PGI® CDK™ 7.1
Cluster Development Kit®

Release Notes

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

PGI CDK 7.1 Cluster Development Kit
Installation & Release Notes

First Printing:	Release 7.1-2	November, 2007
Second Printing:	Release 7.1-3	December, 2007
Third Printing:	Release 7.1-4	January, 2008
Fourth Printing:	Release 7.1-5	February, 2008

Technical support: *www.pgroup.com*

Table of Contents

1 PGI Release 7.1 Introduction

Welcome to Release 7.1 of *PGI Cluster Development Kit*, or *PGI CDK*, a set of Fortran, C and C++ compilers and development tools for 32-bit and 64-bit *x86*-compatible processor-based workstations and servers running versions of the Linux* operating systems.

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

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

1.1 Product Overview

Release 7.1 of PGI CDK includes the following components:

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

- PGDBG graphical OpenMP/multi-thread symbolic debugger.

- *MPICH MPI libraries, version 1.2.7,* for both 32-bit and 64-bit development environments*. (Note:* 64-bit *linux86-64* MPI *messages* are limited to *<2GB* size each*).*
- *MPICH2 MPI libraries, version 1.0.5p3,* for both 32-bit and 64-bit development environments.
- *MVAPICH MPI libraries, version 0.9.9,* for both 32-bit and 64-bit development environments.
- TORQUE 1.2.0 resource management system, a continuation of the product known as *OpenPBS*, or Portable Batch System. See `www.supercluster.org/projects/torque` for more information.
- *ScaLAPACK* linear algebra math library for distributed-memory systems, including *BLACS* version 1.1- the Basic Linear Algebra Communication Subroutines) and *ScaLAPACK* version 1.7 for use with *MPICH* or *MPICH2* and the PGI compilers on Linux systems with a kernel revision of 2.4.20 or higher. This is provided in both *linux86* and *linux86-64* versions for AMD64 or EM64T CPU-based installations. (Note: *linux86-64* versions are limited).
- *FlexLM* license utilities.

The release contains the following documentation and tutorial materials:

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

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

1.2 Terms and Definitions

Following are definitions of terms used in the context of these release notes.

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. Includes the AMD* Athlon64* , AMD Opteron* and AMD Turion* processors.

Barcelona – In this document the Quad-Core AMD Opteron(TM) Processor (i.e. Opteron Rev x10) is referred to as Barcelona.

driver – the compiler *driver* controls the compiler, linker, and assembler and adds objects and libraries to create an executable. The *–dryrun* option illustrates operation of the driver. `pgf77`, `pgf95`, `pghpf`, `pgcc`, and pgCC are drivers for the PGI compilers. A `pgf90` driver is retained for compatibility with existing makefiles, even though `pgf90` and `pgf95` drivers are identical.

Dual-, Quad-, or Multi-core – some x64 CPUs incorporate two or four complete processor cores (functional units, registers, level 1 cache, level 2 cache, etc) on a single silicon die. These are referred to as Dual-core or Quad-core (in general, Multi-core) processors. For purposes of OpenMP or auto-parallel threads, or MPI process parallelism, these cores function as distinct processors. However, the processing cores are on a single chip occupying a single socket on the system motherboard. In PGI 7.1, there are no longer software licensing limits on OpenMP threads for Multi-core.

EM64T – 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.

Hyperthreading (HT) – some IA32 CPUs incorporate extra registers that allow 2 threads to run on a single CPU with improved performance for some tasks. This is called *hyperthreading*, and abbreviated *HT*. Some *linux86* and *linux86-64* environments treat IA32 CPUs with HT as though there were a 2nd *pseudo* CPU, even though there is only one physical CPU. Unless the Linux kernel is *hyperthread-aware*, the second thread of an *OpenMP* program will be assigned to the *pseudo* CPU, rather than a real second physical processor (if one exists in the system). *OpenMP* Programs can run very slowly if the second thread is not properly assigned.

IA32 – an Intel Architecture 32-bit processor designed to be binary compatible with *x86* processors, but incorporating new features such as streaming SIMD extensions (SSE) for improved performance. This includes Intel Pentium 4, Intel Xeon, and Intel Core 2 processors. For simplicity, these release notes refer to *x86* and *IA32* processors collectively as *32-bit x86* processors.

Large Arrays – arrays with aggregate size larger than 2GB**,** which require 64-bit index arithmetic for accesses to elements of arrays. Program units that use *Large Arrays* must be compiled using *–mcmodel=medium*. If *–mcmodel=medium* is not specified, but *–Mlarge_arrays* is specified, the default small memory model is used but all index arithmetic is performed in 64-bits. This mode of execution can be a useful for certain existing 64-bit applications that use the small memory model but allocate and manage a single contiguous data space larger than 2GB.

linux86 – 32-bit Linux operating system running on an *x86, AMD64* or *EM64T* processor-based system, with 32-bit GNU tools, utilities and libraries used by the PGI compilers to assemble and link for 32-bit execution.

linux86-64 – 64-bit Linux operating system running on an *AMD64* or *EM64T* processor-based system, with 64-bit and 32-bit GNU tools, utilities and libraries used by the PGI compilers to assemble and link for execution in either *linux86* or *linux86-64* environments. The 32-bit development tools and execution environment under *linux86-64* are considered a cross development environment for *x86* processor-based applications.

–mcmodel=small – compiler/linker switch to produce *small memory model* format objects/executables in which both code (*.text*) and data *(.bss*) sections are limited to less than 2GB. This format is the default and only possible format for *linux86* 32-bit executables. This format is the default for *linux86-64* executables. Maximum address offset range is 32-bits, and total memory used for OS+Code+Data must be less than 2GB.

–mcmodel=medium – compiler/linker switch to produce *medium memory model* format objects/executables in which code sections are limited to less than 2GB, but data sections can be greater than 2GB. Supported *only* in *linux86-64* environments. This option must be used to *compile* any program unit that will be linked in to a 64-bit executable that will use aggregate data sets larger than 2GB and that will access data requiring address offsets greater than 2GB. This option must be used to *link* any

64-bit executable that will use aggregate data sets greater than 2GB in size. Executables linked using *–mcmodel=medium* can incorporate objects compiled using *–mcmodel=small* as long as the *small* objects are from a shared library.

NUMA – Non-Uniform Memory Access. A type of multi-processor system architecture in which the memory latency from a given processor to a given portion of memory can vary, resulting in the possibility for compiler or programming optimizations to ensure frequently accessed data is "close" to a given processor as determined by memory latency.

OFED – OpenFabrics Enterprise Distribution is the release mechanism for the OpenFabrics software packages.

SFU – Windows Services for Unix is the precursor to *SUA*. *SFU* supports 32-bit applications on *Windows 2000*, *Windows Server 2003*, and *XP*.

Shared library – a Linux library of the form *libxxx.so* containing objects that are dynamically linked into a program at the time of execution.

SSE – collectively, all SSE extensions supported by the PGI compilers.

SSE1 – 32-bit IEEE 754 FPU and associated *streaming SIMD extensions* (SSE) instructions on Pentium III, AthlonXP* and later 32-bit *x86*, *AMD64* and *EM64T* compatible CPUs, enabling scalar and packed vector arithmetic on single-precision floating-point data.

SSE2 – 64-bit IEEE 754 FPU and associated SSE instructions on P4/Xeon and later 32-bit *x86*, *AMD64* and *EM64T* compatible CPUs. SSE2 enables scalar and packed vector arithmetic on double-precision floating-point data.

