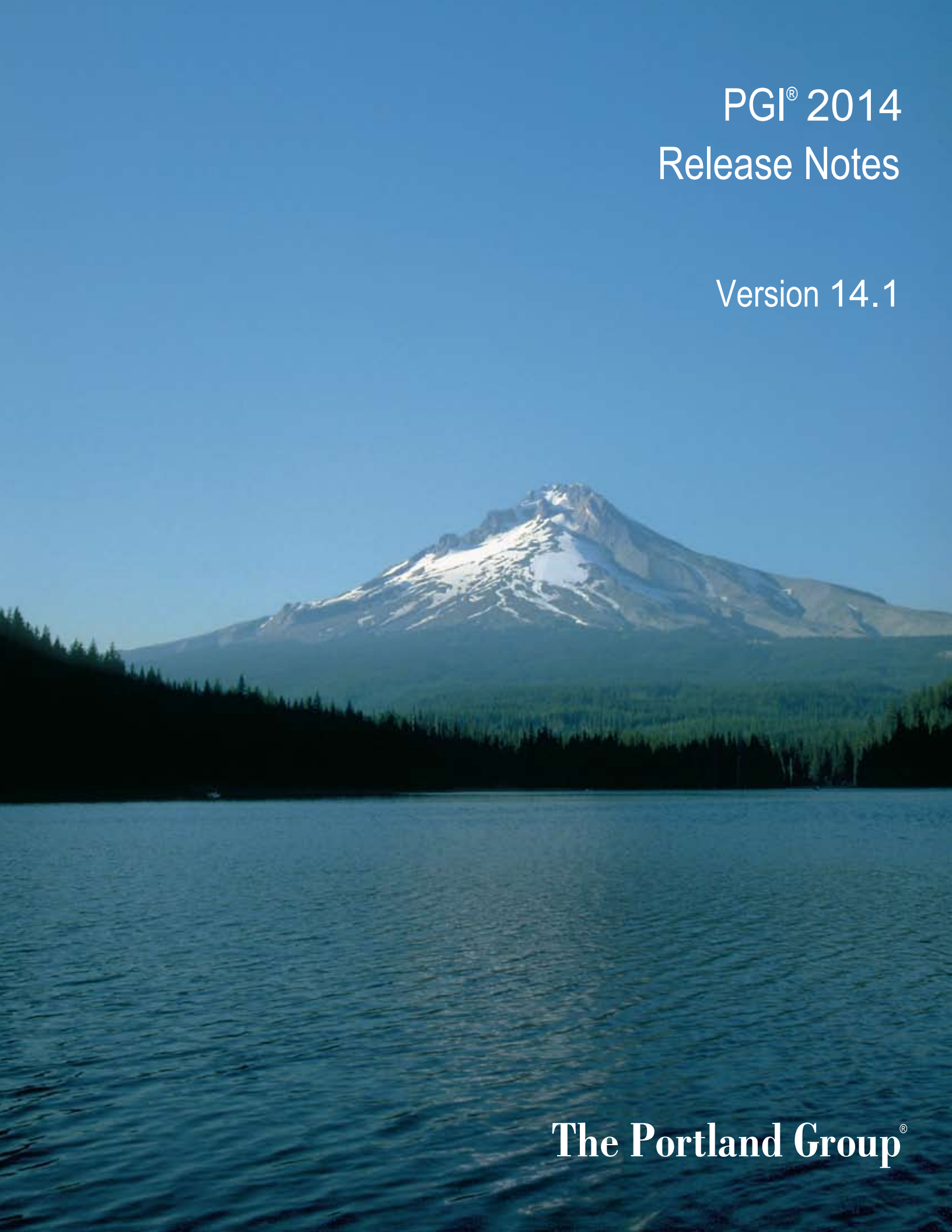
PGI® 2014 Release Notes

Version 14.1



The Portland Group®

PGI® 2014 Release Notes

Printed in the United States of America
First Printing: Release 2014, version 14.1, January 2014

Technical support: trs@pgroup.com
Sales: sales@pgroup.com
Web: www.pgroup.com

ID: 14241736

Contents

Tables

Chapter 1. Release Overview

Welcome to Release 2014 of *PGI Workstation™* , *PGI Server™* , and the *PGI CDK® Cluster Development Kit®*, a set of compilers and development tools for 32-bit and 64-bit x86-compatible processor-based workstations, servers, and clusters running versions of the Linux operating system. *PGI Workstation* and *PGI Server* are also available for the Apple OS X and Micorsoft WIndows operating systems.

This document describes changes between previous versions of the PGI 2014 release as well as late-breaking information not included in the current printing of the *PGI Compiler User's Guide*.

Product Overview

PGI Workstation, PGI Server, and the *PGI CDK* include exactly the same PGI compiler and tools software. The difference is the manner in which the license keys enable the software.

