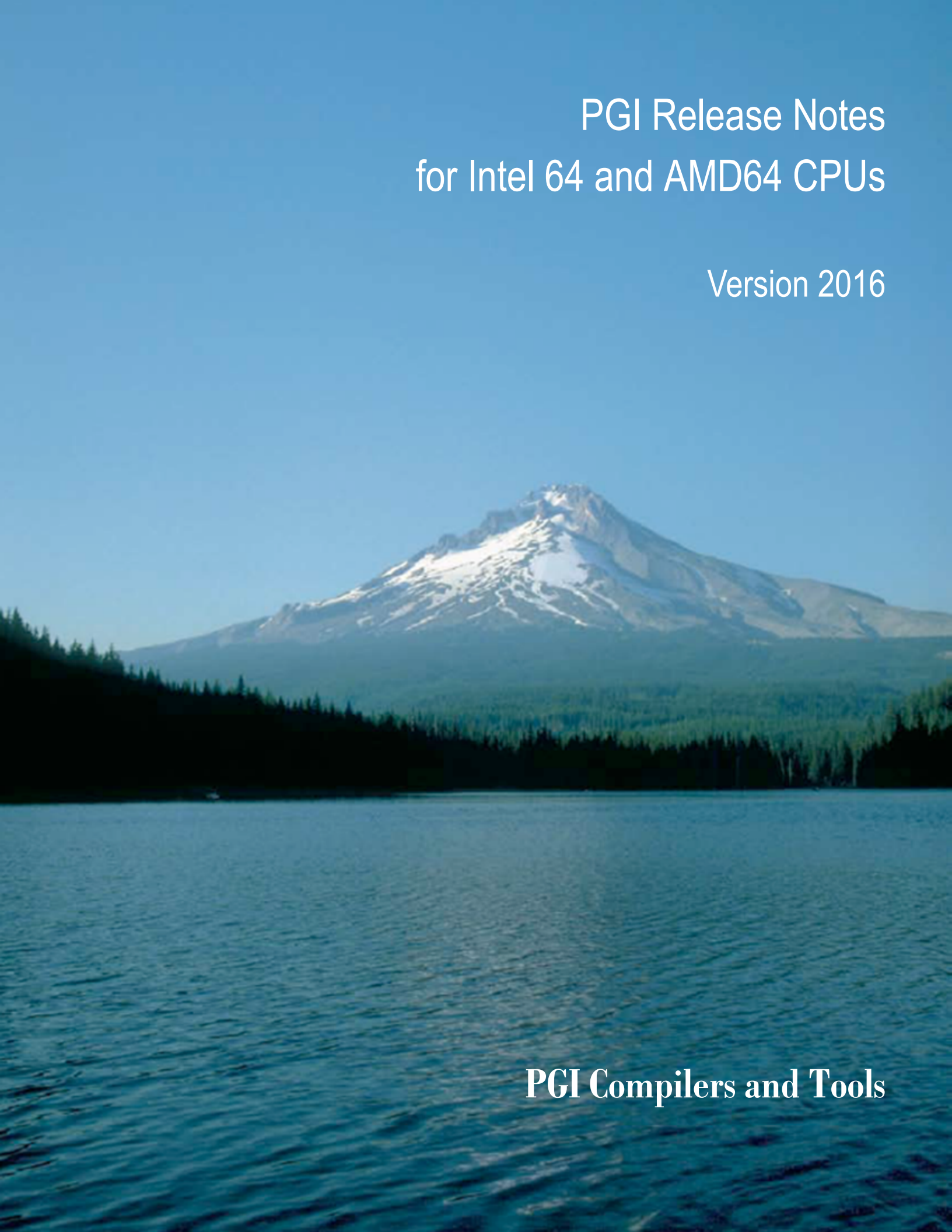
PGI Release Notes for Intel 64 and AMD64 CPUs

Version 2016



PGI Compilers and Tools

TABLE OF CONTENTS

LIST OF TABLES

Chapter 1. RELEASE OVERVIEW

Welcome to Release 2016 of the PGI Accelerator™ C11, C++14 and Fortran 2003 compilers and development tools for 32-bit and 64-bit x86-compatible processor-based workstations, servers, and clusters running versions of the Linux, Apple macOS and Microsoft Windows operating systems.

This document describes changes between previous releases of the PGI compilers and tools as well as late-breaking information not included in the current version of the PGI Compiler User's Guide.

PGI Release 2016 version 16.4 and newer includes FlexNet license daemons updated to version 11.13.1.3. This update addresses a FlexNet security vulnerability. These new license daemons also work with older PGI releases. We recommend all users update their license daemons - see the licensing FAQ for more information. This FlexNet update also requires you to update your PGI FlexNet license keys to a new format. Older keys are incompatible.

1.1. Product Overview

All PGI products include exactly the same PGI compilers and tools software. The difference is the manner in which the license keys enable the software.

