

PGI<sup>®</sup> Server 9.0

PGI<sup>®</sup> Workstation 9.0

# Release Notes

The Portland Group®  
STMicroelectronics, Inc  
Two Centerpointe Drive  
Lake Oswego, OR 97035  
[www.pgroup.com](http://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 Server 9.0 / PGI Workstation 9.0

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](http://www.pgroup.com/support)

# Table of Contents

---

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>INTRODUCTION</b> .....  | <b>1</b>  |
| 1.1      | PGI WORKSTATION AND PGI SERVER.....                                    | 1         |
| 1.1.1    | Licensing Terminology.....   | 1         |
| 1.1.2    | Workstation and Server Comparison.....                                 | 2         |
| 1.2      | PRODUCT OVERVIEW.....  | 2         |
| 1.3      | TERMS AND DEFINITIONS.....   | 3         |
| <b>2</b> | <b>PGI RELEASE 9.0 OVERVIEW</b> .....                                  | <b>5</b>  |
| 2.1      | PGI RELEASE 9.0 CONTENTS.....  | 6         |
| 2.2      | SUPPORTED PROCESSORS.....  | 6         |
| 2.3      | SUPPORTED OPERATING SYSTEMS.....                                       | 8         |
| <b>3</b> | <b>NEW OR MODIFIED COMPILER FEATURES</b> .....                         | <b>11</b> |
| 3.1      | WHAT'S NEW IN PGI RELEASE 9.0.....                                     | 11        |
| 3.2      | GETTING STARTED.....   | 13        |
| 3.3      | USING -FAST, -FASTSSE, AND OTHER<br>PERFORMANCE-ENHANCING OPTIONS..... | 13        |
| 3.4      | NEW OR MODIFIED COMPILER OPTIONS.....                                  | 14        |
| 3.5      | NEW OR MODIFIED DIRECTIVES AND PRAGMAS.....                            | 15        |
| 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 Intrinsics.....                                | 17        |
| 3.6.4    | New Intrinsic Module.....  | 19        |
| 3.6.5    | Array-Related Enhancements.....  | 19        |
| 3.6.6    | Additional Fortran Enhancements.....                                   | 20        |
| 3.7      | NEW OR MODIFIED RUNTIME LIBRARY ROUTINES.....                          | 20        |
| 3.8      | NEW OR MODIFIED ENVIRONMENT VARIABLES.....                             | 20        |

|          |   |           |
|----------|---|-----------|
| 3.9      | NEW OR MODIFIED TOOLS SUPPORT .....                 | 21        |
| 3.9.1    | PGDBG.....  | 21        |
| 3.9.2    | PGPROF .....  | 21        |
| 3.10     | NEW OR MODIFIED MPI SUPPORT .....                   | 22        |
| 3.11     | NEW OR MODIFIED OPENMPI PROFILING SUPPORT .....     | 22        |
| 3.11.1   | Instructions for Linux .....                        | 22        |
| 3.11.2   | Compiler Wrapper data Files .....                   | 23        |
| 3.11.3   | Configure the Program for PGI Profiling .....       | 24        |
| 3.11.4   | Modified Compiler Wrapper Data File Sample .....    | 25        |
| 3.11.5   | Instructions for Apple OS X .....                   | 27        |
| 3.12     | LIBRARY INTERFACES .....                            | 27        |
| <b>4</b> | <b>PGI WORKSTATION 9.0.....</b>                     | <b>29</b> |
| 4.1      | PGI WORKSTATION 9.0 FOR LINUX.....                  | 29        |
| 4.1.1    | Java Runtime Environment (JRE).....                 | 29        |
| 4.2      | PGI WORKSTATION 9.0 FOR WINDOWS, SFU, AND SUA ..... | 29        |
| 4.3      | PGI WORKSTATION 9.0 FOR MAC OS X.....               | 30        |
| <b>5</b> | <b>PGI ACCELERATOR.....</b>                         | <b>31</b> |
| 5.1      | COMPONENTS .....                                    | 31        |
| 5.2      | AVAILABILITY .....                                  | 31        |
| 5.3      | USER-DIRECTED ACCELERATOR PROGRAMMING .....         | 32        |
| 5.4      | FEATURES NOT COVERED OR IMPLEMENTED .....           | 32        |
| 5.5      | SYSTEM REQUIREMENTS .....                           | 32        |
| 5.5.1    | Supported Processors and GPUs.....                  | 33        |
| 5.6      | INSTALLATION AND LICENSING .....                    | 33        |
| 5.7      | RUNNING AN ACCELERATOR PROGRAM .....                | 33        |
| 5.7.1    | PGI Accelerator Compilers Runtime Libraries .....   | 34        |
| 5.7.2    | Environment Variables .....                         | 35        |
| 5.7.3    | Applicable Command Line Options .....               | 35        |
| 5.8      | PGI UNIFIED BINARY FOR ACCELERATORS .....           | 36        |
| 5.9      | MULTIPLE PROCESSOR TARGETS.....                     | 38        |
| 5.10     | PROFILING ACCELERATOR KERNELS .....                 | 38        |
| 5.11     | SUPPORTED INTRINSICS.....                           | 39        |
| <b>6</b> | <b>DISTRIBUTION AND DEPLOYMENT .....</b>            | <b>41</b> |
| 6.1      | APPLICATION DEPLOYMENT AND REDISTRIBUTABLES .....   | 41        |
| 6.1.1    | PGI Redistributables .....                          | 42        |
| 6.1.2    | Linux Redistributables .....                        | 42        |
| 6.1.3    | Microsoft Redistributables.....                     | 42        |