Licensing Terminology

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

- **License** – a legal agreement between NVIDIA and PGI end-users, to which users assent upon installation of any PGI product. The terms of the License are kept up-to-date in documents on pgroup.com and in the $PGI/<platform>/<rel_number> directory of every PGI software installation.
- **License keys** – ASCII text strings that enable use of the PGI software and are intended to enforce the terms of the License. License keys are generated by each PGI end-user on pgroup.com using a unique hostid and are typically stored in a file called `license.dat` that is accessible to the systems for which the PGI software is licensed.
- **PIN** – Personal Identification Number; a unique 6-digit number associated with a license which is included in your PGI order confirmation. The PIN can also be found in your PGI license file after `VENDOR_STRING=`.
- **License PIN code** – The unique 16-digit number associated with each PIN that enables users to "tie" the PIN to their pgroup.com user account. Provided by PIN owners to others whom they wish tied to their PIN(s).

License Options

PGI offers licenses for either x64+accelerator or x64 only platforms. *PGI Accelerator*™ products, the x64+accelerator platform products, include support for the directive-based OpenACC programming model, CUDA Fortran and PGI CUDA-x86. PGI Accelerator compilers are supported on all Intel and AMD x64 processor-based systems with either CUDA-enabled NVIDIA GPUs or select AMD GPUs and APUs, running Linux, OS X, or Windows. OS X accelerator support is available only on NVIDIA GPUs.

PGI Workstation and PGI Server Comparison

- All *PGI Workstation* products include a node-locked single-user license, meaning one user at a time can compile on the one system on which the *PGI Workstation* compilers and tools are installed. The product and *license server* are on the same local machine.
- *PGI Server* products are offered in configurations identical to *PGI Workstation*, but include 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 *license server*, that is, the system on which the *PGI Server* license keys are installed. There can be multiple installations of the *PGI Server* compilers and tools on machines connected to the license server; and the users can use the product concurrently, provided they are issued a license key by the license server.

PGI CDK Cluster Development Kit

A cluster is a collection of compatible computers connected by a network. The *PGI CDK* supports parallel computation on clusters of 64-bit x86-compatible AMD and Intel processor-based Linux workstations or servers with or without accelerators and 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.

Release Components

Release 2014 includes the following components:

- PGFORTRAN™ native OpenMP and auto-parallelizing Fortran 2003 compiler.
- PGCC® native OpenMP and auto-parallelizing ANSI C99 and K&R C compiler.
- PGC++® native OpenMP and auto-parallelizing ANSI C++ compiler.
- *PGPROF*® MPI, OpenMP, and multi-thread graphical profiler.
- *PGDBG*® MPI, OpenMP, and multi-thread graphical debugger.
- MPICH MPI libraries, version 3.0.4, for 64-bit development environments (Linux and OS X only).

  #### Note

  64-bit linux86-64 MPI messages are limited to <2GB size each.

- Microsoft HPC Pack 2012 MS-MPI Redsitributable Pack (version 4.1) for 64-bit and 32-bit development environments (Windows only).

- LAPACK linear algebra math library for shared-memory vector and parallel processors, version 3.4.2, supporting Level 3 BLACS (Basic Linear Algebra Communication Subroutines) for use with PGI compilers. This library is provided in both 64-bit and 32-bit versions for AMD64 or Intel 64 CPU-based installations running Linux, OS X, or Windows.
- ScaLAPACK 2.0.2 linear algebra math library for distributed-memory systems for use with MPICH, Open MPI, MVAPICH, and the PGI compilers on 64-bit Linux and OS X for AMD64 or Intel 64 CPU-based installations.
- A UNIX-like shell environment for 32-bit and 64-bit Windows platforms.
- FlexNet license utilities.
- Documentation in PDF and man page formats.

Additional components for PGI CDK

The *PGI CDK* for Linux includes additional components available for download from the PGI website, but *not* contained in the installation package:

- MVAPICH2 MPI libraries, version 1.9 available for 64-bit development environments.
- Open MPI libraries, version 1.7.3 for 64-bit development environments.

MPI Support

You can use PGI products to develop, debug, and profile MPI applications. The PGPROF® MPI profiler and PGDBG® debugger included with *PGI Workstation* are limited to eight local processes. The versions included with *PGI Server* are limited to 16 local processes. The MPI profiler and debugger included with *PGI CDK* supports up to 64 or 256 remote processes, depending on the purchased capabilities.

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 an unfamiliar term in these notes, please refer to the online glossary at

www.pgroup.com/support/definitions.htm

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

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

Supported Platforms

There are six platforms supported by the *PGI Workstation* and *PGI Server* compilers and tools. Currently, *PGI CDK* supports only 64-bit Linux clusters.

