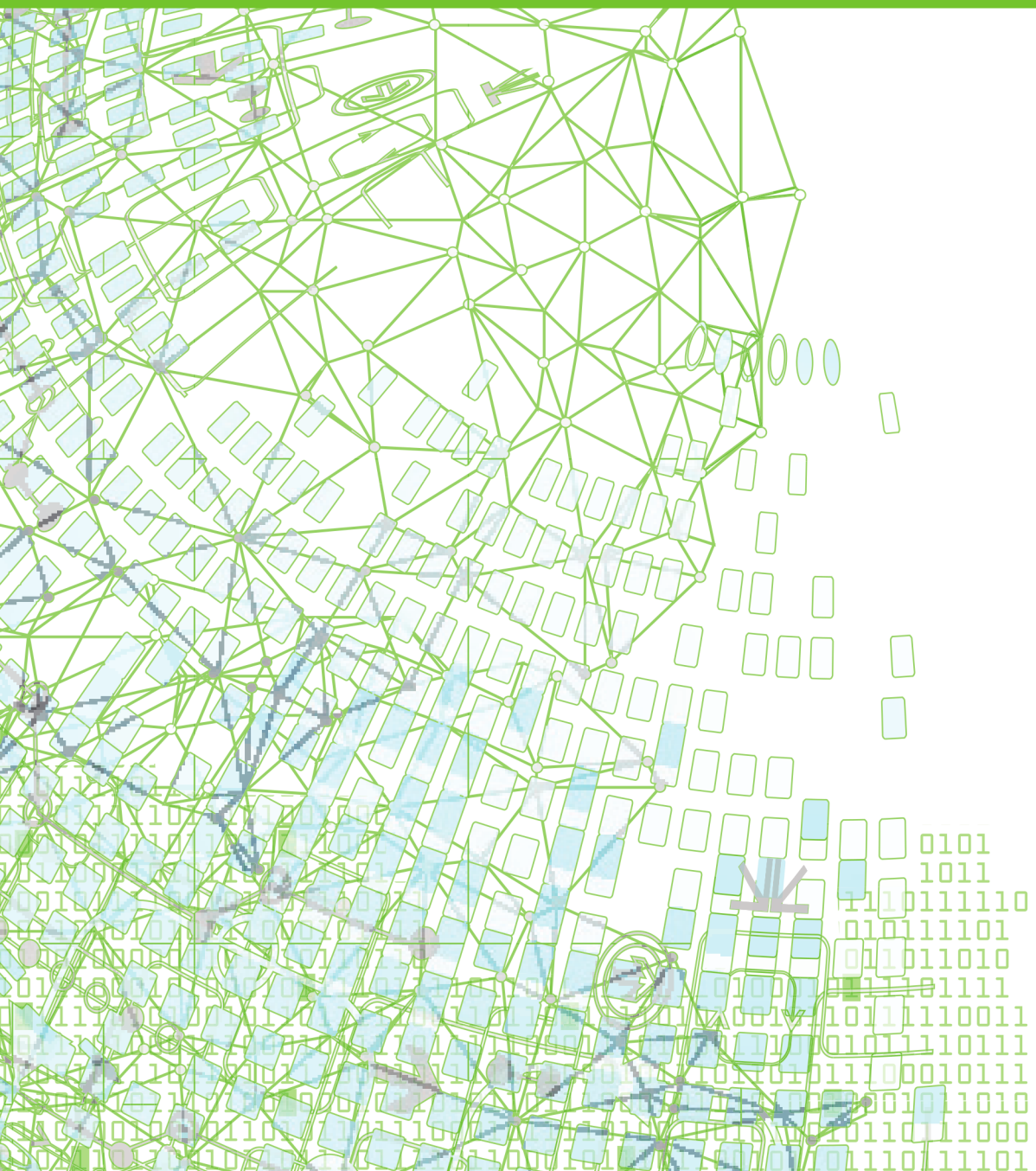
PGI® COMPILERS & TOOLS

INSTALLATION AND RELEASE NOTES FOR OPENPOWER CPUS AND TESLA GPUS

Version 2019



NVIDIA

TABLE OF CONTENTS

Chapter 1. WHAT'S NEW IN PGI 2019

Welcome to Release 2019 of the PGI compilers and tools!