|          |  |           |
|----------|--|-----------|
| <b>7</b> | <b>THE PGI WINDOWS CDK .....</b>                           | <b>43</b> |
| 7.1      | BUILD MPI APPLICATIONS WITH MSMPI.....                     | 43        |
| 7.1.1    | Using MSMPI libraries .....                                | 43        |
| 7.1.2    | Generate MPI Profile Data.....                             | 43        |
| 7.2      | DEBUG MSMPI APPLICATIONS WITH PGDBG .....                  | 44        |
| 7.2.1    | Bash Shell Example .....                                   | 45        |
| 7.2.2    | DOS Shell Example .....                                    | 45        |
| <b>8</b> | <b>TROUBLESHOOTING TIPS AND KNOWN<br/>LIMITATIONS.....</b> | <b>47</b> |
| 8.1      | GENERAL ISSUES .....                                       | 47        |
| 8.2      | PLATFORM-SPECIFIC ISSUES .....                             | 48        |
| 8.2.1    | Linux .....  | 48        |
| 8.2.2    | Apple Mac OS X.....  | 48        |
| 8.2.3    | Windows .....  | 49        |
| 8.3      | PGDBG-RELATED ISSUES .....                                 | 50        |
| 8.4      | PGPROF-RELATED ISSUES.....                                 | 52        |
| 8.5      | CORRECTIONS.....   | 53        |
| <b>9</b> | <b>CONTACT INFORMATION AND DOCUMENTATION .....</b>         | <b>55</b> |



# 1

# Introduction

---

Welcome to Release 9.0 of *PGI Workstation* and *PGI Server*, 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, Windows, and Mac OS operating systems.

These release notes apply to all workstation-class and server-class compiler products from The Portland Group.

## 1.1 PGI Workstation and PGI Server

The PGI Workstation and PGI Server include exactly the same software. The difference in the two is the manner in which the user gets a license to run the software.

### 1.1.1 Licensing Terminology

The PGI compilers and tools are license-managed. It is useful to have common terminology, These two terms are often confused, so they are clarified here:

- **License** - is a legal agreement between ST 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 SW installation.
- **License keys** - are 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 each PGI end-user on pgroup.com using a unique hostid and 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.

## 1.1.2 Workstation and Server Comparison

- All workstation-class compilers and tools products from The Portland Group, such as *PGI Fortran Workstation*, are subsets of the *PGI Workstation Complete* product. These workstation-class products provide a node-locked single-user license, meaning one user at a time can compile on the one system on which the *PGI Workstation* compilers and tools are installed. The product and license server are on the same local machine.
- *PGI Server* products are offered in configurations identical to the workstation-class products, but provide network-floating multi-user licenses. This means that two or more users can use the *PGI* compilers and tools concurrently on any compatible system networked to the system on which the *PGI Server* compilers are installed. There can be multiple installations on machines connected to the server and the users can use the product concurrently, provided they are issued a license key.

## 1.2 Product Overview

Release 9.0 of PGI Workstation and PGI Server 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 MPI/OpenMP/multi-thread performance profiler.
- *PGDBG* graphical MPI/OpenMP/multi-thread symbolic debugger
- MPICH MPI libraries, version 1.2.7, for both 32-bit and 64-bit development environments (Linux only)
- Online documentation in PDF, HTML and man page formats.
- A UNIX\*-like shell environment for Win32 and Win64 platforms.

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

The MPI profiler and debugger included with PGI Workstation are limited to processes on a single node. PGI Workstation can be installed on a single

computer, and that computer can be used to develop, debug, and profile MPI applications. The PGI CDK Cluster Development Kit supports general development on clusters.

## 1.3 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 online glossary at

[www.pgroup.com/support/definitions.htm](http://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, and AMD Istanbul 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, Intel Penryn, and Intel Core i7 (Nehalem) processors.



## 2

# PGI Release 9.0 Overview

---

This document describes changes between previous releases and Release 9.0 of the PGI compilers, as well as late-breaking information not included in the current printing of the *PGI User's Guide*. There are nine platforms supported by the *PGI Workstation* and *PGI Server* 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.
- *32-bit Windows* – supported on *32-bit Windows operating systems* running on either a 32-bit *x86* compatible or an *x64* compatible processor.
- *64-bit/32-bit Windows* – includes all features and capabilities of the 32-bit Windows version, and is also supported on *64-bit Windows operating systems* running an *x64* compatible processor.
- *32-bit SFU* – supported on *32-bit Windows operating systems* running on either a 32-bit *x86* compatible or an *x64* compatible processor.
- *32-bit SUA* – supported on *32-bit Windows operating systems* running on either a 32-bit *x86* compatible or an *x64* compatible processor.
- *64-bit/32-bit SUA* – includes all features and capabilities of the 32-bit SUA version, and is also supported on *64-bit Windows operating systems* running an *x64* compatible processor.
- *32-bit Apple Mac OS X* – supported on 32-bit Apple Mac operating systems running on either a 32-bit or 64-bit Intel-based Mac system.
- *64-bit Apple Mac OS X* – supported on 64-bit Apple Mac operating systems running on a 64-bit Intel-based Mac system.

These release notes distinguish these versions where necessary.

