PGI® Server 8.0
PGI® Workstation 8.0

Release Notes

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

PGI Server 8.0 / PGI Workstation 8.0
Release Notes

First Printing Release 8.0-1 November 2008

Technical support: www.pgroup.com/support

Table of Contents

1 Introduction

Welcome to Release 8.0 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.

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.

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.

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

1.1 Product Overview

Release 8.0 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.

1.2 Terms and Definitions

These release notes contain a number of terms and definitions with which you may or may not be familiar. If you encounter a term in these notes with which you are not familiar, please refer to the online glossary at

www.pgroup.com/support/definitions.htm

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

AMD64 – a 64-bit processor from AMD designed to be binary compatible with 32-bit *x86* processors, and incorporating new features such as additional registers and 64-bit addressing support for improved performance and greatly increased memory range. This term includes the AMD Athlon64™, AMD Opteron™, AMD Turion™, AMD Barcelona, and AMD Shanghai processors.

Intel 64 – a 64-bit IA32 processor with *Extended Memory 64-bit Technology* extensions designed to be binary compatible with AMD64 processors. This includes Intel Pentium 4, Intel Xeon, and Intel Core 2 processors.

2 PGI Release 8.0 Overview

This document describes changes between previous releases and Release 8.0 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 8.0 Contents

Release 8.0 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 8.0 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 8.0 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 8.0

Brand	CPU	Cores	<target>	Memory Address	Floating Point HW					
					SSE1	SSE2	SSE3	SSSE3	SSE4	ABM and SSE4a
AMD	Opteron/Quadcore	4	shanghai-64	64-bit	Yes	Yes	Yes	No	No	Yes
AMD	Opteron/Quadcore	4	shanghai	64-bit	Yes	Yes	Yes	No	No	Yes
AMD	Opteron/Quadcore	4	barcelona-64	64-bit	Yes	Yes	Yes	No	No	Yes
AMD	Opteron/Quadcore	4	barcelona	32-bit	Yes	Yes	Yes	No	No	Yes
AMD	Opteron/Athlon64	2	k8-64	32-bit	Yes	Yes	Yes	No	No	No
AMD	Opteron/Athlon64	2	k8-32	32-bit	Yes	Yes	Yes	No	No	No
AMD	Opteron Rev E/F Turion /Athlon64	2	k8-64e	64-bit	Yes	Yes	Yes	No	No	No
AMD	Opteron Rev E/F	2	k8-32	32-bit	Yes	Yes	No	No	No	No
AMD	Turion64 Turion /Athlon64	1	k8-64e	64-bit	Yes	Yes	Yes	No	No	No
AMD	Turion64	1	k8-32	32-bit	Yes	Yes	No	No	No	No
Intel	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 8.0 of the PGI compilers and tools supports.

To determine if Release 8.0 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 *libpthreads* 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 2.

