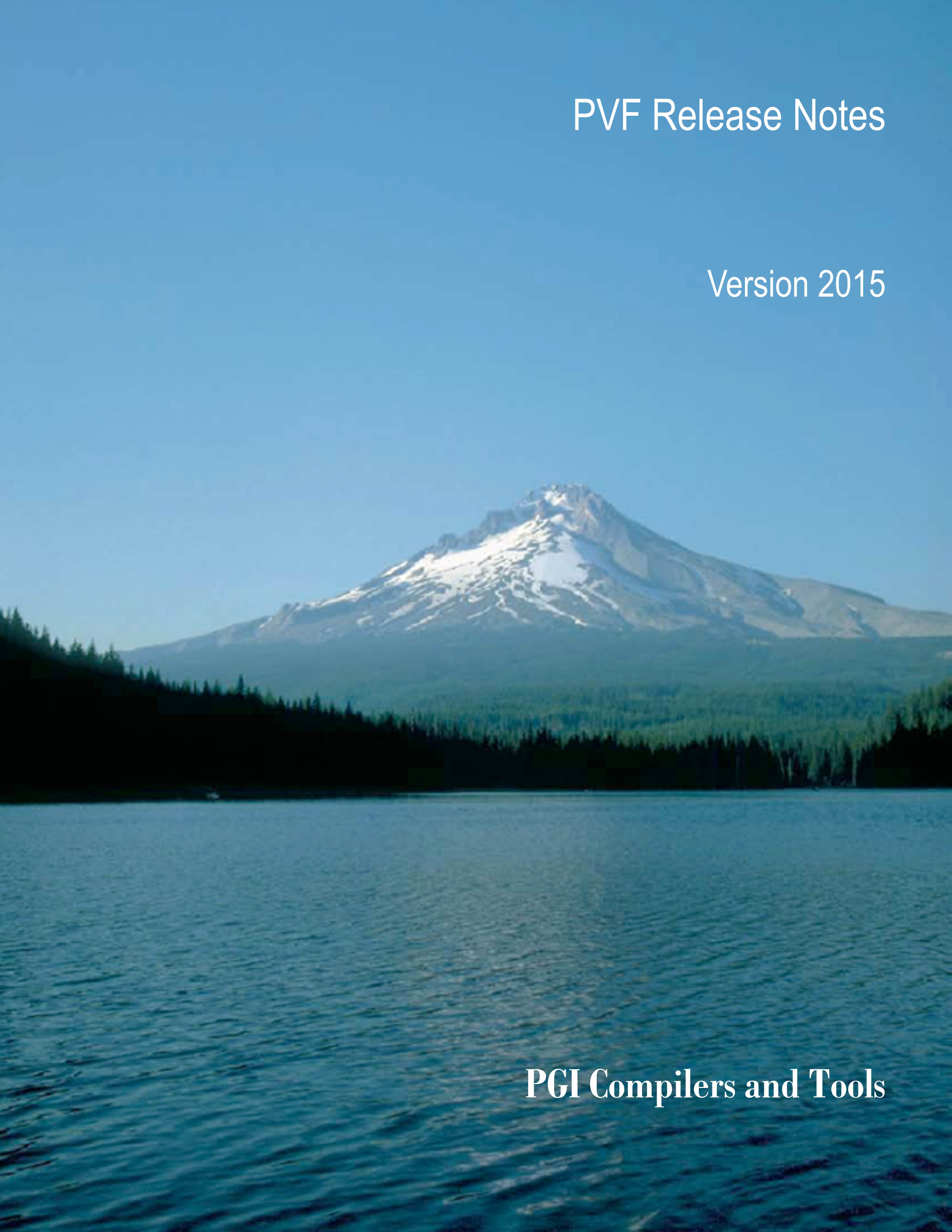

PVF Release Notes

Version 2015

PGI Compilers and Tools

TABLE OF CONTENTS

Chapter 1. PVF RELEASE OVERVIEW

Welcome to Release 2015 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 2015 compilers and tools from previous releases, and late-breaking information not included in the standard product documentation.