## 2.1 PGI Release 9.0 Contents

Release 9.0 of PGI Workstation and PGI Server includes 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.  
Note: PGHPF is supported only on Linux platforms.
- PGCC native OpenMP and auto-parallelizing ANSI C99 and K&R C compiler.
- PGC++ native OpenMP and auto-parallelizing ANSI C++ compiler.
- PGPROF multi-thread, OpenMP and MPI graphical profiler.
- PGDBG multi-thread, OpenMP and MPI graphical debugger.
- MPICH MPI libraries, version 1.2.7, for both 32-bit and 64-bit development environments (Linux only)
- Complete online documentation in PDF, HTML and UNIX man page formats.
- A UNIX-like shell environment for Win32 and Win64 environments.

Depending on the product you purchased, you may not have licensed all of the above components.

## 2.2 Supported Processors

The following table contains 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 AMD and Intel x64 CPUs. The `-tp <target>` option must be used to produce unified binary files.

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<br>Turion<br>/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<br>Turion<br>/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/Pentium4                          | 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 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 table lists the operating systems, and their equivalents, that Release 9.0 of the PGI compilers and tools supports.

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 and Windows 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.
- If you attempt to install the 64-bit Windows version on a system running 32-bit Windows, the installation fails.

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 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. For more information on these terms, refer to Terms and Definitions on page 3.

## 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> |
|--------------------------|-----------------|---------------|-----------|--------------|--------------|-------------|-------------|--------------|------------|
| <b>RHEL 5.3</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | No          | 2.5          | 4.1.2      |
| <b>RHEL 5.0</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | No          | 2.5          | 4.1.2      |
| <b>RHEL 4.0</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | No          | 2.3.4        | 3.4.3      |
| <b>RHEL 3.0</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | No          | 2.3.2        | 3.2.3      |
| <b>Fedora 11</b>         | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.9          | 4.3.3      |
| <b>Fedora 10</b>         | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.9          | 4.3.2      |
| <b>Fedora 9</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.8          | 4.3.0      |
| <b>Fedora 8</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.7          | 4.1.2      |
| <b>Fedora 7</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.6          | 4.1.2      |
| <b>Fedora 6</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.5          | 4.1.1      |
| <b>Fedora 5</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.4          | 4.1.0      |
| <b>Fedora 4</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | No          | 2.3.5        | 4.0.0      |
| <b>Fedora 3</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | No          | 2.3.3        | 3.4.2      |
| <b>Fedora 2</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | No          | 2.3.3        | 3.3.3      |
| <b>SuSE 11.1</b>         | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.9          | 4.3.3      |
| <b>SuSE 11.0</b>         | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.8          | 4.3.1      |
| <b>SuSE 10.3</b>         | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.6.1        | 4.2.1      |
| <b>SuSE 10.2</b>         | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.5          | 4.1.0      |
| <b>SuSE 10.1</b>         | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.4          | 4.1.0      |
| <b>SuSE 10.0</b>         | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.3.5        | 4.0.2      |
| <b>SuSE 9.3</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.3.4        | 3.3.5      |
| <b>SuSE 9.2</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.3.3        | 3.3.4      |
| <b>SLES 11</b>           | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.9          | 4.3.3      |
| <b>SLES 10</b>           | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.4          | 4.1.0      |
| <b>SLES 9</b>            | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | No          | Yes         | 2.3.3        | 3.3.3      |
| <b>SuSE 9.1</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | No          | 2.3.3        | 3.3.3      |
| <b>SuSE 9.0</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | No          | No          | 2.3.2        | 3.3.1      |
| <b>SuSE 8.2</b>          | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | No          | No          | 2.3.2        | 3.3        |
| <b>RedHat 9.0</b>        | <b>Linux</b>    | No            | No        | Yes          | Yes          | Yes         | No          | 2.3.2        | 3.2.2      |
| <b>Ubuntu 9.04</b>       | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.9          | 4.3.3      |
| <b>Ubuntu 8.10</b>       | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.8          | 4.3.2      |
| <b>Ubuntu 8.04</b>       | <b>Linux</b>    | Yes           | Yes       | Yes          | Yes          | Yes         | Yes         | 2.7          | 4.2.1      |
| <b>Microsoft Windows</b> | <b>XP</b>       | No            | Yes       | Yes          | Yes          | NA          | Yes         | NA           | NA         |
|                          | <b>XP x64</b>   | Yes           | Yes       | Yes          | Yes          | NA          | Yes         | NA           | NA         |
|                          | <b>2003</b>     | No            | No        | Yes          | Yes          | NA          | Yes         | NA           | NA         |
|                          | <b>2003 x64</b> | Yes           | Yes       | Yes          | Yes          | NA          | Yes         | NA           | NA         |
|                          | <b>2008</b>     | No            | Yes       | Yes          | Yes          | NA          | Yes         | NA           | NA         |
|                          | <b>2008 x64</b> | Yes           | Yes       | Yes          | Yes          | NA          | Yes         | NA           | NA         |
|                          | <b>SFU</b>      | No            | Yes       | Yes          | Yes          | NA          | Yes         | SFU          | 3.3        |
|                          | <b>SUA</b>      | No            | Yes       | Yes          | Yes          | NA          | Yes         | SUA          | 3.3        |
|                          | <b>SUA x64</b>  | Yes           | Yes       | Yes          | Yes          | NA          | Yes         | SUA          | 3.3        |
| <b>Vista</b>             | No              | Yes           | Yes       | Yes          | NA           | Yes         | NA          | NA           |            |
| <b>Vista x86</b>         | Yes             | Yes           | Yes       | Yes          | NA           | Yes         | NA          | NA           |            |

