *PGI CDK* software is installed on a working cluster - it is not the purpose of this product to create a cluster, or to troubleshoot one. The *PGI CDK* release can be installed on a single node, and the node can be treated as if it is a cluster.

Support for cluster programming does not extend to clusters combining AMD64 or Intel 64 CPU-based systems with IA32 CPU-based systems, unless all are running 32-bit applications built for a common set of working x86 instructions.

For multi-process programming of message-passing applications that execute on a cluster, we provide both a 32-bit and a 64-bit set of MPICH and MPICH2 libraries. These libraries include versions that can collect additional information useful in the cluster debugger *PGDBG* and the cluster profiler *PGPROF*, and they implement the MPI inter-process communication standard.

Introduction

You can view the online HTML interface to the *PGI CDK* using any web browser.

Once you have downloaded the *PGI CDK* from www.pgroup.com, you can view the documentation by loading the file `index.htm` from the directory in which you unpack the PGI tar file. After installation, the file `index.htm` is available in the top-level PGI installation directory.

Generally, clusters are configured with two types of nodes:

- A "master" node from which jobs are launched
- "slave" nodes that are used only for computation

Typically, the master node is accessible from the general-purpose or "public" network and shares a file system with the other computers on your network using NFS. The master node and all of the slave nodes are interconnected using a second "private" network that is only accessible from computers that are part of the cluster.

There are two common cluster configurations:

1. The master node is used only for compilation and job submission, and only the slave nodes are used for computation.
2. All nodes are used for computation, including the master node.

One way to use MPICH in the first configuration is to manage job scheduling. By default, the `mpirun` command uses the master node as one of the computation nodes. It is possible to exclude the master node as a computation node in the second configuration if `mpirun` is invoked with the `-nolocal` option. For more information this, refer to the man page for `mpirun`.

If you are using the first configuration, it is possible to install MPICH and run parallel MPI or HPF jobs without installing any of the other components. However, if you have multiple users running jobs on your cluster simultaneously, you should determine how to ensure your cluster nodes are allocated and used efficiently.

Typically, a master node has two network cards to allow communication to the outside network as well as to the cluster nodes themselves, which may be on their own subnet. If this is the case on your cluster, then when the installation script prompts you for the name of the master node, you should use the name associated with the network card connected to the same network as the cluster nodes.

For MPICH to run correctly, access from each node to every other node must be available via the `rsh` or `ssh` command. For example, if a 3-node cluster consists of a master, named *master*, and two slaves named *node1* and *node2*, then from *node1* as a user you should be able to issue the commands:

```
% rsh master date
% rsh node2 date
or
% ssh master date
% ssh node2 date
```

You can issue similar commands from *node2* and *master*.

By default, all of the PGI compilers and tools will be installed on your system. You will select which of the open source components to install.

At this point, before you start the installation, you must determine:

- Which *PGI CDK* open source components - MPICH, MPICH2, and MVAPICH - you will install.
- The hostnames of all the nodes that will be included in your cluster - you will need a list of these during the installation.
- The type of cluster configuration - that is, whether the master node will participate as a compute node or will be strictly a front-end for compilation, job launching, and so on.
- Whether the compute nodes can share files with the master node, which is strongly recommended.

[Chapter 4, “PGI CDK Installations on Linux”](#) describes how to install the PGI Fortran, C and C⁺⁺ compilers and tools on Linux using the *installcdk* script from PGI.

Some, but not all, parts of the *PGI CDK* require root access to successfully execute the *installcdk* script.

For multi-platform cluster installations, you need root access. Without root access you can create a compilers, tools and MPICH library installation on a single platform that allows multi-process/multi-threaded development. [Chapter 4, “PGI CDK Installations on Linux”](#) also includes the steps for configuring and starting the FlexNet license daemon, required to make the PGI software operational.

The FlexNet license daemon enables use of the PGI compilers and tools by any user on any system networked to the system on which the PGI software is installed. For example, users can compile, debug, and profile using the *PGI CDK* compilers and tools on any system on your general-purpose network, subject to the constraints on concurrent usage for the product you have purchased.

Open Source Component Overview

[Chapter 5, “Using the Open Source Cluster Utilities”](#) describes basic usage of the open source components of the *PGI CDK*, including MPICH, ScaLAPACK libraries, and the example benchmark programs and tutorials.

For the first 60 days after your purchase, you may submit technical questions about the *PGI CDK* compilers and tools to the online problem reporting site at www.pgroup.com/support/index.htm. If you have purchased PGI's Subscription Service, you will have access to service for an additional 12 months and will be notified by email when maintenance releases occur. For more information, refer to [“Product Support,” on page 6](#).

MPICH, MPICH2, MVAPICH, and ScaLAPACK are all open source software packages that are not formally supported by The Portland Group. All source code for these components is included in the `cdk` subdirectory. Along with the source code, each of these components has end-user and implementer documentation, generally in the form of printable PostScript. Support for these products is generally provided by their respective user communities, which you can learn more about at the following URLs:

- MPICH - www.mcs.anl.gov/research/projects/mpi/mpich1 contains a wealth of information, including online documentation, tutorials, FAQ files, patch distributions, and information on how to submit bug reports to the MPICH developers.
- MPICH2 - www.mcs.anl.gov/research/projects/mpi/mpich2 contains a wealth of information, including online documentation, tutorials, FAQ files, patch distributions, and information on how to submit bug reports to the MPICH2 developers.
- MVAPICH - MVAPICH 1.1 supports many features for high performance, scalability, portability, and fault tolerance. It also supports a wide range of platforms. mvapich.cse.ohio-state.edu/overview/mvapich contains a wealth of information, including online documentation, tutorials, FAQ files, patch distributions, and how to submit bug reports to the MVAPICH developers.

Note

To use the *PGI CDK* version of MVAPICH, the prerequisite OpenFabrics (OFED) software must be installed. See www.openfabrics.org for details on OFED.

- ScaLAPACK - www.netlib.org/scalapack contains FAQ files and current distributions of ScaLAPACK

The PGI compilers and tools are license-managed, which is described in the next chapter. Further, the [“Installation Steps for Linux,” on page 15](#) provides specific information about how to use your personalized account to generate trial or permanent license keys.

Chapter 3. Licensing

The PGI compilers and tools are license-managed.