Operating Systems and Features Supported in PGI 8.0									
Distribution	***Type***	***64-bit***	***HT***	***pgC++***	***pgdbg***	***NPTL***	***NUMA***	***glibc***	***GCC***
RHEL 5.0	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*2.5*	*4.1.2*
RHEL 4.0	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*2.3.4*	*3.4.3*
RHEL 3.0	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*2.3.2*	*3.2.3*
Fedora 9	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.8*	*4.3.0*
Fedora 8	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.7*	*4.1.2*
Fedora 7	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.6*	*4.1.2*
Fedora 6	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.5*	*4.1.1*
Fedora 5	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.4*	*4.1.0*
Fedora 4	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*2.3.5*	*4.0.0*
Fedora 3	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*2.3.3*	*3.4.2*
Fedora 2	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*2.3.3*	*3.3.3*
SuSE 11	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.8*	*4.3.0*
SuSE 10.3	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.6.1*	*4.2.1*
SuSE 10.2	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.5*	*4.1.0*
SuSE 10.1	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.4*	*4.1.0*
SuSE 10.0	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.3.5*	*4.0.2*
SuSE 9.3	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.3.4*	*3.3.5*
SuSE 9.2	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.3.3*	*3.3.4*
SLES 10	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.4*	*4.1.0*
SLES 9	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*Yes*	*2.3.3*	*3.3.3*
SuSE 9.1	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*2.3.3*	*3.3.3*
SuSE 9.0	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*No*	*2.3.2*	*3.3.1*
SuSE 8.2	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*No*	*2.3.2*	*3.3*
RedHat 9.0	**Linux**	*No*	*No*	*Yes*	*Yes*	*Yes*	*No*	*2.3.2*	*3.2.2*
Ubuntu 8	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.7*	*4.2.1*
Microsoft Windows	**XP**	*No*	*Yes*	*Yes*	*Yes*	*NA*	*Yes*	*NA*	*NA*
	XP x64	*Yes*	*Yes*	*Yes*	*Yes*	*NA*	*Yes*	*NA*	*NA*
	2003	*No*	*No*	*Yes*	*Yes*	*NA*	*Yes*	*NA*	*NA*
	2003 x64	*Yes*	*Yes*	*Yes*	*Yes*	*NA*	*Yes*	*NA*	*NA*
	2008	*No*	*Yes*	*Yes*	*Yes*	*NA*	*Yes*	*NA*	*NA*
	2008 x64	*Yes*	*Yes*	*Yes*	*Yes*	*NA*	*Yes*	*NA*	*NA*
	SFU	*No*	*Yes*	*Yes*	*Yes*	*NA*	*Yes*	*SFU*	*3.3*
	SUA	*No*	*Yes*	*Yes*	*Yes*	*NA*	*Yes*	*SUA*	*3.3*
	SUA x64	*Yes*	*Yes*	*Yes*	*Yes*	*NA*	*Yes*	*SUA*	*3.3*
	Vista	*No*	*Yes*	*Yes*	*Yes*	*NA*	*Yes*	*NA*	*NA*
	Vista x86	*Yes*	*Yes*	*Yes*	*Yes*	*NA*	*Yes*	*NA*	*NA*
	HPC Server 2008	*Yes*	*Yes*	*Yes*	*Yes*	*NA*	*Yes*	*NA*	*NA*
Apple Mac OS X	Tiger	*No*	*No*	*Yes*	*Yes*	*NA*	*NA*	*NA*	*4.0.1*
	Leopard	*Yes*	*No*	*Yes*	*Yes*	*NA*	*NA*	*NA*	*4.0.1*

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

The following list contains the new features of Release 8.0 of the PGI compilers and tools as compared to prior releases.

- **OpenMP 3.0** parallel programming for multi-core x64 CPUs and multi-socket servers, including full support for OpenMP 3.0 and TASKs in PGF95 and PGCC®, , as described in Enhancements to OpenMP on page 14.
- **Common Compiler Feedback Format (CCFF)** sections are now generated by all PGI 8.0 compilers; CCFF is a standardized format for storing and reporting of optimization information and hints to users, described in Common Compiler Feedback Format (CCFF) on page 13.
- **New and Improved PGPROF** performance analysis and tuning tool.
  - All new look-and-feel with intuitive navigation and analysis features
  - Browse-able CCFF information correlated with source code
  - Improved multi-core scalability analysis using HW counters on Linux
  - Calculates Compute Intensity to identify regions of code suitable for Multi-core parallelization and/or offloading to a GPU/Accelerator
  - Improved performance of PGPROF timer-instrumented executables
- **Provisional x64+GPU Support** on 64-bit Linux for CUDA-enabled NVIDIA GPUs using the new high-level PGI Accelerator Compilers programming model
- **Compiler Optimizations and Features**
  - Computation and reporting of compute intensity of loops in all languages
  - Packed SSE code generation for unrolled loops
  - SSE vectorization of generalized reduction loops
  - Improved scalar prefetching, spill tuning and live range splitting
  - Improved static estimation of block execution frequencies
  - PGC++® STL is now thread safe; based on STLPort 4.6.2

- Incremental support for OpenMP 3.0 features in PGC++
- GCC variadic macro extensions
- Auto-generation of DWARF for improved tools interoperability
- Enhanced Fortran 95 DWARF generation