### 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>   |
|-----------------------|------------------------|---------------|------------|--------------|--------------|-------------|-------------|--------------|--------------|
|                       | <b>HPC Server 2008</b> | <i>Yes</i>    | <i>Yes</i> | <i>Yes</i>   | <i>Yes</i>   | <i>NA</i>   | <i>Yes</i>  | <i>NA</i>    | <i>NA</i>    |
| <i>Apple Mac OS X</i> | <i>Tiger</i>           | <i>No</i>     | <i>No</i>  | <i>Yes</i>   | <i>No</i>    | <i>NA</i>   | <i>NA</i>   | <i>NA</i>    | <i>4.0.1</i> |
|                       | <i>Leopard</i>         | <i>Yes</i>    | <i>No</i>  | <i>Yes</i>   | <i>Yes</i>   | <i>NA</i>   | <i>NA</i>   | <i>NA</i>    | <i>4.0.1</i> |

**Note:** <http://www.pgroup.com/support/install.htm> lists any new Linux, Apple or Windows 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`.

**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 original 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 included 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* and *-fastsse*, the *-Mipa=fast* option for inter-procedural analysis and optimization can improve performance. You may also 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*, *-Mpfi/-Mpfo*, and so on. However, increased speeds using these options are typically application- and system-dependent. 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 Core i7 (Nehalem) processor, 32-bit mode |
| nehalem-32  | Intel Core i7 (Nehalem) processor, 32-bit mode |
| nehalem-64  | Intel Core i7 (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:
 

|                 |   |
|-----------------|---|
| <i>analysis</i> | Perform loop analysis only. Do not generate code. |
| <i>cc10</i>     | Generate code for compute capability 1.0.         |
| <i>cc11</i>     | Generate code for compute capability 1.1.         |
| <i>cc13</i>     | Generate code for compute capability 1.3.         |
| <i>nofma</i>    | Do not generate fused- multiply-add instructions. |
| <i>time</i>     | 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 PGI Accelerator Compilers Runtime Libraries on page 34.

## 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 Environment Variables on page 35.

## 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 symbols and source lines in C/C++ include files, Fortran `INCLUDE` files, 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, MPI support 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, MVAPICH, or OpenMPI. This section describes how to use these capabilities and some of their limitations. The Using MPI chapter of the PGI User's Guide provides details and examples for MPICH-1, MPICH-2, MVAPICH, HP\_MPI on Linux, and MSMPI. The following section provides information on OpenMPI.

## 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.** MPI profiling is included automatically when you build with `-Mprof=time|lines|func|hwcts`.

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:

|                                      |                                      |
|--------------------------------------|--------------------------------------|
| <code>mpicc-wrapper-data.txt</code>  | <code>mpic++-wrapper-data.txt</code> |
| <code>mpif77-wrapper-data.txt</code> | <code>mpif90-wrapper-data.txt</code> |

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  
-lnsl -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 -lnsl -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}
```

## 3.11.5 Instructions for Apple OS X

This section contains instructions on how to install and configure your system for OpenMPI profiling on Apple OS X. In release 9.0-1, OpenMPI profiling is only supported on 32-bit systems (osx86).

On Apple OS X, there is no need to install or configure OpenMPI, PGI Workstation for Apple includes a pre-configured version of OpenMPI.

1. **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=lines|func`, MPI profiling is included automatically.
2. **Run your program:** Run as you normally would. One or more files named `pgprof.out` is created in your working directory.
3. **Run the profiler:** Invoke the profiler to see the results of your profiling run.

