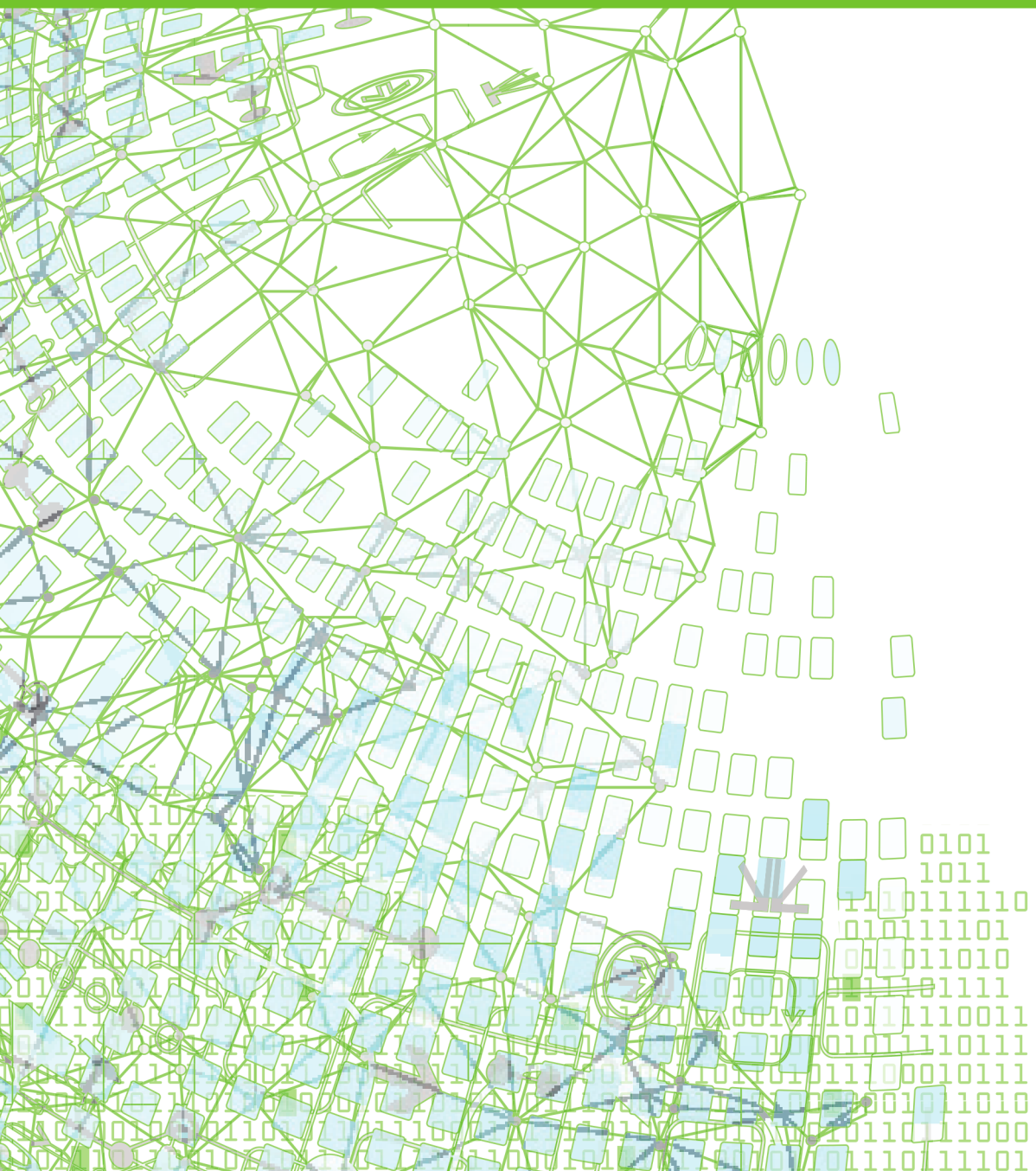
PGI® COMPILERS & TOOLS

RELEASE NOTES FOR X86 CPUS

Version 2017



NVIDIA

TABLE OF CONTENTS

Chapter 1. Release Overview....1
1.1. Product Overview....1
1.1.1. Licensing Terminology....1
1.1.2. Bundled License Key....2
1.1.3. Node-locked and Network Floating Comparison....2
1.2. Release Components....2
1.2.1. Additional Components for PGI Network Floating Licenses....3
1.2.2. MPI Support....3
1.3. Terms and Definitions....3
1.4. Supported Platforms....4
1.5. Supported Operating System Updates....4
1.5.1. Linux....4
1.5.2. Apple macOS....4
1.5.3. Microsoft Windows....5
1.6. Precompiled Open-Source Packages....5
1.7. Getting Started....6
Chapter 2. New and Modified Features....8
2.1. What's New in Release 2017....8
2.2. OpenACC....16
2.3. OpenMP....19
2.4. C++ Compiler....20
2.4.1. C++ and OpenACC....20
2.4.2. C++ Compatibility....20
2.5. Runtime Library Routines....21
2.6. Library Interfaces....21
2.7. Environment Modules....21
2.8. Beta LLVM Code Generator....21
Chapter 3. Distribution and Deployment....23
3.1. Application Deployment and Redistributables....23
3.1.1. PGI Redistributables....23
3.1.2. Linux Redistributables....23
3.1.3. Microsoft Redistributables....24
Chapter 4. Troubleshooting Tips and Known Limitations....25
4.1. Platform-specific Issues....25
4.1.1. Linux....25
4.1.2. Apple macOS....25
4.1.3. Microsoft Windows....25
4.2. Issues Related to Debugging....26
4.3. Profiler-related Issues....26
4.4. OpenACC Issues....26

