

PGI CDK[®] 9.0

Cluster Development Kit[®]

Release Notes

The Portland Group[®]
STMicroelectronics, Inc
Two Centerpointe Drive
Lake Oswego, OR 97035
www.pgroup.com

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, Inc. and/or The Portland Group and may be used or copied only in accordance with the terms of the license agreement (“EULA”).

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, Inc. and/or The Portland Group.

PGI CDK 9.0 Cluster Development Kit

Release Notes

Copyright © 2009

The Portland Group®

STMicroelectronics, Inc. - All rights reserved.

Printed in the United States of America

First Printing Release 9.0-1 June 2009

Technical support: www.pgroup.com/support

Table of Contents

1	PGI RELEASE 9.0 INTRODUCTION.....	1
1.1	PRODUCT OVERVIEW.....	1
1.2	TERMS AND DEFINITIONS	3
2	PGI RELEASE 9.0 OVERVIEW	5
2.1	PGI RELEASE 9.0 CONTENTS.....	5
2.2	SUPPORTED PROCESSORS	6
2.3	SUPPORTED OPERATING SYSTEMS.....	8
3	NEW OR MODIFIED COMPILER FEATURES.....	11
3.1	WHAT'S NEW IN PGI RELEASE 9.0.....	11
3.2	GETTING STARTED	13
3.3	USING -FAST, -FASTSSE, AND OTHER PERFORMANCE-ENHANCING OPTIONS.....	13
3.4	NEW OR MODIFIED COMPILER OPTIONS.....	14
3.5	NEW OR MODIFIED DIRECTIVES AND PRAGMAS	16
3.6	FORTRAN ENHANCEMENTS.....	16
3.6.1	Enhanced Fortran Interoperability with C.....	16
3.6.2	New or Modified Fortran Statements and Assignment	17
3.6.3	New or Modified Fortran Intrinsic	18
3.6.4	New Intrinsic Module	19
3.6.5	Array-Related Enhancements.....	20
3.6.6	Additional Fortran Enhancements.....	21
3.7	NEW OR MODIFIED RUNTIME LIBRARY ROUTINES	21
3.8	NEW OR MODIFIED ENVIRONMENT VARIABLES	21

3.9	NEW OR MODIFIED TOOLS SUPPORT	21
3.9.1	PGDBG.....	21
3.9.2	PGPROF	22
3.10	NEW OR MODIFIED MPI SUPPORT	22
3.11	NEW OR MODIFIED OPENMPI PROFILING SUPPORT	23
3.11.1	Instructions for Linux	23
3.11.2	Compiler Wrapper data Files	24
3.11.3	Configure the Program for PGI Profiling.....	25
3.11.4	Modified Compiler Wrapper Data File Sample	26
3.12	LIBRARY INTERFACES	28
3.13	ENVIRONMENT MODULES	28
4	DISTRIBUTION AND DEPLOYMENT	29
4.1	APPLICATION DEPLOYMENT AND REDISTRIBUTABLES	29
4.1.1	PGI Redistributables	30
5	TROUBLESHOOTING TIPS AND KNOWN LIMITATIONS.....	31
5.1	GENERAL ISSUES	31
5.2	LINUX-SPECIFIC ISSUES	32
5.3	PGDBG-RELATED ISSUES	32
5.4	PGPROF-RELATED ISSUES.....	34
5.5	CORRECTIONS.....	35
6	CONTACT INFORMATION AND DOCUMENTATION	37

1 PGI Release 9.0

Introduction

Welcome to Release 9.0 of *PGI Cluster Development Kit*, or *PGI CDK*, a set of Fortran, C and C++ compilers and development tools for 32-bit and 64-bit x86-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.

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

1.1 Product Overview

Release 9.0 of PGI CDK includes the following components:

- *PGF95* OpenMP* and auto-parallelizing Fortran 90/95 compiler.
- *PGF77* OpenMP and auto-parallelizing FORTRAN 77 compiler.
- *PGHPF* data parallel High Performance Fortran compiler.
Note. PGHPF is supported only on Linux platforms.
- *PGCC* OpenMP and auto-parallelizing ANSI C99 and K&R C compiler.
- *PGC++* OpenMP and auto-parallelizing ANSI C++ compiler.
- *PGPROF* graphical OpenMP/multi-thread performance profiler.
- *PGDBG* graphical OpenMP/multi-thread symbolic debugger.

- *MPICH MPI libraries, version 1.2.7*, for both 32-bit and 64-bit development environments. (Note: 64-bit *linux86-64* MPI messages are limited to <2GB size each).
- *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.
- *TORQUE 1.2.0* resource management system, a continuation of the product known as *OpenPBS*, or Portable Batch System. See www.clusterresources.com/pages/products/torque-resource-manager.php for more information.
- *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 EM64T CPU-based installations. (Note: *linux86-64* versions are limited).
- *FlexLM* license utilities.

The release contains the following documentation and tutorial materials:

- *OSC Training Materials* –a set of HTML-based parallel and scientific programming training materials developed by the Ohio Supercomputer Center.
- Online documentation in PDF, HTML and man page formats.
- Online HPF tutorials that provide insight into cluster programming considerations.
- The CD-ROM media kit including the *PGI User's Guide*, the *PGI Tools Guide*, *MPI – The Complete Reference, Volume 1*, *How to Build a Beowulf*, and a printed copy of these release notes.

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.

1.2 Terms and Definitions

These release notes contain a number of terms and definitions with which you may or may not be familiar. If you encounter a term in these notes with which you are not familiar, please refer to the our 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 includes the AMD* Athlon64*, AMD Opteron* AMD Turion*, and Barcelona 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, and Intel Core 2 processors.

2

PGI Release 9.0 Overview

This document describes changes between Release 9.0 and previous releases of the *PGI CDK*, as well as late-breaking information not included in the current printing of the *PGI User's Guide*. There are two platforms supported by the *PGI CDK* compilers and tools:

- *32-bit Linux* – supported on *32-bit Linux operating systems* running on either a 32-bit x86 compatible or an x64 compatible processor.
- *64-bit/32-bit Linux* – includes all features and capabilities of the 32-bit Linux version, and is also supported on *64-bit Linux operating systems* running an x64 compatible processor.