1.1.1. Licensing Terminology

The PGI compilers and tools are license-managed. Before discussing licensing, it is useful to have common terminology.

- **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 http://www.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** – Product Identification Number, a unique 6-digit number associated with a license. This PIN is included in your PGI order confirmation. The PIN can also be found in your PGI license file after `VENDOR_STRING=`.
- **License PIN code** – A unique 16-digit number associated with each PIN that enables users to "tie" that PIN to their http://www.pgroup.com user account. This code is provided by PIN owners to others whom they wish tied to their PIN(s).

1.1.2. 16.10 PGI Community License

PGI 16.10 is a Community Edition release. Installation may place a Community license key in a file named `license.dat` in the PGI installation directory if no such file already exists.

If you use a separate license server, for example `LM_LICENSE_FILE=port@server.domain.com`, that supports use of PGI 16.10, it is recommended that you remove or rename the local Community license key file that is installed with 16.10.

Tip `FlexNet Licensing error:-96,7` This error may be encountered when using one or more license servers and you also have a `license.dat` file in your PGI installation directory. An anomaly in the license manager software can cause this error if a bad or failed license server is last in a list of servers, even if other, functioning license servers are listed. To remedy, either adjust your list of license servers or remove the local `license.dat` file.

1.1.3. Node-locked and Network Floating Comparison

- With a PGI node-locked single-user license, one user at a time can compile on the one system on which the PGI compilers and tools are installed. The product and license server are on the same local machine.
- PGI network floating products are offered in configurations identical to PGI node-locked products, but include network-floating licenses. This means that one 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 network floating license keys are installed. There can be multiple installations of the PGI compilers and tools on machines connected to the license server; and the users can use the product concurrently, provided they are issued a license seat by the license server.

1.1.4. 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 Intel or AMD processor-based Linux workstations or servers with or without accelerators and interconnected by a TCP/IP-based network, such as Ethernet or InfiniBand.

Support for cluster programming does not extend to clusters combining 64-bit processor-based systems with 32-bit processor-based systems.

1.2. Release Components

Release 2016 includes the following components:

- PGFORTRAN™ native OpenMP and OpenACC Fortran 2003 compiler.
- PGCC® native OpenMP and OpenACC ANSI C11 and K&R C compiler.
- PGC++® native OpenMP and OpenACC ANSI C++14 compiler.
- PGPROF® OpenACC, CUDA, OpenMP, and multi-thread graphical profiler.
- PGDBG® MPI, OpenMP, and multi-thread graphical debugger.
- Open MPI version 1.10.2 for 64-bit Linux including support for NVIDIA GPUDirect. GPUDirect requires CUDA 7.0 or later. Note that 64-bit linux86-64 MPI messages are limited to < 2 GB size each. As NVIDIA GPUDirect depends on InfiniBand support, Open MPI is also configured to use InfiniBand hardware if it is available on the system. InfiniBand support requires OFED 3.18 or later.
- MPICH libraries, version 3.2, for 64-bit macOS development environments.
- ScaLAPACK 2.0.2 linear algebra math library for distributed-memory systems for use with Open MPI, MPICH or MVAPICH, and the PGI compilers on 64-bit Linux and macOS for Intel 64 or AMD64 CPU-based installations.
- Microsoft HPC Pack 2012 MS-MPI Redistributable Pack (version 4.1) for 64-bit and 32-bit development environments (Windows only).
- BLAS and LAPACK library based on the customized OpenBLAS project source.
- A UNIX-like shell environment for 32-bit and 64-bit Windows platforms.
- FlexNet license utilities.
- Documentation in man page format and online PDFs.

1.2.1. Additional Components for PGI Network Floating Licenses

PGI floating licenses for Linux, including the PGI CDK, include additional components available for download from the PGI website, but not contained in the installation package including:

- MVAPICH2 MPI libraries, version 2.0, available for 64-bit development environments.
- MPICH MPI libraries, version 3.2, available for 64-bit development environments.

1.2.2. MPI Support

You can use PGI products to develop and debug MPI applications. The PGDBG® MPI debugger included with PGI node-locked licenses is limited to 16 local processes. The MPI debugger included with PGI network floating licenses supports up to 256 remote processes.

1.3. Terms and Definitions

This document contains 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 http://www.pgroup.com/support/definitions.htm.

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

Intel 64

A 64-bit Intel Architecture 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 2, Intel Core 2 Duo (Penryn), Intel Core (i3, i5, i7), both first generation (Nehalem) and second generation (Sandy Bridge) processors, as well as Ivy Bridge and Haswell 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, AMD Bulldozer, and AMD Piledriver processors.

1.4. Supported Platforms

There are six platforms supported by the PGI compilers and tools for x86 processor-based systems. The *PGI CDK* supports only 64-bit Linux clusters.

Support for 32-bit development is deprecated in PGI 2016 and will no longer be available as of the PGI 2017 release. PGI 2017 will only be available for 64-bit operating systems and will not include the ability to compile 32-bit applications for execution on either 32- or 64-bit operating systems.