Licensing Terminology

Before discussing licensing, it is useful to have common terminology. These two terms are often confused, so they are clarified here:

- **License** - a legal agreement between STMicroelectronics and PGI end-users to which users assent upon installation of any PGI product. The terms of the License are kept up-to-date in documents on pgroup.com and in the `$PGI/<platform>/<rel_number>` directory of every PGI software installation.
- **License keys** - ASCII text strings that enable use of the PGI software and are intended to enforce the terms of the License. License keys are generated by the PGI end-user on pgroup.com using a unique `hostid`. They are typically stored in a file called `license.dat` that is accessible to the systems for which the PGI software is licensed at a given site.

There are two types of license keys: permanent and trial.

Permanent and Trial License Keys

PGI CDK includes the PGI License Setup tool to help automate your license retrieval and installation process. Use this tool to obtain your license keys, either trial or permanent.

Note

You must install the PGI software before you obtain your license keys because the license key generation process requires information that is generated during the software installation.

- **Permanent License Keys** - When you purchase a *permanent* PGI license, the email order confirmation you receive includes complete instructions for logging on to the pgroup.com web page and generating permanent license keys.
- **Trial License Keys** - When you register for a *trial* license, you generate trial keys using the web page: www.pgroup.com/login.php.

Note

At the conclusion of the 15-day trial period, the PGI compilers and tools and any executable files generated prior to the installation of permanent license keys will cease to function.

Any executables, object files, or libraries created using the PGI compilers with trial license keys must be recompiled once permanent license keys in place.

For more detailed information on how to do obtain license keys, refer to Step 5 in [“Installation Steps for Linux,”](#) on page 15.

License Keys and System Configurations

Executable files generated with permanent license keys in place are unconstrained, and run on any compatible system regardless of whether the PGI compilers are installed.

Important

If you change the configuration of your system by adding or removing hardware, your license keys may become invalid. Please contact license@pgroup.com if you expect to reconfigure your system to ensure that you do not temporarily lose the use of your PGI compilers and tools.

CDK Licensing

PGI CDK supports multi-user, network floating licenses. Multiple users can use the PGI compilers and tools concurrently from multiple systems on a network when those systems have a properly configured version of *PGI CDK* installed. The number of seats purchased for the license determines the limitation on the number of concurrent users.

Note

We recommend the license services run on the cluster master node.

The FlexNet License Manager

PGI CDK software licensing uses the FlexNet Publisher (FNP) license management system from Flexera Software. As part of the process of installing the PGI compilers and tools, you install and configure the FlexNet license management software. The instructions in the following chapters of this guide describe how to configure license daemons for Linux, including installation and start-up of the license services, and proper initialization of the `LM_LICENSE_FILE` and, for Windows, `FLEXLM_BATCH` environment variables.

License Support

All new PGI licenses include 60 days of PGI Subscription Service. For more information about this service and how to extend it, refer to [“Product Support,”](#) on page 6.

Chapter 4. PGI CDK Installations on Linux

This chapter describes how to install *PGI CDK* in a generic manner on a Linux system. It is applicable to permanent or trial installations.

- For installations on 32-bit x86 systems, the PGI installation script installs only the linux86 versions of the PGI compilers and tools.
- For installations on 64-bit x64 systems running a linux86-64 execution and development environment, the PGI installation script installs the linux86-64 version of the PGI compilers and tools.
- If the 32-bit gcc development package is already installed on the system, the 32-bit linux86 tools are also installed on a 64-bit x64 system.

The 32-bit and 64-bit compilers, tools, and supporting components have the same command names, and the environment you target by default (linux86-64 or linux86) depends on the version of the compiler that comes first in your path settings.

Preparing to Install on Linux

To prepare for the installation:

1. Download the software.

Unlike other PGI software, the CDK isn't generally available. To download the software, first log in to your PGI account at www.pgroup.com/login. Once you are logged in, click the Download Software link. If you don't see the *PGI CDK* listed as a product option on the downloads page, contact PGI at sales@pgroup.com.

2. Bring up a shell command window on your system. The installation instructions assume you are using csh, sh, ksh, bash, or some compatible shell. If you are using a shell that is not compatible with one of these shells, appropriate modifications are necessary when setting environment variables.
3. Log in as root on the master node to install and verify your cluster.

Note

If you fail to do this, software is installed only on the master node.

4. Verify you have enough free disk space.

The uncompressed installation package requires 700 MB of free disk space.

The linux86 platform requires up to 450 MB of free disk space, depending on the number of packages installed.

The linux86-64 platform requires up to 1.4 GB of free disk space, depending on the number of packages installed.

5. Install a version of Python. (Optional)

If you want to debug 32-bit MPICH-2 programs on a 64-bit system, a 32-bit Python release must be installed on the 64-bit system. The CDK includes a pre-compiled 32-bit Python in `$PGI/linux86/11.1/mpi2/mpich/Python32` (where `$PGI` is the PGI installation directory), but it is recommended that you install a version of Python built specifically for your system.

6. Install MVAICH. (Optional)

If you want to install MVAICH, verify that you already have OpenFabrics (OFED) software installed on your system. MVAICH is an InfiniBand library. See section 2.2 for more information.

Before you begin the actual installation, review the CDK Linux installation process illustrated in [Figure 4.1](#). Also, be certain that you know this information:

- Locate your PGI order confirmation email.

This email contains instructions on how to download the PGI software from the PGI web page as well as other information you may need for generating your permanent license keys. It also contains your PGI Product Identification Number (PIN) that you need if you contact PGI.

- Know how your computer accesses the Internet - directly or through some sort of proxy connection.