- *32-bit Linux* – includes all features and capabilities of the *32-bit Linux operating systems* running an *x64* compatible processor. 64-bit Linux compilers will *not* run on these systems.
- *64-bit Linux* – includes all features and capabilities of the *64-bit Linux operating systems* running an *x64* compatible processor. 32-bit Linux compilers do run on these systems.
- *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 OS X* – supported on 32-bit Apple operating systems running on either a 32-bit or 64-bit Intel-based Mac system.
- *64-bit OS X* – supported on 64-bit Apple operating systems running on a 64-bit Intel-based Mac system.

Supported Operating Systems

This section describes updates and changes to PGI 2014 that are specific to Linux, OS X, and Windows.

Linux

Linux download packages are reorganized in PGI 2014. You can download a 64-bit Linux compiler package for installation on 64-bit Linux machines, and/or you can download a 32-bit package that installs on 32-bit and 64-bit Linux systems.

- RHEL 4.8+, including RHEL 6.5
- Fedora 4+, including Fedora 20
- SuSE 9.3+, including SuSE 13.1
- SLES 10+, including SLES 11 SP 3
- Ubuntu 8.04+, including Ubuntu 13.10

OS X

PGI 2014 for OS X supports most of the features of the 32-bit 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 OS X function identically to their Linux counterparts.

- Supported versions are OS X versions 10.6 (Snow Leopard) and newer, including 10.9 (Mavericks).

Windows

PGI 2014 for Windows supports most of the features of the 32-bit and 64-bit versions for linux86 and linux86-64 environments.

Note

Starting January 2015, PGI releases will no longer include support for Windows XP, Windows Server 2003, or Windows Server 2008.

These releases are supported in PGI 2014, and require that the *Microsoft Windows 8.1 Software Development Kit (SDK)* be installed prior to installing the compilers.

- Windows Server 2008 R2
- Windows 7
- Windows 8
- Windows 8.1
- Windows Server 2012

PGI products on all Windows systems include the Microsoft Open Tools. On the systems being deprecated in 2015, it contains all the tools needed for building executables. On the newer Windows systems, the Open Tools also needs the SDK to build executables.

Note

PGI 2014 requires 8.1 SDK, even on Windows 7 and Windows 8. The Windows 8 SDK requires the PGI 2013 release.

Getting Started

By default, the PGI 2014 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`.

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
`–Mpre`	Indicates partial redundancy elimination

`–fast` for 64-bit targets and `–fastsse` for both 32- and 64-bit targets also typically include:

`–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 PGFORTRAN 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 that are described in the *PGI Compiler Reference Manual*, 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.

Chapter 2. New or Modified Features

This chapter provides information about the new or modified features of Release 2014 of the PGI compilers and tools.

What's New in Release 2014

14.1 Updates and Additions

- Updates to PGI Accelerator OpenACC Fortran/C/C++ compilers, including:
  - Support for CUDA 5.5 and NVIDIA Kepler K40 GPUs
  - Support for AMD Radeon GPUs and APUs
  - Native compilation for NVIDIA and AMD GPUs
  - Ability within CUDA Fortran to generate dwarf information and debug on the host, device, or both
  - Additional OpenACC 2.0 features supported, including procedure calls (routine directive), unstructured data lifetimes; create and device_resident clauses for the Declare directive; multidimensional dynamically allocated C/C++ arrays; ability to call CUDA Fortran atomic functions on NVIDIA; and complete run-time API support.
  - PGI Unified Binary for OpenACC programs across NVIDIA and AMD GPUs

  For more information, refer to "PGI Accelerator, OpenACC, and CUDA Fortran Enhancements," on page 14.
- Full Fortran 2003 and incremental Fortran 2008 features including long integers, recursive I/O, type statement for intrinsic types, ISO_FORTRAN_ENV and ISO_C_BINDING module updates as well as support for F2008 `contiguous` attribute and keyword.

  For more information, refer to "New or Modified Fortran Functionality," on page 12.
- Extensive updates to libraries:
  - Updated versions of MPICH, OpenMPI and MVAPICH pre-built and validated with the PGI compilers. For more information, refer to "Using MPI," on page 13.
  - Pre-packaged open source libraries downloadable from the PGI website including NetCDF and HDF5