```
pgprof -exe your_program
```

For information on how to start an OpenMPI debugging session with PGDBG on Mac OS X 10.5 (Leopard), refer to Platform-specific Issues on page 48.

## 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*.



# 4 PGI Workstation 9.0

---

This chapter describes the updates and changes to PGI Workstation 9.0 that are specific to Linux, Windows, and Mac OS X, such as using the module load command on Linux.

## 4.1 PGI Workstation 9.0 for Linux

### 4.1.1 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 Windows 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 | libXext  |        |

## 4.2 PGI Workstation 9.0 for Windows, SFU, and SUA

*PGI Workstation 9.0 for Windows* supports most of the features of the 32- and 64-bit versions for *linux86* and *linux86-64* environments.

*PGI Workstation 9.0 for Windows*, during the point-and-click installation, now supports automatic license generation from [www.pgroup.com](http://www.pgroup.com), and license server setup.

## 4.3 PGI Workstation 9.0 for Mac OS X

*PGI Workstation 9.0* for *Mac OS X* supports most of the features of the 32- and 64-bit versions for *linux86* and *linux86-64* environments. Except where noted in these release notes or the user manuals, the PGI compilers and tools on *Mac OS X* function identically to their *Linux* counterparts.

# 5

# PGI Accelerator

---

An accelerator is a special-purpose co-processor attached to a CPU and to which the CPU can offload data and executable kernels to perform compute-intensive calculations. This chapter describes the new PGI Accelerator compilers, including the collection of compiler directives used to specify regions of code in Fortran and C programs that can be offloaded from a *host* CPU to an attached *accelerator*.

**Note.** For more information and more details about the PGI Accelerator compilers, the programming model and directives, refer to Chapter 7, *Using an Accelerator* and Chapter 18, *PGI Accelerator Compilers Reference*, in the PGI User's Guide.

## 5.1 Components

The PGI Accelerator compiler technology includes the following components:

- PGF95 auto-parallelizing accelerator-enabled Fortran 90/95 compiler.
- PGCC auto-parallelizing accelerator-enabled ANSI C99 and K&R C compiler.
- A simple command-line tool to detect whether the system has an appropriate GPU or accelerator card.

No accelerator-enabled debugger or profiler is included with this release.

## 5.2 Availability

The PGI 9.0 Fortran & C Accelerator compilers are available only on x64 processor-based workstations and servers running 64-bit Linux operating systems, with an attached NVIDIA CUDA-enabled GPU or Tesla card. These compilers target only a limited number and type of x64+GPU platforms. All examples included in this chapter are developed and

presented on such a platform. For a list of supported GPUs, refer to section 2.2, Supported Processors, on page 6.

## 5.3 User-directed Accelerator Programming

In user-directed accelerator programming the user specifies the regions of a host program to be targeted for offloading to an accelerator device. The bulk of a user's program, as well as regions containing constructs that are not supported on the targeted accelerator, are executed on the host.

## 5.4 Features Not Covered or Implemented

Currently the PGI Accelerator compilers do not include features for automatic detection and offloading of regions of code to an accelerator by a compiler or other tool, nor does it cover targeting of accelerator regions to multiple accelerators attached to a single host. While future versions of the PGI compilers may allow for automatic offloading, multiple accelerators of the same type, or multiple accelerators of different types, these features are not currently supported.

## 5.5 System Requirements

To use the PGI Accelerator compiler features, you must install the NVIDIA CUDA components for 64-bit Linux:

- NVIDIA Driver
- CUDA Toolkit
- CUDA SDK

You may download these components from the NVIDIA website at [www.nvidia.com/cuda](http://www.nvidia.com/cuda). These are not PGI products, and are licensed and supported by NVIDIA.

Further, you must be using a 64-bit Linux operating system that is supported by both the current PGI release and by the CUDA software and drivers.

## 5.5.1 Supported Processors and GPUs

This PGI Accelerator compiler release supports all AMD64 and Intel 64 host processors supported by Release 9.0 or higher of the PGI compilers and tools. Further, you can use the `-tp <target>` flag as documented in that release.

You can also use the `-ta=nvidia` flag to enable the accelerator directives and target the NVIDIA GPU. You can then use the generated code on any system with CUDA installed that has a CUDA-enabled GeForce, Quadro, or Tesla card.

For more information on these flags as they relate to accelerator technology, refer to the PGI User's Guide. For a complete list of supported GPUs, refer to the NVIDIA website at:  
[www.nvidia.com/object/cuda\\_learn\\_products.html](http://www.nvidia.com/object/cuda_learn_products.html)

## 5.6 Installation and Licensing

The PGI Accelerator compilers require a separate license key in addition to a normal PGI Workstation, Server, or CDK license key. For specific information related to installation, refer to the PGI Workstation Installation Guide.

## 5.7 Running an Accelerator Program

Running a program that has accelerator directives and was compiled and linked with the `-ta=nvidia` flag is the same as running the program compiled without the `-ta=nvidia` flag. The program looks for and dynamically loads the CUDA libraries. If the libraries are not available, or if they are in a different directory than they were when the program was compiled, you may need to append the CUDA library directory to your `LD_LIBRARY_PATH` environment variable.

When your program reaches its first accelerator region, there is a 0.5 to 1.5 second pause, which is the cost to acquire a handle to the GPU and allocate static resources. After that, there is no overhead to execute accelerator regions.

If you run an accelerated program on a system without a CUDA-enabled NVIDIA GPU, or without the CUDA software installed in a directory where the runtime library can find it, the program fails at runtime with an error message.

If you set the environment variable `ACC_NOTIFY` to a nonzero integer value, the runtime library prints a line to standard error every time it launches a kernel..

## 5.7.1 PGI Accelerator Compilers Runtime Libraries

PGI Accelerator Compilers provide user-callable functions and library routines that are available for use by programmers to query the accelerator features and to control behavior of accelerator-enabled programs at runtime. In Fortran, none of the PGI Accelerator compilers runtime library routines may be called from a `PURE` or `ELEMENTAL` procedure.

To access accelerator libraries, you must *link* an accelerator program with the `-ta` flag described on page 35.

There are separate runtime library files for C and for Fortran.

- **C Runtime Library Files** - In C, prototypes for the runtime library routines are available in a header file named `accel.h`. All the library routines are extern functions with “C” linkage. This file defines:
  - The prototypes of all routines in this section.
  - Any data types used in those prototypes, including an enumeration type to describe types of accelerators.
- **Fortran Runtime Library Files** - In Fortran, interface declarations are provided in a Fortran include file named `accel_lib.h` and in a Fortran module named `accel_lib`. These files define:
  - Interfaces for all routines in this section chapter.
  - Integer parameters to define integer kinds for arguments to those routines.
  - Integer parameters to describe types of accelerators.  
The integer parameter `accel_version` with a value `yyyymm` where `yyyy` and `mm` are the year and month designations of the version of the Accelerator programming model supported. This value matches the value of the preprocessor variable `_ACCEL`.

The following list briefly describes the supported PGI Accelerator compilers runtime library routines that PGI currently supports. For a complete description of these routines, refer to “PGI Accelerator Runtime Routines” section of the PGI User’s Guide.

- **`acc_get_device`** - returns the type of accelerator device being used.
- **`acc_get_num_devices`** - returns the number of accelerator devices of the given type attached to the host.

- **acc\_init** - connects to and initializes the accelerator device and allocates control structures in the accelerator library.
- **acc\_set\_device\_num** - tells the runtime which device to use among those attached of the given type.

## 5.7.2 Environment Variables

PGI supports environment variables that modify the behavior of accelerator regions. This section defines the user-setable environment variables used to control behavior of accelerator-enabled programs at execution. These environment variables must comply with these rules:

- The names of the environment variables must be upper case.
- The values assigned environment variables are case insensitive and may have leading and trailing white space.
- The behavior is implementation-defined if the values of the environment variables change after the program has started, even if the program itself modifies the values.

The following list briefly describes the Accelerator environment variables that PGI supports. For more information on these variables, refer to the PGI User's Guide.

- **ACC\_DEVICE** - controls which accelerator device to use when executing PGI Unified Binary for accelerators. The value of this environment variable may be the string NVIDIA or HOST
- **ACC\_DEVICE\_NUM** - controls the default device number to use when executing accelerator regions. The value of this environment variable must be a nonnegative integer between zero and the number of devices attached to the host.
- **ACC\_NOTIFY** - when set to a non-negative integer, indicates to print a message for each kernel launched on the device.

## 5.7.3 Applicable Command Line Options

There are command line options that apply specifically when working with accelerators.

- **-tp** - use this option to specify the target host processor architecture.
- **-Minfo** or **-Minfo=accel** - use this option to see messages about the success or failure of the compiler in translating the accelerator region into GPU kernels.

- `-ta=nvidia(,nvidia_suboptions),host` - enables 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.
```