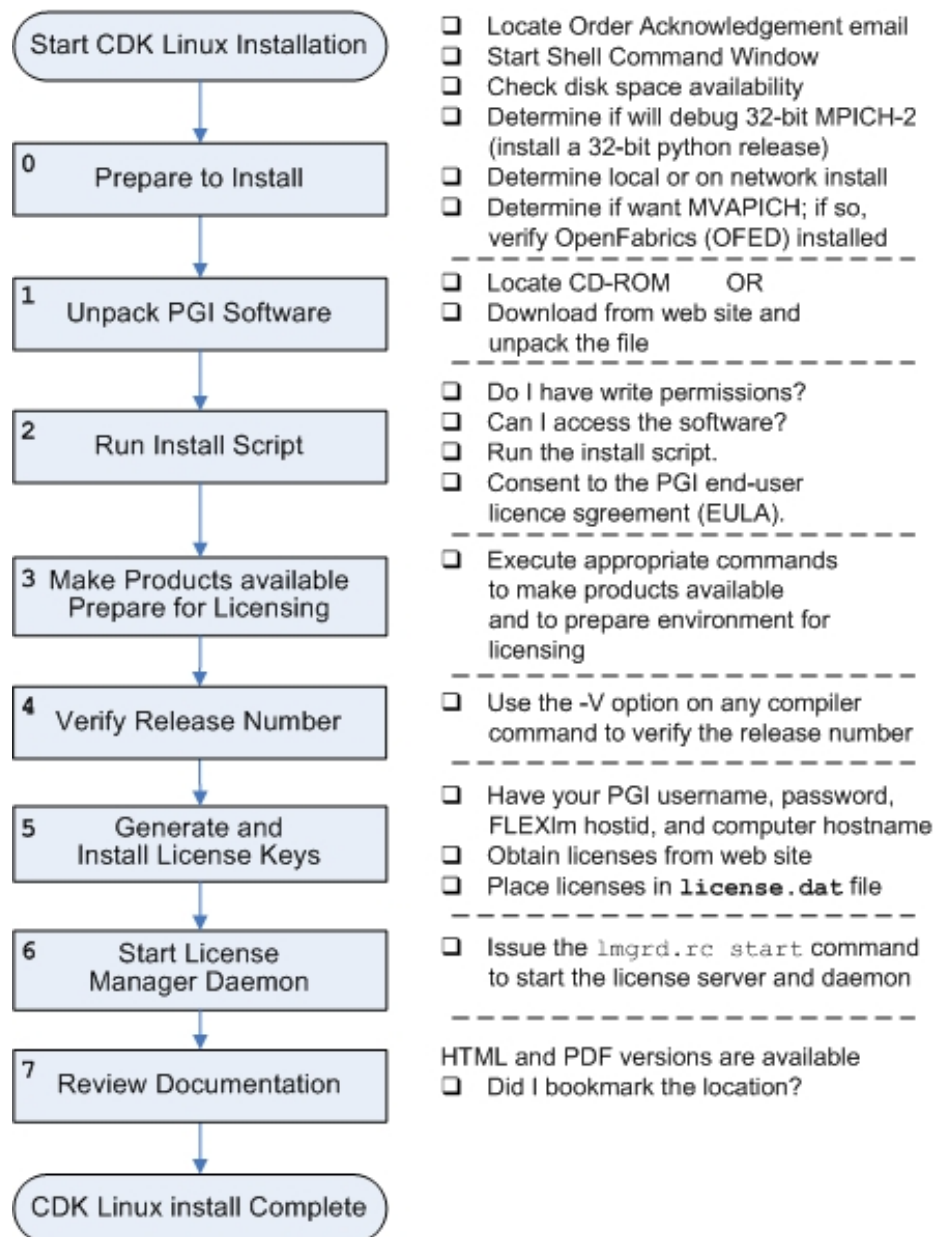
Note

If this computer is behind a firewall at your site, please make sure it can access the Internet.

If a proxy is used, you need this additional information:

- The address (URL) of the proxy server.
- Whether the proxy requires authentication - and if so, what is the required username and password.
- To login to the PGI website to generate license keys, you need either a PGI web account (username and password) or a PIN code from the PGI order confirmation email message from PGI Sales.

Figure 4.1. CDK Linux Installation Overview



Installation Steps for Linux

Follow these instructions to install the software:

1. Unpack the PGI software.

Download the software from www.pgroup.com or another electronic distribution site. In the instructions that follow, replace <tarfile> with the name of the file that you downloaded.

Note

The PGI products cannot be installed into the same directory where the tar file is unpacked.

Use the following command sequence to unpack the tar file in a temporary directory before installation:

```
% mkdir /tmp/pgi
% mv <tarfile>.tar.gz /tmp/pgi
% cd /tmp/pgi
% tar xpfz <tarfile>.tar.gz
```

2. Run the installation script.

The *installcdk* script *must* run to completion to properly install the software.

- If you are not logged in as root, the *PGI CDK* compilers and tools and *MPICH*, *MPICH2*, and *MVAPICH* libraries can be installed on your current machine; and be configured for that machine only. This will create a single node cluster and/or a cross-development installation.
- If you are logged in as root, the entire cluster can be configured.

Note

If you are updating a previous release, or wish to reinstall, we recommend that you run the *uninstallcdk* script before running *installcdk*.

Execute the following script in the directory where you unpacked the tar file:

```
% ./installcdk
```

To successfully run this script to completion, do the following:

- Consent to the PGI end-user license agreement (EULA).
- Determine whether to install the optional ACML math library components.
- Determine whether to install the optional NVIDIA components.
- Determine whether to install the optional *MPICH* components.
- Define where to place the installation directory.
- Determine whether to use the builtin utility to generate license keys.

After the software is installed, the *installcdk* script performs system-specific customization and then initializes the licensing.

The *installcdk* script installs all of the binaries for the PGI compilers and tools, *MPICH*, *MPICH2*, *MVAPICH* and *ScaLAPACK* in the \$PGI directory tree in the appropriate *bin*, *include*, *lib*, and *man* subdirectories. You are prompted for various information about how to configure your cluster as the script executes. Once the installation script has completed, exit the root shell.

3. Make PGI products accessible.

When the *installcdk* script has completed, execute the following commands to make the PGI products accessible and to initialize your environment for use by FlexNet.

Note

Each user must issue the follow sequence of commands to initialize the shell environment prior to using the PGI compilers and tools.

For linux86-64:

To use the linux86-64 version of the compilers and tools, execute the following commands, assuming you have installed in the default `/opt/pgi` directory.

In csh, use these commands:

```
% setenv PGI /opt/pgi
% set path=(/opt/pgi/linux86-64/11.1/bin $path)
% setenv MANPATH "$MANPATH":/opt/pgi/linux86-64/11.1/man
% setenv LM_LICENSE_FILE "$LM_LICENSE_FILE":/opt/pgi/license.dat
```

In bash, sh, or ksh, use these commands:

```
$ PGI=/opt/pgi; export PGI
$ PATH=/opt/pgi/linux86-64/11.1/bin:$PATH; export PATH
$ MANPATH=$MANPATH:/opt/pgi/linux86-64/11.1/man; export MANPATH
$ LM_LICENSE_FILE=$LM_LICENSE_FILE:/opt/pgi/license.dat;export LM_LICENSE_FILE
```

For linux86:

To use only the linux86 version of the compilers and tools, or to target linux86 as the default, use a setup similar to the previous one, changing the path settings as illustrated in the following commands.