- **PGDBG Debugger Enhancements**
  - OpenMP 3.0 debugging, including support for tasks
  - PGDBG OpenMPI debugging on Linux clusters with the PGI CDK
  - PGDBG OpenMPI debugging on MacOS (a version of OpenMPI is now bundled with PGI Workstation 8.0 for MacOS)
  - Debug OpenMP & Auto-parallelized multi-core applications on Mac OS X.
  - Numerous PGDBG reliability and performance enhancements
- **Automatic licensing generation** from www.pgroup.com, and license server setup during the with point-and-click Windows installation.
- **Expanded Platform Support**
  - Support for the New Quad-Core AMD Opteron processor (AMD Shanghai)
  - PGI Unified Binary™ support for the latest AMD and Intel processors
  - Fedora 8, Fedora 9, SuSE 10.3, SuSE 11.0, and Ubuntu 8 Linux
  - Microsoft Vista 32-bit and 64-bit, Microsoft HPC Server 2008
  - Mac OS X Leopard for x86 32-bit and 64-bit
- **Updated Documentation** including the PGI Users Guide, PGI Tools Guide and PVF Users Guide.

3.1 Getting Started

By default, the PGI 8.0 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*.

3.2 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.

3.3 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 8.0:

- *–Mnodwarf* is a new switch that specified not to add DWARF debug information
- *–Mnofpapprox* is a new switch that specifies not to use low-precision fp approximation operations
- *–Minfo* has a number of new suboptions:

all	Implies a number of suboptions: *–Minfo* =accel,inline,ipa,loop,lre,mp,opt,par,unified,vect
accel	Enable Accelerator information
ccff	Append information, such as optimization info, to object file
ftn	Enable Fortran-specific information
hpf	Enable HPF-specific information
inline	Enable inliner information
lre	Enable LRE information
par	Enable parallelizer information
pfo	Enable profile feedback information
vect	Enable vectorizer information

- *–Mconcur* has two new suboptions: *allcores* and **bind**.

allcores	Use all available cores; specify at link time
bind	Bind threads to cores; specify at link time

- *–Mprof* has a new suboption:

[no]ccff	Enable [disable] CCFF information

- *–Msmartalloc* has a new suboption:

hugebss	Put the BSS section in huge pages

- *–Mvect* has a new suboption:

[no]short	Enable [disable] short vector operations.

–Mvect=short enables generation of packed SSE instructions for short vector operations that arise from scalar code outside of loops or within the body of a loop iteration.

- *–mp* has 2 new suboption:

allcores	Use all available cores; specify at link time
bind	Bind threads to cores; specify at link time

- *–Mneginfo* has a number of new suboptions:

all	Implies a number of suboptions: –Mneginfo=accel,inline,ipa,loop,lre,mp,opt,par,vect
accel	Enable Accelerator information
ftn	Enable Fortran-specific information
hpf	Enable HPF-specific information
inline	Enable inliner information
ipa	Enable IPA information
lre	Enable LRE information
mp	Enable OpenMP information
opt	Enable optimizer information
par	Enable parallelizer information
pfo	Enable profile feedback information
vect	Enable vectorizer information

- *–tp* has 3 new target cpu types:

shanghai	AMD Shanghai processor, 32-bit mode
shanghai-32	AMD Shanghai processor, 32-bit mode
shanghai-64	AMD Shanghai processor, 64-bit mode

- --pedantic is a new switch that prints warnings from included <system header files>

3.4 Common Compiler Feedback Format (CCFF)

Using the Common Compiler Feedback Format (CCFF), PGI compilers save information about how your program was optimized, or why a particular optimization was not made, in the executable file. To append this information to the object file, use the compiler option *–Minfo=ccff*. Using the compiler option *–Mprof=ccff* also appends this information.

The PGPROF performance profiler provides an interface for browsing CCFF compiler feedback and associating it with the performance of components of your program.

3.5 Enhancements to OpenMP

OpenMP is a specification for a set of compiler directives, an applications programming interface (API), and a set of environment variables that you can use to specify shared memory parallelism in FORTRAN and C/C++ programs. You may use OpenMP to obtain most of the parallel performance you can expect from your code, or to serve as a stepping stone to parallelizing an entire application with MPI.