  - Updated BLAS and LAPACK pre-compiled libraries based on LAPACK 3.4.2
    - LAPACK linear algebra math library for shared-memory vector and parallel processors, version 3.4.2, supporting Level 3 BLACS (Basic Linear Algebra Communication Subroutines) for use with PGI compilers. This library is provided in both 64-bit and 32-bit versions for AMD64 or Intel 64 CPU-based installations running Linux, OS X, or Windows.
    - ScaLAPACK 2.0.2 linear algebra math library for distributed-memory systems for use with MPICH, Open MPI, MVAPICH, and the PGI compilers on 64-bit AMD64 or Intel 64 CPU-based installations running Linux and OS X.
- Support for the latest Operating Systems including Ubuntu 13.04, Ubuntu 13.10, Fedora 18, Fedora 19, Fedora 20, CentOS 6.4, RHEL 5, RHEL 6, Windows 7, Windows 8, Windows 8.1, OS X Mountain Lion and OSX Mavericks.
- GNU compatible C++ improved inlining, Boost and Trilinos correctness as well as OpenACC robustness; full C++11 coming in PGI 14.4.
- The `-ta` and `-acc` flags include additional options and functionality. The `-tp` flag functionality is now primarily for processor selection. For more information, refer to "New or Modified Compiler Options," on page 8.
- PGI Visual Fortran fully integrated with Visual Studio 2013, supported on Windows 8.1, including support for OpenACC and CUDA Fortran on NVIDIA Tesla GPUs, and full native OpenACC on AMD Radeon GPUs.
- A comprehensive suite of new and updated code examples and tutorials covering Fortran 2003, CUDA Fortran, CUDA-x86, OpenACC, OpenMP parallelization, auto-parallelization and MPI.
- These Windows releases are supported in PGI 2014, but will be deprecated in PGI 2015.
  - Windows XP
  - Windows Server 2003
  - Windows Server 2008

New or Modified Compiler Options

PGI 14.1 supports a number of new command line options as well as new keyword suboptions for existing command line options.

Required suboption

The default behavior of the OpenACC compilers has changed in 14.1 from previous releases. The OpenACC compilers now issue a compile-time error if accelerator code generation fails. You can control this behavior with the `required` suboption.

In previous releases, the compiler would issue a warning when accelerator code generation failed. Then it would generate code to run the compute kernel on the host. This previous behavior generates incorrect results if the compute kernels are inside a data region and the host and device memory values are inconsistent.

`-acc=required`, `-ta=tesla:required`, and `-ta=radeon:required` are the defaults.

You can enable the old behavior by using the `nonrequired` suboption with the `-ta` or `-acc` flags.

Accelerator Options

Note

The `-ta=nvidia` option is deprecated in PGI 2014. Users are urged to change their build commands and makefiles to use `-ta=tesla` in place of `-ta=nvidia`.

The `-acc` option enables the recognition of OpenACC directives. In the absence of any explicit `-ta` option, `-acc` implies `-ta=tesla,host`.

`-ta` Option

The `-ta` option defines the target accelerator and the type of code to generate. This flag is valid for Fortran, C, and C++ on supported platforms.

Syntax

```
–ta=tesla(:tesla_suboptions),radeon(:radeon_suboptions),host
```

There are three major suboptions:
`tesla(:tesla_suboptions)`
`radeon(:radeon_suboptions)`
`host`

Default

The default is `–ta=tesla,host`.

Select Tesla Accelerator Target

Use the `tesla(:tesla_suboptions)` option to select the Tesla accelerator target and, optionally, to define the type of code to generate.

In the following example, Tesla is the accelerator target architecture and the accelerator generates code for compute capability 3.0:

```
$ pgfortran -ta=tesla:cc30
```

The `–ta=tesla` flag has these suboptions:

`cc10`	Generate code for compute capability 1.0.
`cc11`	Generate code for compute capability 1.1.
`cc12`	Generate code for compute capability 1.2.
`cc13`	Generate code for compute capability 1.3.
`cc1x`	Generate code for the lowest 1.x compute capability possible.
`cc1+`	Is equivalent to cc1x, cc2x, cc3x.

`cc20`	Generate code for compute capability 2.0.
`cc2x`	Generate code for the lowest 2.x compute capability possible.
`cc2+`	Is equivalent to cc2x, cc3x.
`cc30`	Generate code for compute capability 3.0.
`cc35`	Generate code for compute capability 3.5.
`cc3x`	Generate code for the lowest 3.x compute capability possible.
`cc3+`	Is equivalent to cc3x.
`[no]debug`	Enable [disable] debug information generation in device code.
`fastmath`	Use routines from the fast math library.
`fermi`	Is equivalent to cc2x.
`fermi+`	Is equivalent to cc2+.
`[no]flushz`	Enable[disable] flush-to-zero mode for floating point computations in the GPU code.
`keep`	Keep the kernel files.
`kepler`	Is equivalent to cc3x.
`kepler+`	Is equivalent to cc3+.
`llvm`	Generate code using the llvm-based back-end
`maxregcount:n`	Specify the maximum number of registers to use on the GPU.
`nofma`	Do not generate fused multiply-add instructions.
`noL1`	Prevent the use of L1 hardware data cache to cache global variables.
`pin`	Set default to pin host memory.
`[no]rdc`	Generate [do not generate] relocatable device code.
`[no]required`	Generate [do not generate] a compiler error if accelerator device code cannot be generated.