These versions are distinguished in these release notes where necessary.

2.1 PGI Release 9.0 Contents

Release 9.0 of PGI CDK is comprised of the following components:

- *PGF95* native OpenMP and auto-parallelizing Fortran 95 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 Multi-process/multi-threaded* graphical profiler.
- *PGDBG Multi-process/multi-thread* graphical debugger.
- *MPICH MPI libraries, version 1.2.7*, for both 32-bit and 64-bit development environments.
- *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.
- *TORQUE 1.2.0* resource management system, a continuation of the product known as *OpenPBS*, or *Portable Batch System*. See www.clusterresources.com/pages/products/torque-resource-manager.php for more information.
- *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* and the PGI compilers on Linux systems with a kernel revision of 2.4.2 or higher. This is provided in both *linux86* and *linux86-64* versions for AMD64 or EM64T CPU-based installations.

The release contains the following documentation and tutorial materials:

- *OSC Training Materials* – an extensive set of HTML-based parallel and scientific programming training materials developed by the Ohio Supercomputer Center.
- Complete online documentation in PDF, HTML and UNIX man page formats.
- Online HP Fort tutorials that provide insight into cluster programming considerations.
- The hard-copy CD-ROM media kit including the *PGI User's Guide*, *PGI Tools Guide*, *MPI – The Complete Reference, Volume 1*, *How to Build a Beowulf*, and a printed copy of these release notes.

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

2.2 Supported Processors

The following table lists the processors on which Release 9.0 of the PGI compilers and tools is supported.

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 AMD and Intel 64-bit AMD64 compatible CPUs.

In addition to the capability to generate binaries optimized for specific AMD or Intel processors, the PGI 9.0 compilers can produce PGI Unified Binary object or executable files containing code streams fully optimized and supported for both the 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 the following table. The table also includes the CPUs available and supported in multi-core versions.

Processors Supported by PGI 9.0

Brand CPU	Cores	<target>	Memory Address	Floating Point HW					
				SSE1	SSE2	SSE3	SSSE3	SSE4	ABM and SSE4a
AMD Istanbul	4	istanbul-64	64-bit	Yes	Yes	Yes	No	Yes	Yes
AMD Istanbul	4	istanbul	64-bit	Yes	Yes	Yes	No	Yes	Yes
AMD Opteron/Quadcore	4	shanghai-64	64-bit	Yes	Yes	Yes	No	No	Yes
AMD Opteron/Quadcore	4	shanghai	64-bit	Yes	Yes	Yes	No	No	Yes
AMD Opteron/Quadcore	4	barcelona-64	64-bit	Yes	Yes	Yes	No	No	Yes
AMD Opteron/Quadcore	4	barcelona	32-bit	Yes	Yes	Yes	No	No	Yes
AMD Opteron/Athlon64	2	k8-64	32-bit	Yes	Yes	Yes	No	No	No
AMD Opteron/Athlon64	2	k8-32	32-bit	Yes	Yes	Yes	No	No	No
AMD Opteron Rev E/F Turion /Athlon64	2	k8-64e	64-bit	Yes	Yes	Yes	No	No	No
AMD Opteron Rev E/F	2	k8-32	32-bit	Yes	Yes	No	No	No	No
AMD Turion64 Turion /Athlon64	1	k8-64e	64-bit	Yes	Yes	Yes	No	No	No
AMD Turion64	1	k8-32	32-bit	Yes	Yes	No	No	No	No
Intel Core i7 (Nehalem)	4	nehalem-64	64-bit	Yes	Yes	Yes	Yes	Yes	Yes
Intel Core i7 (Nehalem)	4	nehalem	32-bit	Yes	Yes	Yes	Yes	Yes	Yes
Intel Penryn	4	penryn-64	64-bit	Yes	Yes	Yes	Yes	Yes	No
Intel Penryn	4	penryn	32-bit	Yes	Yes	Yes	Yes	Yes	No
Intel Core 2	2	core2-64	64-bit	Yes	Yes	Yes	Yes	Yes	No
Intel Core 2	2	core2	32-bit	Yes	Yes	Yes	Yes	Yes	No
Intel P4/Xeon EM64T	2	p7-64	64-bit	Yes	Yes	Yes	Yes	No	No
Intel P4/Xeon EM64T	2	p7	32-bit	Yes	Yes	Yes	Yes	No	No
Intel Xeon/Pentium 4	1	p7	32-bit	Yes	Yes	No	No	No	No

AMD	Athlon XP/MP	1	athlonxp	32-bit	Yes	No	No	No	No	No
Intel	Pentium m III	1	piii	32-bit	Yes	No	No	No	No	No
AMD	Athlon	1	athlon	32-bit	No	No	No	No	No	No
AMD	K6	1	k6	32-bit	No	No	No	No	No	No
Intel	Pentium II	1	p6	32-bit	No	No	No	No	No	No
Generic	Generic x86	1	p5 or px	32-bit	No	No	No	No	No	No

2.3 Supported Operating Systems

The following table lists the operating systems, and their equivalents, on which Release 9.0 of the PGI compilers and tools.

To determine if Release 9.0 will install and run under a Linux equivalent version, such as Mandrake*, Debian*, Gentoo*, 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.

- Newer distributions of the Linux operating systems include support for x64 compatible processors and 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 newer Linux distributions support the *Native Posix Threads Library* (NPTL), a new threads library that can be utilized in place of the *libpthread* library available in earlier versions of Linux. 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 thread library 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, and
 NUMA = Non-Uniform Memory Access.