Chapter 5. Contact Information..... 28

LIST OF TABLES

Table 1 Typical -fast Options 7

Table 2 Additional -fast Options 7

Table 3 Pool Allocator Environment Variables 17

Chapter 1. RELEASE OVERVIEW

Welcome to Release 2017 of the PGI Accelerator™ C11, C++14 and Fortran 2003 compilers and development tools for 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, http://www.pgicompilers.com/resources/docs/17.10/pdf/pgi17ug-x86.pdf.

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, https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-8277. These new license daemons also work with older PGI releases. We recommend all users update their license daemons—see the licensing FAQ, https://www.pgicompilers.com/support/faq.htm for more information. This FlexNet update also requires you to update your PGI FlexNet license keys, https://www.pgicompilers.com/license/pin_manage.php?view=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** – also known as the End-user License Agreement (EULA), this is a legal agreement between NVIDIA and PGI end-users, to which users assent upon installation of any PGI product. The terms of the License, http://www.pgicompilers.com/LICENSE are kept up-to-date in documents on the PGI

website. You can also find a copy in the `$PGI/<platform>/<rel_number>/doc` 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. For PGI Professional, License keys are generated by each PGI end-user on the PGI website 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 PGI Professional license. This PIN is included in your order confirmation. The PIN can also be found in your license key 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 user account, https://www.pgicompilers.com/account/index.php for administrative purposes. This code is provided by PIN owners to others whom they wish tied to their PIN(s).

1.1.2. Bundled License Key

Installation may place a temporary 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 this version, it is recommended that you remove or rename the license key file in the installation directory.

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.

1.2. Release Components

Release 2017 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.
- PGI Profiler® OpenACC, CUDA, OpenMP, and multi-thread graphical profiler.
- PGI Debugger® MPI, OpenMP, and multi-thread graphical debugger.

- Open MPI version 2.1.2 for 64-bit Linux including support for NVIDIA GPUDirect. GPUDirect requires CUDA 7.5 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 development environments (Windows only).
- BLAS and LAPACK library based on the customized OpenBLAS project source.
- A UNIX-like shell environment for 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 (formerly the PGI CDK Cluster Development Kit) include additional components available for download from the PGI website, but not contained in the installation package including:

- MVAPICH2 MPI libraries, version 2.0.
- MPICH MPI libraries, version 3.2.

1.2.2. MPI Support

You can use PGI products to develop and debug MPI applications. PGI node-locked licenses support debugging of up to 16 local MPI processes. PGI network floating licenses provide the ability to debug up to 256 local or remote MPI 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 PGI online glossary located at https://www.pgicompilers.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, Haswell, and Broadwell processors.

AMD64
A 64-bit processor from AMD™ incorporating 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 three platforms supported by the PGI compilers and tools for x86-64 processor-based systems.

PGI 2017 supports 64-bit operating systems only. Compiling 32-bit applications for execution on either 32-bit or 64-bit operating systems is no longer supported on any platform.

- **64-bit Linux** – supported on 64-bit Linux operating systems running on a 64-bit x86 compatible processor.
- **64-bit macOS** – supported on 64-bit Apple macOS operating systems running on a 64-bit Intel processor-based Macintosh computer.
- **64-bit Windows** – supported on 64-bit Microsoft Windows operating systems running on a 64-bit x86-compatible processor.

1.5. Supported Operating System Updates

This section describes updates and changes to PGI 2017 that are specific to Linux, macOS, and Windows.

1.5.1. Linux

- CentOS 5+, including CentOS 7
- Fedora 6+, including Fedora 25
- OpenSUSE 11+, including OpenSUSE Leap 42.2
- RHEL 5+, including RHEL 7.3
- SLES 11+, including SLES 12 SP 2
- Ubuntu 12.04+, including Ubuntu 16.10

1.5.2. Apple macOS

PGI 2017 for macOS supports most of the features of the version 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.

- The compilers, debugger, and profiler are supported on macOS versions 10.9 (Mavericks) through 10.13 (High Sierra).

1.5.3. Microsoft Windows

PGI products for Windows support most of the features of the PGI products 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 2017 requires the Windows 10 SDK, even on Windows 7, 8 and 8.1.

These Windows operating systems are supported in PGI 2017:

- Windows Server 2008 R2
- Windows 7
- Windows 8
- Windows 8.1
- Windows 10
- Windows Server 2012
- Windows Server 2016

1.6. Precompiled Open-Source Packages

Many open-source software packages have been ported for use with PGI compilers on Linux x86-64.