In csh, use these commands:

```
% setenv PGI /opt/pgi
% set path=(/opt/pgi/linux86/11.1/bin $path)
% setenv MANPATH "$MANPATH":/opt/pgi/linux86/11.1/man
% setenv LM_LICENSE_FILE "$LM_LICENSE_FILE":/opt/pgi/license.dat
```

In bash, sh, or ksh, use these commands:

```
$ PATH=/opt/pgi/linux86/11.1/bin:$PATH; export PATH
$ MANPATH=$MANPATH:/opt/pgi/linux86/11.1/man; export MANPATH
$ LM_LICENSE_FILE=$LM_LICENSE_FILE:/opt/pgi/license.dat;export LM_LICENSE_FILE
```

Note

You should add these commands to your shell startup files to ensure that you have access to the PGI products in future login sessions.

4. Verify the release number of the installed software.

To verify the release number of the products you have installed, use the `-v` option on any of the compiler commands, as illustrated in the following examples. If you use `-v` instead, you can also see the sequence of steps the compiler uses to compile and link programs for execution on your system.

For Fortran 77, use: **pgf77 -V x.f**
For Fortran 2003, use: **pgfortran -V x.f**
For HPE, use: **pghpf -V x.f**
For C++, use: **pgCC -V x.c** or **pgcpp -V x.c**
For ANSI C, use: **pgcc -V x.c**

Note

These commands can be successfully executed even if the files `x.f` or `x.c` do not exist and you have not completed the licensing phase of the installation. Use it to check that you have installed the proper version of the compilers and have initialized your environment to enable access to that version.

5. Generate and install license keys.

Note

This step is necessary only if you chose not to allow the installation script to perform these tasks for you.

All of the PGI compilers and tools are license-managed and require installation of license keys to make the PGI software operational. The other components of the *PGI CDK*, including *MPICH*, *MPICH2*, *MVAPICH*, and *ScaLAPACK* are open source products that are not license-managed.

To obtain license keys, you need the following information:

- An account on the PGI website.
 - If you purchased a license without creating an account, one was created for you when your order was processed. Please check for an activation email from `accounts-noreply@pgroup.com`.
 - If you don't have an account, you can create one at: `www.pgroup.com/register`. Without an account you have no access to the CDK software.
- The FlexNet `hostid` and `hostname` of the computer on which the software is installed, which is echoed to your screen by the installer.

Note

You can also obtain your FlexNet `hostid` by using the following command after you have installed the products and initialized the environment variables:

```
% lutil lmhostid
```

You should see a message similar to the following with one or more `hostids` displayed.

```
The FlexNet host ID of this machine is "12345678abcd edcba9876543".
```

You can use either `12345678abcd` or `edcba9876543`, but not both, as the `hostid`.

Tip

Hostids come from configured network cards. If you use your computer in multiple environments, you may want to run the following command in each environment to see what hostids are configured. Then, to reduce potential license problems, choose the hostid that occurs in *all* your environments.

Generate License Keys

Log on to your PGI web account at www.pgroup.com/login. You should see a screen similar to the following:

Welcome

Use the links below to manage your PGI account.

[Download software](#) – For updating or evaluation.

[Manage PGI products](#) – Purchase, subscription, and ownership information

[Create permanent keys](#) – Create permanent software license keys

[Create trial keys](#) – For a two-week evaluation of the PGI product of your choice.

[Display a PIN code](#) – Use your old (pre-2008) PIN-based username and password to display the new PIN code.

[Tie a PIN to this account](#) – Use a PIN code to tie a PIN to your account and create permanent license keys.

[Update account](#) – Update contact information, change password, or modify email preferences.

[FAQ](#) – Answers to common questions.

To generate permanent license keys:

1. Click Create permanent keys.
2. Click the PIN associated with the product for which you wish to generate license keys. If you don't see any PINs listed, you first need to tie one to your account.
 - a. Obtain your PIN code from your original PGI order confirmation email.
 - b. Click the link: Tie a PIN to this account.
 - c. Follow the instructions provided.
3. Click License keys to generate the keys.

To generate trial license keys:

1. Click Create trial keys.
2. Accept the terms of the agreement.
3. Enter the hostid exactly as it appears in the message that is displayed during installation or when you issue the command:

```
% lmutil lmhostid
```

If multiple host ids are displayed, select any *one* of them to use as the hostid.

4. Click the *Generate license key* button.

Install License Keys

Once you have generated your trial or permanent license keys, copy, and then paste them into the file: `/opt/pgi/license.dat`, substituting the appropriate installation directory path if you have not installed in the default `/opt/pgi` directory.

For example, if you have purchased PGI CDK, the `license.dat` file should look similar to the following:

```
SERVER server 123456789012 27000
DAEMON pgroupd
PACKAGE PGI2011-cdk-linux pgroupd 2011.1231 5D2F7BAEC27D \
  COMPONENTS="pgi-hpf-lin64 pgi-f95-lin64 pgi-f77-lin64 \
  pgi-cc-lin64 pgi-cpp-lin64 pgi-hpf-lin32 pgi-f95-lin32 \
  pgi-f77-lin32 pgi-cc-lin32 pgi-cpp-lin32 pgi-f95-win64 \
  pgi-f77-win64 pgi-cc-win64 pgi-cpp-win64 pgi-f95-win32 \
  pgi-f77-win32 pgi-cc-win32 pgi-cpp-win32 pgi-f95-osx64 \
  pgi-f77-osx64 pgi-cc-osx64 pgi-cpp-osx64 pgi-f95-osx32 \
  pgi-f77-osx32 pgi-cc-osx32 pgi-cpp-osx32 pgi-prof pgi-dbg \
  pgi-dbg-gui pgi-pvf " OPTIONS=SUITE_RESERVED SIGN="18F4 F66B 0F20 \
  86F2 7494 0A21 5FCF 80E1 51EC 2DA1 2692 6A25 C913 C51A B0BD \
  18B5 0C0E DAC3 665F 0EE4 6501 481B 342C 0564 DE6B FDF3 C4E8 \
  0117 142B 279C"
FEATURE PGI2011-cdk-linux pgroupd 2011.1231 permanent 2 6461BAE004A8 \
  VENDOR_STRING=987654:2:cdk:accel DUP_GROUP=U \
  SUITE_DUP_GROUP=U BORROW=336 SIGN="1E3D 8B4F 8A97 5193 C039 \
  6A4D 7A2C 726D 68CC 7CE7 FAC6 53AB 5A6C B72F 742E 1D0F 5867 \
  DF88 BBC7 FD45 8852 819D 5CCB E498 0621 7F1A 1561 346A F417 \
  E6A3"
PACKAGE PGI71-cdk-linux pgroupd 7.1 E103E0016D48 \
  COMPONENTS="pghpf-linux86-64:7.1 pgf90-linux86-64:7.1 \
  pgf77-linux86-64:7.1 pgcc-linux86-64:7.1 pgcpp-linux86-64:7.1 \
  pghpf-linux86:7.1 pgf90-linux86:7.1 pgf77-linux86:7.1 \
  pgcc-linux86:7.1 pgcpp-linux86:7.1 pghpf-linux86:7.1 \
  pgf90-linux86:7.1 pgf77-linux86:7.1 pgcc-linux86:7.1 \
  pgcpp-linux86:7.1 pgprof:7.1 pgdbg:7.1 pgdbg-gui:7.1 \
  pgdbg-linux86-64:7.1" OPTIONS=SUITE_RESERVED SIGN="19B4 78BC \
  4323 9E65 730F 74DD D99A 07B7 4A61 98E0 89C1 2823 A55B C4C4 \
  B0A6 0FD5 EF8D 2A95 6C4E D8C2 EE23 6CD5 D6EC 08B2 535E 99B8 \
  2ED0 C9C4 7BDF 9E75"
FEATURE PGI71-cdk-linux pgroupd 7.1 permanent 2 E05A155E64C8 \
  VENDOR_STRING=987654:2:cdk:DUP_GROUP=U SUITE_DUP_GROUP=U \
  SIGN="1489 142D D7E6 719A FE4F 3669 15A9 F23A 37A1 E57E 7712 \
  1565 CAE3 1C09 287A 1392 1790 574F 1AE6 EDCE 3EFB 6366 A815 \
  942B 5418 DB07 F659 640A 8668 9744"
```