SSE3 – additional 32-bit and 64-bit SSE instructions to enable more efficient support of arithmetic on complex floating-point data on 32-bit *x86*, *AMD64* and *EM64T* compatible CPUs with so-called *Prescott New Instructions* (PNI), such as Intel IA32 processors with EM64T extensions and newer generation (Revision E and beyond) AMD64 processors.

SSE4A and ABM – AMD Instruction Set enhancements for the Quad-Core AMD Opteron Processor. Support for these instructions is enabled by the –tp barcelona or –tp barcelona-64 switch.

SSSE3 – an extension of the SSE3 instruction set found on the Intel Core 2.

Static linking – a method of linking:

- On Linux, use *–Bstatic* to ensure all objects are included in a generated executable at link time. Static linking causes objects from static library archives of the form libxxx.a to be linked in to your executable, rather than dynamically linking the corresponding libxxx.so shared library.
- On Windows, the Windows linker links statically or dynamically depending on whether the libraries on the link-line are DLL import libraries or static libraries. By default, the static PGI libraries are included on the link line. To link with DLL versions of the PGI libraries instead of static libraries, compile and link with the *–Bdynamic option*.

SUA – the Subsystem for Unix-based Applications is a source-compatible subsystem for compiling and running 32- and 64-bit UNIX-based applications on a computer running Windows. *SUA* is supported on *Windows 2003 R2* and *Vista*. *SUA* is bundled with Windows; however, the full *SUA Utilities SDK* must be installed to link programs.

Win32 – any of the 32-bit Microsoft* Windows* Operating Systems (XP/2000/Server 2003) running on an *x86, AMD64* or *EM64T* processor-based system. On these targets, the PGI compiler products include additional Microsoft tools and libraries needed to build executables for 32-bit Windows systems.

Win64 – any of the 64-bit Microsoft Windows Operating Systems (XP Professional / Windows Server 2003 x64 Editions) running on an *x64* processor-based system. On these targets, the PGI compiler products include additional Microsoft tools and libraries needed to build executables for either Win32 or Win64 environments.

Windows – collectively, all *Win32* and *Win64* platforms supported by the PGI compilers.

x64 – collectively, all AMD64 and EM64T processors supported by the PGI compilers.

x86 – a processor designed to be binary compatible with i386/i486 and previous generation processors from Intel* Corporation. Refers collectively to such processors up to and including 32-bit variants.

x87 – 80-bit IEEE stack-based floating-point unit (FPU) and associated instructions on *x8*6–compatible CPUs.

2 PGI Release 7.1 Overview

This document describes changes between Release 7.1 and previous releases of the *PGI CDK*, as well as late-breaking information not included in the current printing of the *PGI User's Guide*. There are two platforms supported by the *PGI CDK* compilers and tools:

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

These versions are distinguished in these release notes where necessary.

2.1 PGI Release 7.1 Contents

Release 7.1 of PGI CDK is comprised of the following components:

- *PGF95* native OpenMP and auto-parallelizing Fortran 95 compiler.
- *PGF77* native OpenMP and auto-parallelizing FORTRAN 77 compiler.
- *PGHPF* data parallel High Performance Fortran compiler.
- *PGCC* native OpenMP and auto-parallelizing ANSI C99 and K&R *C* compiler.
- *PGC++* native OpenMP and auto-parallelizing ANSI *C++* compiler.
- *PGPROF Multi-process/multi-threaded* graphical profiler.
- *PGDBG Multi-process/multi-thread* graphical debugger.

- *MPICH MPI libraries, version 1.2.7,* for both 32-bit and 64-bit development environments.
- *MPICH2 MPI libraries, version 1.0.5p3,* for both 32-bit and 64-bit development environments.
- *MVAPICH MPI libraries, version 0.9.9,* for both 32-bit and 64-bit development environments.
- *TORQUE* 1.2.0 resource management system, a continuation of the product known as *OpenPBS*, or *Portable Batch System*. See www.supercluster.org/projects/torque for more information.
- *ScaLAPACK* linear algebra math library for distributed-memory systems, including *BLACS* version 1.1 (the Basic Linear Algebra Communication Subroutines) and *ScaLAPACK* version 1.7 for use with *MPICH* and the PGI compilers on Linux systems with a kernel revision of 2.4.2 or higher. This is provided in both *linux86* and *linux86-64* versions for AMD64 or EM64T CPU-based installations.

The release contains the following documentation and tutorial materials:

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

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

2.2 Supported Processors

The following table lists the processors on which Release 7.1 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 7.1 compilers can produce PGI Unified Binary object or executable files containing code streams fully optimized and supported for both AMD and Intel x64 CPUs. To produce unified binary files, you use one of the following *–tp* command-line options: *–tp x64* or *–tp <target1>,<target2>,<target3>…,* where <target> is any of the valid values in the following table. The table also includes the CPUs available and supported in multi-core versions.

Processors Supported by PGI 7.1

Brand	CPU	Cores	<target>	Memory Address	Floating Point HW					
					SSE1	SSE2	SSE3	SSSE3	SSE4	ABM SSE4a
AMD	*Opteron/Quadcore*	*4*	*barcelona-64*	*64-bit*	*Yes*	*Yes*	*Yes*	*No*	*No*	*Yes*
AMD	*Opteron/Quadcore*	*4*	*barcelona*	*32-bit*	*Yes*	*Yes*	*Yes*	*No*	*No*	*Yes*
AMD	*Opteron/Athlon64*	*2*	*k8-64*	*32-bit*	*Yes*	*Yes*	*Yes*	*No*	*No*	*No*
AMD	*Opteron/Athlon64*	*2*	*k8-32*	*32-bit*	*Yes*	*Yes*	*Yes*	*No*	*No*	*No*
AMD	*Opteron Rev E/F Turion /Athlon64*	*2*	*k8-64e*	*64-bit*	*Yes*	*Yes*	*Yes*	*No*	*No*	*No*
AMD	*Opteron Rev E/F*	*2*	*k8-32*	*32-bit*	*Yes*	*Yes*	*No*	*No*	*No*	*No*
AMD	*Turion64 Turion /Athlon64*	*1*	*k8-64e*	*64-bit*	*Yes*	*Yes*	*Yes*	*No*	*No*	*No*
AMD	*Turion64*	*1*	*k8-32*	*32-bit*	*Yes*	*Yes*	*No*	*No*	*No*	*No*
Intel	*Core 2*	*2*	*core2*	*32-bit*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*No*
Intel	*Core 2*	*2*	*core2-64*	*64-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 following table lists the operating systems, and their equivalents, on which Release 7.1 of the PGI compilers and tools.

To determine if Release 7.1 will install and run under a Linux equivalent version, such as Mandrake*, Debian*, Gentoo*, and so on, check the table for a supported system with the same glibc and gcc versions. Version differences in other operating system components can cause difficulties, but often these can be overcome with minor adjustments to the PGI software installation or operating system environment.

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

Most newer Linux distributions support the *Native Posix Threads Library* (NPTL), a new threads library that can be utilized in place of the *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 will automatically make use of NPTL on distributions when it is available. In addition, the *PGDBG* debugger is capable of debugging executables built using either NPTL or earlier thread library implementations.

Multi-processor 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, see Terms and Definitions *on page 3*.