PGI Visual Fortran (PVF®) is licensed using FLEXnet, the flexible license management system from Flexera Software®. Instructions for obtaining a permanent license are included in your order confirmation. More information on licensing is available in the PVF Installation Guide for this release.

1.1. Product Overview

PVF is integrated with several versions of Microsoft Visual Studio. Currently, Visual Studio 2008, 2010, 2012, and 2013 are supported. Throughout this document, "PGI Visual Fortran" refers to PVF integrated with any of the four supported versions of Visual Studio. Similarly, "Microsoft Visual Studio" refers to Visual Studio 2008, VS 2010, VS 2012, and VS 2013. 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 2015 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.
- AMD Core Math Library (ACML), version 5.24.0 for Windows x64 and version 4.4.0 for 32-bit Windows.
- 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 2013 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 8.1 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:

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.

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.

Chapter 2. NEW AND MODIFIED FEATURES

This section provides information about the new and modified features of Release 2015 of PGI Visual Fortran.

2.1. What's New in Release 2015

Further information about most of these topics is included in the following sections.

15.9 Updates and Additions

- Beta support for the OpenACC parallel programming model on multicore x86 CPUs; whereas the `-ta=tesla` and `-ta=radeon` compiler options offload OpenACC parallel loops and regions to execute on a GPU, the new `-ta=multicore` compiler option maps OpenACC loops and regions to execute in parallel on multicore x86 CPUs.
    - Compile OpenACC programs for parallel execution across all cores of a multicore CPU or multi-socket CPU server
    - Incrementally parallelize applications for multicore CPUs and GPUs using the OpenACC KERNELS directive and PGI compiler optimization and parallelization feedback feature
    - Use a uniform parallel programming model across CPUs and GPUs in Fortran, C and C++
    - Write scalable OpenACC source code that will compile and run in parallel on NVIDIA GPUs, Radeon GPUs or multicore CPUs
- CUDA 7.0 toolkit support is now default; the CUDA 7.5 toolkit is integrated with this release and supported using a compile- and link-time option
- Numerous PGI Accelerator compiler improvements including:
    - Auto-privatization of scalar and array variables in OpenACC compute regions
    - New conditional sentinels for CUDA Fortran (!@CUF) and OpenACC (!@acc) with functionality similar to the OpenMP !$ conditional sentinel

  - Support for CUDA 7.0 and 7.5 Thrust headers
  - A new cuRAND Fortran module to enable calls to cuRAND with CUDA Fortran DEVICE arrays
  - An updated version of the CUDA 7.0 cuFFT library
  - Support for Fortran computed goto in OpenACC regions
- Platform updates
  - Support for Windows 10
  - OpenMPI 1.8.8 is now included in the PGI CDK Cluster Development Kit
  - 32-bit applications are no longer supported with CUDA-x86
- Over 30 user-requested fixes and updates.

15.7 Updates and Additions

- PGI Fortran/C/C++ Compilers
  - Incremental Fortran 2008 features: new iso_c_binding function C_sizeof, new iso_Fortran_env functions compiler_options and compiler_version, allocating polymorphic variables using the MOLD= specifier, associating non-pointer actual arguments with pointer dummy arguments in procedure calls.
  - Improved SIMD vectorization of loops with logical operations and expressions involving constants
  - Improved performance of intrinsic function calls where one or more input arguments are compile-time constants
- PGI Accelerator Fortran/C/C++ OpenACC Compilers
  - Support for Maxwell GPUs including Titan X
  - The default compute capability list for OpenACC and CUDA Fortran compilation for NVIDIA Tesla targets is now cc20, cc30, cc35, and cc50, generating device code for Fermi, Kepler K10 and K20, and Maxwell. The generation of device code can be time consuming, so you may notice an increase in compile time. The default can be overridden; refer to Default Compute Capability for details.
  - The `-ta=tesla:beta` flag in PGI 15.7 will enable 128-bit load and store operations in the generated code, which can produce dramatic performance improvements when working on short structures in the innermost loop. This feature is currently under beta test, and may become the default in a future release.