Operating Systems and Features Supported in PGI 9.0									
<i>Distribution</i>	<i>Type</i>	<i>64-bit</i>	<i>HT</i>	<i>pgC++</i>	<i>pgdbg</i>	<i>NPTL</i>	<i>NUMA</i>	<i>glibc</i>	<i>GCC</i>
<i>RHEL 5.3</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>2.5</i>	<i>4.1.2</i>
<i>RHEL 5.0</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>2.5</i>	<i>4.1.2</i>
<i>RHEL 4.0</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>2.3.4</i>	<i>3.4.3</i>
<i>RHEL 3.0</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>2.3.2</i>	<i>3.2.3</i>
<i>Fedora 11</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.9</i>	<i>4.3.3</i>
<i>Fedora 10</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.9</i>	<i>4.3.2</i>
<i>Fedora 9</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.8</i>	<i>4.3.0</i>
<i>Fedora 8</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.7</i>	<i>4.1.2</i>
<i>Fedora 7</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.6</i>	<i>4.1.2</i>
<i>Fedora 6</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.5</i>	<i>4.1.1</i>
<i>Fedora 5</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.4</i>	<i>4.1.0</i>
<i>Fedora 4</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>2.3.5</i>	<i>4.0.0</i>
<i>Fedora 3</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>2.3.3</i>	<i>3.4.2</i>
<i>Fedora 2</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>2.3.3</i>	<i>3.3.3</i>
<i>SuSE 11.1</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.9</i>	<i>4.3.3</i>
<i>SuSE 11.0</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.8</i>	<i>4.3.1</i>
<i>SuSE 10.3</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.6.1</i>	<i>4.2.1</i>
<i>SuSE 10.2</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.5</i>	<i>4.1.0</i>
<i>SuSE 10.1</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.4</i>	<i>4.1.0</i>
<i>SuSE 10.0</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.3.5</i>	<i>4.0.2</i>
<i>SuSE 9.3</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.3.4</i>	<i>3.3.5</i>
<i>SuSE 9.2</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.3.3</i>	<i>3.3.4</i>
<i>SLES 11</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.9</i>	<i>4.3.3</i>
<i>SLES 10</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.4</i>	<i>4.1.0</i>
<i>SLES 9</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>2.3.3</i>	<i>3.3.3</i>
<i>SuSE 9.1</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>2.3.3</i>	<i>3.3.3</i>
<i>SuSE 9.0</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>2.3.2</i>	<i>3.3.1</i>
<i>SuSE 8.2</i>	Linux	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>2.3.2</i>	<i>3.3</i>
<i>RedHat 9.0</i>	Linux	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>2.3.2</i>	<i>3.2.2</i>
<i>Ubuntu 9.04</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.9</i>	<i>4.3.3</i>
<i>Ubuntu 8.10</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.8</i>	<i>4.3.2</i>
<i>Ubuntu 8.04</i>	Linux	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>2.7</i>	<i>4.2.1</i>

Note. www.pgroup.com/support/install.htm lists new distributions that may be explicitly supported by the PGI compilers. If your operating system is newer than any of those listed in the preceding table, the installation may still be successful.

3 New or Modified Compiler Features

This chapter provides information about the new or modified compiler features of Release 9.0 of the PGI compilers and tools as compared to prior releases.

3.1 What's New in PGI Release 9.0

- **PGI Accelerator x64+GPU Fortran and C99 compilers** supporting directive-based programming of x64+NVIDIA Linux systems. PGF95 and PGCC® accelerator compilers are supported on all Intel and AMD x64 processor-based systems with CUDA-enabled NVIDIA GPUs and enable incremental porting and tuning of Fortran and C programs from multi-core x64 to x64+GPU platforms.
- **Fortran 2003 incremental features** including: `IMPORT`, `POINTER` reshaping, `ISO_C_BINDING` `C_F_POINTER`, `ENUM`, `MOVE_ALLOC()`, `ISO_FORTRAN_ENV` module, optional `KIND` to intrinsics, allocatable scalars, `VOLATILE` attribute and statement, `PROCEDURE` pointers and statements, and high-speed `POPCNT`, `POPPAR`, and `LEADZ` intrinsics on architectures with explicit instructions to support these functions.
 - **Enhanced Fortran Intrinsics**
A number of Fortran intrinsics now have the `KIND` argument as an optional argument. For these intrinsics, when the `KIND` argument is present, the return value is of the specified kind. These intrinsics are listed in this document on page 16, and described in detail in the Intrinsics chapter of the PGI Fortran Reference.

- Enhanced Fortran Interoperability with C
Fortran 2003 provides a mechanism for interoperating with C. Any entity involved must have equivalent declarations made in both C and Fortran. . These enhancements are listed in this document on page 16, and described in detail in the *Interoperability with C* chapter of the PGI Fortran Reference.
- **Expanded Platform Support**
 - **Intel Core i7 (Nehalem) support** and optimizations including support for vectorization of loops using SSE 4.1/4.2 instructions.
 - **AMD Istanbul Six-Core AMD Opteron support** and optimizations.
- **PGC++/ PGCC enhancements** including full support for OpenMP 3.0 with tasks in C++, GNU linkonce support, C++ compile speed improved by up to 20%, and support for the `_m128` data type in the PGCC C99 compiler.
- **PGDBG parallel MPI/OpenMP debugger all-new graphical user interface (GUI)**
 - All new look-and-feel with intuitive navigation and usage features
 - Single easy-to-use menu bar replaces multiple menu bars in previous GUI
 - Tabbed panes for fast and easy switching between source/assembly/mixed debugging views, debugging contexts for multiple threads/processes, and graphical/textual multi-process/multi-thread state information
 - Improved DWARF generation for C/C++ include files, Fortran INCLUDE processing, and Fortran MODULES
- **PGPROF parallel MPI/OpenMP performance analysis and tuning tool**
 - New data collection mechanism via `pgcollect` enables profiling without re-compiling or any special software privileges
 - Support for profiling of code in shared object files
 - Updated GUI with tabbed access to multiple source files and improved drill-down to assembly code
 - Support for profiling of binaries compiled by non-PGI compilers
- **Updated Documentation** including the PGI Users Guide, PGI Tools Guide, and PGI Fortran Reference.

3.2 Getting Started

By default, the PGI 9.0 compilers generate code that is optimized for the type of processor on which compilation is performed, the compilation host. If you are unfamiliar with the PGI compilers and tools, a good option to use by default is *-fast* or *-fastsse*.

3.3 Using *-fast*, *-fastsse*, and Other Performance-Enhancing Options

These aggregate options incorporate a generally optimal set of flags for targets that support SSE capability. These options incorporate optimization options to enable use of vector streaming SIMD instructions for 64-bit targets. They enable vectorization with SSE instructions, cache alignment, and flushz.