If you read only one thing about this PGI release, make it this chapter. It covers all the new, changed, deprecated, or removed features in PGI products released this year. It is written with you, the user, in mind.

Every PGI release contains user-requested fixes and updates. We keep a complete list of these fixed Technical Problem Reports online for your reference.

1.1. What's New in 19.7

OpenACC

The compiler will now issue an error message for OpenACC **`data`**, **`enter data`**, and **`exit data`** constructs with no data clauses. Previously, these had no effect. These are almost certainly programming errors, and the error message will help avoid such errors.

Tightened the conditions under which OpenACC **`kernels`** loops containing pointer references can be parallelized. This change may prevent the compiler from parallelizing a **`kernels`** loop it previously parallelized unsafely, which could result in a loss of performance. If you are certain the loop can be parallelized safely, changing **`kernels`** to **`parallel`** or adding the **`independent`** loop clause will instruct the compiler to override the safety check and parallelize the loop.

Changed how the compilers treat aggregate members (struct and derived type members) used in loop bounds or loop limits. Previously, the compiler would generally read the aggregate member from host memory. In 19.7, the compiler follows the OpenACC spec more carefully by reading the device memory value of the aggregate member, except for values used in the limits of the outermost loop or loops collapsed with the outermost loop. This means that the value in device memory of any aggregate members used in loop limits must be up-to-date with the host memory values, or the generated code will behave differently.

Improved support for OpenACC **host_data use_device** constructs. The compilers now accept references to aggregate data types in the variable-list as defined in the OpenACC specification.

Added support for capture of the ***this** pointer by value to the OpenACC C++ compiler.

The CUDA 10.1 version bundled with PGI has been upgraded to CUDA 10.1 update 1. More information can be found in the CUDA 10.1 release notes.

CUDA Fortran

Added support to CUDA Fortran for the half precision floating point data type, real(2), as a first class type. This type, represented as IEEE binary16, gives the programmer the ability to declare and use half precision data for programming the Tensor Cores in select NVIDIA GPUs using CUDA Fortran. Real(2) support is available in Fortran compilers on Linux for x86 and OpenPOWER.

Improved performance of CUDA Fortran data transfers between host and device memory using assignment statements, for array sections that can be mapped onto cudaMemcpy2D.

Fortran

Added support for the Fortran 2008 **g0** editor descriptor.

Improved generation of DWARF debug information for allocatable arrays within modules.

Added support for allocations from multiple sources via the **SOURCE** argument (e.g. **ALLOCATE(a, b, SOURCE=c)**).

Extended **MAXLOC** and **MINLOC** intrinsics to accept the optional **BACK** argument.

Implemented **NORM2** intrinsic to calculate magnitude of a Euclidean vector.

Libraries

We have added two new optimized intrinsics: **atan** and **atan2** to our intrinsic math library for x86 processors. This includes both single and double precision as well as scalar and vector versions.

1.2. What's New in 19.5

All Compilers

The PGI 19.5 release contains all the new features found in PGI 19.4 and a few key updates for important user-reported problems.

1.3. What's New in 19.4

All Compilers

Version 19.4 of the PGI Compilers & Tools is a Community Edition release. New, refined and improved support for new features introduced in 2019 include: Tensor Core support in CUDA Fortran, LLVM-based code generation, full C++17 language support, support for printf() in OpenACC regions, support for CUDA 10.0 and 10.1, improved SIMD vectorization, GNU interoperability through GCC 8.1, pre-compiled Open MPI 3.1.3 libraries, support for the latest operating systems, and more. Please refer to the TPR list located at https://www.pgroup.com/support/release-tprs-2019.htm for a detailed summary of bug fixes.

