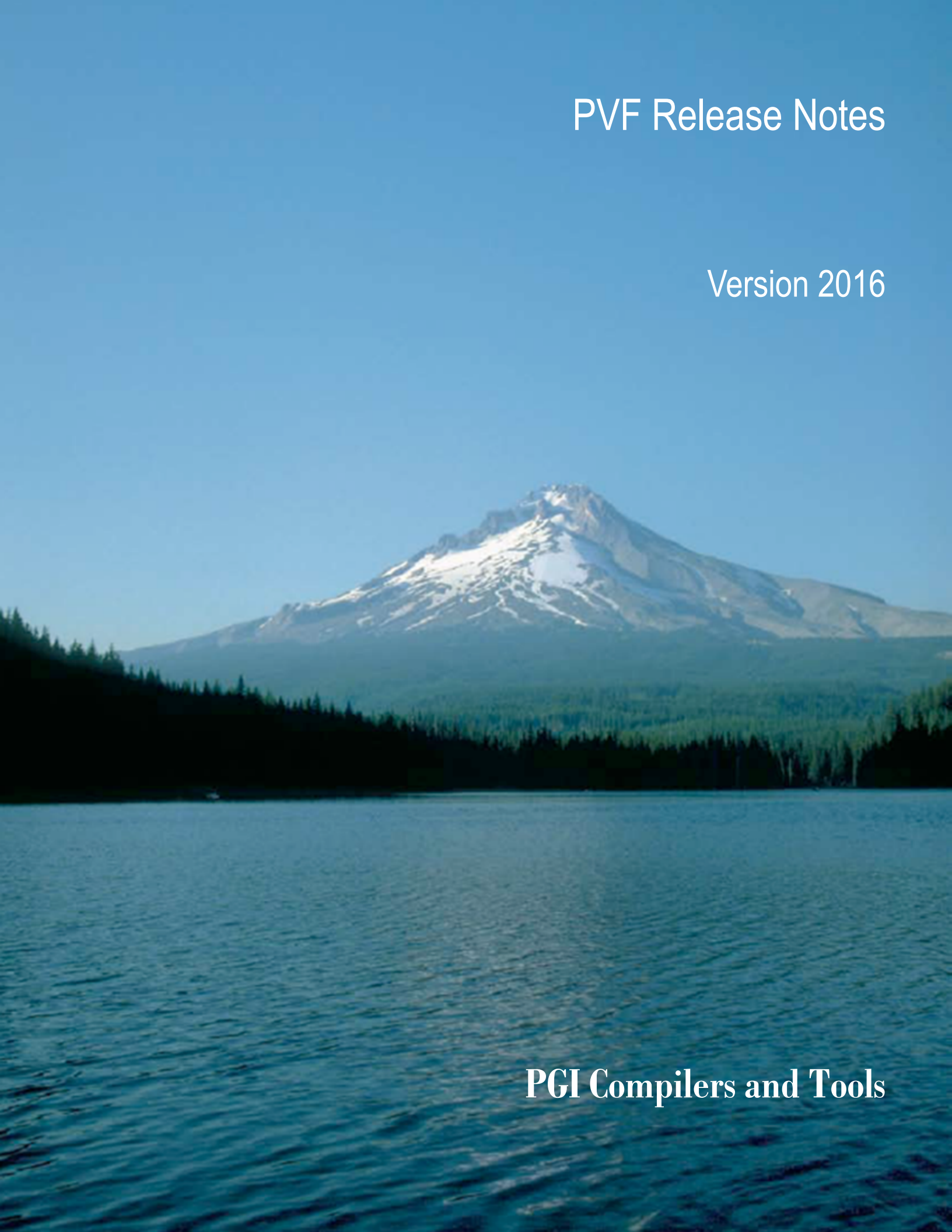
PVF Release Notes

Version 2016



PGI Compilers and Tools

TABLE OF CONTENTS

Chapter 1.
PVF RELEASE OVERVIEW

Welcome to Release 2016 of PGI Visual Fortran®, a set of Fortran compilers and development tools for 32-bit and 64-bit Windows integrated with Microsoft® Visual Studio.

This document describes the new features of the PVF IDE interface, differences in the PVF 2016 compilers and tools from previous releases, and late-breaking information not included in the standard product documentation.