Select Radeon Accelerator Target

Use the `radeon(:radeon_suboptions)` option to select the Radeon accelerator target and, optionally, to define the type of code to generate.

In the following example, Radeon is the accelerator target architecture and the accelerator generates code for Radeon Cape Verde architecture:

```
$ pgfortran -ta=radeon:capeverde
```

The `–ta=radeon` flag has these suboptions:

`buffercount:n`	Set the maximum number of OpenCL buffers in which to allocate data.
`capeverde`	Generate code for Radeon Cape Verde architecture.
`keep`	Keep the kernel files.

`llvm`	Generate code using the llvm-based back-end
`[no]required`	Generate [do not generate] a compiler error if accelerator device code cannot be generated.
`spectre`	Generate code for Radeon Spectre architecture.
`tahiti`	Generate code for Radeon Tahiti architecture.

Host option

Use the `host` option to generate code to execute OpenACC regions on the host.

The `–ta=host` flag has no suboptions.

Multiple Targets

Specifying more than one target, such as `–ta=tesla,radeon` generates code for multiple targets. When host is one of the multiple targets, such as `–ta=tesla,host`, the result is generated code that can be run with or without an attached accelerator.

Relocatable Device Code

A `rdc` option is available for the `–ta` and `–Mcuda` flags that specifies to generate relocatable device code. Starting in PGI 14.1, on Linux the default code generation and linking mode for NVIDIA-target OpenACC and CUDA Fortran is `rdc`, relocatable device code.

You can disable the default and enable the old behavior and non-relocatable code by specifying any of the following: `–ta=tesla:nordc`, `–Mcuda=nordc`, or by specifying any 1.x compute capability or any Radeon target.

LLVM/SPIR and Native GPU Code Generation

For accelerator code generation, PGI 14.1 has two options.

- In legacy mode, which continues to be the default, PGI generates low-level CUDA C or OpenCL code.
- Beginning in PGI 14.1, PGI can generate an LLVM-based intermediate representation. To enable this code generation, use `–ta=tesla:llvm` on NVIDIA Tesla hardware or `–ta=radeon:llvm` on AMD Radeon hardware. `–ta=tesla:llvm` implies and requires CUDA 5.5 or higher.

PGI's debugging capability for Tesla uses the LLVM back-end.

DWARF Debugging Formats

PGI 14.1 has initial support for generating dwarf information in GPU code. To enable dwarf generation, just as in host code, you use the `–g` option.

Dwarf generation requires use of the LLVM code generation capabilities. Further, it is possible to generate dwarf information and debug on the host, device, or both. Further, for NVIDIA, the LLVM code generation requires CUDA 5.5.

If you don't want –g to apply to both targets, PGI supports the `debug` and `nodebug` suboptions. For example:

`-acc -g` implies `-ta=tesla,host -O0 -g` on the host and `-g llvm` on the device with cuda5.5.

`-acc -ta=tesla:debug` implies debug on the device; use llvm and cuda5.5

`-acc -g -ta=tesla:nodebug` implies debug on the host and no llvm code generation

-tp Modifications

The `-tp` switch now truly indicates the target processor. In prior releases a user could use the `-tp` flag to also indicate use of 32-bit or 64-bit code generation. For example, the `-tp shanghai-32` flag was eqivalent to the two flags: `-tp shanghai` and `-m32`.

The `-tp` flag interacts with the `-m32` and `-m64` flags to select a target processor and 32-bit or 64-bit code generation. For example, specifying `-tp shanghai -m32` compiles 32-bit code that is optimized for the AMD Shanghai processor, while specifying `-tp shanghai -m64` compiles 64-bit code.

Specifying `-tp shanghai` without a `-m32` or `-m64` flag compiles for a 32-bit target if the PGI 32-bit compilers are on your path, and for a 64-bit target if the PGI 64-bit compilers are on your path.

New or Modified Fortran Functionality

PGI 14.1 contains additional Fortran functionality such as full Fortran 2003 and incremental Fortran 2008 features including long integers, recursive I/O, type statement for intrinsic types, as well as ISO_FORTRAN_ENV and ISO_C_BINDING module updates and support for F2008 `contiguous` attribute and keyword.