15.5 Updates and Additions

- Improved OpenACC user experience via enhanced compiler messages.
- Bounds-checking is not supported in device code; the -Mbounds option has been disabled.
- Argument removal during IPA linking is now disabled for accelerator code.

- OpenACC Fortran for NVIDIA GPUs and CUDA Fortran now supports list-directed i/o to the default output unit (`PRINT *` or `WRITE(*,*)`) in device kernels. See the CUDA Fortran Programming Guide and Reference, section 3.6.7, for currently supported data types.
- A number of problems are corrected 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.

15.4 Updates and Additions

- PGI Fortran Compiler
  - Incremental Fortran 2008 features: transformational Bessel functions, storage size intrinsic, complex inverse trigonometric intrinsics, function for C sizeof in iso_c_binding, find location in an array
- PGI Accelerator Fortran/C/C++ OpenACC Compilers
  - CUDA 6.5 targeted by default
  - Integrated CUDA 7.0 toolkit
  - New OpenACC SDK examples

15.3 Updates and Additions

- A number of problems are corrected 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.

15.1 Updates and Additions

- PGI Fortran Compiler
  - Incremental Fortran 2008 features
  - Support for automatic arrays in OpenACC routines and CUDA Fortran
  - New CUDA Fortran intrinsics
  - New CUDA Fortran cuSPARSE module
- PGI Accelerator OpenACC Fortran/C/C++ Compilers
  - Comprehensive OpenACC 2.0 support
  - Support for OpenACC CUPTI-based profiling
  - Support for CUDA 6.0/6.5 and NVIDIA Kepler K40/K80 GPUs
  - New OpenACC SDK examples

2.2. New and Modified Compiler Options

Accelerator device code generation for the 64-bit compilers has changed to use nvvm by default. Use `-ta=tesla:nollvm` or `-Mcuda=nollvm` to use the legacy CUDA-C code generator.

Release 2015 supports a number of new command line options as well as new keyword suboptions for existing command line options.

New compiler options include the following:

- `-M[no]idiom` — Enable [disable] loop idiom recognition.
- `--install` — Rerun makelocalrc.

Modifications to the `-Mcuda` suboptions include the deprecation of `cc1x`, `cc1+`, `tesla` and `tesla+` as well as these changes:

- `charstring` — Enable limited support for character strings in GPU kernels.
- `[no]lineinfo` — Generate GPU line information (replaces `[no]lineninfo`).
- `[no]llvm` — The LLVM back end is now the default in 64-bit mode.

Modifications to the `-ta=tesla` suboptions include the deprecation of `cc1x`, `cc1+`, `tesla`, `tesla+` and `[no]required` as well as these changes:

- `[no]lineinfo` — Generate GPU line information (replaces `[no]lineninfo`).
- `[no]llvm` — The LLVM back end is now the default in 64-bit mode.

Modifications to the `-ta=radeon` suboptions include the deprecation of `[no]required` as well as these changes:

- `[no]lineinfo` — Generate GPU line information (replaces `[no]lineninfo`).
- `[no]llvm` — The LLVM/SPIR back end is now the default in 64-bit mode.
- `spir` — Use the LLVM/SPIR back end, which is now the default in 64-bit mode.

Modifications to the `-acc` suboptions include the deprecation of the `[no]required` suboption.

2.3. New and Modified Fortran Functionality

- Incremental Fortran 2008 features including SIMPLY CONTIGUOUS, ERF/ERFC, inverse hyperbolic functions, elemental and transformational Bessel functions, storage size intrinsic, gamma, log_gamma, hypot, complex inverse trigonometric intrinsics, function for C sizeof in iso_c_binding, and find location in an array.
- Enhanced ANSI-like preprocessing for Fortran.
- Enhanced SIMD vectorization: conditionals, vectorization profitability analysis.

2.4. New and Modified CUDA Fortran Functionality

NVIDIA is dropping support for 32-bit CUDA toolkits. PGI's CUDA support plan will follow:

32-bit Systems

- Linux: CUDA 6.5 will be the default version of CUDA through PGI 15.10
- OS X: None
- Windows: CUDA 6.5 will be the default version of CUDA through PGI 15.10

PGI will stop supporting CUDA on 32-bit systems in 2016.