Operating Systems and Features Supported in PGI 7.1									
Distribution	***Type***	***64-bit***	***HT***	***pgC++***	***pgdbg***	***NPTL***	***NUMA***	***glibc***	***GCC***
RHEL 5.0	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.5*	*4.1.2*
RHEL 4.0	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*2.3.4*	*3.4.3*
Fedora C-7	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.6*	*4.1.2*
Fedora C-6	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.5*	*4.1.1*
Fedora C-5	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*2.4*	*4.1.0*
Fedora C-4	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*2.3.5*	*4.0.0*
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*
SuSE 9.1	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*2.3.3*	*3.3.3*
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*
RHEL 3.0	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*2.3.2*	*3.2.3*
SuSE 9.0	**Linux**	*Yes*	*Yes*	*Yes*	*Yes*	*No*	*No*	*2.3.2*	*3.3.1*
RedHat 9.0	**Linux**	*No*	*No*	*Yes*	*Yes*	*Yes*	*No*	*2.3.2*	*3.2.2*

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

2.4 New System Calls

Release 7.1 of the PGI run-time libraries makes use of Linux system libraries to implement, for example, OpenMP and Fortran I/O. The PGI run-time libraries make use of several additional system library routines in this release, as listed below.

On 64-bit Linux systems, the additional system library routines are:

aio_error
aio_read
aio_return
aio_suspend
aio_write
calloc
getrlimit
pthread_attr_init
pthread_mutex_init
pthread_mutex_lock
pthread_mutex_unlock
setrlimit
sleep

On 32-bit Linux systems, the additional system library routines are:

aio_error	*calloc*
aio_read	*getrlimit*
aio_return	*pthread_attr_init*
aio_suspend	*setrlimit*
aio_write	*Sleep*

3 New or Modified Compiler Features

Following are the new features of Release 7.1 of the PGI compilers and tools as compared to prior releases.

- *MPICH2 1.0.5p3 support* – Updated from version 1.0.3.
- *MVAPICH (InfiniBand) support* – The PGI CDK 7.1 includes MVAPICH 0.9.9 configured for use with OpenFabrics (OFED). The compilers, debugger, and profiler all support this version of MPI.
- *PGI Unified Binaries* – When using the *–tp x64* target option, PGI compilers produce PGI Unified Binary programs containing code streams fully optimized and supported for both AMD and Intel x64 CPUs.
- *Multiple PGI Unified Binary Targets* – The *–tp* switch now supports a comma-separated list of targets allowing programs to be optimized for more than two 64-bit targets. Unified Binary directives and pragmas may be applied to functions, subroutines, or whole files, directing the compiler to generate unified binary code optimized for one or more targets.
- *PGI Unified Binary Culling* – Only those functions and subroutines where the target affects the generated code will have unique binary images. This change results in a code-size savings of 10-90% compared to *PGI 6.2,* without any adverse affect on performance.
- *Fortran 2003 ISO_C_BINDING* – The *ISO_C_BINDING* module is partially implemented. 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. The procedure *C_LOC* is supported, which returns the C address of an object. Other procedures in *ISO_C_BINDING*, such as *C_ASSOCIATED*, are not yet implemented.
- *Fortran 2003 Allocatable Regularization* – Fortran 2003 allocatable regularization is implemented in *PGF95* and 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* – Fortran 2003 allocatable array assignment is available in *PGF95*. The default is to use the Fortran 95 assignment semantics; however, the option *-Mallocatable=03* enables the Fortran 2003 assignment semantics.

- *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, a* synchronous 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.

- *ANSI C99* – The default value of *__STDC_VERSION__* is now defined as 199901L. Designated initializers and compound literals are implemented.

- *C Preprocessor* – Files with an upper-case .S suffix are treated as assembler source that must be preprocessed before passing the file to the assembler. A macro is not expanded by the preprocessor if it is preceded by '$' because a 'C' identifier can contain a '$'. When the source being preprocessed is an assembly file and the input file suffix is .s or .S, '$' is excluded from an identifier in the preprocessor.

- *C++ __restrict type qualifier* – The *__restrict* type qualifier indicates that all data accessed through the pointer will only be accessed through that pointer for the scope of the restricted pointer. The *__restrict* type qualifier may appear in the same context as a *const* type qualifier. This qualifier allows the compiler to perform additional optimizations.

- *Expanded gcc compatibility – PGC++ now supports* extended asm, the inline intrinsic libraries, GNU statement expressions and these redefinable predefined macros: `__PGIC__`, `__PGIC_MINOR__`, and `__PGIC_PATCHLEVEL__`.
  On Linux, PGI compilers are now using the *.ctor* and *.dtor* sections for constructors and destructors instead of the *.init* and *.fini* sections.

- *OpenMP 3.0 Stack Size* – The OpenMP 3.0 *OMP_STACK_SIZE* environment variable and the stack-size API are supported to control the size of the stack for newly created threads. API functions *omp_set_stack_size* and *omp_get_stack_size* are implemented.

- *OpenMP 3.0 Wait Policy* – The OpenMP 3.0 environment variable *OMP_WAIT_POLICY* affects the behavior of idle threads, in particular whether they spin or sleep when idle. Unemployed threads during a serial region can either busy wait using the barrier (*ACTIVE*) or politely wait using a mutex (*PASSIVE*). The choice is set by OMP_WAIT_POLICY. The default is *ACTIVE*.
- *Support for SSSE3, SSE4a, and ABM* – Support for Intel Core 2 SSSE3 instructions and AMD SSE4a and ABM instruction is now incorporated into the C and C++ inline intrinsic packages. On Linux, these instructions are available through the platform *binutils as*.
- *Improved Performance* – Code-generation optimizations include propagation and elimination of sign-extension, removal of useless zero extension, dead-store elimination, use of two-byte returns when branching to a return, optimized 32- and 64-bit shift operations, and better allocation of registers.
- *Fortran/C/C++ Performance* – High-level optimizations include serially-nested redundant conditional elimination, better analysis that creates more opportunities for vectorization and auto-parallelization, LRE enabled with pointer variables, extending the range of local common-subexpression elimination, adding LRE to the C++ compiler, and using an exit heuristic where certain function names are considered to be invoked only on a rare path and a branch to a target that has a low probability.
- *Auto-parallelization for multi-core processors* – Enhanced heuristic for auto-parallelization gives less consideration to loops with few iterations and more consideration to loops which contain nested loops when generating serial alt-code for auto-parallelized loops.
- *Enhanced vectorization* – Further tuning of the vectorizer provides additional idiom recognition and vectorization of loops with type conversions. For 64-bit targets, *-fast* now includes SSE code-generation and vectorization.
- *ACML 3.6* – The latest edition of the *AMD Core Math Library, ACML 3.6*, is bundled with the PGI 7.1 compilers on Linux.
- *Expanded OS support* – Added support for Fedora Core 6 and SuSE 10.2.
- *Environment Modules* – For users of the environment modules package, a script is included to set up the appropriate module files.

3.1 Getting Started

By default, the PGI 7.1 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*. This option incorporates optimization options to enable use of vector streaming SIMD (SSE/SSE2) instructions for 64-bit targets. The contents of the *–fast* switch are host-dependent, but usually includes the options *–O2 –Munroll –Mnoframe -Mlre*. For 64-bit targets, *-fast* also includes *–Mvect=sse –Mscalarsse –Mcache_align –Mflushz*.