Contiguous Pointers

PGI 14.1 supports the `contiguous` attribute as well as the `is_contiguous` intrinsic inquiry function.

contiguous Attribute

Here is an example of a declaration using the contiguous keyword:

```
    real*4, contiguous, pointer, dimension(:,:) :: arr1_ptr, arr2_ptr, arr3_ptr
```

It is the responsibility of the programmer to assure proper assignment and use of contiguous pointers. Contiguous pointers can result in improved performance, such as this example of using contiguous pointers as the arguments to the matmul intrinsic function.

```
    arr3_ptr = matmul(arr1_ptr,arr2_ptr)
```

is_contiguous Intrinsic Inquiry Function

The `is_contiguous()` intrinsic function takes a pointer argument and returns a value of type logical. It returns true if the pointer is associated with a contiguous array section, false otherwise.

New or Modified Tools Functionality

This section provides information about the debugger, *PGDBG*, and the profiler, *PGPROF*.

Debug and Profile SGI MPI Programs

In PGI 14.1 PGDBG and PGPROF support debugging and profiling of MPI programs built with SGI MPI. To debug an SGI MPI program, use the PGDBG `-sgimpi` option, which has the same syntax as the `-mpi` option.

To profile an SGI MPI program, build it with `-Mprof=func,sgimpi`, `-Mprof=lines,sgimpi`, or with `-Mprof=time,sgimpi`. You must specify `sgimpi` even if you use `mpicc` or `mpif90` to build your program.

Local and Remote Debugging

PGDBG is licensed software available from The Portland Group. *PGDBG* supports debugging programs running on local and remote systems. The PGI license keys that enable *PGDBG* to debug must be located on the same system where the program you want to debug is running.

Local debugging
: If you want to debug a program running on the system where you have launched *PGDBG*, you are doing local debugging and you need license keys on that local system.

Remote debugging
: If you want to debug a program running on a system other than the one on which *PGDBG* is launched, then you are doing remote debugging and you need license keys on the remote system. The remote system also needs an installed copy of PGI Workstation, PGI Server, or PGI CDK.

Using MPI

The PGI compilers provide an option, `-Mmpi`, to make building MPI applications with some MPI distributions more convenient by adding the MPI include and library directories to the compiler's include and library search paths. The compiler determines the location of these directories using various mechanisms.

Table 2.1 lists the sub-options supported by `-Mmpi`.

Table 2.1. MPI Distribution Options

This MPI implementation...	Requires compiling and linking with this option...
MPICH1	Deprecated. `-Mmpi=mpich1`
MPICH2	Deprecated. `-Mmpi=mpich2`
MPICH v3	`-Mmpi=mpich`
MS-MPI	`-Mmpi=msmpi`
MVAPICH1	Deprecated. `-Mmpi=mvapich1`
MVAPICH2	Use MVAPICH2 compiler wrappers.
Open MPI	Use Open MPI compiler wrappers.
SGI MPI	`-Mmpi=sgimpi`

For more extensive information on using each of these MPI implementations, refer to *Using MPI* in the PGI Compiler User's Guide.

For distributions of MPI that are not supported by the `-Mmpi` compiler option, use the MPI-distribution-supplied compiler wrappers mpicc, mpic++, mpif77, or mpif90 to compile and link.

PGI Accelerator, OpenACC, and CUDA Fortran Enhancements

OpenACC Directive Summary

PGI now supports the following OpenACC directives:

Parallel Construct
: Defines the region of the program that should be compiled for parallel execution on the accelerator device.

Kernels Construct
: Defines the region of the program that should be compiled into a sequence of kernels for execution on the accelerator device.

Data Directive
: Defines data, typically arrays, that should be allocated in the device memory for the duration of the data region, whether data should be copied from the host to the device memory upon region entry, and copied from the device to host memory upon region exit.

Enter Data and Exit Data Directives
: The Enter Data directive defines data, typically arrays, that should be allocated in the device memory for the duration of the program or until an exit data directive that deallocates the data, and whether data should be copied from the host to the device memory at the enter data directive.

  The Exit Data directive defines data, typically arrays, that should be deallocated in the device memory, and whether data should be copied from the device to the host memory.

Host_Data Construct
: Makes the address of device data available on the host.

Loop Directive
: Describes what type of parallelism to use to execute the loop and declare loop-private variables and arrays and reduction operations. Applies to a loop which must appear on the following line.

Combined Parallel and Loop Directive
: Is a shortcut for specifying a loop directive nested immediately inside an accelerator parallel directive. The meaning is identical to explicitly specifying a parallel construct containing a loop directive.

Combined Kernels and Loop Directive
: Is a shortcut for specifying a loop directive nested immediately inside an accelerator kernels directive. The meaning is identical to explicitly specifying a kernels construct containing a loop directive.

Cache Directive
: Specifies array elements or subarrays that should be fetched into the highest level of the cache for the body of a loop. Must appear at the top of (inside of) the loop.

Declare Directive
: Specifies that an array or arrays are to be allocated in the device memory for the duration of the implicit data region of a function, subroutine, or program.