1.4. What's New in 19.3

OpenACC and CUDA Fortran

Added support for the CUDA Toolkit version 10.1. Read more about support for CUDA Toolkits in CUDA Toolkit Versions.

1.5. What's New in 19.1

All Compilers

The LLVM backend has been upgraded from version 6.0 to 7.0 and is now the default code generator for Linux x86-64 platforms.

Upgraded FlexNet Publisher licensing to version 11.16.2 from 11.14.1.3 for x86-64. The newer daemons are backwards-compatible with previous versions of the PGI compilers. The reverse is not true. This release, and future releases, of the PGI compilers are not compatible with the older daemons and require version 11.16.2 at a minimum.

PCAST (PGI Compiler-Assisted Software Testing) now supports C and Fortran directives that are equivalent to the existing library API calls.

Added support for the following versions of Linux/x86-64:

- CentOS 7.6
- Fedora 29
- RHEL 7.6
- Ubuntu 18.10

C/C++

Updated support for SIMD intrinsic functions in the **pgc++** compiler for Linux/x86-64 platforms. Support is comprehensive for SSE, SSE2, SSE3, SSSE3, SSE4.1 and SSE4.2 intrinsics. Many of the AVX, AVX2 and AVX-512 intrinsics are now implemented as well. We expect to add comprehensive support for the AVX intrinsics in a future release.

The **pgc++** compiler now implements full support for the C+17 language standard.

Fortran

Added support for the **ERROR STOP** statement.

Added run-time checks to enforce **CONTIGUOUS** behavior. These checks only occur in code running on the CPU; in particular, they do not occur in code running on a GPU.

Improved Fortran reshape performance, and handling of derived type c-binding for C++ objects.

OpenACC and CUDA Fortran

Added initial support for use of NVIDIA GPU Tensor Cores in CUDA Fortran programs.

The C and C++ compilers now include support for formatted output using **printf()** statements in OpenACC compute regions. Most common format specifiers for flag, width, precision, size and type are supported. Using **printf()** in OpenACC regions is useful for basic debugging and programmer-driven tracing during development and tuning of OpenACC applications on both multicore CPUs and GPUs.

Added support for Turing architecture with compute capability version 7.5.

CUDA 9.2 is now the default target version if there is no GPU or CUDA driver found on the system.

Removed support for legacy PGI Accelerator Directives. Attempting to compile code with such directives will now give an error instead of a warning.

Renamed the **[no]llvm** sub-option to **-Mcuda** and **-ta=tesla** to **[no]nvvm**.

Various compiler improvements, including bug fixes related to subkernels when compiling with **-Mallocatable=03** (default allocation behavior since 18.7), and improved scalar replacement optimizations.

Consolidated the accelerator runtime libraries by removing non-thread-safe versions previously used for serial execution.

Performance

Improved performance of several intrinsic functions for AVX2 and newer processors: `sin, cos, log, log10, cexp, acos, ains, atan, tanh`.

Improved performance of memory idiom functions: `mset, mcopy, mzero`.

Enhanced vectorizer to recognize more opportunities for SIMD code generation across all CPU targets.

Known Issues and Limitations

Passing **-M[no]llvm** to MPI wrappers (mpicc, mpifort, etc.) is not supported. Doing so will cause unresolved symbol errors and segmentation faults when compiling.

Using **-V<version>** with the PGI 2019 compilers to target PGI 2018 or earlier releases is not supported. This is a known limitation that is the result of the switch to using LLVM compilers as the default code generator for the PGI 2019 compilers.

Programs built with OpenMPI 3.1.3 that use only one MPI rank will hang if invoked directly. You must now run the program with **mpirun -np 1 ./executable** or set the environment variable **OMPI_MCA_ess_singleton_isolated=1**. This is a known and intended behavior change in the 3.1.3 version of OpenMPI.