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

PVF is integrated with two versions of Microsoft Visual Studio. Currently, Visual Studio 2013 and 2015 are supported. Throughout this document, "PGI Visual Fortran" refers to PVF integrated with either of the two supported versions of Visual Studio. Similarly, "Microsoft Visual Studio" refers to Visual Studio 2013 and VS 2015. When it is necessary to distinguish among the products, the document does so.

Single-user node-locked and multi-user network floating license options are available for both products. When a node-locked license is used, one user at a time can use PVF on the single system where it is installed. When a network floating license is used, a system is selected as the server and it controls the licensing, and users from any of the client machines connected to the license server can use PVF. Thus multiple users can simultaneously use PVF, up to the maximum number of users allowed by the license.

PVF provides a complete Fortran development environment fully integrated with Microsoft Visual Studio. It includes a custom Fortran Build Engine that automatically derives build dependencies, Fortran extensions to the Visual Studio editor, a custom PGI Debug Engine integrated with the Visual Studio debugger, PGI Fortran compilers, and PVF-specific property pages to control the configuration of all of these.

Release 2016 of PGI Visual Fortran includes the following components:

- PGFORTRAN OpenMP and auto-parallelizing Fortran 2003 compiler.
- PGF77 OpenMP and auto-parallelizing FORTRAN 77 compiler.
- PVF Visual Studio integration components.
- OpenACC and CUDA Fortran tools and libraries necessary to build executables for Accelerator GPUs, when the user's license supports these optional features.
- PVF documentation.

If you do not already have Microsoft Visual Studio on your system, be sure to get the PVF installation package that contains the Visual Studio 2015 shell.

1.2. Microsoft Build Tools

PVF on all Windows systems includes Microsoft Open Tools. These files are required in addition to the files Microsoft provides in the Windows SDK.

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.

Chapter 2.
NEW AND MODIFIED FEATURES

This section provides information about the new and modified features of Release 2016 of PGI Visual Fortran.

2.1. What's New in Release 2016

16.7 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 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 currently supported on Pascal GPUs; 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.

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.

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 Visual Fortran
  - Added support for Visual Studio 2015 including the VS 2015 Shell; continued support for VS 2013.

- Dropped support for Visual Studio 2012, 2010 and 2008.

- 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 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.
- 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
  - 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).
  - 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.

- `-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. 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, OS X 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 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.

Targeting a CUDA Toolkit Version

- The CUDA 7 Toolkit is set as the default in PGI 16.7. 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
```

2.5. 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.

2.6. 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.

Chapter 3.
SELECTING AN ALTERNATE COMPILER

Each release of PGI Visual Fortran contains two components — the newest release of PVF and the newest release of the PGI compilers and tools that PVF targets.

When PVF is installed onto a system that contains a previous version of PVF, the previous version of PVF is replaced. The previous version of the PGI compilers and tools, however, remains installed side-by-side with the new version of the PGI compilers and tools. By default, the new version of PVF will use the new version of the compilers and tools. Previous versions of the compilers and tools may be uninstalled using Control Panel | Add or Remove Programs.

There are two ways to use previous versions of the compilers:

- Use a different compiler release for a single project.
- Use a different compiler release for all projects.

The method to use depends on the situation.

3.1. For a Single Project

To use a different compiler release for a single project, you use the compiler flag –V<ver> to target the compiler with version <ver>. This method is the recommended way to target a different compiler release.

For example, –V13.8 causes the compiler driver to invoke the 13.8 version of the PGI compilers if these are installed.

To use this option within a PVF project, add it to the Additional options section of the `Fortran | Command Line and Linker | Command Line` property pages.

3.2. For All Projects

You can use a different compiler release for all projects.

The `Tools | Options` dialog within PVF contains entries that can be changed to use a previous version of the PGI compilers. Under `Projects and Solutions | PVF`

`Directories`, there are entries for Executable Directories, Include and Module Directories, and Library Directories.

- For the x64 platform, each of these entries includes a line containing `$(PGIToolsDir)`. To change the compilers used for the x64 platform, change each of the lines containing `$(PGIToolsDir)` to contain the path to the desired `bin`, `include`, and `lib` directories.
- For the 32-bit Windows platform, these entries include a line containing `$(PGIToolsDir)` on 32-bit Windows systems or `$(PGIToolsDir32)` on 64-bit Windows systems. To change the compilers used for the 32-bit Windows platform, change each of the lines containing `$(PGIToolsDir)` or `$(PGIToolsDir32)` to contain the path to the desired `bin`, `include`, and `lib` directories.

Warning: The debug engine in PVF 2016 is not compatible with previous releases. If you use `Tools | Options` to target a release prior to 2016, you cannot use PVF to debug. Instead, use the `-V` method described earlier in this section to select an alternate compiler.

Chapter 4. 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.

4.1. 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.

4.1.1. PGI Redistributables

PGI Visual Fortran includes redistributable directories which contain all of the PGI dynamically linked libraries that can be re-distributed by PVF 2016 licensees under the terms of the PGI End-User License Agreement (EULA). For reference, a copy of the PGI EULA in PDF form is included in the release.

The following paths for the redistributable directories assume '`C:`' is the system drive.

- On a 32-bit Windows system, the redistributable directory is:

```
C:\Program Files\PGI\win32\16.7\REDIST
```

- On a 64-bit Windows system, there are two redistributable directories:

```
C:\Program Files\PGI\win64\16.7\REDIST
C:\Program Files(x86)\PGI\win32\16.7\REDIST
```

The redistributable directories contain the PGI runtime library DLLs 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 the requirement that end-users of the executable have properly initialized their environment to use the relevant version of the PGI DLLs.

4.1.2. Microsoft Redistributables

PGI Visual Fortran includes Microsoft Open Tools, the essential tools and libraries required to compile, link, and execute programs on Windows. PVF 2016 installed for Microsoft Visual Studio 2015 includes the latest version, version 14.0, of the Microsoft Open Tools; PVF 2016 installed for Microsoft Visual Studio 2013 includes the previous version, version 12.0, of the Microsoft Open Tools in order to maintain compatibility with Visual C++ 2013.

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 associated license agreements.

On Windows, runtime libraries built for debugging (e.g. `msvcrtd` and `libcmtd`) are not included with PGI Visual Fortran. 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.

Chapter 5. 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.

5.1. PVF IDE Limitations

The issues in this section are related to IDE limitations.

- When moving a project from one drive to another, all `.d` files for the project should be deleted and the whole project should be rebuilt. When moving a solution from one system to another, also delete the solution's Visual Studio Solution User Options file (`.suo`).
- The Resources property pages are limited. Use the `Resources | Command Line` property page to pass arguments to the resource compiler. Resource compiler output must be placed in the intermediate directory for build dependency checking to work properly on resource files.
- Dragging and dropping files in the Solution Explorer that are currently open in the Editor may result in a file becoming "orphaned." Close files before attempting to drag-and-drop them.

5.2. PVF Debugging Limitations

The following limitations apply to PVF debugging:

- Debugging of unified binaries is not fully supported. The names of some subprograms are modified in the creation of the unified binary, and the PVF debug engine does not translate these names back to the names used in the application source code. For more information on debugging a unified binary, refer to www.pgroup.com/support/tools.htm.
- In some situations, using the Watch window may be unreliable for local variables. Calling a function or subroutine from within the scope of the watched local variable may cause missed

events and/or false positive events. Local variables may be watched reliably if program scope does not leave the scope of the watched variable.

- Rolling over Fortran arrays during a debug session is not supported when Visual Studio is in Hex mode. This limitation also affects Watch and Quick Watch windows.

  Workaround: deselect Hex mode when rolling over arrays.

5.3. PGI Compiler Limitations

- Take extra care when using `-Mprof` with PVF runtime library DLLs. To build an executable for profiling, use of the static libraries is recommended. The static libraries are used by default in the absence of `-Bdynamic`.
- Using `-Mpfi` and `-mp` together is not supported. The `-Mpfi` flag disables `-mp` at compile time, which can cause runtime 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. Using the `-Mpfo` flag does not disable OpenMP processing.

5.4. 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.

5.5. 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 6. 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
Support: trs@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

All technical support is by email or submissions using an online form at:

http://www.pgroup.com/support

Phone support is not currently available.

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

PGI®