PGI Workstation 8.0 supports OpenMP 3.0. There are a number of enhancements that are associated with this support.

Important. The C++ Standard Template library has been made thread-safe to the extent allowed in the STLport code. Simultaneous accesses to distinct containers are safe, simultaneous reads to shared containers are also safe. However, simultaneous writes to shared containers must be protected by *#pragma omp critical* sections. Users must compile and link with the –mp switch to get the thread-safe version of the template header files and library.

Note. For more information and examples related to the directives, pragmas, clauses, run-time library routines, environment variables, and other information in this section, refer to Chapter 5 of the PGI Workstation User's Guide: *Using OpenMP*.

3.5.1 New or Modified OpenMP Directives and Pragmas

Every part of an OpenMP program is part of a task. In this release, PGI supports the following directives and clauses that are part of OpenMP 3.0.

- The Task directive and omp task pragma, which define an explicit task.
- The Taskwait directive and omp taskwait pragma. Which specify a wait on the completion of child tasks generated since the beginning of the current task.
- The Collapse (n) clause, associated with the DO...END DO, PARALLEL DO, and PARALLEL WORKSHARE directives. This clause specifies how many loops are associated with a loop construct.

3.5.2 New or Modified OpenMP Run-time Library Routines

PGI Workstation 8.0 supports these new run-time library routines.

- **omp_get_ancestor_thread_num**
  This routine returns, for a given nested level of the current thread, the thread number of the ancestor.
- **omp_get_active_level**
  This routine returns the number of enclosing active parallel regions enclosing the task that contains the call. PGI currently supports only one level of active parallel regions, so the return value currently is 1.
- **omp_get_level**
  This routine returns the number of parallel regions enclosing the task that contains the call.
- **omp_get_team_size**
  This routine returns, for a given nested level of the current thread, the size of the thread team to which the ancestor belongs.
- **omp_set_schedule**
  This routine sets the value of the run_sched_var.
- **omp_get_schedule**
  This routine retrieves the value of the run_sched_var.

3.5.3 New or Modified OpenMP Environment Variables

PGI Workstation 8.0 supports these OpenMP environment variables.

- **OMP_MAX_ACTIVE_LEVELS**
  This variable currently has no effect. It typically enables (TRUE) or disables (FALSE) nested parallelism.
- **OMP_THREAD_LIMIT**
  This variable, whose default is 64, specifies the absolute maximum number of threads that can be used in a program.
- **OMP_STACKSIZE**
  Formally OMP_STACK_SIZE, this variable overrides the default stack size for a newly created thread Formally Omp_Stack_Size

3.6 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 8.0-1, the local MPI capability continues to expand. You can debug and profile MPI applications for MPICH-1 (using the included version of MPICH-1), HP-MPI for Linux, MPICH-2, or MVAPICH. This chapter describes how to use these capabilities and some of their limitations.

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

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

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

3.7 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 8.0

This chapter describes the updates and changes to PGI Workstation 8.0 that are specific to Linux, Windows, and Mac OS X, such as using the module load command on Linux.

4.1 PGI Workstation 8.0 for Linux

4.1.1 Provisional x64+GPU Support

PGI Workstation 8.0 for *Linux* supports provisional x64+GPU Support on 64-bit Linux for CUDA-enabled NVIDIA GPUs using the new high-level PGI Accelerator Compilers programming model.

4.1.2 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 X Windows support generally includes these libraries:

libXau	libXdmcp	libxcb
libX11	libXext	

4.2 PGI Workstation 8.0 for Windows, SFU, and SUA

PGI Workstation 8.0 for *Windows* supports most of the features of the 32- and 64-bit versions for *linux86* and *linux86-64* environments.

4.2.1 All-new Windows Licensing Installation

PGI Workstation 8.0 for Windows, during the point-and-click installation, now supports automatic license generation from www.pgroup.com, and license server setup.

4.3 PGI Workstation 8.0 for Mac OS X

PGI Workstation 8.0 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 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.*

5.1 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.