The following PGI-compiled open-source software packages are included in the PGI Linux x86-64 download package:

- OpenBLAS 0.2.19 – customized BLAS and LAPACK libraries based on the OpenBLAS project source.
- Open MPI 2.1.2 – open-source MPI implementation.
- ScaLAPACK 2.0.2 – a library of high-performance linear algebra routines for parallel distributed memory machines. ScaLAPACK uses Open MPI 2.1.2.

The following list of open-source software packages have been precompiled for execution on Linux x86-64 targets using the PGI compilers and are available to download from the PGI website at https://www.pgicompilers.com/support/downloads.php.

- MPICH 3.2 – open-source MPI implementation.
- MVAPICH2 2.2 – open-source MPI implementation.
- ESMF 7.0.0 for Open MPI 2.1.2 – The Earth System Modeling Framework for building climate, numerical weather prediction, data assimilation, and other Earth science software applications.
- ESMF 7.0.0 for MPICH 3.2
- ESMF 7.0.0 for MVAPICH2 2.2
- NetCDF 4.4.1.1 for GCC 4.x – A set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing

of array-oriented scientific data, written in C. Included in this package are the following components:

  - NetCDF-C++ 4.3.0 – C++ interfaces to NetCDF libraries.
  - NetCDF-Fortran 4.4.4 – Fortran interfaces to NetCDF libraries.
  - HDF5 1.10.0-patch1 – data model, library, and file format for storing and managing data.
  - CURL 7.46.0 – tool and a library (usable from many languages) for client-side URL transfers.
  - SZIP 2.1 – extended-Rice lossless compression algorithm.
  - ZLIB 1.2.8 – file compression library.
- NetCDF 4.4.1.1 for GCC 5.x – includes all the components listed in NetCDF for GCC 4.x above.
- Parallel NetCDF 1.7.0 for MPICH 3.2
- Parallel NetCDF 1.7.0 for MVAPICH2 2.2
- Parallel NetCDF 1.7.0 for Open MPI 2.1.2

In addition, these software packages have also been ported to PGI on Linux x86-64 but due to licensing restrictions, they are not available in binary format directly from PGI. You can find instructions for building them in the Porting & Tuning Guides section of the PGI website at https://www.pgicompilers.com/resources/tips.htm.

- FFTW 2.1.5 – version 2 of the Fast Fourier Transform library, includes MPI bindings built with Open MPI 2.1.2.
- FFTW 3.3.4 – version 3 of the Fast Fourier Transform library, includes MPI bindings built with Open MPI 2.1.2.

For additional information about building these and other packages, please see the Porting & Tuning Guides section of the PGI website at https://www.pgicompilers.com/resources/tips.htm.

1.7. Getting Started

By default, the PGI 2017 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.

Aggregate options incorporate a generally optimal set of flags for targets that support SSE capability. These options incorporate optimization options to enable use of vector streaming SIMD instructions . They enable vectorization with SSE instructions, cache alignment, and flushz.

The content of the -fast option is host-dependent.

The following table shows the typical -fast options.

Table 1 Typical -fast 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 also typically includes the options shown in the following table:

Table 2 Additional -fast Options

Use this option...	To do this...
-Mvect=simd	Generates packed SSE and AVX instructions.
-Mcache_align	Aligns long objects on cache-line boundaries.
-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 -fast option.

In addition to -fast, 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 2017 of the PGI compilers and tools.

Many user-requested fixes and updates are implemented in each PGI release. Refer to the Technical Problem Reports, https://www.pgicompilers.com/support/release_tprs.htm webpage for a complete and up-to-date table of TPRs fixed in recent PGI releases. This table contains a summary description of each problem as well as the version in which it was fixed.

2.1. What's New in Release 2017

17.10 Updates and Additions

- Integrated full support for the official release of the CUDA Toolkit 9.0; resolved PGI 17.9 limitation involving changes to warp shuffle instructions in CUDA 9.0. CUDA 7.5 remains the default version used by the compilers. To use either CUDA 8.0 or CUDA 9.0 instead, add either the `cuda8.0` or `cuda9.0` sub-option to the `-ta=tesla` or `-Mcuda` compile- and link-time options.
- Implemented initial support for CUDA 9.0 cooperative groups in CUDA Fortran. Refer to the CUDA Fortran Programming Guide for details.
- Implemented support for the OpenMP 4.5 **`taskloop`** construct and all its clauses except **`firstprivate`** and **`lastprivate`**. OpenMP 4.5 programs currently support parallel execution across all the cores of a multicore CPU or server; OpenMP 4.5 is supported on Linux/x86 with the Beta LLVM code generator.
- Updated the version of the PGI-built Open MPI libraries to 2.1.2 for Linux.
- Added support for macOS version 10.13 (High Sierra) and Xcode 9.

17.9 Updates and Additions

- PGI Accelerator Compilers

Integrated support for the CUDA Toolkit 9.0 Release Candidate (RC). CUDA 7.5 remains the default version used by the compilers. To use either CUDA 8.0 or CUDA 9.0 instead, add either the `cuda8.0` or `cuda9.0` sub-option to the `-ta=tesla` or `-Mcuda` compile- and link-time options.