3.2 Using –fast, –fastsse, and Other Performance-Enhancing Options

These options create a generally optimal set of flags for targets that support SSE/SSE2 capability. These options incorporate optimization options to enable use of vector streaming SIMD (SSE/SSE2) instructions for 64-bit targets. They enable vectorization with SEE 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 loop during each iteration.
`-Mnoframe`	Indicates to not generate code to set up a stack frame. *Note*. With this option, a stack trace does not work.
`-Mlre`.	Indicates loop-carried redundancy elimination

- These additional options are also typically available when using *–fast* for 64-bit targets and *–fastsse* for both 32- and 64-bit targets:

`-Mvect=sse`	Generates SSE instructions
`-Mscalarsse`	Generates scalar SSE code with xmm registers; implies -Mflushz

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

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

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

3.3 New or Modified Compiler Options

The following compiler options have been added or modified in PGI 7.1:

- *–Bdynamic* – Compiles for and links to the DLL version of the PGI runtime libraries. For more information, refer to *–Bdynamic* on page 29.
- *–Bstatic* – Compiles for and links to the static version of the PGI runtime libraries. For more information, refer to *–Bstatic* on page 29.
- *–Bstatic_pgi* – Statically links only the PGI libraries, allowing users to run applications on other hosts without installing compilers or the Portability Package. For more information, refer to *–Bstatic_pgi* on page 29.
- *–Mmpi=<suboption>* – The *–Mmpi* option has been modified to accept suboptions. These suboptions distinguish the version of MPI from which the compiler gets the header files and libraries. The suboptions include:

  mpich1 – Use MPICH1 headers and libraries

  mpich2 – Use MPICH2 headers and libraries

  mvapich – Use MVAPICH and OFED headers and libraries

  Note. –Mmpi without any suboption is still accepted for MPICH1, but this option is deprecated.

- *–Mprof=<suboption>* – The *–Mprof* option has been modified to accept suboptions. These suboptions have two functions:
  - They imply –Mmpi=<suboption>.
  - They enable MPI profiling.

    -MPI profiling must be used in conjunction with another –Mprof suboption, such as "time" or "func". Profile output is generated from an executable that is generated with the following option, and can be viewed using the PGI profiler, PGPROF.

    ```
    -Mprof=[mpich1|mpich2|mvapich1]
    ```

    -Mprof=mpi is still accepted to enable MPICH2 profiling, but the mpi suboption is deprecated.
- *–Mscalapack* – Use *–Mscalapack* with *–Mmpi* to include MPICH or use it with *–Mmpi2* to include MPICH2-compatible ScaLAPACK libraries on the linker command line.
- *–fast* – For 64-bit targets, *–fast* now includes the same options as the -fastsse option. The new *–fast* enables vectorization with SEE instructions, cache alignment, and flushz. The C/C++ compilers enable *–Mautoinline* with *–fast* or *–fastsse*.
- *–tp* – The *–tp* switch now allows a list of comma-separated targets. Previous releases allowed just one. If multiple targets are given, a unified binary is generated with code optimized for each of the targets.
- *–O4* – A new optimization level, *–O4*, enables hoisting of guarded invariant floating point expressions.
- *–d[D|I|M|N]* – The *–d* option prints additional information from the preprocessor:
  - *-dD* Print macros and values from source files.
  - *-dI* Print include file names.
  - *-dM* Print macros and values, including predefined and command-line macros.
  - *-dN* Print macro names from source files.
- *–soname library.so* – The compiler recognizes the *–soname* option and passes it to the linker. (Linux only.)
- *–flagcheck* – This option returns zero status if all flags are ok.
- *–E* – *pgcc –E* now preprocesses .h files.

- *–M[no]dse* – Enable [disable] a dead-store elimination phase that is useful for C++ programs that rely on extensive use of inline function calls for performance. The default is *–Mnodse.*
- *–Mallocatable=[95|03]* – The *–Mallocatable* 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.

4 Environment Modules

On Linux, if you use the Environment Modules package (e.g., the `module load` command), PGI 7.1 includes a script to set up the appropriate module files.

Assuming your installation base directory is `/opt/pgi`, and your MODULEPATH environment variable is `/usr/local/Modules/modulefiles`, execute the command:

```
/opt/pgi/linux86/7.1/etc/modulefiles/pgi.module.install \
       -all -install /usr/local/Modules/modulefiles
```

This command creates module files for all installed versions of the PGI compilers. You must have write permission to the `modulefiles` directory. This will enable the module commands:

```
module load pgi32/7.1

module load pgi64/7.1

module load pgi/7.1
```

where "pgi/7.1" uses the 32-bit compilers on a 32-bit system and uses the 64-bit compilers on a 64-bit system. To see what versions are available, use this command:

```
module avail pgi
```

The `module load` command will set or modify the environment variables as follows:

PGI	*the base installation directory*
CC	*full path to pgcc*
FC	*full path to pgf90*
F90	*full path to pgf90*
F77	*full path to pgf77*
CPP	*full path to pgCC*
CXX	*path to pgCC*
C++	*path to pgCC*

PATH	*prepends the PGI compiler and tools bin directory*
MANPATH	*prepends the PGI man page directory*
LD_LIBRARY_PATH	*prepends the PGI library directory*

Pgi does not support the Environment Modules package. For more information about the package, go to *http://modules.sourceforge.net*.

5 New or Modified PGDBG and PGPROF Features

PGI Workstation 7.11 and PGI CDK 7.1 include several new features and enhancements in the *PGDBG* parallel debugger and the PGPROF parallel profiler. This chapter describes those features.

5.1 PGDBG New and Modified Features

PGDBG is supported as a graphical and command line debugger in the *linux86* and *linux86-64* execution and development environments. Like the compilers, *PGDBG* for *linux86-64* must run in a *linux86-64* execution environment. *PGDBG* for *linux86* environments is a separate version, and it also runs in the *linux86-64* execution environment, but only with *linux86* executables. The *linux86-64* version of *PGDBG* only debugs executables built to run as *linux86-64* executables.

PGDBG 7.1 new features include:

- PGDBG 7.1 now supports debugging of applications consisting of up to four MPICH-1 processes running on the same machine as the PGDBG. For more information, refer to the *PGI Workstation 7.1 Release Notes.*
- PGDBG 7.1 supports debugging of MSMPI applications running on Microsoft Windows CCS clusters.
- PGDBG has a new debugger option: -mpi. This option, when placed on the command line prior to the name of the program being debugged, debugs an MPI application except MPICH-1.

- As described earlier, the compiler options *–Mmpi* and *–Mprof* now have suboptions that distinguish the version of MPI from which the compiler gets the header files and libraries. For more information, refer to New or Modified
  Compiler Features on page 13 or to the PGI User's Guide.

PGDBG 7.1 enhancements include:

- Improved stack trace capability
- Improved interoperability with Microsoft Visual C++
- Improved interoperability with gcc/g++
- Fast disassembly and overall performance improvements

The *PGDBG* graphical user interface (GUI) is invoked by default. To use a command-line interface, invoke the tool with the *–text* option.

Note. When running PGDBG, I/O from MPICH2 is sent to the window where PGDBG was invoked, not to the Program I/O window.

For a description of the usage and capabilities of PGDBG, see the *PGI Tools Guide*.

For limitations and workarounds, see *www.pgroup.com/support/faq.htm*.

5.2 PGPROF New and Modified Features

PGPROF is supported as a graphical and command line profiler in both the *linux86* and *linux86-64* environments. The same version works in any of these environments to process a trace file of profile data created by executing the instrumented program. Program instrumentation is either line-level (*–Mprof=lines*) on function-level (*–Mprof=func*). Additionally, on Linux, *PGPROF* supports *gprof*-style (*–pg*) sample-based and trace profiling, and hardware counters (hwcts).

PGDBG parallel profiler new features and enhancements are these:

- PGPROF 7.1 supports MPI profiling of collective communication routines, including MPICH1, MPICH2, and MVAPICH profiling. For more information, refer to the CDK 7.1 Installation Notes.
- PGPROF 7.1 supports MSMPI profiling on Microsoft Windows CCS clusters.
- PGPROF 7.1 includes some graphical user interface improvements.
- MPICH profiling is supported using *–Mprof=mpi*. To link with the correct MPICH library, *–Mprof=mpi* must be used with either the *–Mmpi=mpich1* or *–Mmpi=mpich2* option.