5.1.1 PGI Redistributables

The PGI 8.0 release includes these directories:

- $PGI/linux86/8.0/REDIST
- $PGI/linux86-64/8.0/REDIST
- $PGI/win64/8.0-6/REDIST
- $PGI/win32/8.0/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 8.0 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 8.0 directory.

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.

5.1.2 Microsoft Redistributables

The PGI products on Windows include Microsoft Open Tools. The Microsoft Open Tools directory contains a subdirectory named “redist”. PGI 8.0 licensees may redistribute the files contained in this directory in accordance with the terms of the PGI End-User License Agreement.

6 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.

6.1 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.

6.1.1 Using MSMPI libraries

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

6.1.2 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. `-Mprof=msmpi` must be used in conjunction with `-Mprof=func` or `-Mprof=lines`. When invoked using this type of profile data, PGPROF automatically displays MPI statistics.

6.2 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\8.0-6\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.

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

6.2.1 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.

6.2.2 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
```

7 Troubleshooting Tips and Known Limitations

This chapter contains information about known limitations, documentation errors, and corrections that have occurred to PGI Workstation 8.0.

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.

7.1 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 8.0* 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 8.0*.
- The *–i8* option can make programs incompatible with the bundled ACML library. Visit *developer.amd.com* to check for compatible libraries.
- The *–i8* option can make programs incompatible with MPI and ACML; use of any INTEGER*8 array size argument can cause failures with these libraries.
- Using *–Mipa=vestigial* in combination with *–Mipa=libopt* with *PGCC*, you may encounter unresolved references at link time. This problem is due to the erroneous removal of functions by the *vestigial* sub-option to *–Mipa*. You can work around this problem by listing specific sub-options to *–Mipa*, not including *vestigial.*

- *OpenMP* programs compiled using *–mp* and run on multiple processors of a *SuSE 9.0* system can run very slowly. These same executables deliver the expected performance and speed-up on similar hardware running *SuSE 9.1* and above.
- ACML 4.1-0 is built using the *–fastsse* compile/link option, which includes *–Mcache_align*. When linking with ACML using the *–lacml* option on 32-bit targets, all program units must be compiled with *–Mcache_align*, or an aggregate option such as *–fastsse,* which incorporates *–Mcache_align*. This process is not an issue on 64-bit targets where the stack is 16-byte aligned by default. The lower-performance, but fully portable, *libblas.a* and *liblapack.a* libraries can be used on CPUs that do not support SSE instructions.
- When compiling with *–fPIC* and linking with *–lacml*, you may get the message "error while loading shared libraries: libacml_mv.so: cannot open shared object file: No such file or directory." In this case, you must add *-lacml_mv* library to the link line.
- Using *-Mpfi* and *-mp* together is not supported. The *-Mpfi* flag will disable *-mp* at compile time, which can cause run-time errors in programs that depend on interpretation of OpenMP directives or pragmas. Programs that do not depend on OpenMP processing for correctness can still use profile feedback. The *-Mpfo* flag does not disable OpenMP processing.

7.2 Platform-specific Issues

7.2.1 Linux

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

7.2.2 Apple Mac OS X

- On Apple Mac OS platform, the *PGI Workstation 8.0* compilers do not support static linking of user binaries. For compatibility with future Apple updates, the compilers support dynamic linking of user binaries.

7.2.3 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 workaround 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 8.0 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.*
- *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.

7.3 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 introduces 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 `watch`ed local variable may cause missed events and/or false positive events. Local variables may be `watch`ed reliably if program scope does not leave the scope of the `watch`ed 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*.

7.4 PGPROF-related Issues

- Using *–Mprof=func*, *–mcmodel=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.

7.5 Corrections

A number of problems have been corrected in the *PGI Workstation 8.0* release. Most were reported in *PGI Workstation 7.1* or previous releases. Problems found in *PGI Workstation 7.1* may not have occurred in previous releases.

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

8 Contact Information and Documentation

You can contact The Portland Group at:

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

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

www.pgroup.com/userforum/index.php

Or contact us electronically using any of the following means:

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

All technical support is by 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*.

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