- **32-bit Linux** – includes all features and capabilities of the 32-bit Linux operating systems running on 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 on an x64 compatible processor. Both 64-bit and 32-bit Linux compilers run on these systems.
- **32-bit Windows** – includes all features of the 32-bit Windows operating systems running on either a 32-bit x86 compatible or an x64 compatible processor. 64-bit Windows compilers will not run on these systems.
- **64-bit Windows** – includes all features and capabilities of the 64-bit Windows version running on an x64 compatible processor. Both 64-bit and 32-bit Windows compilers run on these systems.
- **32-bit macOS** – supported on 32-bit Apple operating systems running on either a 32-bit or 64-bit Intel-based Mac system. 64-bit macOS compilers will not run on these systems.
- **64-bit macOS** – supported on 64-bit Apple operating systems running on a 64-bit Intel-based Mac system. Both 64-bit and 32-bit macOS compilers run on these systems.

1.5. Supported Operating System Updates

This section describes updates and changes to PGI 2016 that are specific to Linux, macOS, and Windows.

1.5.1. Linux

Linux download packages are organized in PGI 2016 so that 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.

- CentOS 5+, including CentOS 7
- Fedora 6+, including Fedora 23
- RHEL 5+, including RHEL 7.2
- SLES 11+, including SLES 12 SP 1
- SuSE 11+, including openSuSE 13.2
- Ubuntu 12.04+, including Ubuntu 15.10

1.5.2. Apple macOS

PGI 2016 for macOS supports most of the features of the 32-bit and 64-bit versions for Linux environments. Except where noted in these release notes or the user manuals, the PGI compilers and tools on macOS function identically to their Linux counterparts.

- Supported versions are OS X versions 10.7 (Lion) through 10.11 (El Capitan).

1.5.3. Microsoft Windows

PGI products for Windows support most of the features of the 32-bit and 64-bit versions for Linux environments. PGI products on all Windows systems include the Microsoft Open Tools but also require that a Microsoft Windows Software Development Kit (SDK) be installed prior to installing the compilers.

PGI 2016 requires the Windows 10 SDK, even on Windows 7, 8 and 8.1.

These Windows operating systems are supported in PGI 2016:

- Windows Server 2008 R2
- Windows 7
- Windows 8
- Windows 8.1
- Windows 10
- Windows Server 2012

The PGI C++ compiler for Windows is no longer available as of the PGI 16.1 release.

1.6. Getting Started

By default, the PGI 2016 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 the aggregate option `-fast` or `-fastsse`.

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 instructionsfor 64-bit targets. They enable vectorization with SSE instructions, cache alignment, and flushz.

The contents of the `-fast` or `-fastsse` options are host-dependent.

The following table shows the typical `-fast` and `-fastsse` options.

Table 1 Typical `-fast` and `-fastsse` Options

Use this option...	To do this...
`-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-bit and 64-bit targets also typically include the options shown in the following table:

Table 2 Additional `-fast` and `-fastsse` Options

Use this option...	To do this...
`-Mvect=simd`	Generates packed SSE and AVX instructions.
`-Mcache_align`	Aligns long objects on cache-line boundaries. **Note** On 32-bit systems, if one file is compiled with the `-Mcache_align` option, then all files should be compiled with it. This is not necessary on 64-bit systems.
`-Mflushz`	Sets flush-to-zero mode.
`-M[no]vect`	Controls automatic vector pipelining.

For best performance on processors that support SSE and AVX 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 interprocedural 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 AND MODIFIED FEATURES

This chapter provides information about the new or modified features of Release 2016 of the PGI compilers and tools.

2.1. What's New in Release 2016

16.10 Updates and Additions

Many user-requested fixes and updates are implemented in this release. Refer to http://www.pgroup.com/support/release_tprs.htm for a complete and up-to-date table of technical problem reports fixed in recent releases of PGI compilers and tools. This table contains a summary description of each problem as well as the version in which it was fixed.

- PGI Accelerator OpenACC Compilers for NVIDIA GPUs
  - Integrated support for the production release of CUDA Toolkit 8. The 16.10 PGI compilers use CUDA 7 by default. CUDA 7.5 or CUDA 8 can be used instead by adding a sub-option (`cuda7.5` or `cuda8.0`, respectively) to the `-ta=tesla` or `-Mcuda` compile- and link-time options.
  - Improved support for the OpenACC cache directive.
  - Improved support for structs, unions, derived types in accelerator regions.
  - Extended data types supported by atomic operations.
- PGI C, C++ and Fortran Compilers
  - Implemented support for macOS Xcode 8.0.

16.9 Updates and Additions

- PGI Accelerator OpenACC Compilers for NVIDIA GPUs
  - Enabled the `-ta=tesla:managed` compiler option for Pascal GPUs; this option can now be combined with the `cc60` sub-option.
  - Added support for native atomic instructions for Pascal GPUs.