64-bit Systems

- Linux: Support for CUDA 6.5, 7.0, 7.5
- OS X: Support for CUDA 7.0, 7.5
- Windows: Support for CUDA 7.0, 7.5

Beginning in PGI 15.4, changes and enhancements have been made to CUDA Fortran default stream handling:

- Two new constants which can be used as stream values have been added to the cudafor module:

  ```
  INTEGER(KIND=CUDA_STREAM_KIND), PARAMETER :: cudaStreamLegacy    = 1
  INTEGER(KIND=CUDA_STREAM_KIND), PARAMETER :: cudaStreamPerThread = 2
  ```
- The `cudaGetStreamDefault` function has been renamed to `cudaforGetDefaultStream`.
- The `cudaSetStreamDefault` function has been renamed to `cudaforSetDefaultStream`.
- Streams for the sum(), maxval(), and minval() intrinsics which operate on device data can now be controlled with the functions `cudaforReductionSetStream` and `cudaforReductionGetStream`.
- If a nonzero default stream is set using `cudaforSetDefaultStream`, it will now be used for CUF Kernels and global subroutine launches if no other stream is explicitly set in the launch configuration.

For further information, please refer to the CUDA Fortran Programming Guide.

Other changes made in PGI 15.4:

- Support for CUDA 6.5/7.0 and NVIDIA Kepler K40/K80 GPUs.
- Updated cuSPARSE module with interfaces for changes in CUDA 7.0 cuSPARSE library.

These changes were made in PGI 15.1:

- New tuned host intrinsics which operate on device or managed data: sum, maxval, minval.
- New cuSPARSE module with interfaces to cuSPARSE library.
- Support for CUDA 6.0/6.5 and NVIDIA Kepler K40/K80 GPUs.
- Support for Fortran automatic arrays in CUDA Fortran kernels.
- Support for global static data in Accelerator device code.
- Full support for CUDA Unified Memory on NVIDIA GPUs.

2.5. New Features in PGI Accelerator OpenACC Compilers

Multicore Support

PGI 15.9 includes the Beta 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 `-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` may not be used in conjunction with `-ta=tesla`, `-ta=radeon` or `-ta=host`.

Default Compute Capability

As of the 15.7 release, the default compute capability list for OpenACC and CUDA Fortran compilation for NVIDIA Tesla targets is cc20, cc30, cc35, and cc50. 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.

The PGI 2015 release implements the OpenACC 2.0 release, except as described below. In addition, the PGI 2015 release has implemented some features scheduled for the next version of OpenACC.

New OpenACC 2.0 Features

- Cache directive
- Auto loop clause
- Device_type clause
- Collapse clause
- Firstprivate clause
- Gang(num:) and gang(static:) loop subclauses
- Reduction in routine clause
- Use of complex data types in reduction clause

OpenACC 2.0 Missing Features

- 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.

Present_or_

The PGI 2015 release has changed the behavior of the **`copy`**, **`copyin`**, **`copyout`** and **`create`** clauses so they will behave the same as **`present_or_copy`**, **`present_or_copyin`**, **`present_or_copyout`** and **`present_or_create`**. This will not change the behavior of any existing correct OpenACC program. A program that would fail with a runtime error message that data in a data clause is already present on the device will now continue to execute. This behavior will be part of the next OpenACC specification version.

Shortloop Clause

The PGI compilers allow an additional loop clause, **`shortloop`**. If the **`shortloop`** clause appears on a loop directive with the **`vector`** clause, it tells the compiler that the loop trip count is less than or equal to the number of vector lanes created for that loop. This means the value of the **`vector()`** clause on the loop directive in a kernels region, or the value of the **`vector_length()`** clause on the parallel directive in a parallel region will be greater than or equal to the loop trip count. This allows the compiler to generate more efficient code for the loop.