There are known problems using PGI 17.9 compilers with CUDA 9.0 installations. In particular, CUDA 9.0 introduces new synchronizing warp-shuffle instructions. The compilers in this PGI release still use older and now unreliable warp shuffle instructions to share values across threads in a warp for reduction operations and to copy values from vector lane zero to all vector lanes, in routines with the **`acc routine vector`** directive. This problem can occur even on Kepler and Maxwell GPUs. This issue is corrected in the PGI 17.10 release.

17.7 Updates and Additions

- PGI Accelerator Compilers
  - Added support for Tesla V100 GPUs in PGI OpenACC and CUDA Fortran. Based on the new NVIDIA Volta GV100 GPU, Tesla V100 offers more memory bandwidth, more streaming multiprocessors, next generation NVLink and new microarchitectural features that add up to better performance and programmability. For OpenACC and CUDA Fortran programmers, Tesla V100 offers improved hardware support and performance for CUDA Unified Memory features on both x86-64 and OpenPOWER processor-based systems. Use the sub-option `cc70` with the `-ta=tesla` or `-Mcuda` compile- and link-time options to generate code for the Tesla V100 GPUs; the CUDA 9 toolkit must be used when targeting Volta GPUs.
  - Added initial support for the CUDA 9 toolkit. Requirements for using CUDA 9 with the PGI 17.7 compilers and profiler:
    - Install the release candidate (RC) or early access (EA) version of the CUDA 9 toolkit.
    - Specify the location of the CUDA 9 toolkit using the compiler option CUDAROOT; for example, if CUDA 9 is installed in `/opt/cuda-9.0`, add **`CUDAROOT=/opt/cuda-9.0`** when compiling, linking, or invoking the profiler.
    - Use the sub-option `cuda9.0` with the `-ta=tesla` or `-Mcuda` compile- and link-time options.
  - Added support for CUDA Unified Memory. PGI 17.7 compilers leverage Pascal and Volta GPU hardware features, NVLink, and CUDA Unified Memory to simplify OpenACC and CUDA Fortran programming on GPU-accelerated x86-64 and OpenPOWER processor-based servers. When CUDA Fortran or OpenACC allocatable data is placed in CUDA Unified Memory,

no explicit data movement or data directives are needed. This simplifies GPU acceleration of applications that make extensive use of allocatable data, and allows you to focus on parallelization and scalability of your algorithms. Use the `-ta=tesla:managed` compiler option to enable this feature. In previous PGI releases, compiler support for CUDA Unified Memory was provided as an optionally-installed, evaluation feature.

- Added a pool allocator to minimize the number of calls to cudaMallocManaged() for CUDA Unified Memory support. The pool allocator is enabled by default when `-ta=tesla:managed` or `-ta=tesla:pinned` is used.
- Added Beta support for automatic deep copy of Fortran derived types. This feature allows you to port applications with modern deeply nested data structures to Tesla GPUs using OpenACC. The PGI 17.7 compilers allow you to list aggregate Fortran data objects in OpenACC **`copy`**, **`copyin`**, **`copyout`** and **`update`** directives to move them between host and device memory including traversal and management of pointer-based objects within the aggregate data object. When enabled, full deep copy ensures that, when moving Fortran variables of derived type from host to device or device to host, the entire data structure, including pointers and allocatable arrays, is copied back and forth between host and device, or device and host, memory. To enable deep copy, use the `deepcopy` sub-option to `-ta=tesla`. Two things to note: polymorphic data types are not supported, and the presence of overlapping pointers may cause runtime errors.
- The cuSOLVER library is based on the cuBLAS and cuSPARSE libraries, and supports matrix factorization and triangular solve routines for dense matrices, a sparse least-squares solver and an eigenvalue solver, and provides a refactorization library useful for solving sequences of matrices with a shared sparsity pattern. You can now call optimized cuSolverDN routines from CUDA Fortran and OpenACC Fortran using the PGI-supplied interface module and the PGI-compiled version of the cuSOLVER libraries bundled with PGI 17.7 for Linux/x86-64 and OpenPOWER. This same cuSolver library is callable from PGI OpenACC C/C++ as they are built using PGI compilers and are compatible with and use the PGI OpenMP runtime.

- All PGI Compilers
  - Added a Beta LLVM code generator and OpenMP runtime to the PGI Linux/x86-64 compilers. The PGI Beta compilers are available as a separate download and installation package. While there are some limitations relative to the production PGI compilers, the Beta release includes support for all OpenACC, CUDA Fortran and OpenMP 4.5 features and delivers significant performance improvements on many C++ applications. For more details, see Beta LLVM Code Generator.

- Added initial support for OpenMP 4.5 syntax and features. The PGI Fortran, C, and C++ compilers will now compile most OpenMP 4.5 programs for parallel execution across all the cores of a multicore CPU or server. **`target`** regions are implemented with default support for the multicore host as the target, and **`parallel`** and **`distribute`** loops are parallelized across all OpenMP threads. This feature is supported on Linux/x86 platforms with the Beta LLVM code generator only.
- Improved inlining with the `-Minline` option. You may see different functions inlined with this release than you observed in prior releases. In some cases, compilation may take noticeably longer to complete because of an increase in the number of functions inlined. Added the `smallsize:number` sub-option tomanage `-Minline` which you can use to always inline functions of size smaller than number regardless of other size limits.
- Added a new feature for program exploration called Program Analysis Summary Output (PASO). PASO enables the exporting of information gathered from the internal representation (IR) of various compilation stages in an organized, human readable format. This data forms a detailed summary of the program's internal structure, spanning not just one but all the files that make up a project. To enable PASO, compile and link with the `-Msummary` option. To find out more about how PASO works and how you might leverage it, refer to Program Analysis Using Summary Output, https://www.pgicompilers.com/blogs/posts/paso.htm. PASO is available on all platforms supported by PGI except macOS.