These options incorporate optimization options to enable use of vector streaming SIMD (SSE/SSE2) instructions for 64-bit targets. The contents of the *-fast* switch are host-dependent, but usually includes the options *-O2 -Munroll -Mnoframe -Mlre*. For 64-bit targets, *-fast* also includes *-Mvect=sse -Mscalarsse -Mcache_align -Mflushz*.

Note. The contents of the *-fast* and *-fastsse* options are host-dependent.

- *-fast* and *-fastsse* typically include these options:
 - O2* Specifies a code optimization level of 2.
 - Munroll=c:1* Unrolls loops, executing multiple instances of the loop during each iteration.
 - Mnoframe* Indicates to not generate code to set up a stack frame.
Note. With this option, a stack trace does not work.
 - Mlre.* Indicates loop-carried redundancy elimination
- These additional options are also typically available when using *-fast* for 64-bit targets and *-fastsse* for both 32- and 64-bit targets:
 - Mvect=sse* Generates SSE instructions
 - Mscalarsse* Generates scalar SSE code with xmm registers; implies *-Mflushz*

- `-Mcache_align` Aligns long objects on cache-line boundaries.
Note. On 32-bit systems, if one file is compiled with the `-Mcache_align` option, all files should be compiled with it. This is not true on 64-bit systems.
- `-Mflushz` Sets SSE to flush-to-zero mode
- `-M[no]vect` Controls automatic vector pipelining.

Note. For best performance on processors that support SSE instructions, use the *PGF95* compiler, even for FORTRAN 77 code, and the `-fastsse` option.

In addition to `-fast`, the `-Mipa=fast` option for inter-procedural analysis and optimization can improve performance. You may be able to obtain further performance improvements by experimenting with the individual `-Mpgflag` options detailed in the *PGI User's Guide*, such as `-Mvect`, `-Munroll`, `-Minline`, `-Mconcur`, `-Mpfil`/`-Mpfo`, and so on. However, increased speeds using these options are typically application- and system-dependent, so it is important to time your application carefully when using these options to ensure no performance degradations occur.

3.4 New or Modified Compiler Options

Unknown options are treated as errors instead of warnings. This feature means it is a compiler error to pass switches that are not known to the compiler; however, you can use the switch `-noswitcherror` to issue warnings instead of errors for unknown switches.

The following compiler options have been added or modified in PGI 9.0:

- `-tp` has 6 new target cpu types:

istanbul	AMD Istanbul processor, 32-bit mode
istanbul-32	AMD Istanbul processor, 32-bit mode
istanbul-64	AMD Istanbul processor, 64-bit mode
nehalem	Intel Nehalem processor, 32-bit mode
nehalem-32	Intel Nehalem processor, 32-bit mode
nehalem-64	Intel Nehalem processor, 64-bit mode

- `-ta=nvidia(,nvidia_suboptions),host` is a new switch associated with the PGI Accelerator compilers. `-ta` defines the target architecture. Use this option to enable recognition of the `!$ACC` directives in Fortran, and `#pragma acc` directives in C. [C, Fortran only]

It has these suboptions:

- `nvidia` - select NVIDIA accelerator target.
This option has the following `nvidia`-suboptions:
 - `analysis` Perform loop analysis only. Do not generate code.
 - `cc10` Generate code for compute capability 1.0.
 - `cc11` Generate code for compute capability 1.1.
 - `cc13` Generate code for compute capability 1.3.
 - `nofma` Do not generate fused- multiply-add instructions.
 - `time` Link in a limited-profiling library
- `host` - select NO accelerator target. Generate PGI Unified Binary code.
- `-Minfo` has a new suboption:
 - `accel` Shows messages about the success or failure of the compiler in translating the accelerator region into GPU kernels.
- `-Msmartalloc` has a new suboption:
 - `nohuge` Overrides a `-Msmartalloc=huge` setting
- `-M[no]m128` is a new flag that recognizes `__m128`, `__m128d`, and `__m128i` datatypes. [C only]
- `-Mfprelaxed` has a new suboption:
 - `recip` Perform reciprocal with relaxed precision
- `-O` has a new suboption:
 - `-o4` Performs all level 1, 2 and 3 optimizations; in addition it enables hoisting of guarded invariant floating point expressions
- `-Minstrument [=functions]` is a new switch that enables instrumentation of functions. `-Minstrument=functions` is the same as `-Minstrument`. This option implies `-Minfo=ccff` and `-Mframe`. [linux86-64 only]

- `-Mipa` has a new suboption:
`nopfo` Enable profile feedback information. The `nopfo` option is valid only immediately following the `inline` suboption.
`-Mipa=inline,nopfo` tells IPA to ignore PFO information when deciding what functions to inline, if PFO information is available.
- `-Mpre` no longer supports the `all` suboption..
- `-Mallocatable=[95|03]` option controls how the compiler treats assignment of allocatables. The default behavior is to use Fortran 95 semantics; the `03` option instructs the compiler to use Fortran 2003 semantics.

3.5 New or Modified Directives and Pragmas

In this release, PGI supports the following additional directive.

- **IGNORE_TKR**
This directive indicates to the compiler to ignore the type, kind, and/or rank of the specified dummy arguments in an interface of a procedure. The compiler also ignores the type, kind, and/or rank of the actual arguments when checking all the specifics in a generic call for ambiguities.

3.6 Fortran Enhancements

3.6.1 Enhanced Fortran Interoperability with C

Fortran 2003 provides a mechanism for interoperating with C. Any entity involved must have equivalent declarations made in both C and Fortran. In this release, PGI has expanded Fortran interoperability with C by adding these components:

- Enumerators - a set of integer constants. The kind of enumerator corresponds to the integer type that C would choose for the same set of constants.
- `ISO_C_BINDING` module – implemented in this release:

- The `BIND` attribute is supported for derived types and constant kind definitions are provided that map to C types.
- For procedures, the `VALUE` and `BIND` attributes are supported, as well as the `BIND` attribute for global data.
- Procedure `C_LOC` is supported, which returns the C address of an object.
- Pointer types – a derived type `c_ptr`, that is interoperable with C pointer types. The named constant `c_null_ptr` corresponds to the null value in C.
- `c_f_pointer` – a subroutine that assigns the C pointer target, `cptr`, to the Fortran pointer, `fptr`, and optionally specifies its shape, `shape`. The syntax is:

```
c_f_pointer (cptr, fptr [,shape])
```

For more information on these components, refer to the *Interoperability with C* chapter of the PGI Fortran Reference.