The compiler automatically invokes the necessary CUDA software tools to create the kernel code and embeds the kernels in the Linux object file.

To access accelerator libraries, you must *link* an accelerator program with the `-ta` flag.

## 5.8 PGI Unified Binary for Accelerators

PGI compilers support the PGI Unified Binary feature to generate executables with functions optimized for different host processors, all packed into a single binary. This release extends the PGI Unified Binary technology for accelerators. Specifically, you can generate a single binary that includes two versions of functions:

- one version is optimized for the accelerator
- other version runs on the host processor when the accelerator is not available or when you want to compare host to accelerator execution.

To enable this feature, use the extended `-ta` flag: `-ta=nvidia,host`. This flag tells the compiler to generate two versions of functions that have valid accelerator regions.

- A compiled version that targets the accelerator.
- A compiled version that ignores the accelerator directives and targets the host processor.

- If you use the `-Minfo` flag, you get messages similar to the following:

```

s1:
    12, PGI Unified Binary version for
    -tp=barcelona-64 -ta=host
    18, Generated an alternate loop for the inner
loop
    Generated vector sse code for inner loop
    Generated 1 prefetch instructions for this
loop
s1:
    12, PGI Unified Binary version for
    -tp=barcelona-64 -ta=nvidia
    15, Generating copy(b(:,2:90))
    Generating copyin(a(:,2:90))
    16, Loop is parallelizable
    18, Loop is parallelizable
    Parallelization requires privatization of
array t(2:90)
    Accelerator kernel generated
    16, !$acc do parallel
    18, !$acc do parallel, vector(256)
    Using register for t

```

The PGI Unified Binary message shows that two versions of the subprogram `s1` were generated:

- one for no accelerator (`-ta=host`)
- one for the NVIDIA GPU (`-ta=nvidia`)

At run time, the program tries to load the NVIDIA CUDA dynamic libraries and test for the presence of a GPU. If the libraries are not available or no GPU is found, the program runs the host version.

You can also set an environment variable to tell the program to run on the NVIDIA GPU. To do this, set `ACC_DEVICE` to the value `NVIDIA` or `nvidia`. Any other value of the environment variable causes the program to use the host version.

The only supported `-ta` targets for this release are `nvidia` and `host`.

## 5.9 Multiple Processor Targets

You can use the `-tp` flag with multiple processor targets along with the `-ta` flag. You will see the following behavior:

- If you specify one `-tp` value and one `-ta` value:  
You see one version of each subprogram generated for that specific target processor and target accelerator.
- If you specify one `-tp` value and multiple `-ta` values:  
The compiler generates two versions of subprograms that contain accelerator regions for the specified target processor and each target accelerator.
- If you specify multiple `-tp` values and one `-ta` value:  
If 2 or more `-tp` values are given, the compiler generates up to that many versions of each subprogram, for each target processor, and each version also targets the selected accelerator.
- If you specify multiple `-tp` values and multiple `-ta` values:  
With 'N' `-tp` values and two `-ta` values, the compiler generates up to N+1 versions of the subprogram. It first generates up to N versions, for each `-tp` value, ignoring the accelerator regions, equivalent to `-ta=host`. It then generates one additional version with the accelerator target.

## 5.10 Profiling Accelerator Kernels

This release supports the command line option:

```
-ta=nvidia,time
```

The `time` suboption links in a timer library, which collects and prints out simple timing information about the accelerator regions and generated kernels.