16.7 Updates and Additions

- PGI Accelerator OpenACC Compilers for NVIDIA GPUs
  - Integrated Beta support for the CUDA Toolkit 8 RC. Support for CUDA 8 is a Beta feature in 16.7 because the production version of CUDA 8 has not yet been released. The 16.7 PGI compilers use CUDA 7 by default. CUDA 7.5 or CUDA 8 can be used instead by adding a sub-option (`cuda7.5` or `cuda8.0`, respectively) to the `-ta=tesla` or `-Mcuda` compile- and link-time options.
  - Added initial support for NVIDIA Pascal GPUs (compute capability 6.0). The CUDA Toolkit 8 RC must be used to target Pascal GPUs. To compile for execution on a Pascal GPU, use the `cuda8.0` or the `cc60` sub-option to either `-ta=tesla` or `-Mcuda`.

    The `-ta=tesla:managed` option is not supported on Pascal GPUs in the 16.7 release; this option has been disabled for the `cc60` sub-option. Compute capability 6.0 can be set by explicit use of `cc60` or by the implicit addition of compute capability 6.0 to the default compute capability list by using the `cuda8.0` sub-option.
  - The PGI OpenACC C, C++ and Fortran compilers now support num_gangs, num_workers and vector_length clauses on the kernels construct, an OpenACC 2.5 feature. The clause argument is used to determine the corresponding number of gangs, workers or vector lanes that will be launched for the kernels region and any subroutines called therein.
  - The behavior for automatic arrays and arrays included in a private loop clause inside an `acc routine` has changed in PGI 16.7. Prior to this release, automatic arrays and loop private arrays were allocated one per GPU thread. The OpenACC specification says that an automatic array in `acc routine gang` or `acc routine worker` or a private array on a `loop gang` should be allocated once for the gang, and shared across all the workers and vector lanes in the gang. An automatic array in `acc routine vector` or a private array on a `loop worker` should be allocated once for each worker, and shared across all the vector lanes of the worker. A private array on a `loop vector` should be allocated once for each vector lane. The 16.7 release honors this behavior for automatic arrays and for fixed size private arrays on `loop vector`. Programs that depend on the previous erroneous behavior may no longer work.
- PGI Debugger

  Added variable rollover to the debugger's graphical interface.

16.5 Updates and Additions

- PGI Accelerator OpenACC Compilers for NVIDIA GPUs

  Using the cache directive to put arrays of variable size in shared memory is now allowed only under the new `safecache` sub-option to `-ta=tesla`. This change affects the default behavior of the compiler and may have performance implications. The `safecache` sub-option should be used only when the user is certain that the amount of data placed in shared

memory will not exceed the available size; exceeding the size of shared memory at runtime will cause a kernel launch failure.

- PGI Profiler

  Added compatibility with CUDA 8.0 drivers to the profiler while retaining compatibility with CUDA 7.5 drivers.

- Open MPI

  - Updated the version of Open MPI shipping with PGI products for Linux from 1.10.1 to 1.10.2.
  - Changed the installation of Open MPI so users can disable Open MPI's CUDA-aware features if they prefer.

16.4 Updates and Additions

- Licensing

  - Updated FlexNet license daemons to version 11.13.1.3 to address a FlexNet security vulnerability.
  - PGI license keys have changed format in version 16.4; all users will need to regenerate keys before using this release.

16.3 Updates and Additions

The 16.3 release was the first PGI release for Windows in 2016; all updates and additions listed for PGI 16.1 apply to 16.3 on Windows as well unless otherwise noted.

- PGI C++ Compiler

  Dropped support for C++ in PGI products for Windows.

- Interprocedural Analysis (IPA)

  Disabled support for Interprocedural Analysis, invoked using the `-Mipa` compiler option, on the Windows platform.

- PGI Accelerator OpenACC Compilers for NVIDIA GPUs

  When compiling an OpenACC parallel construct containing a call to a procedure declared as 'routine gang' or 'worker' or 'vector', or when compiling a procedure declared as such, the compiler will limit the `vector_length` to 32, the length of one CUDA warp. When compiling an OpenACC kernels construct containing a call to such a procedure, the compiler will limit the length of the vector clause to 32. This is to address performance issues when synchronizing vector lanes after a 'loop vector'.

- Libraries

  Dropped support for the Windows version of the IMSL Fortran Numerical Library.

16.1 Updates and Additions

PGI 16.1 was released for Linux/x86 and Mac OS X.