The *PGPROF* graphical user interface (GUI) is invoked by default. To use a command-line interface, invoke the tool with the *–text* option.

For a description of the usage and capabilities of PGPROF, see the *PGI Tools Guide*. For limitations and workarounds, see *www.pgroup.com/support/faq.htm*.

MPICH profiling is supported using *–Mprof=mpi*. To link with the correct MPICH library, *–Mprof=mpi* must be used with either the *–Mmpi=mpich1* or *–Mmpi=mpich2* option.

6 Distribution and Deployment

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

6.1 Generating PGI Unified Binaries

All PGI compilers can produce PGI Unified Binary programs containing code streams fully optimized and supported for both AMD64 and Intel EM64T processors using the *-tp* target option. The compilers generate and combine into one executable multiple binary code streams each optimized for a specific platform. At runtime, this one executable senses the environment and dynamically selects the appropriate code stream.

Different processors have differences, some subtle, in hardware features, such as instruction sets and cache size. The compilers make architecture-specific decisions about such things as instruction selection, instruction scheduling, and vectorization. PGI unified binaries provide a low-overhead means for a single program to run well on a number of hardware platforms.

Executable size is automatically controlled via unified binary culling. Only those functions and subroutines where the target affects the generated code will have unique binary images, resulting in a code-size savings of 10-90% compared to generating full copies of code for each target.

Programs can use PGI Unified Binary even if all of the object files and libraries are not compiled as unified binaries. Like any other object file, you can use PGI Unified Binary object files to create programs or libraries. No special start up code is needed; support is linked in from the PGI libraries.

The *-Mpfi* option disables generation of PGI Unified Binary. Instead, the default target auto-detect rules for the host are used to select the target processor.

6.1.1 Unified Binary Command-line Switches

The PGI Unified Binary command-line switch is an extension of the target processor switch, *-tp*, which may be applied to individual files during compilation.

The target processor switch, *-tp*, accepts a comma-separated list of 64-bit targets and generates code optimized for each listed target. The following example generates optimized code for three targets.

```
-tp k8-64,p7-64,core2-64
```

A special target switch, *-tp x64*, is the same as *-tp k8-64,p7-64*.

6.1.2 Unified Binary Directives and Pragmas

Unified binary directives and pragmas may be applied to functions, subroutines, or whole files. The directives and pragmas cause the compiler to generate PGI Unified Binary code optimized for one or more targets. No special command line options are needed for these pragmas and directives to take effect.

The syntax of the Fortran directive is

```
!pgi$[g|r| ] pgi tp [target]...
```

where the scope is g (global), r (routine) or blank. The default is r, routine.

For example, the following syntax indicates that the whole file, represented by *g,* should be optimized for both *k8_64* and *p7_64*.

```
!pgi$g pgi tp k8_64 p7_64
```

The syntax of the C/C++ pragma is

```
#pragma [global|routine| ] tp [target]...
```

where the scope is global or routine or blank. The default is routine.

For example, the following syntax indicates that the next function should be optimized for *k8_64*, *p7_64*, and *core2_64*.

```
#pragma routine tp k8_64 p7_64 core2_64
```

6.2 Static and Dynamic Linking with PGI Compilers

Prior to the 7.1 release, on Windows, all executables were linked against the PGI runtime DLL called pg.dll and the multi-threaded DLL version of the Microsoft runtime libraries. All executables were therefore dependent upon pg.dll and the Microsoft runtime. PGI now provides two compiler options that allow you to select static or dynamic linking:

6.2.1 –Bdynamic

Use this option to compile for and link to the DLL version of the PGI runtime libraries. This flag is required when linking with any DLL built by the PGI compilers. This flag corresponds to the /MD flag used by Microsoft's cl compilers.

On Windows, –Bdynamic must be used for *both* compiling and linking.

When you use the PGI compiler flag –Bdynamic to create an executable that links to the DLL form of the runtime, the executable built is smaller than one built without –Bdynamic. The PGI runtime DLLs, however, must be available on the system where the executable is run. The –Bdynamic flag must be used when an executable is linked against a DLL built by the PGI compilers.

6.2.2 –Bstatic

Use this option to explicitly compile for and link to the static version of the PGI runtime libraries.

On Windows, –Bstatic must be used for *both* compiling and linking.

For more information and usage examples for –Bdynamic and –Bstatic as well as information on using and creating static and dynamically-linked libraries, refer to the *PGI User's Guide*.

6.2.3 –Bstatic_pgi

Use this option to statically link only the PGI libraries. This option allows users to run applications on other hosts of similar glibc and gcc type, without installing the compilers or the Portability Package.

6.3 The REDIST Directory

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

6.3.1 PGI Redistributables

The PGI 7.1 release includes these directories:

- *$PGI/linux86/7.1/REDIST*
- *$PGI/linux86-64/7.1/ REDIST*
- *$PGI/win64/7.1-5/REDIST*
- *$PGI/win32/7.1/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 7.1 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 7.1 directory.

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

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

6.3.2 Microsoft Redistributables

The PGI products on Windows include Microsoft Open Tools. The Microsoft Open Tools directory contains a subdirectory named “redist”. PGI 7.1 licensees may redistribute the files contained in this directory in accordance with the terms of the PGI End-User License Agreement.

7 Known Limitations and Corrections

7.1 Documentation Clarifications

This section clarifies information available in the current documentation in the *PGI Tools Guide*.

7.1.1 -Mprof suboptions

In the *PGI Tools Guide*, Chapter 2, section *Compilation*, the description of –Mprof suboptions errorneously refers to the MVAPICH option as mvapich rather than mvapich1. The correct use of this compiler option is:

```
-Mprof=mvapich1
```

7.1.2 Configure SSH

In the *PGI Tools Guide*, Chapter 1, section *SSH and RSH*, the description of how to configure SSH so that it is unnecessary to enter passwords on the remote nodes when debugging MPI applications is incomplete.

The current **incomplete** description in the manual is this:

****

The following set of steps provides one way to configure `SSH` to eliminate this prompt.

```
$ ssh-keygen -t dsa
$ eval `ssh-agent -s`
$ ssh-add
<make sure that $HOME is not group-writable>
$ cd $HOME/.ssh
$ cp id_dsa.pub authorized_keys
```

Then for each cluster node you use in debugging, use:

```
$ ssh <host>
```

Once you answer the prompts to make the initial connection, subsequent connections should not require further prompting.

****

Additions and Corrections related to Configure SSH:

Please note the following additions and corrections:

First, the `ssh-keygen` command does prompt for a passphrase. This passphrase is used to authenticate to the ssh-agent during future sessions. The passphrase can be anything you choose.

Second, the example uses `'ssh-agent -s'`, which is correct for the `sh` or `bash` shells. For csh shells, use `'ssh-agent -c'`.

Finally, at the end of this description, the following should be added:

After logging out and logging back in, the ssh-agent must be restarted and reauthorized. For example, in a bash shell, this is accomplished as follows:

```
$ eval `ssh-agent -s`
$ ssh-add
```

You must enter the passphrase that was initially given to ssh-add to authenticate to the ssh-agent.

For further information consult your SSH documentation.

7.2 Known Limitations

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

- PGDBG - When running PGDBG, I/O from MPICH2 is sent to the window where PGDBG was invoked, not to the Program I/O window.
- PGDBG - MPICH-2 debugging on 32-bit systems doesn't work well with `mpiexec -pgi`. Launching an application in this way causes the application to hang. Most of the time using the `halt` button, then running to a breakpoint, results in a functional debug session. Workaround: Launch MPICH-2 applications using `pgdbg -mpi`. *Note.* This problem does not occur on 64-bit systems.