3.6.2 New or Modified Fortran Statements and Assignment

These statements are new or modified in Release 9.0. For complete descriptions of these statements, refer to the PGI Fortran Reference.

- **IMPORT** - used only in an interface body, this statement gives access to the named entities of the containing scope.
- **Pointer Assignment** - Fortran 2003 extends pointer assignment for arrays allowing lower bounds and possibly upper bounds to be specified.

Syntax: `p(0:,0:) => a`

The lower bounds may be any scalar integer expressions when upper bounds are specified. Further, remapping of the elements of a target array is permitted, as shown in this example: `p(1:m,1:2*m) => a(1:2*m)`

- **Volatile Attribute** - in Fortran 2003, is used in a type declaration statement to indicate to the compiler that, at any time, the variable might change or be examined from outside the Fortran program.

Syntax: `datatype, volatile :: var_name`
 or `datatype :: var_name`
 `volatile :: var_name`

The following example declares both the integer variable `xyz` and the real variable `abc` to be volatile.

```
integer, volatile :: xyz
real :: abc
volatile :: abc
```

3.6.3 New or Modified Fortran Ininsics

An intrinsic is a function available in a given language whose implementation is handled specifically by the compiler. Since the compiler has an intimate knowledge of the intrinsic function, it can better integrate it and optimize it for the situation. In this release, PGI enhanced the following intrinsics as described.

- **LEADZ(I)** - an elemental function that returns the number of leading zero bits in `I`.
- **POPCNT(I)** - an elemental function that returns the number of one bits in the argument `I`.
- **POPPAR(I)** - an elemental function that returns the bitwise parity of the argument `I`.
- **MOVE_ALLOC(TO, FROM)** – is a subroutine with no return value that move an allocation from one allocatable object to another.
- **Added the KIND argument to each of the following intrinsics.**
 The **KIND** argument for these intrinsics is a scalar integer initialization expression. For these intrinsics, when the optional **KIND** argument is present, the return value is of the specified kind.
 - **ACHAR(I [, KIND])** – returns the character in the ASCII collating position specified by the argument `I`.
 - **IACHAR(C [, KIND])** – returns the position of the specified character, `C`, in the ASCII collating sequence.
 - **ICHAR(C [, KIND])** – returns the position of the specified character, `C`, in the character set's collating sequence.

- INDEX(*STRING*, *SUBSTRING* [, *BACK*[, *KIND*]]) – returns the starting position of a substring, *SUBSTRING*, within a string, *STRING*.
- LBOUND(*ARRAY* [, *DIM*[, *KIND*]]) – returns the lower bounds of an array, *ARRAY*, or the lower bound for the specified dimension, *DIM*.
- LEN(*STRING* [, *KIND*]) – returns the length of the supplied string, *STRING*.
- LEN_TRIM(*STRING* [, *KIND*]) – returns the length of the supplied string, *STRING*, minus the number of trailing blanks.
- MAXLOC(*ARRAY* [, *DIM*] [, *MASK*] [, *KIND*]) – Determines and returns the first position in the specified array that has the maximum value of the values in the array. The test elements may be limited by a dimension argument, *DIM*, a logical mask argument, *MASK*, or a kind, *KIND*.
- MINLOC(*ARRAY* [, *DIM*] [, *MASK*] [, *KIND*]) – Determines and returns the first position in the specified array, *ARRAY*, that has the minimum value of the values in the array. The test elements may be limited by a dimension argument, *DIM*, a logical mask argument, *MASK*, or a kind, *KIND*.
- SCAN(*STRING*, *SET* [, *BACK*[, *KIND*]]) – Search the supplied string, *STRING*, for a character in a set of characters, *SET*.
- SHAPE(*SOURCE* [, *KIND*]) – returns the shape of the supplied argument, *SOURCE*.
- SIZE(*ARRAY* [, *DIM*[, *KIND*]]) – returns either the total number of elements in the array, *ARRAY*, or the number of elements along a specified dimension, *DIM*.
- UBOUND(*ARRAY* [, *DIM*[, *KIND*]]) – returns the lower bounds of an array, *ARRAY*, or the lower bound for the specified dimension, *DIM*.
- VERIFY(*STRING*, *SET* [, *BACK*[, *KIND*]]) – verifies that a character string, *STRING*, contains all characters from a set of characters, *SET*; and returns an integer the is the position of the first (or last, if *BACK* is present) character that is not in the set.

3.6.4 New Intrinsic Module

PGI Workstation 9.0 now supports the Fortran intrinsic module, `iso_fortran_env`. This intrinsic module provides information about the Fortran environment through use of named constants.

Each named constant is a default integer scalar.

- `character_storage_size` - the size, in bits, of a character storage unit
- `error_unit` - the unit number for a preconnected output unit suitable for reporting errors. This may be the same as the output-unit.
- `file_storage_size` - the size, in bits, of a file storage unit.
- `input_unit` - the unit number for the preconnected external unit used for input.
- `iostat_end` - the value returned by `IOSTAT=` that indicates an end-of-file condition occurs during execution of a `READ` statement.
- `iostat_eor` - the value returned by `IOSTAT=` that indicates an end-of-record condition occurs during execution of a `READ` statement.
- `numeric_storage_size` - the size, in bits, of a numeric storage unit.
- `output_unit` - The unit number for the preconnected external unit used for output.

These special unit numbers may be negative, though they are never -1, since -1 is reserved for another purpose.

For more information on using intrinsic modules, refer to the *Intrinsics Modules* section of the PGI Fortran Reference.

3.6.5 Array-Related Enhancements

There are Fortran 2003 enhancements for arrays:

- **Allocatable attribute** – specifies that an array with fixed rank but deferred shape is available for a future `ALLOCATE` statement. An `ALLOCATE` statement allocates space for the allocatable object or scalar.
- **Fortran 2003 allocatable regularization** - implemented in PGF95, this is always enabled. These changes allow allocatable arrays to be passed as dummy arguments, to be returned from functions, and to be components of derived types.
- **Fortran 2003 Allocatable Array Assignment** - Available in PGF95, the default is to use the Fortran 95 assignment semantics; however, the option `-Mallocatable=03` enables the Fortran 2003 assignment semantics.