- PGI C++ Compiler
  - Performance on a spectrum of C++ codes we use for internal performance benchmarking is improved by an average of over 10% in this release compared to PGI 15.1
  - Comprehensive support for C++14; use of GCC version 5.0 or greater is required. Enable C++14 features with the PGC++ compiler option `--c++14`. PGC++ supports all C++14 features except:
    - generalized `constexpr` and `constexpr` member functions and implicit const
    - variable templates
    - clarifying memory allocation (merged allocation)
  - Integrated EDG release 4.10.1 providing interoperability with GCC 5.1. To take full advantage of the new C++11 features supported by GCC 5.1, use the PGC++ compiler option `--c++11`.
  - GNU changed the GCC STL header files in version 5.0. These changes may introduce compatibility issues with code compiled with earlier GCC versions. The GCC-supplied workaround to support compatibility with pre-GCC 5.0 compiled objects and libraries is to compile with `-D_GLIBCXX_USE_CXX11_ABI=0`. For more information, refer to https://gcc.gnu.org/onlinedocs/libstdc++/manual/using_dual_abi.html.
- PGI Fortran Compiler
  - Improved AVX SIMD vectorization and other Fortran compiler optimizations have improved performance of some WRF workloads by 15–20%.
  - Further optimizations of scalar/vector single precision `pow`, `exp`, `log`, and `atan` intrinsic functions for Intel Haswell processors.
  - Improved support for the F2003 associate clause and derived types.
- PGI Accelerator OpenACC Compilers
  - The following OpenACC 2.5 features are now supported:
    - The profiler/trace tools interface specified in OpenACC 2.5 has replaced the previously supported interface.
    - The default(present) clause is supported for C, C++ and Fortran compute constructs.
    - Reference counting is now used to manage the correspondence and lifetime of device data.
    - The `copy`, `copyin`, `copyout` and `create` data clauses now behave like `present_or_copy`, etc.
    - The `acc_copyin`, `acc_create`, `acc_copyout` and `acc_delete` API routines now behave like `acc_present_or_copyin`, etc.
    - The declare create directive with a Fortran allocatable has new behavior.
    - Added the API routine `acc_memcpy_device`.

- Added asynchronous versions of the data API routines.
- Reductions on orphaned gang loops are now explicitly disallowed.
- Reductions on array and struct elements, and C++ class members, are now explicitly disallowed.
- The use of `acc` routine without `gang/worker/vector/seq` is now flagged as an error.

- Improved vectorization of inner loops with `-ta=multicore`.
- Added support for use of F90 pointers in OpenACC kernels regions.
- Replaced the compiler option `-ta=tesla:pin` with `-ta=tesla:pinned`. The `pin` suboption dynamically 'pinned' user data before using that in data movement to the GPU. The new `pinned` suboption changes a program's allocates to allocate (and free) pinned memory instead. The OpenACC runtime will then test whether the data memory is pinned, and if so, will do direct transfers to and from the pinned space. The functionality of the pinned suboption is much more stable than the previous pin.
- All kernel launches when using `-ta=tesla:managed` are now synchronous by default. In other words, the default setting of `PGI_ACC_SYNCHRONOUS` has been changed to 1 with `-ta=tesla:managed`. This behavior is safer but may impact performance; to change kernel launches to be asynchronous, set `PGI_ACC_SYNCHRONOUS` to 0.
- The `ACC_BIND` environment variable is now set by default with `-ta=multicore`; `ACC_BIND` has similiar behavior to `MP_BIND` for OpenMP.

- PGI Debugger and Profiler
  - The all-new PGPROF profiler can profile CPU code or CPU+GPU code. PGPROF has both graphical and command-line modes. The new PGPROF is not compatible with previous versions and replaces the PGI `optopgprof`, `pgcollect`, `pgevtofq`, `pgoprun` and `pgsampt` profiling tools.
  - Added support for the disassembly of AVX3 instructions to the PGDBG debugger.
  - Enhanced the debugger's display of Fortran character types and named commons.
  - Improved support for debugging shared objects on OS X.
  - Significantly reduced the debugger load time of large applications on all platforms.
- Libraries
  - All PGI products now include a BLAS and LAPACK library based on the customized OpenBLAS project source and built with PGI compilers. To link against this library, simply add the `-lblas` and `-llapack` options to the link line.
  - PGI products for Linux now ship with Open MPI 1.10.1 instead of MPICH.
  - Users with PGI network floating licenses including the *PGI CDK* can access separate downloads of MVAPICH 2.1 and MPICH 3.2.
  - PGI-built versions of NetCDF 4.3.3.1 and Parallel NetCDF 1.6.1 are available online.
  - PGI-built versions of the Earth System Modeling Framework (ESMF) 6.3.0rp1, one per PGI-built MPI distribution, are available online.