### Sample Accelerator Kernel Timing Data

```
Accelerator Kernel Timing data
/proj/qa/tests/accel/bb04.f90
s1
  15: region entered 1 times
      time(us): total=1490738
                  init=1489138 region=1600
                  kernels=155 data=1445
      w/o init: total=1600 max=1600
```

```
min=1600 avg=1600
18: kernel launched 1 times
time(us): total=155 max=155 min=155 avg=155
```

In this example, a number of things are occurring:

- For each accelerator region, the file name `/proj/qa/tests/accel/bb04.f90` and subroutine or function name `s1` is printed, with the line number of the accelerator region, which in the example is **15**.
- The library counts how many times the region is entered (1 in the example) and the microseconds spent in the region (in this example 1490738), which is split into initialization time (in this example 1489138) and execution time (in this example 1600).
- The execution time is then divided into kernel execution time and data transfer time between the host and GPU.
- For each kernel, the line number is given, (18 in the example), along with a count of kernel launches, and the total, maximum, minimum, and average time spent in the kernel, all of which are 155 in this example.

## 5.11 Supported Intrinsic

PGI Accelerator compilers support Fortran and C intrinsics. For complete descriptions of these intrinsics, refer to the “Supported Intrinsic” section of the Using an Accelerator chapter of the PGI User’s Guide.

In addition to the ones listed in the Pgi User’s Guide, PGI plans to support additional intrinsics in future releases.



# 6

# 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, 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*.

## 6.1 Application Deployment and Redistributables

Programs built with PGI compilers 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.

## 6.1.1 PGI Redistributables

The PGI 9.0 release includes these directories:

- \$PGI/linux86/9.0/REDIST
- \$PGI/linux86-64/9.0/REDIST
- \$PGI/win64/9.0-0/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.

## 6.1.2 Linux Redistributables

The Linux 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 will 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.

## 6.1.3 Microsoft Redistributables

The PGI products on Windows include Microsoft Open Tools. The Microsoft Open Tools directory contains a subdirectory named “redist”. PGI 9.0 licensees may redistribute the files contained in this directory in accordance with the terms of the PGI End-User License Agreement.

# 7 The PGI Windows CDK

---

If you have a PGI Windows CDK (Cluster Development Kit) license, then your PGI software includes support for working with Microsoft Compute Cluster Server and MSMPI. Specifically, this software includes support for these things:

- Building MPI applications with MSMPI
- Using PGPROF to do MPI profiling of MSMPI applications
- Using PGDBG to do MPI debugging of MSMPI applications
- This chapter provides information on these tasks.

## 7.1 Build MPI Applications with MSMPI

*Note.* For the options `-Mprof=msmpi` and `-Mmpi=msmpi` to work properly, the `CCP_HOME` environment variable must be set. This variable is typically set when the Microsoft Compute Cluster SDK is installed.

### 7.1.1 Using MSMPI libraries

To build an application using the MSMPI libraries, use the option `-Mmpi=msmpi`. This option inserts options into the compile and link lines to pick up the MSMPI headers and libraries.

### 7.1.2 Generate MPI Profile Data

To build an application that generates MPI profile data, use the `-Mprof=msmpi` option. This option performs MPICH-style profiling for Microsoft MSMPI. For Microsoft Compute Cluster Server only, using this option implies `-Mmpi=msmpi`.

The profile data generated by running an application built with the option `-Mprof=msmpi` contains information about the number of sends and receives, as well as the number of bytes sent and received, correlated with the source location associated with the sends and receives. You must use `-Mprof=msmpi` in conjunction with either the option `-Mprof=func` or `-Mprof=lines`. When invoked using this type of profile data, PGPROF automatically displays MPI statistics.

## 7.2 Debug MSMPI Applications with PGDBG

To invoke the PGDBG debugger to debug an MSMPI application, use the `pgdbg -mpi` option:

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

The location of `mpiexec` should be part of your `PATH` environment variable. Otherwise, you should specify the pathname for `mpiexec`, or another similar launcher, as `<path>` in `-mpi[:<path>]`.

To start a distributed debugging session, you must use the `job submit` command on the command line, as illustrated in the example that follows. You must also ensure that the debugger has access to the `pgserv.exe` remote debug agent on all nodes of the cluster used for the debug session.

To make `pgserv.exe` available, copy it from the PGI installation directory, such as `C:\Program Files\PGI\win64\9.0-0\bin\pgserv.exe`, into a directory or directories such that the path to `pgserv.exe` is the same on all nodes in the debug session. Then you can start the debug session as follows:

```
$ pgdbg -pgserv:<path_to_pgserv.exe>  
-mpi[:<job submit command>]
```

If you use a command similar to the following one, it copies `pgserv.exe` to the current directory and also sets the path to `pgserv.exe` to this copy.

## 7.2.1 Bash Shell Example

Suppose you wanted to debug the following job invoked in a bash shell:

```
PGI$ "job.cmd" submit /numprocessors:4  
/workdir:\\\\cce-head\d\srt /stdout:sendrecv.out  
mpiexec sendrecv.exe
```

You use this command:

```
$ pgdbg -pgserv "-mpi:c:\Program Files\Microsoft  
Compute Cluster Pack\Bin\job.cmd" submit  
/numprocessors:4 /workdir:\\cce-head\d\srt  
/stdout:sendrecv.out mpiexec sendrecv.exe
```

**Important.** For this command to execute properly, a copy of `pgserv.exe` must be located in `\\cce-head\d\srt`.

Since the CCP installation updates the default PATH, the following command is equivalent to the previous one:

```
$ pgdbg -pgserv -mpi:job.cmd submit  
/numprocessors:4 /workdir:\\cce-head\d\srt  
/stdout:sendrecv.out mpiexec sendrecv.exe
```

**Note.** The use of quotes around the `-mpi` option varies, depending on the type of shell you are using. In the example, or if you are using `cmd`, specify the option as `"-mpi:..."`, including the quotes around the option as well as around the optional `job submit` command. When invoking in a Cygwin bash shell, you can specify the `-mpi` option as `-mpi:"..."`, using the quotes around only the `job submit` command.