Specifies whether the data values are to be transferred from the host to the device memory upon entry to the implicit data region, and from the device to the host memory upon exit from the implicit data region.

Creates a visible device copy of the variable or array.

Update Directive
: Used during the lifetime of accelerator data to update all or part of a host memory array with values from the corresponding array in device memory, or to update all or part of a device memory array with values from the corresponding array in host memory.

Routine Directive
: Used to tell the compiler to compile a given procedure for an accelerator as well as the host. In a file or routine with a procedure call, the routine directive tells the implementation the attributes of the procedure when called on the accelerator.

Wait Directive
: Specifies to wait until all operations on a specific device async queue or all async queues are complete.

For more information on each of these directives and which clauses they accept, refer to the "*Using an Accelerator*" chapter of the *PGI Compiler User's Guide*.

CUDA Toolkit Version

The PGI Accelerator x64+accelerator compilers with OpenACC and CUDA Fortran compilers support the CUDA 5.0 toolkit as the default. The compilers and tools also support the CUDA 5.5 Toolkit. To specify the version of the CUDA Toolkit that is targeted by the compilers, use one of the following properties:

In OpenACC directives:

For CUDA Toolkit 5.0: `-ta=tesla:cuda5.0`
For CUDA Toolkit 5.5: `-ta=tesla:cuda5.5`

For CUDA Fortran:

For CUDA Toolkit 5.0: `-Mcuda=cuda5.0`
For CUDA Toolkit 5.5: `-Mcuda=cuda5.5`

You may also specify a default version by adding a line to the `siterc` file in the installation `bin/` directory or to a file named `.mypgirc` in your `home` directory. For example, to specify CUDA Toolkit 5.5, add the following line to one of these files:

```
set DEFCUDAVERSION=5.5;
```

Support for CUDA Toolkit versions 4.2 and earlier has been removed.

C++ Compiler

C++ and OpenACC

This release includes the OpenACC directives for the C++ compilers, pgcpp and (Linux only) pgc++. There are limitations to the data that can appear in data constructs and compute regions:

- Variable-length arrays are not supported in OpenACC data clauses; VLAs are not part of the C++ standard.
- Variables of class type that require constructors and destructors do not behave properly when they appear in data clauses.
- Exceptions are not handled in compute regions.
- Any function call in a compute region must be inlined. This includes implicit functions such as for I/O operators, operators on class type, user-defined operators, STL functions, lambda operators, and so on.

C++ Compatibility

PGI 2014 C++ object code is incompatible with prior releases.

All C++ source files and libraries that were built with prior releases must be recompiled to link with PGI 2014 or higher object files.

New or Modified Runtime Library Routines

PGI 2014 supports new runtime library routines associated with the PGI Accelerator compilers. For more information, refer to the ”Using an Accelerator” chapter of the *PGI Compiler User's Guide*.

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 Compiler User's Guide*.

Environment Modules

Note

This section is only applicable to *PGI CDK*

On Linux, if you use the Environment Modules package (e.g., the **module load** command), then PGI 2014 includes a script to set up the appropriate module files.

Chapter 3. Distribution and Deployment

Once you have successfully built, debugged and tuned your application, you may want to distribute it to users who need to run it on a variety of systems. This chapter addresses how to effectively distribute applications built using PGI compilers and tools.

Application Deployment and Redistributables

Programs built with PGI compilers may depend on runtime 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.

PGI Redistributables

The PGI 2014 release includes these directories:

$PGI/linux86/14.1/REDIST
$PGI/linux86-64/14.1/REDIST
$PGI/win32/14.1/REDIST
$PGI/win64/14.1/REDIST

These directories contain all of the PGI Linux runtime library shared object files, OS X dynamic libraries, or Windows dynamically linked libraries that can be re-distributed by PGI 2014 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 2014 directory.

Linux Redistributables

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

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

Microsoft Redistributables

The PGI products on Windows include Microsoft Open Tools. The Microsoft Open Tools directory contains a subdirectory named "redist". PGI 2014 licensees may redistribute the files contained in this directory in accordance with the terms of the PGI End-User License Agreement.

Microsoft supplies installation packages, `vcredist_x86.exe` and `vcredist_x64.exe`, containing these runtime files. These files are available in the `redist` directory.

Chapter 4. Troubleshooting Tips and Known Limitations

This chapter contains information about known limitations, documentation errors, and corrections.

For up-to-date information about the state of the current release, visit the frequently asked questions (FAQ) section on pgroup.com at: www.pgroup.com/support/index.htm

General Issues

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

- Object files created with prior releases of PGI compiler are incompatible with object files from PGI 2014 and should be recompiled.
- Using –g option to generate debug information for CUDA Fortran has these limitations:
  - Only linux 64-bit platform supports this feature

    No debug information is generated for boundaries for Fortran automatic arrays, adjustable dummy arrays, or assumed-shape dummy arrays.