In your license file:

- `<hostid>` should match the `hostid` you submitted above when you generated your license keys.
- If necessary, you can enter or edit the `<hostname>` entry manually, but you cannot edit the `<hostid>` entry or you will invalidate the license keys.
- The date in the file, in this example 2011.1231, represents the expiration date for your subscription service.

For example, if your subscription date for your PGI PIN (Product Identification Number) is August 1, 2011, then the date in your file is 2011.0801. For information on how to renew your license, refer to [“Product Support,” on page 6](#)

- The six digits immediately following the = in the feature line component, 987654 of `VENDOR_STRING=987654:2:cdk` in this example, represent the PIN for this installation.

You have a similar unique PIN for your installation.

Note

Please include your PIN when contacting PGI for technical support for the products you have purchased. This PIN is also in your order confirmation email.

6. Start the license manager daemon.

Important

If you used the installation script to do this or if you are evaluating PGI software with trial license keys, you do not need to perform this step and can proceed to Step 7.

Installations in a directory other than the default `/opt/pgi`

Note

The following refers to the shell script template for linux86-64. If you have installed only linux86, please substitute linux86 for linux86-64.

If you installed the compilers in a directory other than `/opt/pgi`, do this:

1. Edit the shell script template `$PGI/linux86-64/11.1/bin/lmgrd.rc`.
2. Substitute the correct installation directory for `/opt/pgi` in the section of the script entitled *Where to find the PGI Software*.
3. Save the file and exit the editor.

Issue the following command to start the license server and pgroup license daemon running on your system:

```
% cd $PGI/linux86-64/11.1/bin/
% ./lmgrd.rc start
```

If you wish to stop the license server and pgroup license daemon at a later time, you can do so with the command:

```
% cd $PGI/linux86-64/11.1/bin/
% ./lmgrd.rc stop
```

Start license server upon reboot:

To start the license server and pgroupd license daemon each time your system is booted:

1. Log in as root.

Note

You **must** be logged in as root to successfully execute these commands.

2. Verify you have set the PGI environment variable as described in Step 3 of this installation process.
3. Execute the following two commands:

```
% cp $PGI/linux86/11.1/bin/lmgrd.rc /etc/init.d/lmgrd
% ln -s /etc/init.d/lmgrd /etc/rc.d/rc3.d/S90lmgrd
```

There are two values in this example that may be different on your system:

- Your rc files may be in a directory other than the one in the example: `/etc/init.d`. If the rc files are in a directory such as `/etc/rc.d/init.d`, then substitute that location in the example.
- Your system's default `runlevel` may be something other than '3', the level used in this example. You can run `/sbin/runlevel` to check the system's runlevel. If the runlevel on your systems is different, then you must set the correct subdirectory; use your system's runlevel in place of the "3" in the preceding example.

chkconfig(8) Utility

Most Linux distributions include the `chkconfig(8)` utility which manages the `runlevel` scripts. If your system has this tool and you wish to use it, then run the following commands:

```
% cp $PGI/linux86/11.1/bin/lmgrd.rc /etc/init.d/
% /sbin/chkconfig --add lmgrd
```

These commands create the appropriate links in the `/etc/init.d` directory hierarchy. For more information on `chkconfig`, please refer to the manual page.

Important

You can co-install Release 2011 with Release 2010, 9.x, 8.x, 7.x, 6.x and/or 5.2; and you can use any of these versions of the compilers and tools with the latest versions of `lmgrd` and `pgroupd` and a single Release 2011 license file.

If you use the `lmgrd.rc` file to start `lmgrd` automatically after a reboot of your system, you need to modify your `lmgrd` script in the `/etc/rc.d` or `/etc/init.d` directory to use the latest `lmgrd` daemon.

For example, your `lmgrd` script may look like this, where `<target>` is replaced appropriately with `linux86` or `linux86-64`.

```

## Path to master daemon lmgrd
# Commented out previous path to 5.2:
#LMGRD=$PGI/<target>/5.2/bin/lmgrd
LMGRD=$PGI/<target>/11.1/bin/lmgrd

## Command to stop lmgrd
#Commented out previous path to 5.2:
#LMUTIL=$PGI/<target>/5.2/bin/lmutil
LMUTIL=$PGI/<target>/11.1/bin/lmutil

```

7. Review documentation.

You can view the online HTML and PDF documentation using any web browser by opening the file:

```
$PGI/linux86-64/11.1/doc/index.htm
```

or

```
$PGI/linux86/11.1/doc/index.htm
```

You may want to bookmark this location for easy future reference to the online manuals.

Note

The **makelocalrc** command does allow the flexibility of having local directories with different names on different machines. However, using the same directory on different machines allows users to easily move executables that use PGI-supplied shared libraries between systems.

Installation of the PGI products for Linux is now complete. For assistance with difficulties related to the installation, send email to trs@pgroup.com.

The following two sections contain information detailing the directory structure of the PGI installation, and instructions for PGI end-users to initialize environment and path settings to use the PGI compilers and tools.