3.6.6 Additional Fortran Enhancements

Fortran 2003 Asynchronous Input/Output - Fortran 2003 asynchronous I/O is partially implemented in PGF77 and PGF95 compilers.

- For external files opened with `ASYNCHRONOUS='YES'` in the `OPEN` statement, asynchronous I/O is allowed.
 - Asynchronous I/O operations are indicated by `ASYNCHRONOUS='YES'` in `READ` and `WRITE` statements.
 - The compilers do not implement the `ASYNCHRONOUS` attribute or `ASYNCHRONOUS` statement.
- **Fortran 2003 Stream Input/Output** - Fortran 2003 Stream access I/O is implemented.

3.7 New or Modified Runtime Library Routines

PGI Workstation 9.0 supports new run-time library routines associated with the PGI Accelerator compilers. For more information, refer to *Using an Accelerator* in the PGI User's Guide.

3.8 New or Modified Environment Variables

PGI Workstation 9.0 supports new environment variables associated with the PGI Accelerator compilers. For more information, refer to the *Using an Accelerator* in the PGI User's Guide.

3.9 New or Modified Tools Support

The PGI Tools Guide describes the tools in detail as well as explains the new features highlighted in this section.

3.9.1 PGDBG

The PGDBG graphical MPI/OpenMP/multi-thread symbolic debugger has these enhancements in this release:

- All new graphical user interface (GUI)
 - Intuitive navigation and usage features
 - Single easy-to-use menu bar replaces multiple menu bars in previous GUI
 - Tabbed panes
Facilitate fast and easy switching between source/assembly/mixed debugging views, debugging contexts for multiple threads/processes, and graphical/textual multi-process/multi-thread state information
- Improved debugging of symbol and source lines in C/C++ include files, Fortran INCLUDE processing, and Fortran MODULES

3.9.2 PGPROF

PGPROF graphical MPI/OpenMP/multi-thread performance analysis and tuning profiler has these enhancements in this release:

- New data collection mechanism via pgcollect enables profiling without re-compiling or any special software co-installation requirements for OProfile. You can use pgcollect in standalone mode for time-based sampling using only PGI software – both on Linux and on Mac OS X 10.5 (Leopard).
- Support for profiling of code in shared object files – on Linux. Dynamic libraries are not yet supported on Mac OS X.
- Updated GUI for easier navigation with tabbed access to multiple source files and improved drill-down to assembly code
- Support for profiling of binaries compiled by non-PGI compilers

3.10 New or Modified MPI Support

Prior to PGI Release 7.1, PGI provided MPI support only in the PGI CDK. In release 7.1, a version of MPICH1 was included with PGI Workstation on Linux, and the debugger and profiler were enabled to support MPI applications running locally with a limited number of processes.

In this release, PGI Workstation 9.0-1, the local MPI capability continues to expand. You can debug and profile MPI applications for MPICH-1 (using the included version of MPICH-1), HP-MPI for Linux, MPICH-2, or MVAPICH. This chapter describes how to use these capabilities and some of their limitations.

The PGI Tools Guide describes the MPI-enabled tools in detail:

- PGPROF graphical MPI/OpenMP/multi-thread performance profiler.
- PGDBG graphical MPI/OpenMP/multi-thread symbolic debugger

For specific information about how to use MPI, refer to Chapter 6, *Using MPI*, of the PGI User's Guide.

3.11 New or Modified OpenMPI Profiling Support

Pgi now provides performance profiling of MPI message passing support for OpenMPI applications on Linux and Mac OS X. On Apple systems, no special configuration is necessary. On Linux systems you must configure the OpenMPI installation to work with the PGI profiling system.

3.11.1 Instructions for Linux

This section contains instructions on how to install and configure your system for OpenMPI profiling on Linux. The basic steps include:

1. **Build and Install PGI-built OpenMPI:** Build the OpenMPI software distribution with PGI compilers and install it.
2. **Configure OpenMPI for PGI profiling.** Refer to subsection 3.11.3 for details.
3. **Build your program:** Build using the OpenMPI compiler wrappers (`mpicc`, `mpic++`, `mpif77`, and/or `mpif90`) with one of the PGI profiling options.

Note. When you build with `-Mprof=time|lines|func|hwcts`, then MPI profiling is included automatically.

4. **Run your program:** Run as you normally would. One or more files named `pgprof.out` is created in your working directory.
5. **Run the profiler:** Invoke the profiler to see the results of your profiling run.

```
pgprof -exe your_program
```

3.11.2 Compiler Wrapper data Files

To configure your OpenMPI installation for PGI profiling, you must make a few simple modifications to some configuration files, which we refer to as compiler wrapper data files.

The compiler wrapper data files are located in the `/share/openmpi` directory of your OpenMPI installation. Example compiler wrapper data files located in your PGI `/etc` directory are available for you to direct modifications of the wrapper data files generated when you built OpenMPI.

The wrapper file names are:

```
m      picc-wrapper-data.txt
m      pic++-wrapper-data.txt
m      pif77-wrapper-data.txt
m      pif90-wrapper-data.txt
```

A sample wrapper file includes a block of data similar to the following.

Note: The lines in bold are ones you modify when you configure your OpenMPI installation for PGI profiling.

```
compiler_args=
project=Open MPI
project_short=OMPI
version=1.2.8
language=C
compiler_env=CC
compiler_flags_env=CFLAGS
compiler=pgcc
extra_includes=
preprocessor_flags=-D_REENTRANT
compiler_flags=
linker_flags=
libs=-lmpi -lopen-rte -lopen-pal -lrt -ldl
-Wl,--export-dynamic -lnsl -lutil -lpthread -ldl
required_file=
includedir=${includedir}
libdir=${libdir}
```

3.11.3 Configure the Program for PGI Profiling

To configure the program for PGI profiling, you edit the compiler wrapper data files. The lines that you modify are in bold in the sample wrapper data file in the previous section.

Important. Before you begin, make backup copies of your original wrapper data files.

Make these modifications:

Step 1. Add the line `'compiler_args='` before any other configuration lines.