- Other Features and Additions
  - PGI license keys have changed format in 2016; all users will need to regenerate keys before using this release.
  - PGI's new license key format allows combining multiple floating licenses of varying types and seat counts into a single license file.
  - New operating system support includes CentOS 7, Fedora23, RHEL 7.2, SLES 12, Ubuntu 15.10 and OS X El Capitan.
- Deprecations and Eliminations
  - The oldest version of glibc supported by PGI compilers on Linux is now 2.5.
  - With the 2016 release, support for PGI Accelerator features has been removed from PGI's 32-bit compilers on all platforms. The 32-bit compilers retain support for the `-ta` suboptions `host` and `multicore`.
  - The PGI C++ compiler compatible with the STLPort STL, and invoked via `pgcpp` or `pgCC`, is no longer provided in the PGI compiler suite on any platform. The GCC-compatible PGI C++ compiler invoked with `pgc++` has replaced the older version on Linux and OS X.
  - The PGI profiling tools `optopgprof`, `pgcollect`, `pgevtofq`, `pgoprun` and `pgsampt` have been removed; all sampling and profiling activities are now performed by the all-new PGPROF profiler.
  - Support for 32-bit development is deprecated in PGI 2016 and will no longer be available as of the PGI 2017 release. PGI 2017 will only be available for 64-bit operating systems and will not include the ability to compile 32-bit applications for execution on either 32- or 64-bit operating systems.
  - Support for PGI Accelerator features on OS X including CUDA Fortran, OpenACC and CUDA-x86 is deprecated in PGI 2016 and will no longer be available as of the PGI 2017 release.
  - Support for the CUDA 6.5 toolkit has been removed.
  - PGI products no longer bundle the AMD Core Math Library (ACML).
  - Support for RHEL 4 and SLES 10 have been dropped.
  - All documentation for PGI compilers and tools is available online at http://www.pgroup.com/resources/docs.htm but the PDFs are no longer bundled with PGI installation packages.

2.2. New and Modified Compiler Options

Release 2016 supports new and updated command line options and keyword suboptions.

New compiler options include the following:

- `-rdynamic` — This GNU-compatiblity switch for PGC++ passes `-export-dynamic` to the linker.

Modifications to the `-ta=tesla` suboptions include the replacement of `pin` with `pinned`:

- `pinned` — Use CUDA Pinned Memory; the previous `pin` suboption made pinning host memory the default behavior.

The all-new PGPROF profiler does not require recompilation to enable profiling. The `-Mprof` suboptions which are no longer necessary have been removed; these suboptions include: `func`, `lines`, `mpich`, `mpich1`, `mpich2`, `mvapich1`, `sgimpi` and `time`.

2.3. New and Modified Fortran Functionality

- Improved AVX SIMD vectorization and other Fortran compiler optimizations have improved performance of some WRF workloads by 15–20%.
- Further optimizations of scalar/vector single precision `pow`, `exp`, `log`, and `atan` intrinsic functions for Intel Haswell processors.
- Improved support for the F2003 associate clause and derived types.

2.4. New and Modified C/C++ Functionality

Comprehensive support for C++14 is included in the 2016 release:

- PGC++ supports all C++14 features except generalized constexpr and constexpr member functions and implicit const, variable templates, and clarifying memory allocation (merged allocation).
- Enable C++14 features with the PGC++ compiler option `--c++14`; use of GCC version 5.0 or greater is required.

The integrated EDG release 4.10.1 provides interoperability with GCC 5.1. To take full advantage of the new C++11 features supported by GCC 5.1, use the PGC++ compiler option `--c++11`.

GNU changed the GCC STL header files in GCC version 5.0. These changes may introduce compatibility issues with code compiled with earlier GCC versions. The GCC-supplied workaround to support compatibility with pre-GCC 5.0 compiled objects and libraries is to compile with `-D_GLIBCXX_USE_CXX11_ABI=0`. For more information, refer to https://gcc.gnu.org/onlinedocs/libstdc++/manual/using_dual_abi.html.

OpenACC implicit deep copy using CUDA Unified Memory on NVIDIA GPUs is available as an optionally-installed evaluation package on Linux. To enable this preview feature, use the compiler option `-ta=tesla:managaed`.

The PGI C++ compiler compatible with the STLPort STL, and invoked via `pgcpp` or `pgCC`, is no longer provided in the PGI compiler suite on any platform. The GCC-compatible PGI C++ compiler invoked with `pgc++` has replaced the older version on Linux and macOS.

2.5. New and Modified Tools Functionality

The 2016 release of the PGPROF profiler introduces a brand new profiler to the PGI suite. PGPROF now has the ability to profile CPU code or CPU plus GPU code, and profiling no longer requires recompilation. The all-new PGPROF provides both graphical and command-line modes. The new profiler is not compatible with previous releases of PGI's profiling tools.

Updates to the PGDBG debugger in this release include:

- Added support to PGDBG for the disassembly of AVX3 instructions.
- Enhanced the debugger's display of Fortran character types and named commons.
- Improved support for debugging shared objects on macOS.
- Significantly reduced debugger load time of large applications on all platforms.

2.6. Updates to CUDA Toolkit Support

As of PGI 16.7, CUDA 7.0, CUDA 7.5 and the CUDA 8.0 RC are supported on Linux, macOS and Windows. CUDA 7.0 is the default. Support for CUDA 8.0 is a Beta feature; the production version of CUDA 8.0 has not been released.