Typical Directory Structure for Linux

If you specify `/opt/pgi` as the base directory for installation, the following directory structure is created by the PGI installation script:

Table 4.1. Linux Directory Structure Sample

This directory...	Contains...
<code>/opt/pgi/linux86/11.1/bin</code>	linux86 32-bit compilers & tools
<code>/opt/pgi/linux86/11.1/liblf</code>	linux86 32-bit large-file support libs (used by <code>-Mlfs</code>)
<code>/opt/pgi/linux86/11.1/include</code>	linux86 32-bit header files
<code>/opt/pgi/linux86-64/11.1/bin</code>	linux86-64 compilers & tools
<code>/opt/pgi/linux86-64/11.1/lib</code>	linux86-64 <code>-mmodel=small</code> libs
<code>/opt/pgi/linux86-64/11.1/libso</code>	linux86-64 <code>-fpic</code> shared libraries for <code>-mmodel=medium</code> development

This directory...	Contains...
/opt/pgi/linux86-64/11.1/include	linux86-64 header files
/opt/pgi/linux86/11.1/REDIST /opt/pgi/linux86-64/11.1/REDIST	Re-distributable runtime libraries
/opt/pgi/linux86/11.1/EXAMPLES /opt/pgi/linux86-64/11.1/EXAMPLES	Compiler examples
/opt/pgi/linux86/11.1/doc /opt/pgi/linux86-64/11.1/doc	Documentation
/opt/pgi/linux86/11.1/man /opt/pgi/linux86-64/11.1/man	UNIX-style man pages
/opt/pgi/linux86/11.1/jre /opt/pgi/linux86-64/11.1/jre	JAVA environment for <i>PGDBG</i> and <i>PGPROF</i> graphical user interfaces
/opt/pgi/linux86/11.1/src /opt/pgi/linux86-64/11.1/src	<i>PGHPF</i> MPI interface file, <code>mpi.c</code>
/opt/pgi/linux86/11.1/mpi/mpich /opt/pgi/linux86-64/11.1/mpi/mpich	MPICH1 scripts and libraries.
/opt/pgi/linux86/11.1/mpi2/mpich /opt/pgi/linux86-64/11.1/mpi2/mpich	MPICH2 scripts and libraries.
/opt/pgi/linux86/11.1/mpi/mvapich /opt/pgi/linux86-64/11.1/mpi/mvapich	MVAPICH scripts and libraries.
/opt/pgi/linux86/11.1/mpi/mpich/share /opt/pgi/linux86-64/11.1/mpi/mpich/share	MPICH machines .LINUX files
/opt/pgi/linux86/11.1/EXAMPLES/mpi /opt/pgi/linux86-64/11.1/EXAMPLES/mpi	MPI Examples
/opt/pgi/linux86/11.1/mpi/mpich/sbin /opt/pgi/linux86-64/11.1/mpi/mpich/sbin	MPICH <i>tstmachines</i>

PGI CDK End-user Environment Settings

With either the trial or permanent license keys in place, choose the applicable commands from the following lists and execute them to make the products you have purchased accessible.

Note

Each user must issue the appropriate following sequence of commands to initialize their shell environment before using the PGI compilers and tools.

Important

For the following path settings, the installation is in the default directory: `/opt/pgi`

Set Environment Variables for Licensing

Execute the following commands to initialize your environment for use of FlexNet licensing.

In `csh`, use these commands:

```
% setenv PGI /opt/pgi
% setenv LM_LICENSE_FILE "$LM_LICENSE_FILE":/opt/pgi/license.dat
```

In `bash`, `sh` or `ksh`, use these commands:

```
$ PGI=/opt/pgi; export PGI
$ LM_LICENSE_FILE=$LM_LICENSE_FILE:$PGI/license.dat; export LM_LICENSE_FILE
```

Set End-User PATH Variables

This section describes the commands required to access *MPICH*, *MPICH2*, and *MVAPICH* as well as the man pages associated with each of these.

Note

If you install only the `linux86` versions of the compilers or wish to target `linux86` as the default, use a setup similar to the `x64 linux86-64` one shown here, substituting *linux86* in place of *linux86-64* in all path settings.

MPICH access

To access *MPICH* and the *MPICH* man pages for `x64 linux86-64`, execute these commands:

In `csh`, use these commands:

```
% set path = (/opt/pgi/linux86-64/11.1/bin \
             /opt/pgi/linux86-64/11.1/mpi/mpich/bin $path)
% setenv MANPATH "$MANPATH":/opt/pgi/linux86-64/11.1/mpi/mpich/man
```

In `bash`, `sh` or `ksh`, use these commands:

```
$ PATH=/opt/pgi/linux86-64/11.1/bin:/opt/pgi/linux86-64/11.1/mpi/mpich/bin:$PATH
$ MANPATH=$MANPATH:/opt/pgi/linux86-64/11.1/mpi/mpich/man
$ export PATH MANPATH
```

MPICH2 access

To access *MPICH2* and the *MPICH2* man pages for `x64 linux86-64`, execute these commands:

In `csh`, use these commands:

```
% set path = (/opt/pgi/linux86-64/11.1/bin \
             /opt/pgi/linux86-64/11.1/mpi2/mpich/bin $path)
% setenv MANPATH "$MANPATH":/opt/pgi/linux86-64/11.1/mpi2/mpich/man
```

In `bash`, `sh` or `ksh`, use these commands:

```
$ PATH=/opt/pgi/linux86-64/11.1/bin:/opt/pgi/linux86-64/11.1/mpi2/mpich/bin:$PATH
$ MANPATH=$MANPATH:/opt/pgi/linux86-64/11.1/mpi2/mpich/man
$ export PATH MANPATH
```

MVAPICH access

To access MVAPICH and the MVAPICH man pages for x64 linux86-64, execute these commands:

In csh, use these commands:

```
% set path = (/opt/pgi/linux86-64/11.1/bin \
             /opt/pgi/linux86-64/11.1/mpi/mpich/bin $path)
% setenv MANPATH "$MANPATH":/opt/pgi/linux86-64/11.1/mpi/mpich/man
```

In bash, sh or ksh, use these commands:

```
$ PATH=/opt/pgi/linux86-64/11.1/bin:/opt/pgi/linux86-64/11.1/mpi/mvapich/bin:$PATH
$ MANPATH=$MANPATH:/opt/pgi/linux86-64/11.1/mpi/mvapich/man
$ export PATH MANPATH
```

Common Linux Installation Issues

If you are having problems with the installation, you might want to check out the [Java Runtime Environment](#).