Step 2. Copy the entire data block in the sample file *twice*. You need a data block for each of these compiler options:
one for `-Mprof=func | lines` and
one for `-Mprof=time | hwcts` (hwcts is linux86-64 only.)

Step 3. In the second data block, modify the `'compiler_args='` line and the `'compiler_flags='` lines. The PGI profiling options are shown just to the right of the equal sign. The compiler flags you select immediately follow the equal sign, with a space between each flag.

Your lines should look similar to these:

```
compiler_args=-Mprof=func;-Mprof=lines
...
compiler_flags=
...
```

Step 4. In the third data block, modify the `'compiler_args='` line and the `'compiler_flags='` lines. The PGI profiling options are shown just to the right of the equal sign. The compiler flags in this data block should include: `-w0, -profile, lines` at the beginning of the list of flags you select.

Your lines should look similar to these:

```
compiler_args=-Mprof=time;-Mprof=hwcts
...
compiler_flags=-w0,-profile,lines
...
```

Step 5. In both the second and third data blocks, modify the 'libs=' line so that '-lpgnod_prof_openmpi' comes just before '-lmpi'.

Note. Do not modify any other lib values.

The new 'libs=' line looks similar to this:

```
libs=-lpgnod_prof_openmpi -lmpi -lopen-rte
-lopen-pal -lrt -ldl -Wl,--export-dynamic
-lns1 -lutil -lpthread -ldl
```

3.11.4 Modified Compiler Wrapper Data File Sample

When you complete your modifications, your new wrapper data file has three data blocks that look similar to these. The lines you modified are in bold.

```
compiler_args=
project=Open MPI
project_short=OMPI
version=1.2.8
language=C
compiler_env=CC
compiler_flags_env=CFLAGS
compiler=pgcc
extra_includes=
preprocessor_flags=-D_REENTRANT
compiler_flags=
linker_flags=
libs=-lmpi -lopen-rte -lopen-pal -lrt -ldl -Wl,
--export-dynamic -lns1 -lutil -lpthread -ldl
required_file=
includedir=${includedir}
libdir=${libdir}
```

```

compiler_args=-Mprof=func;-Mprof=lines
project=Open MPI
project_short=OMPI
version=1.2.8
language=C
compiler_env=CC
compiler_flags_env=CFLAGS
compiler=pgcc
extra_includes=
preprocessor_flags=-D_REENTRANT
compiler_flags=
linker_flags=
libs=-lpgnod_prof_openmpi -lmpi -lopen-rte -lopen-pal
-lrt -ldl -Wl,--export-dynamic -lnsl -lutil -lpthread
-ldl
required_file=
includedir=${includedir}
libdir=${libdir}
compiler_args=-Mprof=time;-Mprof=hwcts
project=Open MPI
project_short=OMPI
version=1.2.8
language=C
compiler_env=CC
compiler_flags_env=CFLAGS
compiler=pgcc
extra_includes=
preprocessor_flags=-D_REENTRANT
compiler_flags=-W0,-profile,lines
linker_flags=
libs=-lpgnod_prof_openmpi -lmpi -lopen-rte -lopen-pal
-lrt -ldl -Wl,--export-dynamic -lnsl -lutil -lpthread
-ldl
required_file=
includedir=${includedir}
libdir=${libdir}

```

3.12 Library Interfaces

PGI provides access to a number of libraries that export C interfaces by using Fortran modules. These libraries and functions are described in Chapter 8 of the *PGI User's Guide*.

3.13 Environment Modules

On Linux, if you use the Environment Modules package (e.g., the `module load` command), PGI 9.0 includes a script to set up the appropriate module files.

4 Distribution and Deployment

This chapter contains a number of topics that are related to using the compilers, including optimizing through the use of PGI Unified Binary Technology, using the linking options on Windows, using the module load command on Linux, distributing the files, and customizing with `siterc` and `user rc` files.

- For more information on generating PGI Unified Binaries, including PGI Unified Binary command-line switches, directives, and pragmas, refer to Chapter 9, *Distributing Files – Deployment*, of the PGI User’s Guide.
- For more information and usage examples of the PGI compiler options that allow you to select static or dynamic linking, as well as information on using and creating static and dynamically-linked libraries, refer to Chapter 8, *Creating and Using Libraries*, of the PGI User’s Guide.
- For examples and information on customizing with `siterc` and `user rc` files to tailor a given installation for a particular purpose, refer to Chapter 1 of the PGI User’s Guide, specifically, *Examples of Using `siterc` and `User rc Files`*.

4.1 Application Deployment and Redistributables

Programs built with PGI compiler may depend on run-time library files. These library files must be distributed with such programs to enable them to execute on systems where the PGI compilers are not installed. There are PGI redistributable files for all platforms. On Windows, PGI also supplies Microsoft redistributable files.

4.1.1 PGI Redistributables

The PGI 9.0 release includes these directories:

- *\$PGI/linux86/9.0/REDIST*
- *\$PGI/linux86-64/9.0/REDIST*
- *\$PGI/win64/9.0/REDIST*
- *\$PGI/win32/9.0/REDIST*

These directories contain all of the PGI Linux runtime library shared object files or Windows dynamically linked libraries that can be re-distributed by PGI 9.0 licensees under the terms of the PGI End-user License Agreement (EULA). For reference, a text-form copy of the PGI EULA is included in the 9.0 directory.

The REDIST directories contain the PGI runtime library shared objects for all supported targets. This enables users of the PGI compilers to create packages of executables and PGI runtime libraries that execute successfully on almost any PGI-supported target system, subject to these requirements:

- End-users of the executable have properly initialized their environment
- On Linux, users have set LD_LIBRARY_PATH to use the relevant version of the PGI shared objects.

5 Troubleshooting Tips and Known Limitations

This chapter contains information about known limitations, documentation errors, and corrections that have occurred to PGI Workstation 9.0.

The frequently asked questions (FAQ) section of the *pgroup.com* web page at <http://www.pgroup.com/support/index.htm> provides more up-to-date information about the state of the current release.

5.1 General Issues

Most issues in this section are related to specific uses of compiler options and suboptions.