OpenMP profiling is not supported for Fortran applications that use both OpenACC and OpenMP, and use critical sections, OpenMP lock API calls, or performs I/O. OpenACC profiling of these applications is still possible by disabling OpenMP profiling with the '**--openmp-profiling off**' option to pgprof.

Deprecations and Eliminations

CUDA 9.1 is no longer included as part of the PGI compilers installation package.

The PGI compilers installation packages no longer include pre-compiled versions of MPICH or MVAPICH.

The **pgf77** driver is deprecated. Use **pgfortran** to compile F77 Fortran.

Chapter 2. RELEASE OVERVIEW

This chapter provides an overview of Release 2019 of the PGI Accelerator™ compilers hosted on and targeting OpenPOWER+Tesla processor-based servers and clusters running versions of the Linux operating system.

2.1. About This Release

These PGI compilers generate host CPU code for 64-bit little-endian OpenPOWER CPUs, and GPU device code for NVIDIA Kepler and Pascal GPUs.

These compilers include all GPU OpenACC features available in the PGI C/C++/Fortran compilers for x86-64.

Documentation includes the `pgcc`, `pgc++` and `pgfortran` man pages and the -help option. In addition, you can find both HTML and PDF versions of these installation and release notes, the *PGI Compiler User's Guide* and *PGI Compiler Reference Manual* for OpenPOWER on the PGI website, pgicompilers.com/docs.

2.2. Release Components

Release 2019 includes the following components:

- PGFORTRAN™ native OpenMP and OpenACC Fortran 2003 compiler.
- PGCC® native OpenMP and OpenACC ISO C11 and K&R C compiler.
- PGC++® native OpenMP and OpenACC ISO C++17 compiler.
- PGI Profiler® OpenACC, CUDA, OpenMP, and multi-thread profiler.
- Open MPI version 3.1.3 including support for NVIDIA GPUDirect. 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.
- ScaLAPACK 2.0.2 linear algebra math library for distributed-memory systems for use with Open MPI and the PGI compilers.
- BLAS and LAPACK library based on the customized OpenBLAS project source.

- Documentation in man page format and online, pgicompilers.com/docs, in both HTML and PDF formats.

2.3. Command-line Environment

The PGI compilers for OpenPOWER are command-line compatible with the corresponding PGI products on Linux x86-64, meaning target-independent compiler options should behave consistently across the two platforms. The intent and expectation is that makefiles and build scripts used to drive PGI compilers on Linux x86-64 should work with little or no modification on OpenPOWER. The -help compiler option lists all compiler options by default, or can be used to check the functionality of a specific option. For example:

```
% pgcc -help -fast
Reading rcfile /opt/pgi/linuxpower/19.7/bin/.pgccrc
-fast               Common optimizations; includes -O2 -Munroll=c:1 -Mlre -
Mautoinline
                    == -Mvect=simd -Mflushz
-M[no]vect[=[no]simd|[no]assoc|[no]fuse]
                    Control automatic vector pipelining
    [no]simd        Generate [don't generate] SIMD instructions
    [no]assoc       Allow [disallow] reassociation
    [no]fuse        Enable [disable] loop fusion
-M[no]flushz        Set SSE to flush-to-zero mode
%
```

2.4. Supported Platforms

These OpenPOWER hardware/software platforms have been used in testing:

- CPUs: POWER8, POWER8E, POWER8NVL, POWER9
- Linux distributions:
  - Fedora 26
  - RHEL 7.3, 7.4, 7.5, 7.6-ALT
  - Ubuntu 14.04, 16.04, 18.04
- GCC versions: 4.8.4, 4.8.5, 5.3.1, 5.4.0, 7.2.1, 7.3.0
- CUDA Toolkit versions:
  - 9.2 driver version 396.26, 396.44
  - 10.0 driver version 410.45, 410.48, 410.73, 410.79
  - 10.1 driver version 418.39

2.5. CUDA Toolkit Versions

The PGI compilers use NVIDIA's CUDA Toolkit when building programs for execution on an NVIDIA GPU. Every PGI installation package puts the required CUDA Toolkit components into a PGI installation directory called `2019/cuda`.