Java Runtime Environment (JRE)

Although the PGI installation on Linux includes a 32-bit version of the Java Runtime Environment (JRE), sufficient 32-bit X Window System support must be available on the system for the JRE and the PGI software that depends on it to function properly. On some systems, notably recent releases of Fedora Core, these libraries are not part of the standard installation. The required X Windows support generally includes these libraries:

```
libXau
libXdmcp
libxcb
libX11
liXext
```

Chapter 5. Using the Open Source Cluster Utilities

This chapter includes detailed information on using each of the open source components of the *PGI CDK*.

Copy the directory `$PGI/bench` to a local working area so you can try an example program. This directory is only created in `$PGI` when the supplementary materials are selected during installation.

Running an MPICH, MPICH2, or MVAPICH Program

You must either work in a directory which is shared with all of the cluster nodes, or you must copy your MPI executables to a common directory on all compute nodes before invocation of `mpirun`. In particular, this precludes you from working in `/tmp` unless you copy the executable to `/tmp` on each slave node prior to invocation of `mpirun`.

Once you have installed the CDK on the cluster, follow the following process to run an MPICH, MPICH2, or MVAPICH program.

The following example uses these assumptions:

- You are using a generic `machines.cluster` file with a node list.
- You are using `rsh`.

Create "Hello World"

Prepare a "hello world" program. You might try the following "hello world" program, which is a modified version of a typical `mpihello` program because it adds the hostname to the process number of the "Hello world!" statement.

```
% more mpihello.f
  program hello
  include 'mpif.h'
  integer ierr,myproc,hostnm
  character*64 hostname
  call mpi_init(ierr)
  call mpi_comm_rank(MPI_COMM_WORLD, myproc, ierr)
```

```

ierr=setvbuf3f(6,2,0)
ierr=hostnm(hostname)
write(6,100) myproc,hostname
100  format(1x,"hello - I am process",i3," host ",A64)
call mpi_finalize(ierr)
end

```

Prepare to Run a Program

Follow these instructions to prepare to run a program:

1. Set up the environments.

For this example, be certain that you have set the `PGI` and `PGRSH` or `PGSSH` environment variables.

- The `PGI` environment variable specifies, at compile-time, the root directory where the PGI compilers and tools are installed.
- The `PGRSH` or `PGSSH` environment variables should be set to `rsh` or `ssh`, to indicate the desired communication method.

You should also verify that your `PATH` environment variable includes the location of the MPI scripts and libraries, described in [Table 4.1, “Linux Directory Structure Sample,” on page 23](#).

Note

Be certain to use the correct location, based on whether you are using 32- or 64-bit MPICH, MPICH2, or MVAPICH.

2. Compile - Build the executables.

To compile for...	Use this command...
MPICH	<code>% pgf90 -o mpihello_mpich -g -Mmpi=mpich1 mpihello.f</code>
MPICH2	<code>% pgf90 -o mpihello_mpich2 -g -Mmpi=mpich2 mpihello.f</code>
MVAPICH	<code>% pgf90 -o mpihello_mvapich -g -Mmpi=mvapich1 mpihello.f</code>

3. Start any daemons that may be necessary. For example, you may need to use the following command:

```
% mpdboot -r rsh -n 4 -f aFileName
```

Note

If you do start a daemon, be certain after you are done with `mpich2` to execute the following command to turn off the daemons:

```
% mpdallexit
```

You are now ready to execute a program.

Execute a Program Normally

First, make certain that you have added the correct mpi bin directory to your path, as described in the chapter on CDK Installation on page 15. Then, for normal execution of an MPI program, use one of the following commands:

To execute for...	Use this command...
MPICH	% mpirun -np 12 mpihello_mpich
MPICH2	% mpiexec -np 12 mpihello_mpich2
MVAPICH	% mpirun -machinefile aFileName -np 12 mpihello_mvapich

Compile and Execute a Program Summary

You can now put all the information you have together to compile and execute a program.

Important

The following examples show the recommended approach to using MPICH, MPICH2, and MVAPICH. If you choose to use mpi scripts to build, please refer to [“Using mpi Scripts,” on page 33](#).

If you installed MPICH:

Compile the test program with `-Mmpi=mpich1`.

```
% pgf90 -o mpihello_mpich -g -Mmpi=mpich1 mpihello.f
% mpirun -np 4 mpihello
```

The output looks similar to the following, where masternode is your master node, and the “slave nodes” are node01, node02, node03, and so on.

```
hello - I am process 0 host masternode
hello - I am process 1 host node01
hello - I am process 2 host node02
hello - I am process 3 host node03
```

If you installed MPICH2:

If you installed MPICH2 and want to use it instead of MPICH, you must set the PATH environment variable to the MPICH2 bin directory and start the MPD daemon.

Compile the test program with `-Mmpi=mpich2`.

```
% pgf90 -o mpihello_mpich2 -g -Mmpi=mpich2 mpihello.f
% mpdboot
% mpiexec -np 12 mpihello_mpich2
```

Though the commands are different, the output looks the same as the output displayed for MPICH.

If you installed MVAPICH:

If you installed MVAPICH and want to use it instead of MPICH or MPICH2, you must set the PATH environment variable to the MVAPICH bin directory.

Compile the test program with `-Mmpi=mvapich1`:

```
% pgf90 -o mpihello_mvapich -g -Mmpi=mvapich1 mpihello.f
% mpirun -machinefile aFileName -np 12 mpihello_mvapich
```

Though the commands are different, the output looks the same as the output displayed for the MPICH program.

Invoking PGDBG for MPI Debugging

The command to start MPI debugging under MPICH using the *PGDBG* GUI is this:

```
% mpirun -np nprocs -dbg=pgdbg executable [ arg1,...argn ]
```

For example, to run the debugger on an MPICH program `mpihello_mpich`, do this:

```
% mpirun -dbg=pgdbg -np 4 mpihello_mpich
```

To invoke *PGDBG* for debugging with non-MPICH versions of MPI, such as MPICH2 or MVAPICH, use one of these commands:

```
% pgdbg -mpi[:<path>] <mpiexec_args> [ -program_args arg1,...argn ]
```

or

```
% mpiexec -np nprocs -pgi executable [ arg1,...arg ]
```

Note

`mpiexec` must be in your `PATH`, or alternately, the pathname for `mpiexec`, or another similar launcher, should be specified as `<path>` in `-mpi[:<path>]`.

For example, to run the debugger on an MPICH2 program `mpihello_mpich2`, use this command:

```
% pgdbg -mpi:mpiexec -np 12 mpihello_mpich2
```

For MPICH2, as with any other MPICH2 application, the `mpdboot` command must have been run.