    No debug information is generated for GPU code after a `!CUF` directive.
- The –i8 option can make programs incompatible with the ACML libraries; use of any INTEGER*8 array size argument can cause failures. Visit developer.amd.com to check for compatible 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.

Platform-specific Issues

Linux

The following are known issues on 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 of the PGI compilers and tools.

Apple OS X

The following are known issues on Apple OS X:

- On OS X platform, the PGI 2014 compilers do not support static linking of binaries. For compatibility with future Apple updates, the compilers only support dynamic linking of binaries.
- Using `–Mprof=func` or `–Mprof=lines` is not supported.

Microsoft Windows

The following are known issues on Windows:

- Windows XP does not support Visual Studio 2013 with PVF. If you have WIndows XP, you should install an earlier PGI 2013 release with the VS 2010 shell, and then install the 14.1 XP release. This approach results in PVF 14.1 installation with VS 2010.

  ### Note

  Starting January 2015, PGI will no longer include support for Windows XP, Windows Server 2003, or Windows Server 2008.

- For the Cygwin `emacs` editor to function properly, you must set the environment variable `CYGWIN` to the value "`tty`" before invoking the shell in which `emacs` will run. However, this setting is incompatible with the PGBDG command line interface `(-text)`, so you are not able to use `pgdbg -text` in shells using this setting.

  The Cygwin team is working to resolve this issue.

- 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.

- C++ programs on Win64 that are compiled with the option `-tp x64` fail when using PGI Unified Binaries. The `-tp x64` switch is not yet supported on the Windows platform for C++.
- 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.

The following are known issues on Windows and *PGDBG*:

- 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.

The following are known issues on Windows and *PGPROF*:

- Do not use `-Mprof` with PGI 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.

PGDBG-related Issues

The following are known issues on *PGDBG*:

- Before *PGDBG* can set a breakpoint in code contained in a shared library, `.so` or `.dll`, the shared library must be loaded.
- 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.
- Debugging of PGI Unified Binaries, that is, 64-bit programs built with more than one `-tp` option, is not fully supported. The names of some subprograms are modified in the creation, 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 PGI Unified Binary, see www.pgroup.com/support/tools.htm.

PGPROF-related Issues

The following are known issues on *PGPROF*:

- Accelerator profiling via `pgcollect` is disabled in PGI 2014. PGI Accelerator and OpenACC programs profiled using `pgcollect` will not generate any performance data related to the GPU. This capability is expected to be restored in a future release.

  Workaround: Set the environment variable `PGI_ACC_TIME` to '1' for the program when it runs (not when compiling). This setting causes the program to print some performance data to `stdout` on exit.

  CUDA Fortran profiling is still available for CUDA Fortran programs.
- 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 options `-pg` or `-Mprof=time`, are for the master thread only. To obtain profile data on individual threads, PGI-style instrumentation profiling with `-Mprof={lines | func}` or **pgcollect** must be used.

CUDA Fortran Toolkit Issues

The CUDA 5.0 Toolkit is set as the default in PGI 14.1. To use the CUDA 5.0 Toolkit, first download the CUDA 5.0 driver from NVIDIA at www.nvidia.com/cuda.

You can compile with the CUDA 5.5 Toolkit either by adding the `-ta=tesla:cuda5.5` option to the command line or by adding `set CUDAVERSION=5.5` to the `siterc` file.

`pgaccelinfo` prints the driver version as the first line of output. For a 5.0 driver, it prints:

```
CUDA Driver Version 5000
```

OpenACC Issues

This section includes the known limitations to OpenACC directives.

PGI plans to support these features in a future release, though separate compilation and extern variables for Radeon will be deferred until OpenCL 2.0 is released.

ACC routine directive limitations

- The routine directive is not yet supported in C++.
- The routine directive has limited support on AMD Radeon. Separate compilation is not supported on Radeon, and selecting `-ta=radeon` disables `rdc` for `-ta=tesla`.
- The `bind` clause on the routine directive is not supported.
- The `nohost` clause on the routine directive is not supported.
- Extern variables may not be used with `acc routine` procedures.
- Functions that return values are not supported with `acc routine`.
- Reductions in procedures with `acc routine` are not fully supported.
- Fortran assumed-shape arguments are not yet supported.

Clause Support Limitations

- The `wait` clause on OpenACC directives is not supported.
- The `async` clause on the `wait` directive is not supported.
- The `device_type` clause is not supported on any directive.

Corrections

A number of problems have been corrected in the PGI 2014 release. 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 version in which it was fixed.

Chapter 5. Contact Information

You can contact PGI at:

Two Centerpointe Drive, Suite 320
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 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.

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

NOTICE

TRADEMARKS

COPYRIGHT