- C++ Compiler
  - Improved the raw performance on the LCALS loops benchmarks by an average of 20%, as well as eliminated the lambda abstraction penalty associated with these loops.
  - Added support for lambdas with capture in OpenACC GPU-accelerated compute regions.
  - Added incremental support for C++17 features consistent with EDG 4.13. This support includes:
    - new rules for auto deduction from braced-init-list
    - allow typename in template template parameter list
    - attributes for namespaces and enumerators
    - u8 character literals
    - nested namespace definitions
    - extended static_assert
    - generalized range-based for
    - remove deprecated register keyword
    - remove deprecated operator++(bool)
    - __has_include

        - [[fallthrough]]
        - [[nodiscard]]
        - [[maybe_unused]]
        - hexadecimal floating literals
        - using attribute namespaces without repetition
        - standard and non-standard attributes
    - Added support for interoperability with GNU 6.3.
    - Added support for the PGI C++ compilers as a CUDA 9.0 NVCC host compiler.
- Profiler

  The following new profiler features are enabled only when invoking the profiler with CUDA 9. To use CUDA 9 with PGI 17.7, start the profiler with the **CUDAROOT** option set to the CUDA 9 installation location. For example, if CUDA 9 is installed in `/opt/cuda-9.0`, add **CUDAROOT=/opt/cuda-9.0** when launching the profiler.

    - Enhanced unified memory profiling:
        - Added association between unified memory events/CPU page faults and the source location of memory allocation.
        - Added new unified memory profiling events for page thrashing, throttling and remote map.
        - Added support for switching between segment and non-segment timelines.
        - Added filtering of events based on the virtual address, migration reason or page fault access type.
    - Added an OpenACC details table summarizing each recorded event.
    - Added support for the profiling of cooperative kernel launches.
    - Added NVLink events to the timeline.
    - Added memory throughput to the NVLink topology diagram.
    - Increased the options available for multi-hop remote profiling.
    - Added support for OpenACC profiling on all multicore systems.
- Documentation

  Added HTML-formatted documentation for all PGI compilers and tools online at https://www.pgicompilers.com. The new format should make it easier to use any Internet-connected system or device to search or reference PGI documentation. PDF versions of all documents also remain available.
- Operating Systems
    - Added support for Fedora 25 and openSUSE 13.2.

17.5 Updates and Additions

- CUDA Toolkit

  - This version includes an updated nvlink linker utility that addresses an issue with the link ordering of object files on Linux.
- Operating Systems
  - Added support for Ubuntu 17.04 and RHEL 6.9 Linux distributions.

17.4 Updates and Additions

- PGI Accelerator Compilers
  - Added support for atomic add and atomic subtract in device code for Fortran's single precision complex data types.
  - Added device versions of isfinite, isnan, isinf, and round.
  - Added support for CUDA Fortran warp-vote operations.
  - Changed the impact of `-g` on the optimization level used when compiling device code. For host code, `-g` sets the compiler's optimization level to zero unless another optimization level is specified using a `-O` option. For device code, the same is now true. For details about generating debug information for device code, refer to the PGI Compiler Reference Guide's section on DWARF Debugging Formats.
  - Updated the version of nvlink to 8.0.73; this version of the device linker allows an arbitrary ordering of object files on its command line.
- PGI Tools

  Improved debugging support of C++ programs on macOS versions 10.11 (El Capitan) and 10.12 (Sierra).

17.3 Updates and Additions

- Added debugger support for macOS version 10.12 (Sierra).
- Updated the PGI C/C++ compiler plug-in for Eclipse versions Neon and Mars; the Eclipse plug-in is available for Linux platforms.

17.1 Updates and Additions

- All PGI Compilers
  - Updated how floating point divides are computed to guarantee results will be uniform for scalar and vector operations. This change can cause numerical differences between PGI 17.1 and previous PGI releases. In rare cases, this change can cause an increase in execution time.
  - Improved inlining with the `-Minline` option. You may see different functions inlined with this release than you observed in prior releases. In some cases, compilation may take longer to complete because of an increase in the number of functions inlined. Some of the `-Minline` sub-options, which you can use to control inlining, have also changed from previous releases:

        - Added `totalsize:n` to limit inlining to the total size of n where n is the size of the combined program units on a per file basis.
        - Changed `size:n` to `maxsize:n` which allows inlining only of functions smaller than approximately n lines. The compilers silently convert the previous `size:n` to `maxsize:n`.
        - Dropped `levels:n` which limited inlining to n levels of functions. The compilers silently ignore `levels:n`.
    - Added the `-cpp` option as an alias for the `-Mpreprocess` option.
- PGI C++ Compiler
    - Integrated EDG release 4.13 and enabled interoperability with GCC 5.2, 5.3, 5.4, 6.0, 6.1, 6.2, and 6.3. To take full advantage of the new C++11 features supported by GCC 5.1 and later, use the PGC++ compiler option `--c++11`.
    - Comprehensive support for C++14; co-installation with GCC version 5.0 or greater is required. Enable C++14 features with the PGC++ compiler option `--c++14`.
    - Added C++11 support to PGC++ when PGC++ is used as the nvcc host compiler; this support requires a patched post-CUDA-8.0-production version of nvcc. Contact sales@pgroup.com for more information.
    - Improved exception handling support and dropped the following related compiler options: `--[no_]exceptions`, `--need_exception_spec`, `--sjlj_eh`, `--[no]zc_eh`.
    - Added `-M[no]variadic_macros` to allow or disallow variadic macros. Variadic macros are allowed by default.
    - Deprecated the compiler options `--edg` and `--gnu`.
    - 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 Accelerator OpenACC Compilers
    - Dropped support for CUDA 7.0.
    - Changed the default version of the CUDA Toolkit used by the compilers from CUDA 7.0 to 7.5. The CUDA Toolkit 8.0 can be used instead of 7.5 by adding the sub-option `cuda8.0` to the `-ta=tesla` or `-Mcuda` compile- and link-time options.
    - Removed compute capability 2.0 (Fermi) from the compilers' default set of compute capabilities. The `cc20` sub-option to the `-ta=tesla` and `-Mcuda` options is deprecated.
    - Improved data management within the OpenACC **`cache`** directive.

- Added support for additional OpenACC 2.5 features:
  - Changed the behavior of the **exit data** directive to decrement the dynamic reference count.
  - Added the new optional **finalize** clause to set the dynamic reference count to zero.
  - Added the **if_present** clause to the **update** directive which changes the behavior when data is not present from a runtime error to a no-op.
  - Added new **init**, **shutdown**, and **set** directives.
  - Added new API routines to get and set the default async queue value.
  - Added support for the new definition of **routine bind** clause.
  - Updated the value of **_OPENACC** to 201510.

  With the exception of nested parallelism, declare link, and adding restrictions to **cache** clause variable references to variables within a cached region, OpenACC 2.5 feature support is complete in PGI 2017.
- PGI Tools
  - Updated the version of the JRE included in all installation packages to version 1.8.0_112.
  - Added variable rollover to the PGI debugger's graphical interface.
  - The PGI debugger requires libncurses.so.5. Some newer Linux distributions, such as Fedora 24, do not include this version of this library by default. In this situation, install the ncurses compatibility library to use the debugger.
- Libraries
  - PGI products for Linux ship with Open MPI 1.10.2.
  - Users with PGI subscription support can access separate downloads of MVAPICH 2.2 and MPICH 3.2.
  - PGI-built versions of NetCDF 4.4.1.1 and Parallel NetCDF 1.7.0 are available online at http://www.pgicompilers.com/resources/tips.htm
  - PGI-built versions of the Earth System Modeling Framework (ESMF) 7.0.0, one per PGI-built MPI distribution, are available online at the address above.
- Other Features and Additions
  - Added support for these new Linux x86-64 operating systems: Fedora 24, RHEL 7.3, Ubuntu 16.10.
  - Added compiler and profiler support for macOS version 10.12 (Sierra); the PGI debugger is not supported on Sierra in PGI 17.1.
  - Added support for Microsoft Windows Server 2016.
- Deprecations and Eliminations
  - PGI 2017 supports 64-bit operating systems only. Compiling 32-bit applications for execution on either 32-bit or 64-bit operating systems is no longer supported on any platform.

- The PGI 2017 release for macOS no longer supports CUDA Fortran and OpenACC running on GPUs or CUDA-x86 running on CPUs. OpenACC targeting multi-core CPUs is still supported using the `-ta` sub-options `host` and `multicore`. The `-acc` option now implies `-ta=multicore` on macOS.
- Dropped support for the CUDA 7.0 toolkit.
- Dropped support for OS X 10.7 (Lion) and 10.8 (Mountain Lion).

2.2. OpenACC

CUDA Unified Memory

Beginning with the PGI 17.7 release, the use of CUDA Unified Memory for allocatable data moved from a Beta feature to production. This feature, described in detail in the OpenACC and CUDA Unified Memory, https://www.pgicompilers.com/lit/articles/insider/v6n2a4.htm *PGInsider* article, is available with the Linux/x86-64 and Linux/OpenPOWER compilers. It is supported on Linux/x86-64 using both the default PGI code generator and the Beta LLVM-based code generator. To enable this feature, add the option `-ta=tesla:managed` to the compiler and linker command lines.

In the presence of `-ta=tesla:managed`, all C/C++/Fortran explicit allocation statements in a program unit are replaced by equivalent "managed" data allocation calls that place the data in CUDA Unified Memory. Managed data share a single address for CPU/GPU and data movement between CPU and GPU memories is implicitly handled by the CUDA driver. Therefore, OpenACC data clauses and directives are not needed for "managed" data. They are essentially ignored, and in fact can be omitted.