- Object and module files created using *PGI Workstation 9.0* compilers are incompatible with object files from *PGI Workstation 5.x* and prior releases.
- Object files compiled with *-Mipa* using *PGI Workstation 6.1* and prior releases must be recompiled with *PGI Workstation 9.0*.
- The *-i8* option can make programs incompatible with the bundled ACML library. Visit *developer.amd.com* to check for compatible libraries.
- The *-i8* option can make programs incompatible with MPI and ACML; use of any `INTEGER*8` array size argument can cause failures with these libraries.
- Using *-Mipa=vestigial* in combination with *-Mipa=libopt* with *PGCC*, you may encounter unresolved references at link time. This problem is due to the erroneous removal of functions by the *vestigial* sub-option to *-Mipa*. You can work around this problem by listing specific sub-options to *-Mipa*, not including *vestigial*.

- *OpenMP* programs compiled using *-mp* and run on multiple processors of a *SuSE 9.0* system can run very slowly. These same executables deliver the expected performance and speed-up on similar hardware running *SuSE 9.1* and above.
- ACML 4.3.0 is built using the *-fastsse* compile/link option, which includes *-Mcache_align*. When linking with ACML using the *-lacml* option on 32-bit targets, all program units must be compiled with *-Mcache_align*, or an aggregate option such as *-fastsse*, which incorporates *-Mcache_align*. This process is not an issue on 64-bit targets where the stack is 16-byte aligned by default. The lower-performance, but fully portable, *libblas.a* and *liblapack.a* libraries can be used on CPUs that do not support SSE instructions.
- When compiling with *-fPIC* and linking with *-lacml*, you may get the message “error while loading shared libraries: libacml_mv.so: cannot open shared object file: No such file or directory.” In this case, you must add *-lacml_mv* library to the link line.
- Using *-Mpfi* and *-mp* together is not supported. The *-Mpfi* flag will disable *-mp* at compile time, which can cause run-time errors in programs that depend on interpretation of OpenMP directives or pragmas. Programs that do not depend on OpenMP processing for correctness can still use profile feedback. The *-Mpf0* flag does not disable OpenMP processing.

5.2 Linux-specific Issues

- Programs that incorporate object files compiled using *-mmodel=medium* cannot be statically linked. This is a limitation of the *linux86-64* environment, not a limitation of the PGI compilers and tools.

5.3 PGDBG-related Issues

- Before PGDBG can set a breakpoint in code contained in a shared library, *.so* or *.dll*, the shared library must be loaded.
- PGDBG supports debugging Open MPI programs with the caveat that Open MPI be built with static libraries. When Open MPI is built with dynamic libraries, many shared libraries are loaded by the Open

MPI application. With the current implementation of PGDBG's shared object load/unload event handlers, debugging a MPI job linked with Open MPI shared objects causes MPI_Init and MPI_Finalize to execute very slowly. Therefore, we recommend that Open MPI be built via static libraries by configuring the build of Open MPI with the following options:

```
--enable-static --disable-shared
```

- PGDBG 8.0 release introduced better support for debugging MPI programs on newer Linux systems where the loading of shared libraries to randomized addresses is enabled. When this Linux kernel mode is enabled, the current implementation of PGDBG uses significantly more memory, which may degrade performance. PGI currently recommends disabling this mode in the Linux kernel when debugging MPI programs via PGDBG. To disable this Linux kernel mode, run the following command as root:

```
sysctl -w kernel.randomize_va_space=0
```
- Due to problems in PGDBG in shared library load recognition on Fedora Core 6 or RHEL5, breakpoints in processes other than the process with rank 0 may be ignored when debugging MPICH-1 applications when the loading of shared libraries to randomized addresses is enabled.
- Do not use “./” to specify an executable in the current directory when debugging an MPICH-1 application with ‘mpirun -dbg=pgdbg’. For example, if you do use

```
mpirun -dbg=pgdbg -np 2 ./a.out
```

when the Linux kernel mode is enabled, breakpoints may be lost in processes other than the process with rank 0. To work around this problem invoke the job to be debugged using this command:

```
mpirun -dbg=pgdbg -np 2 a.out
```

- When debugging an 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.
- The `watch` family of commands is unreliable when used with local variables. Calling a function or subroutine from within the scope of the watched local variable may cause missed events and/or false positive events. Local variables may be watched reliably if program scope does not leave the scope of the watched variable. Using the `watch` family of commands with global or static variables is reliable.
- Debugging of unified binaries, that is, programs built with the `-tp=x64` option, is not fully supported. The names of some subprograms are modified in the creation of the unified binary, and PGDBG does not translate these names back to the names used in the application source code. For detailed information on how to debug a unified binary, see www.pgroup.com/support/tools.htm.

5.4 PGPROF-related Issues

- Using `-Mprof=func`, `-mcmode=medium` and `-mp` together on any of the PGI compilers can result in segmentation faults by the generated executable. These options should not be used together.
- Programs compiled and linked for `gprof`-style performance profiling using `-pg` can result in segmentation faults on system running version 2.6.4 Linux kernels.
- Times reported for multi-threaded sample-based profiles, that is, profiling invoked with `-pg` or `-Mprof=time` options, are for the master thread only. PGI-style instrumentation profiling with `-Mprof={lines | func}` or hardware counter-based profiling using `-Mprof=hwcts` or `pgcollect` must be used to obtain profile data on individual threads.

5.5 Corrections

The following problems have been corrected in the *PGI 9.0* release. Most were reported in PGI 7.2 or previous releases. Problems found in PGI 7.2 may not have occurred in the previous releases.

Refer to www.pgroup.com/support/release_tprs.htm for a complete and up-to-date table of technical problem reports, TPRs, fixed in recent releases of the PGI compilers and tools. This table contains a summary description of each problem as well as the release in which it was fixed.

6 Contact Information and Documentation

You can contact The Portland Group at:

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

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

Or contact us electronically using any of the following means:

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

All technical support is by e-mail or submissions using an online form at www.pgroup.com/support. Phone support is not currently available. Many questions and problems can be resolved at our frequently asked questions (FAQ) site at www.pgroup.com/support/faq.htm.

Online documentation is available by pointing your browser at either your local copy of the documentation in the release directory doc/index.htm or online at www.pgroup.com/resources/docs.htm.