With the 2016 release, support for PGI Accelerator features has been removed from PGI's 32-bit compilers on all platforms. The 32-bit compilers retain support for the `-ta` suboptions `host` and `multicore`.

Support for PGI Accelerator features for macOS including CUDA Fortran, OpenACC and CUDA-x86 is deprecated in PGI 2016 and will no longer be available as of the PGI 2017 release.

Targeting a CUDA Toolkit Version

- The CUDA 7 Toolkit is set as the default in PGI 16.10. To use the CUDA 7.0 Toolkit, first download the CUDA 7.0 driver from NVIDIA at http://www.nvidia.com/cuda.
- You can compile with the CUDA 7.5 Toolkit either by adding the option `-ta=tesla:cuda7.5` to the command line or by adding `set DEFCUDAVERSION=7.5` to the `siterc` file. To use the CUDA 7.5 Toolkit, you must download and install the CUDA 7.5 driver from NVIDIA.
- You can compile with the CUDA 8 Toolkit RC either by adding the option `-ta=tesla:cuda8.0` to the command line or by adding `set DEFCUDAVERSION=8.0` to the `siterc` file. To use the CUDA 8.0 Toolkit RC, you must download and install the CUDA 8.0 driver from NVIDIA.
- `pgaccelinfo` prints the driver version as the first line of output. For a 7.0 driver, it prints:

```
CUDA Driver Version 7000
```

2.7. New Features in PGI Accelerator OpenACC Compilers

Multicore Support

PGI 16.1 supports the option `-ta=multicore`, to set the target accelerator for OpenACC programs to the host multicore. This will compile OpenACC compute regions for parallel execution across the cores of the host X86 processor or processors. The host multicore will be treated as a shared-memory accelerator, so the data clauses (`copy`, `copyin`, `copyout`, `create`) will be ignored and no data copies will be executed.

By default, `-ta=multicore` will generate code that will use all the available cores of the processor. If the compute region specifies a value in the `num_gangs` clause, the minimum of the `num_gangs` value and the number of available cores will be used. At runtime, the number of cores can be limited by setting the environment variable `ACC_NUM_CORES` to a constant integer value. If an OpenACC compute construct appears lexically within an OpenMP parallel construct, the OpenACC compute region will generate sequential code. If an OpenACC compute region appears dynamically within an OpenMP region or another OpenACC compute region, the program may generate many more threads than there are cores, and may produce poor performance.

The ACC_BIND environment variable is set by default with `-ta=multicore`; `ACC_BIND` has similiar behavior to `MP_BIND` for OpenMP.

The `-ta=multicore` option differs from the `-ta=host` option in that `-ta=host` generates sequential code for the OpenACC compute regions. In this release, `-ta=multicore` is not supported in conjunction with `-ta=tesla`, `-ta=radeon` or `-ta=host`.

Default Compute Capability

The default compute capability list for OpenACC and CUDA Fortran compilation for NVIDIA Tesla targets is cc20, cc30, cc35, and cc50. If CUDA 8.0 is specified using the `cuda8.0` sub-option, cc60 is added to the default compute capability list. The generation of device code can be time consuming, so you may notice an increase in compile time. You can override the default by specifying one or more compute capabilities using either command-line options or an `rcfile`.

To change the default with a command-line option, provide a comma-separated list of compute capabilities to `-ta=tesla:` for OpenACC or `-Mcuda=` for CUDA Fortran.

To change the default with an `rcfile`, set the `DEFCOMPUTECAP` value to a blank-separated list of compute capabilities in the siterc file located in your installation's bin directory:

```
set DEFCOMPUTECAP=20 30 35 50;
```

Alternatively, if you don't have permissions to change the `siterc` file, you can add the `DEFCOMPUTECAP` definition to a separate `.mypgirc` file (`mypgi_rc` on Windows) in your home directory.

New OpenACC 2.5 Features

The PGI 2016 release contains support for the following OpenACC 2.5 features:

- The profiler/trace tools interface specified in OpenACC 2.5 has replaced the previously supported interface.
- The default(present) clause for C, C++ and Fortran compute constructs.
- Reference counting manages the correspondence and lifetime of device data.
- The `copy`, `copyin`, `copyout` and `create` data clauses behave like `present_or_copy`, etc.
- The `acc_copyin`, `acc_create`, `acc_copyout` and `acc_delete` API routines behave like `acc_present_or_copyin`, etc.
- The declare create directive with a Fortran allocatable has new behavior.
- Added the API routine acc_memcpy_device.
- Added asynchronous versions of the data API routines.
- Reductions on orphaned gang loops are not allowed.
- Reductions on array and struct elements, and C++ class members, are not allowed.
- Use of acc routine without gang/worker/vector/seq is an error.

OpenACC 2.0 Features Not Supported

- The declare link directive for global data is not implemented.
- Nested parallelism (parallel and kernels constructs within a parallel or kernels region) is not implemented.