## 7.2.2 DOS Shell Example

Suppose you wanted to debug the following job invoked in a DOS shell:

```
DOS> job submit /numprocessors:4 /workdir:\\cce-  
head\d\srt /stdout:sendrecv.out mpiexec sendrecv.exe
```

You use this command:

```
$ pgdbg -pgserv "-mpi:c:\Program Files\Microsoft  
Compute Cluster Pack\Bin\job.cmd" submit  
/numprocessors:4 /workdir:\\cce-head\d\srt  
/stdout:sendrecv.out mpiexec sendrecv.exe
```



# 8 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.

## 8.1 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](http://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 *-Mpf* and *-mp* together is not supported. The *-Mpf* 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 *-Mpfo* flag does not disable *OpenMP* processing.

## 8.2 Platform-specific Issues

### 8.2.1 Linux

- 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.

### 8.2.2 Apple Mac OS X

- On Apple Mac OS platform, the *PGI Workstation 9.0* compilers do not support static linking of user binaries. For compatibility with future Apple updates, the compilers support dynamic linking of user binaries.
- Using *-Mprof=func* or *-Mprof=lines* on *osx86-64*, both Tiger and Leopard is not supported.

- OpenMPI, PGDBG, and PGPROF are supported on Mac OS X 10.5 (Leopard). OpenMPI, PGDBG, and PGPROF are not supported on Mac OS X 10.4 (Tiger).
- To begin an OpenMPI debugging session with PGDBG on Mac OS X 10.5 (Leopard), use the following steps:
  1. Invoke the debugger using the full pathname of the executable. For example, you might use a command similar to this one:
 

```
pgdbg -mpi:mpirun -np 4 /home/user1/a.out
```
  2. Set a breakpoint on `main`.
  3. Continue to the breakpoint.
  4. Begin your debugging session.

## 8.2.3 Windows

The following are known issues on Windows:

- On Windows, the version of `vi` included in Cygwin can have problems when the `SHELL` variable is defined to something it does not expect. In this case, the following messages appear when `vi` is invoked:
 

```
E79: Cannot expand wildcards
Hit ENTER or type command to continue
```

 To workaroud this problem, set `SHELL` to refer to a shell in the cygwin bin directory, e.g. `/bin/bash`.
- On Windows, runtime libraries built for debugging (e.g. `msvcrtd` and `libcmtd`) are not included with *PGI Workstation*. When a program is linked with `-g`, for debugging, the standard non-debug versions of both the PGI runtime libraries and the Microsoft runtime libraries are always used. This limitation does not affect debugging of application code.
- Dynamic Link Libraries (DLLs) built on the Microsoft Windows platform by the PGI Workstation 9.0 compilers have the following known limitations:
  - DLLs cannot be produced with the *PGI Workstation C++* compiler.
  - If a DLL is built with the *PGI Workstation* compilers, the runtime DLLs must be used. The compiler option `-Mmakedll` ensures the correct runtime libraries are used.

- If an executable is linked with any *PGI Workstation*-compiled DLL, the *PGI Workstation* runtime library DLLs must be used; this means the static libraries, which are used by default, cannot be used. To accomplish this, use the compiler option *-Bdynamic* when creating the executable.

These are known issues on Windows and PGDBG:

- In *PGDBG* on the *Windows* platform, use the forward slash (/) character to delimit directory names in file path names.  
*Note.* This requirement does not apply to the *PGDBG debug* command or to target executable names on the command line, although this convention will work with those commands.
- In *PGDBG* on the *Windows* platform, Windows times out *stepi/nexti* operations when single stepping over blocked system calls. For more information on the workaround for this issue, refer to the online FAQs at [www.pgroup.com/support/tools.htm](http://www.pgroup.com/support/tools.htm).
- *On SFU/SUA* – Due operating system limitations, *PGDBG* does not support hardware watchpoints, that is, the "hwatch" command, on *SFU/SUA* systems.
- *On SFU/SUA* – Due to operating system limitations, *PGDBG* supports multi-thread debugging only with 32-bit *SUA* programs, with one restriction: once stopped, the process may be continued by continuing all threads, or a single thread, but not a partial set of the threads. Attempts to continue a partial set of threads results in the entire process, all threads, being continued.

These are known issues on Windows and PGDBG:

- Do not use *-Mprof* with *PGI Workstation* runtime library DLLs. To build an executable for profiling, use the static libraries. When the compiler option *-Bdynamic* is *not* used, the static libraries are the default.

## 8.3 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](http://www.pgroup.com/support/tools.htm).

## 8.4 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.

## 8.5 Corrections

A number of problems have been corrected in the *PGI Workstation 9.0* release. Most were reported in previous releases of *PGI Workstation*. Problems found in *PGI Workstation 8.1* may not have occurred in previous releases.

Refer to [www.pgroup.com/support/release\\_tprs.htm](http://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.



# 9 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](http://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 email or submissions using an online form at <http://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](http://www.pgroup.com/support/faq.htm).

Online documentation is available at [www.pgroup.com/resources/docs.htm](http://www.pgroup.com/resources/docs.htm) or in your local copy of the documentation in the release directory [doc/index.htm](#).