When a program allocates managed memory, it allocates host pinned memory as well as device memory thus making allocate and free operations somewhat more expensive and data transfers somewhat faster. A memory pool allocator is used to mitigate the overhead of the allocate and free operations. The pool allocator is enabled by default for `-ta=tesla:managed` or `-ta=tesla:pinned`. Beginning with the PGI 17.7 release, the presence of `-Mcuda` disables the pool allocator; we are working on lifting that restriction in an upcoming release.

Data movement of managed data is controlled by the NVIDIA CUDA GPU driver; whenever data is accessed on the CPU or the GPU, it could trigger a data transfer if the last time it was accessed was not on the same device. In some cases, page thrashing may occur and impact performance. An introduction to CUDA Unified Memory is available on Parallel Forall.

This feature has the following limitations:

- Use of managed memory applies only to dynamically-allocated data. Static data (C static and extern variables, Fortran module, common block and save variables) and function local data is still handled by the OpenACC runtime. Dynamically allocated

Fortran local variables and Fortran allocatable arrays are implicitly managed but Fortran array pointers are not.

- Given an allocatable aggregate with a member that points to local, global or static data, compiling with `-ta=tesla:managed` and attempting to access memory through that pointer from the compute kernel will cause a failure at runtime.
- C++ virtual functions are not supported.
- The `-ta=tesla:managed` compiler option must be used to compile the files in which variables are allocated, even if there is no OpenACC code in the file.

This feature has the following additional limitations when used with NVIDIA Kepler GPUs:

- Data motion on Kepler GPUs is achieved through fast pinned asynchronous data transfers; from the program's perspective, however, the transfers are synchronous.
- The PGI runtime enforces synchronous execution of kernels when `-ta=tesla:managed` is used on a system with a Kepler GPU. This situation may result in slower performance because of the extra synchronizations and decreased overlap between CPU and GPU.
- The total amount of managed memory is limited to the amount of available device memory on Kepler GPUs.

This feature is not supported on NVIDIA Fermi GPUs.

CUDA Unified Memory Pool Allocator

Dynamic memory allocations are made using cudaMallocManaged(), a routine which has higher overhead than allocating non-unified memory using cudaMalloc(). The more calls to cudaMallocManaged(), the more significant the impact on performance.

To mitigate the overhead of cudaMallocManaged() calls, both `-ta=tesla:managed` and `-ta=tesla:pinned` use a CUDA Unified Memory pool allocator to minimize the number of calls to cudaMallocManaged(). The pool allocator is enabled by default. It can be disabled, or its behavior modified, using these environment variables:

Table 3 Pool Allocator Environment Variables

Environment Variable	Use
PGI_ACC_POOL_ALLOC	Disable the pool allocator. The pool allocator is enabled by default; to disable it, set PGI_ACC_POOL_ALLOC to 0.
PGI_ACC_POOL_SIZE	Set the size of the pool. The default size is 1GB but other sizes (i.e., 2GB, 100MB, 500KB, etc.) can be used. The actual pool size is set such that the size is the nearest, smaller number in the Fibonacci series compared to the provided or default size. If necessary, the pool allocator will add more pools but only up to the PGI_ACC_POOL_THRESHOLD value.

Environment Variable	Use
PGI_ACC_POOL_ALLOC_MAXSIZE	Set the maximum size for allocations. The default maximum size for allocations is 500MB but another size (i.e., 100KB, 10MB, 250MB, etc.) can be used as long as it is greater than or equal to 16B.
PGI_ACC_POOL_ALLOC_MINSIZE	Set the minimum size for allocation blocks. The default size is 16B but other sizes can be used. The size must be greater than or equal to 16B.
PGI_ACC_POOL_THRESHOLD	Set the percentage of total device memory that the pool allocator can occupy. The default is set to 50% but other percentages can be used.

Multicore Support

PGI Accelerator OpenACC compilers support the option `-ta=multicore`, to set the target accelerator for OpenACC programs to the host multicore CPU. This will compile OpenACC compute regions for parallel execution across the cores of the host 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.

Default Compute Capability

The default compute capability list for OpenACC and CUDA Fortran compilation for NVIDIA Tesla targets is 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. Similarly, if CUDA 9.0 is specified using the `cuda9.0` sub-option, cc70 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=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.

OpenACC 2.5

The PGI compilers implement OpenACC 2.5 as defined in *The OpenACC Application Programming Interface*, Version 2.5, August 2013, http://www.openacc.org, with the exception that these features are not yet supported:

- nested parallelism
- declare link
- enforcement of the new **`cache`** clause restriction that all references to listed variables must lie within the region being cached

Support for Profiler/Trace Tool Interface

PGI compilers support 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.

2.3. OpenMP

OpenMP 3.1

The PGI Fortran, C, and C++ compilers support OpenMP 3.1 on all platforms.

OpenMP 4.5

The PGI Fortran, C, and C++ compilers compile most OpenMP 4.5 programs for parallel execution across all the cores of a multicore CPU or server. **`target`** regions are implemented with default support for the multicore host as the target, and **`parallel`** and **`distribute`** loops are parallelized across all OpenMP threads. This feature is supported on Linux/x86 platforms with the Beta LLVM code generator only.