- PGDBG - Debugging of MPI applications on more recent versions of Linux (SuSE 10.x, Fedora 6) results in warning messages like the following:

  ```
  pgserv (myhost:20198): thr_init_mechanism:
  p_td_ta_new: Version of libpthread and libthread_db
  do not match
  ```

  Multi-thread debugging on the hosts where this warning is emitted is disabled. Multi-thread debugging on the debugger host will work if the host is where process 0 runs; on other hosts multi-thread debugging is disabled for this debug session.

- PGDBG - Debugging of MPI applications on older versions of Linux (SuSE9.3, SLES 9) requires that the user execute a `stepi` command before entering a `cont` command to enable multi-thread debugging.

- PGDBG - In PGI CDK 7.1-2 and later, PGDBG does not support message queue dumps for MPICH-2. Also, dumping of MVAPICH message queues is reliable for receive queues, but there are errors associated with dumping post queues.

- PGDBG - When debugging a MPI job that is launched under pgserv, the processes in the job are stopped before the first instruction of the program. Since there is no source level debugging information at this point, issuing the source level next command executes very slowly. To avoid having to run the job until it completes, stops due to an exception, or stops by a PGDBG halt command entered by the user, the user should set an initial breakpoint. If a Fortran program is being debugged, set the initial breakpoint at main, or MAIN_, or at another point on the execution path before issuing the continue command.

- 32-bit MPICH2 debugging does not work with GLIBC2.4 or above.

- Object and module files created using *PGI 7.1* compilers are incompatible with object files from *PGI 5.X* and prior releases.

- Object files compiled with –Mipa using *PGI 6.0* and prior releases must be recompiled with *PGI 7.1*.

- The *–i8* option can make programs incompatible with MPI and the ACML math library. Typically, use of any INTEGER*8 array size argument can cause failures with these libraries.

- The –i8 option can make programs incompatible with the bundled ACML library. Visit *developer.amd.com* to check for compatible libraries.
- 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.
- Using *–Mipa=vestigial* in combination with *–Mipa=libopt* with *PGCC* may result in 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.*
- 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 systems running version 2.6.4 Linux kernels. In addition, the time reported for each program unit by *gprof* and *PGPROF* for such executables run under some Linux distributions can be a factor of 10 higher than the actual time used. This behavior is due to a bug in certain shared object libraries included with those Linux distributions.
- *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 beyond.
- ACML 3.6 is built using the *–fastsse* compile/link option, which includes *–Mcache_align*. When linking in the ACML using the *–lacml* option on 32-bit targets, you must compile/link all program units 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.
- 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* must be used to obtain profile data on individual threads.

- *PGDBG* – 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.
- *PGDBG* – The `call` command does not support the following F90/F95 features: array-valued functions, pointer-valued functions, assumed-shape array arguments, pointer arguments. There is no known workaround to this limitation.
- *PGDBG* – Before PGDBG can set a breakpoint in code contained in a shared library, .so or .dll, the shared library must be loaded.
- *PGDBG* – 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 information on how to debug unified binaries, see *www.pgroup.com/support/tools.htm*.
- 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.

7.3 Corrections

The following problems have been corrected in the *PGI 7.1* release. Most were reported in *PGI 7.0* or previous releases. Problems found in *PGI 7.0* may not have occurred in the previous releases.

The following table provides the summary description of the problem. An *Internal Compiler Error* (ICE) is usually the result of checks the compiler components make on internal data structures, discovering inconsistencies that could lead to faulty code generation. For a complete and up-to-date list of TPRs fixed in recent releases of the PGI compilers and tools, see *www.pgroup.com/support/release_tprs.htm*.

7.3.1 Corrections in 7.1-5

The following TPRs have been resolved in 7.1-5.

TPR	Language / Tool	Description
3662	64-bit Fortran	MPI program fails to complete. Added a new routine, fsync 3F, to help mitigate race conditions in user code that arise because open, seek, write, and close calls may happen in the system kernel buffers but not necessarily on the disk.
4159	32/64-bit Fortran	Use of Module causes 'Severe error - Intrinsic not supported in initialization: mod PGF90 now implements the MOD intrinsic for integer initializations.
4322	64-bit Fortran	pgCC generates vector sse code which produces wrong answers. The optimizer may not have computed the proper alias information when two different pointers access the same class member and that class member is a pointer or array reference.
4352	32/64-bit C/C++	#PRAGMA OMP PARALLEL NUM_THREADS(1) -- ERROR MSG - EXPECTED AN IDENTIFIER PGC++ now allows expressions in an OpenMP directive that uses num_threads.
4353	32/64-bit Fortran	pgf90 program fails to compile ICE - "mk_mem_ptr_shape: no static desc for pointer" Fixed a problem when processing a deallocate statement for an allocatable member of a derived type.
4367	32/64-bit C/C++	pgCC with OpenMP ICE – sym_is_refd: bad sptr Fixed a problem with non-integer reduction variables.
14204	32-bit Fortran	select case(trim(adjustl(ftype))) segfaults on 7.1-2, not 7.0-7 Using trim in a case expression no longer causes a seg fault.
14205	32/64-bit Fortran	OpenMP regression 'FAIL: private copies are not disassociated' Corrected an error that caused private copies of threadprivate pointer variables not to be disassociated.

TPR	Language / Tool	Description
14208	32/64-bit Fortran	pgf90 example exposes an extra deallocation. Corrected an error where some internal variables were not created private in a parallel region.
14210	32/64-bit Fortran	pgf90 example causes ICE 'can't get const value 0' Corrected an error where the MIN intrinsic in an initialization expression might cause an internal compiler error.
14264	32/64-bit Fortran	pgf90 -fast ICE 'mismatched carry-around expression' The optimizer now handles the case when a negated subexpression is redundant with an existing redundant subexpression.
14279	64-bit Fortran	pgf95 -Mallocatable=03 gives wrong answer with -Mbounds and -g flags Corrected the bounds information for dummy arguments when used with -Mbounds and -g.
14285	32-bit C/C++	C++ Generates Bad Assembly code, identifer too long Fixed a problem generating assembly code for long identifiers.
14286	64-bit C/C++	C++ OpenMP fails where C OpenMP works Removed an outdated restriction in PGC++ that prevented sections inside of barrier.
14295	32/64-bit C/C++	Internal Compiler Error peel_last_iteration when using '-Mvect=sse' A compiler saftey check was incorrectly identifying an otherwise legal case as an internal error.
14313	32/64-bit C/C++	Can't inline intrinsics PGC++ failed to inline the call to the intrinsic mm_add_ps. The definition of mm_add_ps and other intrinsics have been updated to match the strict type-matching rules of the C++ inliner.
14319	64-bit Fortran	-O2 64-bit failed on WPS2.2.1 on core-2 in 7.1 The optimizer no longer considers loads of dummy arguments as invariant if the dummy argument is optional.