Expressions for Vector Length

Previously, the expression in the **`vector_length`** clause on the **`parallel`** directive, and the expression in the **`vector`** clause on the loop directive in a kernels region was limited to compile-time constant expressions. The PGI compilers now allow a runtime expression in this clause. If the value exceeds the limits of the target device, the program will fail at runtime. The current limit for NVIDIA CUDA devices is 1024 for the product of the vector length and the number of workers. The current limit for AMD Radeon devices is 256 for the product of the vector length and the number of workers.

Fortran Derived Type Member References In Data Clauses

This release allows Fortran derived type members to appear in OpenACC data clauses. The member may be an allocatable, pointer or direct array. If the member is an allocatable or pointer array and the parent derived type variable is also present on the device, the corresponding pointer in the device copy of the parent derived type variable is also updated to point to the device copy of the member.

```
type objects
  real, allocatable, dimension(:) :: temperature
  integer, allocatable, dimension(:) :: location
end type
    . . .
type(objects) :: x
    . . .
!$acc enter data copy(x, x%temperature)
    . . .
!$acc parallel loop
    do i = 1, n
       x%temperature(i) = findtemp(i)
    enddo
```

Support for Profiler / Trace Tool Interface

This release implements a preliminary 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. New and Modified Runtime Library Routines

PGI 2015 supports new runtime library routines associated with the PGI Accelerator compilers.

For more information, refer to *Using an Accelerator* in the 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 2015 is not compatible with previous releases. If you use `Tools | Options` to target a release prior to 2015, 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 all platforms. 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 2015 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\15.9\REDIST
```

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

```
C:\Program Files\PGI\win64\15.9\REDIST
C:\Program Files(x86)\PGI\win32\15.9\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 2015 installed on Windows 7, 8, 8.1, and Server 2012 includes the latest version, version 12, of the Microsoft Open Tools.

The Microsoft Open Tools directory contains a subdirectory named `REDIST`. PGI 2015 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 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.
- The `–i8` option can make programs incompatible with the ACML library; use of any INTEGER*8 array size argument can cause failures with these libraries. Visit developer.amd.com to check for compatible ACML libraries.
- ACML is built using the `–fastsse` compile/link option, which includes `–Mcache_align`. When linking with ACML on 32-bit Windows, all program units must be compiled with `–Mcache_align`, or an aggregate option such as `–fastsse`, which incorporates `–Mcache_align`. This process is not an issue on 64-bit targets where the stack is 16-byte aligned by default. You can use the lower-performance, but fully portable, BLAS and LAPACK libraries on CPUs that do not support SSE instructions.

5.4. CUDA Toolkit Issues

Targeting a CUDA Toolkit Version

- The CUDA 6.5 Toolkit is set as the default in PGI 15.9. To use the CUDA 6.5 Toolkit, first download the CUDA 6.5 driver from NVIDIA at www.nvidia.com/cuda.
- You can compile with the CUDA 7.0 Toolkit either by adding the option `-ta=tesla:cuda7.0` to the command line or by adding `set DEFCUDAVERSION=7.0` to the `siterc` file.
- `pgaccelinfo` prints the driver version as the first line of output. For a 6.5 driver, it prints:

```
CUDA Driver Version 6050
```

CUDA 6.5 Toolkit Limitations

- Support for CUPTI, and therefore PGI_ACC_TIME, is not provided in 32-bits.
- The CUDA 6.5 Toolkit is not supported on Windows Server 2012 although it is supported on Windows Server 2012 R2.

CUDA 7.0 Toolkit Limitations

- The CUDA 7.0 Toolkit is not available for Windows 32-bit nor is it supported on Windows Server 2012; it is, however, supported on Windows Server 2012 R2.

5.5. OpenACC Issues

This section includes known limitations in PGI's support for 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 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.
- Reductions in gang-scheduled loops in procedures with `acc routine gang` are not supported. This is being added as a restriction in the next OpenACC specification.
- Fortran assumed-shape arguments are not yet supported.

Clause Support Limitations

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

OpenMP Limitations

- There are known problems when using OpenACC for GPU accelerators inside of OpenMP parallel regions. This problem will be fixed in the near future.

5.6. 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 e-mail 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 or in your local copy of the documentation in the release directory `doc/index.htm`.

PGI®