To run the debugger on an MVAPICH program `mpihello_mvapich`, use this command:

```
% pgdbg -mpi:mpirun -machinefile aFileName -np 12 mpihello_mvapich
```

The command to start MPI debugging via *mpirun* or *mpiexec* using *PGDBG* in TEXT mode is the same, except that the `DISPLAY` environment variable must be undefined in the shell that is invoking *mpirun*:

For `sh/bash` users:

```
$ unset DISPLAY
```

For `csh/tcsh` users:

```
% unsetenv DISPLAY
```

When an MPI debug session begins, *PGDBG* will stop the program at the first executable statement in the program. Execution does not need to be started using the `run` command as it does with serial or multi-threaded programs. Execution is started using one of the other control commands, such as `cont`, `next`, or `step`.

You cannot restart an MPI application from within *PGDBG*. You must exit the debugger and start a new debug session.

Note

When debugging a MPI job that is launched under *pgserv*, the processes in the job are stopped before the first instruction of the program. Since there is no source level debugging information at this point, issuing the source level next command executes very slowly.

To avoid having to run the job until it completes, stops due to an exception, or stops by a *PGDBG* halt command entered by the user, the user should set an initial breakpoint. If a Fortran program is being debugged, set the initial breakpoint at `main`, or `MAIN_`, or at another point on the execution path before issuing the continue command.

For more information on *MPICH*, *MPICH2* and *MVAPICH*, refer to the “Multiprocessing MPI Debugging” section in the PGI Tools Guide.

Linking with ScaLAPACK

The ScaLAPACK libraries are automatically installed by the `installcdk` script described in step 2 of section 2.2. You can link with the ScaLAPACK libraries by specifying `-Mscalapack` on any of the *PGI CDK* compiler command lines. For example:

```
% pgf77 myprog.f -Mmpi=mpich -Mscalapack
```

or

```
% pgf77 myprog.f -Mmpi=mpich2 -Mscalapack
```

The `-Mscalapack` option causes the following libraries to be linked into your executable:

```
scalapack.a
blacsCinit_MPI-LINUX-0.a
blacs_MPI-LINUX-0.a
blacsF77init_MPI-LINUX-0.a
libblas.a
libmpich.a
```

These libraries are installed in

```
$PGI/linux86/11.1/mpi/mpich/lib
$PGI/linux86/11.1/mpi2/mpich/lib
$PGI/linux86/11.1/mpi/mvapich/lib.
```

You run a program that uses ScaLAPACK routines just like any other MPI program. The version of ScaLAPACK included in the *PGI CDK* is pre-configured for use with *MPICH*. If you wish to use a different BLAS library, and still use the `-Mscalapack` switch, you will have to copy your BLAS library into `$PGI/linux86/11.1/lib/libblas.a`.

Alternatively, you can just list the above set of libraries explicitly on your link line. You can test that ScaLAPACK is properly installed by running a test program as outlined in the following section.

Testing and Benchmarking

The `bench` directory contains various benchmarks and tests. Copy this directory into a local working directory by issuing the following command:

```
% cp -r $PGI/linux86/11.1/bench .
```

NAS Parallel Benchmarks

The `NPB2.3` subdirectory contains version 2.3 of the NAS Parallel Benchmarks in MPI. Issue the following commands to run the BT benchmark on 4 nodes of your cluster:

```
% cd bench/NPB2.3/BT
% make BT NPROCS=4 CLASS=W
% cd ../bin
% mpirun -np 4 bt.w.4
```

There are several other NAS parallel benchmarks available in this directory. Similar commands are used to build and run each of them. If you want to run a larger problem, try building the Class A version of BT by substituting "A" for "W" in the previous commands.

The `hpfnpb` subdirectory contains versions of five NAS Parallel Benchmarks coded in High Performance Fortran (HPF). README files explain how to build and run each of these benchmarks on various platforms. Use the instructions and makefiles in the `linux86` subdirectories of each benchmark to test these programs on your cluster.

ScaLAPACK

The ScaLaPack test times execution of the 3D PBLAS (parallel BLAS) on your cluster. To run this test, execute the following commands:

```
% cd scalapack
% make
% mpirun -np 4 pdbl3tim
```

Matrix Multiplication

The Matrix Multiplication test times execution of a simple distributed matrix multiply on your cluster. To run this test, execute the following commands

```
% cd matmul
% buildhpf
% mpirun -np 4 matmul_hpf
```

Limitations

The Open Source Cluster utilities, in particular the MPICH and ScaLAPACK libraries, are provided with support necessary to build and define their proper use. However, use of these libraries on `linux86-64` systems is subject to the following limitations:

- MPI libraries are limited to Messages of length < 2GB, and integer arguments are *INTEGER*4* in FORTRAN, and *int* in C.
- Integer arguments for ScaLAPACK libraries are *INTEGER*4* in FORTRAN, and *int* in C.

- Arrays passed must be < 2GB in size.

Using mpi Scripts

For MPICH1 and MVAPICH, if you use mpi scripts, such as `mpicc` to build with option `-fpic` or `-mmodel=medium`, then you must specify `-shlib` to link with the correct libraries. Here are a few examples:

For a static link to the mpi library, use this command:

```
% mpicc hello.f
```

For a dynamic link to the mpi library, use this command:

```
% mpicc hello.f -shlib
```

To compile with `-fpic`, which, by default, is a dynamic link, use this command:

```
% mpicc -fpic -shlib hello.f
```

To compile with `-mmodel=medium`, use this command:

```
% mpicc -mmodel=medium -shlib hello.f
```


Chapter 6. Contact Information

You can contact The Portland Group at:

The Portland Group
STMicroelectronics, Inc.
Two Centerpointe Drive
Lake Oswego, OR 97035 USA

Or electronically using any of the following means:

Fax	+1-503-682-2637
Sales	sales@pgroup.com
Support	trs@pgroup.com
WWW	www.pgroup.com

The PGI User Forum is monitored by members of the PGI engineering and support teams as well as other PGI customers. The forum newsgroups may contain answers to commonly asked questions. Log in to the PGI website to access the forum:

www.pgroup.com/userforum/index.php

Many questions and problems can be resolved by following instructions and the information available at our frequently asked questions (FAQ) site:

www.pgroup.com/support/faq.htm

All technical support is by email or submissions using an online form at www.pgroup.com/support. Phone support is not currently available.

PGI documentation is available at www.pgroup.com/resources/docs.htm or in your local copy of the documentation in the release directory doc/index.htm.