TPR	Language / Tool	Description
14329	32/64-bit C/C++	Boost C++ build gets "no instance of overloaded operator new Added new overloadings of new and delete to the PGI standard library: extern void *operator new(size_t sz, std::locale * ll); extern void operator delete(void *, std::locale * ll);
14331	32/64-bit Fortran	PGF90 example segfaults at 'SELECT CASE(TRIM(ADJUST(pt)))=ac' Duplicate of TPR#14202, also fixed in PGI 7.1-5.
14336	64-bit Fortran	_PGIC_version macros broken on 64-bit Apple OSX Updated the version macros for PGI 7.1.
14348	32/64-bit Fortran	Code that compiles with ifort fails with pgf90 Corrected a problem when passing an array to an elemental intrinsic in an initialization expression.
14351	ALL	Add Installation Guide to release The Installation Guide is now a separate document. The Release Notes no longer include installation instructions. Both the Installation Guide and the Release Notes may be found in the "doc" subdirectory of a release and on http://www.pgroup.com.
14362	32/64-bit Fortran	'pgf90 -Mipa=inline,except:xxx' fails to NOT inline xxx The command-line option expected the so-called decoarated function name. Now, both the user-visible and the decoarated name can be used.
14372	32-bit Fortran	Mac OS X 7.1-4 - mistake in startup script for FLEXlm lmgrd The startup script inadvertently referred to the PGI 7.0 release of lmgrd.
14373	64-bit C/C++	pgcc fails to properly link header file alloca.h on 64-bit Mac OS X An updated version of alloca.h is now in the include directory.

7.3.2 Corrections in 7.1-4

The following TPRs have been resolved in 7.1-4.

TPR	Language / Tool	Description
4217	32/64-bit Fortran	Read from character string succeeds, expected failure. Using formatted read of a double from the string "1.x" now properly returns an error. Previously, the value "1." was read as a valid string.
13258	32/64-bit Fortran	Derived types with allocatables print incorrectly. When printing derived types with allocatable members, the implicit data structures for the allocatables were printed.
14169	All	Complex multiply not vectorized properly. Multiple complex assignments not vectorized properly.
14172	All	PGI 7.1 man pages are out-of-date. Several options, such as -zc_eh, -sjlj_eh, and -traceback, were not documented in the PGI 7.1 man pages.
14220	32/64-bit Fortran	Threadprivate character improperly shared. OpenMP THREADPRIVATE CHARACTER variables were being shared by multiple threads.
14238	C/C++	Regression in tail-recursion elimination. Functions that pass the address of a local variable are not candidates for tail-recursion elimination.
14242	32/64-bit Fortran	Automatic variables managed with heavyweight allocator. Allocations of automatic arrays were done using the standard Fortran 90 allocator. This allocator has overhead to maintain F90 semantics of allocate and deallocate. This extra processing isn't needed for automatics and a lightweight allocator can be used.
14252	32/64-bit Fortran	Add Quad-core Opteron 'fastmath' library. When compiling with '-tp barcelona-64' or natively on a Quad-core AMD Opteron, the compiler generates calls to math library routines optimized for the target.

7.3.3 Corrections in 7.1-3

The following TPRs have been resolved in 7.1-3.

TPR	Language / Tool	Description
3875	32/64-bit Fortran	Code with ambiguous error not flagged as error
3894	32/64-bit Fortran	Program reports errors at the wrong line number
3905	32/64-bit Fortran	Example generates severe error 'Illegal attempt to redefine symbol'
4090	32/64-bit Fortran	Inappropriate error 'Illegal use of symbol max - not public entity of module'
4184	32/64-bit Fortran	Illegal FORALL loop results in ICEs - better to have error message
4234	32/64-bit Fortran	Example causes false severe error 'Illegal implied DO expression'
4291	32/64-bit Fortran	pgf90 does not catch argument mismatches in CONTAINED code.
4305	32/64-bit Fortran	Legal program generates severe 'must be a deferred shape array' error
4326	32/64-bit C	Undefined reference when -O2 and -g used together
4327	32/64-bit Fortran	Return value from a C complex function not handled correctly
4330	32/64-bit Fortran	Internal error in template code
4350	32/64-bit Fortran	Performance regression when sequential data in pointers passed to F77 subroutines
4357	32/64-bit C/C++	PGCC-S-0155-Illegal context for barrier message should not be issued
4358	32/64-bit C/C++	‘#pragma omp barrier’ before '#pragma omp for ordered' causes PGCC-S-0155-Illegal context

7.3.4 Corrections in 7.1-2

The following TPRs have been resolved in 7.1-2.

TPR	Language / Tool	Description
4096	All	Want to be able to link to static or dynamic libraries on Windows
4123	32/64-bit Fortran	OpenMP error ' Statement not allowed in WORKSHARE construct'
4186	32/64-bit Fortran	Problem setting array bounds from intrinsic with constant strings
4196	32/64-bit Fortran	Parameters incorrectly initialized
4256	32/64-bit Fortran	Problem when array constructor depends on an outer loop variable
4315	64-bit C/C++	Boost needs access to GLIBC trunc and strptime
4317	32/64-bit Fortran	Compilation warnings emitted for valid code (PGF90-W-0164-Overlapping data initializations)
4331	32/64-bit Fortran	Array pointer assignment fails within OpenMP WORKSHARE

7.3.5 Corrections in 7.1-1

The following TPRs have been resolved in 7.1-1.

TPR	Language / Tool	Description
3195	32/64-bit Fortran	Would like function than translates Fortran 90 iostat error codes to error strings
3425	64-bit Fortran	Request for directives to ignore possible dependences in array assignments
3448	32/64-bit Fortran	Enhancement request - add support for LEADZ(),POPCNT(),POPPAR()
3564	32/64-bit Fortran	Request: make -Mbackslash the default, or make -Mstandard imply -Mbackslash
3595	32/64-bit C/C++	pgCC build seems to hang in IPA during link step

TPR	Language / Tool	Description
3602	32/64-bit C/C++	pgCC openmp program fails with schedule (static)
3717	32/64-bit C/C++	Enhancement request: support packed struct, __attribute__((packed)), for pgcc
3726	All	DWARF3: use DW_AT_MIPS_LINKAGE_NAME for mangled C++, Fortran 90 names
3754	64-bit C/C++	64-bit pgcc with -mp and omp parallel for cannot handle long iterators
3777	32/64-bit Fortran	Forward reference to module function from earlier module function in character length fails
3778	32/64-bit Fortran	Forward reference to module function from earlier module function in character length fails
3796	32/64-bit Fortran	Fortran use module, only:operator(.x.) gives spurious error message about .neqv.
3818	64-bit Fortran	Win64 -Mchkstk with -fPIC fails at link time
3928	32/64-bit Fortran	Fortran Dwarf2 problem - need DW_AT_type for function as dummy argument
3939	32/64-bit Fortran	Enhancement Request - User's example should not complain 'Non-constant expression'
3970	32/64-bit C/C++	C++ compiler fails (segmentation fault) with > 60 if/else in while loop
3982	32/64-bit Fortran	F90 with multiple directories with two .mod files of the same name confuses pgf90
4001	32/64-bit C/C++	pgcc #pragma omp for with 64-bit index variable not properly implemented
4046	32/64-bit Fortran	Multiple imports of pgf90 shared object using "dlopen" cause runtime failures.
4051	All	ScaLapack example needs revision to work with 6.2-5 CDK
4064	32/64-bit Fortran	Request for support of F2003 GET_COMMAND_ARGUMENT
4070	All	PDF documentation is not well formatted, margins too large