Current limitations include:

- The **simd** construct is ignored, with the exception that the **private**, **lastprivate**, **reduction**, and **collapse** clauses are processed and supported.
- The **declare simd** construct is ignored.
- The **ordered** construct's **simd** clause is ignored.
- The **taskloop** construct's **firstprivate** and **lastprivate** clauses are not supported.
- The **task** construct's **depend** and **priority** clauses are not supported.
- The loop construct's **linear**, **schedule**, and **ordered(n)** clauses are not supported.
- The **declare reduction** directive is not supported.

2.4. C++ Compiler

2.4.1. 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.

2.4.2. C++ Compatibility

PGI 2017 C++ object code is incompatible with PGI 2016 and prior releases.

All C++ source files and libraries that were built with prior releases must be recompiled to link with PGI 2017 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

2.5. Runtime Library Routines

PGI 2017 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.

2.6. 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.

2.7. Environment Modules

On Linux, if you use the Environment Modules package (e.g., the `module load` command), then PGI 2017 includes a script to set up the appropriate module files.

2.8. Beta LLVM Code Generator

Made available with this release is a Beta version of the PGI Linux/x86-64 compilers with an LLVM code generator and OpenMP runtime. The Beta version includes all languages and programming model support found in the default PGI compilers, as well as the PGI debugger and profiler. The PGI optimizer, fast math numerics library, and Fortran runtime are included. The PGI Beta compilers and tools are available as a separate download and install package; they can be installed side-by-side with the default PGI x86-64 compilers.

LLVM is a collection of compiler and toolchain technologies. The LLVM project provides a code generator and OpenMP runtime for many popular CPUs. More details about LLVM can be found on the web site www.llvm.org.

The PGI Linux/x86-64 compilers with the Beta LLVM code generator is offered as a Beta release. For assistance with difficulties related to this beta release, please contact the PGI technical reporting service, https://www.pgicompilers.com/support/support_request.php. We welcome your feedback.

To use the PGI Beta compiler, put its bin directory on your PATH. If the PGI Beta compilers are installed side-by-side with the default PGI compilers and tools, you can put the default PGI compilers on your PATH and use the command-line option `-Mllvm` to invoke the Beta version.

Take care not to mix object files compiled with the default PGI compilers and the PGI Beta compilers. While the generated code is compatible, the OpenMP runtime libraries are not.

Features available in the PGI Linux/x86-64 compilers using the Beta LLVM code generator:

- Optimized OpenMP atomics.
- OpenMP 4.5 features, not including GPU offload.
- Improved performance for some applications compared to the default PGI x86-64 code generator.

Limitations of the PGI Beta:

- Only available for Linux; not available for macOS or Windows.
- Fortran debugging is available only for breakpoints, call stacks, and basic types.
- PGI Unified Binary is not available.
- Interprocedural optimization using the `-Mipa` option is not available.
- CCFF information is not available.
- Some source-code directives, e.g. **`DEC$`**, may not have an effect.
- Some, but not all, C/C++ MMX/SSE/AVX intrinsics are available.
- CUDA Fortran emulation mode, `-Mcuda=emu`, is not available.

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 2017 Release includes these directories:

```
$PGI/linux86-64/17.10/REDIST
$PGI/win64/17.10/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 2017 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 17.10 `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 2017 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, please see the PGI frequently asked questions (FAQ) webpage at https://www.pgicompilers.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 2017 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.

4.2. Issues Related to Debugging

The following are known issues in the PGI debugger:

- Debugging of PGI Unified Binaries, that is, programs built with more than one -tp option, is not fully supported. The names of some subprograms are modified in compilation and the debugger does not translate these names back to the names used in the application source code.
- The debugging of C++ programs is currently limited to macOS versions 10.9 (Mavericks) and 10.10 (Yosemite); there are no limitations on the debugging of C or Fortran programs.
- When debugging on the Windows platform, the Windows operating system times out **`stepi`**/**`nexti`** operations when single stepping over blocked system calls.

4.3. Profiler-related Issues

Some specific issues related to the PGI Profiler:

- The Profiler relies on being able to directly call 'dlsym'. If this system call is intercepted by the program being profiled or by some other library the profiler may hang at startup. We have encountered this specific problem with some implementations of MPI. We recommend you disable any features that may be intercepting the 'dlsym' system call or disable CPU profiling with the --cpu-profiling off option.
  - To disable 'dlsym' interception when using IBM's spectrum MPI omit the following option: -gpu and add the options -x PAMI_DISABLE_CUDA_HOOK=1 and -disbale_gpu_hooks.

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

- Fortran assumed-shape arguments are not yet supported.

Clause Support Limitations

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

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: mailto: sales@pgroup.com
WWW: https://www.pgroup.com or https://www.pgicompilers.com

The PGI User Forum, https://www.pgicompilers.com/userforum/index.php is monitored by members of the PGI engineering and support teams as well as other PGI customers. The forums contain answers to many commonly asked questions. Log in to the PGI website, https://www.pgicompilers.com/account/login.php" to access the forums.

Many questions and problems can be resolved by following instructions and the information available in the PGI frequently asked questions (FAQ), https://www.pgicompilers.com/support/faq.htm.

Submit support requests using the PGI Technical Support Request form, https://www.pgicompilers.com/support/support_request.php .