Support for Profiler/Trace Tool Interface

This release implements the OpenACC 2.5 version of the OpenACC Profiler/Trace Tools Interface. This is the interface used by the PGI profiler to collect performance measurements of OpenACC programs.

2.8. C++ Compiler

The PGI C++ compiler compatible with the STLPort STL, and invoked via `pgcpp` or `pgCC`, is no longer provided in the PGI compiler suite on any platform. The GCC-compatible PGI C++ compiler invoked with `pgc++` has replaced the older version on Linux and macOS.

2.8.1. C++ and OpenACC

This release includes support for OpenACC directives for the PGC++ compiler. 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.

2.8.2. C++ Compatibility

PGI 2016 C++ object code is incompatible with PGI 2015 and prior releases.

All C++ source files and libraries that were built with prior releases must be recompiled to link with PGI 2016 or higher object files.

Optional packages that provide a C++ interface—such as the MPI package included with all PGI products—now require the use of the `pgc++` compiler driver for compilation and linking.

These optional packages include:

- ESMF
- MPICH
- MVAPICH
- NetCDF
- Open MPI
- Parallel NetCDF
- ScaLAPACK

2.9. Runtime Library Routines

PGI 2016 supports runtime library routines associated with the PGI Accelerator compilers. For more information, refer to *Using an Accelerator* in the PGI Compiler User's Guide.

2.10. 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 the PGI Compiler User's Guide.

2.11. Environment Modules

On Linux, if you use the Environment Modules package (e.g., the `module load` command), then PGI 2016 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 section addresses how to effectively distribute applications built using PGI compilers and tools.

3.1. 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 Linux and Windows. On Windows, PGI also supplies Microsoft redistributable files.

3.1.1. PGI Redistributables

The PGI 2016 Release includes these directories:

```
$PGI/linux86/16.10/REDIST
$PGI/linux86-64/16.10/REDIST
$PGI/win32/16.10/REDIST
$PGI/win64/16.10/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 2016 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 16.10 `doc` directory.

3.1.2. Linux Redistributables

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

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

3.1.3. Microsoft Redistributables

The PGI products on Windows include Microsoft Open Tools. The Microsoft Open Tools directory contains a subdirectory named `redist`. PGI 2016 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 section contains information about known limitations, documentation errors, and corrections. Wherever possible, a work-around is provided.

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

4.1. Platform-specific Issues

4.1.1. 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.

4.1.2. Apple macOS

The following are known issues on Apple macOS:

- The PGI 2016 compilers do not support static linking of binaries. For compatibility with future Apple updates, the compilers only support dynamic linking of binaries.

4.1.3. Microsoft Windows

The following are known issues on Windows:

- 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.

- 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 work around this problem, set `SHELL` to refer to a shell in the Cygwin `bin` directory, e.g., `/bin/bash`.
- On Windows, runtime libraries built for debugging (e.g., `msvcrtd` and `libcmtd`) are not included with PGI products. 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.
- If you are building a 32-bit Windows Fortran program that performs Input/Output (I/O) on a namelist with User-Defined Derived type I/O, then you need to compile and link with the `-Munix` option.

The following are known issues with PGDBG on Windows:

- In PGDBG on the Windows platform, Windows times out `stepi`/`nexti` operations when single stepping over blocked system calls. For more information on the work-around for this issue, refer to the online FAQs at http://www.pgroup.com/support/tools.htm.

4.2. PGDBG-related Issues

The following are known issues in PGDBG:

- 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 compilation 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, refer to http://www.pgroup.com/support/tools.htm.
- There is a known issue when debugging on OS X 10.11 (El Capitan); the debugger will generate warning messages when running to a breakpoint. These warning messages do not affect the ability of the debugger to debug your application and will be addressed in a future release.

4.3. PGPROF-related Issues

The 16.1 release of PGPROF introduces an all-new PGI profiling tool. User feedback will be tracked for reported issues.

4.4. OpenACC Issues

This section includes known limitations in PGI's support for OpenACC directives. PGI plans to support these features in a future release.

ACC routine directive limitations

- 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.
- Fortran assumed-shape arguments are not yet supported.

Clause Support Limitations

- Not all clauses are supported after the `device_type` clause.

4.5. Corrections

A number of problems are corrected in this 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:

20400 NW Amberwood Drive Suite 100
Beaverton, OR 97006

Or electronically using any of the following means:

Fax: +1-503-682-2637
Sales: sales@pgroup.com
WWW: http://www.pgroup.com

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:

http://www.pgroup.com/userforum/index.php

Many questions and problems can be resolved by following instructions and the information available at our frequently asked questions (FAQ) site:

http://www.pgroup.com/support/faq.htm

Submit technical support requests through the online form at:

https://www.pgroup.com/support/support_request.php

PGI documentation is available at http://www.pgroup.com/resources/docs.htm.

PGI®