TPR	Language / Tool	Description
4079	32/64-bit Fortran	request for CVF 'sizeof' in fortran
4081	32-bit C/C++	C++ code compiled '-tp px -fastsee' causes internal compiler error "fr_decr_use: bad usect"
4086	64-bit C/C++	C++ compiler does not implement -Mfcon
4088	All	fstat64 fails for Win64 (Fortran)
4111	32-bit Fortran	On SUA32, -Mchkfpstk can give "fp stack is not empty" runtime errors
4128	32/64-bit Fortran	pgf90 fails to recognize constant PARAMETER array elements in array bounds
4132	32/64-bit C/C++	Feature request: define "__PIC__=1" when "-fPIC" is used.
4134	32/64-bit C/C++	Linking with shared and nonshared libs causes ' static object marked for destruction more than once'
4148	32/64-bit C/C++	pgcpp does not properly recognize "__builtin_expect" (gcc compatibility)
4149	32-bit Fortran	Allow for non-English Administrator group name for Windows/SUA installations
4150	32/64-bit Fortran	Fortran Reads of integers in ascii files with (*) format should catch non-integer strings
4151	64-bit C/C++	pgcpp fails with internal compiler error on Win64 with -MD switch
4154	32/64-bit Fortran	time() and ctime() return integer*8 - not integer*4 as documentation indicates
4158	32/64-bit Fortran	Request for F2008 C_SIZEOF function
4160	32/64-bit Fortran	PGF90 fails to inline character function with spurious message about argument count
4163	32/64-bit Fortran	PGF90 PARAMETER array set by a reshaping another array fails
4167	32/64-bit Fortran	pgf90 does not support transfer function in parameter specification
4173	32/64-bit Fortran	OpenMP parallel region IF clause with(.false.) incorrectly runs in parallel

TPR	Language / Tool	Description
4175	All	Linking with -lacml or -lacml_mp requires -lacml_mv as well
4180	32/64-bit Fortran	pgf90 reporting error when attempting to inline pointer function
4185	32-bit C/C++	pgcc fails with internal compiler error with -mp/-Mconcur, 'Unable to allocate a register 22'
4191	32-bit Fortran	32-bit pgf77/pgf90 gives internal compiler error at -O2, 'Unable to allocate a register'
4200	32/64-bit Fortran	SAVE attribute assigned to allocatable, flagged as warning
4201	32/64-bit Fortran	SYSTEM_CLOCK does not have enough resolution
4202	32/64-bit Fortran	NINT() does not agree with other compiler output.
4203	32/64-bit Fortran	Fortran allocatable components of local derived type are not auto-deallocated
4205	32/64-bit Fortran	pghpf fails with internal compiler error - 'mk_mem_ptr_shape: extnt not subs'
4207	32/64-bit Fortran	Please refer to ISO/IEC 1539-1:1997 for Fortran reference
4208	All	64-bit 'Large-array' programming examples should add Win64 limitations
4213	64-bit C/C++	Compiler takes a long time to compile files with lots of functions
4214	32/64-bit Fortran	Fortran compiler fails with Internal compiler error in register allocation.
4219	32/64-bit Fortran	PGF90 USE,ONLY:OPERATOR(.x.) causes spurious error messages
4224	32/64-bit Fortran	OSX compilers do not support static linking
4226	32/64-bit C/C++	C++ link causes execution error "static object has been marked for destruction more than once"
4233	32/64-bit Fortran	Allow expression using PARAMETER array element in dimension of another module array
4235	32/64-bit Fortran	Fortran RAN() function does not work properly with -r8

TPR	Language / Tool	Description
4241	32/64-bit C/C++	pgcc gives spurious error message for barrier in nested OpenMP parallel region
4244	32/64-bit Fortran	Nested OpenMP parallel regions should compile with -mp
4245	32-bit Fortran	pgf77/pgf90 fail with internal compiler error in register allocation, with -O -fPIC
4246	32/64-bit Fortran	Incorrect error message with use, only::usergeneric.
4247	32/64-bit C/C++	Add pgCC -a, like pgCC -A, but generating warnings, not errors
4248	32/64-bit C/C++	pgCC failure with -O3 -g --- undefined label in generated asm code
4249	32/64-bit Fortran	Fortran compiler should give error, not fail, with SUM(1:n)
4251	32/64-bit Fortran	There was no documentation for the $PGI/target/version/src directory.
4253	32/64-bit Fortran	Compiler segfaults with Fortran PARAMETER initialized with implied DO
4254	32/64-bit Fortran	Fortran trim call and // in argument character length attribute causes runtime error
4257	32/64-bit Fortran	Allow non-default-kind logical argument to SUM MASK argument
4259	32/64-bit C/C++	Win32 compiler driver fails for large @objfilelist
4265	32/64-bit C/C++	pgCC -O2 -g example causes "Error: can't resolve `.text' .LBxxxx"
4266	32/64-bit Fortran	Host allocatable not auto-deallocated if only allocated in contained routine
4268	32/64-bit Fortran	Using MS$ATTRIBUTES C without explicit interface
4271	32-bit Fortran	pgf77 with -fpic -tp px generated bad code accessing static variables
4274	All	The install script should modify the default value of $PGI in lmgrd.rc to install directory.
4276	32-bit Fortran	Compiler fails with -mp with adjustable array in private clause of parallel region

TPR	Language / Tool	Description
4279	64-bit C/C++	asm() example with 'm' constraint was not restricted to memory
4280	32/64-bit C/C++	Bad assembly generated with C++ using -O -g
4283	32-bit Fortran	DEC$ATTRIBUTE DLLIMPORT,ALIAS names should not be decorated (Windows)
4284	32-bit Fortran	Request to add GETDAT and GETTIM to lib3f
4286	64-bit Fortran	Fortran LOG function with -Ktrap=denorm generates floating point exception
4287	All	When upgrading a CDK installation with a new compiler release, installcdk should not fail.
4288	All	User Guide should explain -Ktrap=inexact in more detail
4292	32/64-bit Fortran	Passing Dummy allocatable arrays not working
4298	32-bit Fortran	Failure when using -Munroll -tp px switches
4300	32/64-bit Fortran	Dwarf2 information was not being generated for Fortran module variables on Windows
4302	64-bit Fortran	pgf90 shape() fails with -Mlarge_arrays
4306	32/64-bit C/C++	pgcc -Mcpp -Bstatic causes the driver to call compiler with no input file
4308	32/64-bit Fortran	Fortran derived type dummy argument with initialized type component caused compiler failure
4310	32/64-bit C/C++	Add better messages when replacing loop by call to optimized runtime routine
4312	32-bit Fortran	Infinite loop in compiler with LRE example
4314	64-bit Fortran	Incorrect loop address used for Fortran dummy argument when expanded in vector loop
4316	64-bit Fortran	PGF90 syntax error issued for DO construct name immediately following USE
4318	64-bit Fortran	PGF95 fails with internal compiler error compiling program with error in deallocate statement
4319	64-bit Fortran	Use of -tp x64 -Mprof=func with contained subroutines yields link time undefined references

7.3.6 Corrections in 7.1-0

The following problems have been corrected in 7.1-0:

TPR	Lang/ Tool	Description
3717	pgcc	Packed struct support in pgcc
3602	pgCC	pgCC openmp program fails to perform with schedule static
4088		FSTA64 missing from the 64-bit Windows products.
4167	pgf90	pgf90 does not like transfer function in parameter specification
4128	pgf90	pgf90 example generates error 'deferred shape array must have POINTER attribute in a derived type'
4151	pgcpp	pgcpp ICE on Win64 with -MD switch
4163	pgf90	PGF90 PARAMETER array set by a reshaping another array fails
4205	pghpf	pghpf fails to build application - ICE 'mk_mem_ptr_shape: extnt not subs'
4233	pgf90	pgf90 example causes false severe error 'Assumed size array, b, is not a dummy argument'
4253	pgf90	User code causes pgf902 to seg fault in "dinit.c"
4254		User code gets allocation error during run time
3717	pgcc	Packed struct support in pgcc
3602	pgCC	pgCC openmp program fails to perform with schedule static

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

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