An NVIDIA CUDA driver must be installed on a system with a GPU before you can run a program compiled for the GPU on that system. PGI products do not contain CUDA Drivers. You must download and install the appropriate CUDA Driver from NVIDIA. The CUDA Driver version must be at least as new as the version of the CUDA Toolkit with which you compiled your code.

The PGI tool `pgaccelinfo` prints the driver version as its first line of output. You can use it to find out which version of the CUDA Driver is installed on your system.

PGI 19.7 includes the following versions of the CUDA Toolkit:

- CUDA 9.2
- CUDA 10.0
- CUDA 10.1

You can let the compiler pick which version of the CUDA Toolkit to use or you can instruct it to use a particular version. The rest of this section describes all of your options.

If you do not specify a version of the CUDA Toolkit, the compiler uses the version of the CUDA Driver installed on the system on which you are compiling to determine which CUDA Toolkit to use. This auto-detect feature was introduced in the PGI 18.7 release; auto-detect is especially convenient when you are compiling and running your application on the same system. In the absence of any other information, the compiler will look for a CUDA Toolkit version in the PGI `2019/cuda` directory that matches the version of the CUDA Driver installed on the system. If a match is not found, the compiler searches for the newest CUDA Toolkit version that is not newer than the CUDA Driver version. If there is no CUDA Driver installed, the PGI 19.7 compilers fall back to the default of CUDA 9.2.

If the only PGI compiler you have installed is PGI 19.7, then:

- If your CUDA Driver is 10.1, the compilers use CUDA Toolkit 10.1.
- If your CUDA Driver is 10.0, the compilers use CUDA Toolkit 10.0.
- If your CUDA Driver is 9.2, the compilers use CUDA Toolkit 9.2.
- If your CUDA Driver is 9.1, the compilers will issue an error that CUDA Toolkit 9.1 was not found; CUDA Toolkit 9.1 is not bundled with PGI 19.7
- If you do not have a CUDA driver installed on the compilation system, the compilers use the default CUDA Toolkit version 9.2.
- If your CUDA Driver is newer than CUDA 10.1, the compilers will still use the CUDA Toolkit 10.1. The compiler selects the newest CUDA Toolkit it finds that is not newer than the CUDA Driver.

You can change the compiler's default selection for CUDA Toolkit version using one of the following methods:

- Use a compiler option. Add the `cudaX.Y` sub-option to `-Mcuda` or `-ta=tesla` where `X.Y` denotes the CUDA version. For example, to compile a C file with the CUDA 9.2 Toolkit you would use:

  ```
  pgcc -ta=tesla:cuda9.2
  ```

  Using a compiler option changes the CUDA Toolkit version for one invocation of the compiler.

- Use an rcfile variable. Add a line defining `DEFCUDAVERSION` to the `siterc` file in the installation `bin/` directory or to a file named `.mypgirc` in your home directory. For example, to specify the CUDA 9.2 Toolkit as the default, add the following line to one of these files:

  ```
  set DEFCUDAVERSION=9.2;
  ```

  Using an rcfile variable changes the CUDA Toolkit version for all invocations of the compilers reading the rcfile.

When you specify a CUDA Toolkit version, you can additionally instruct the compiler to use a CUDA Toolkit installation different from the defaults bundled with the current PGI compilers. While most users do not need to use any other CUDA Toolkit installation than those provided with PGI, situations do arise where this capability is needed. Developers working with pre-release CUDA software may occasionally need to test with a CUDA Toolkit version not included in a PGI release. Conversely, some developers might find a need to compile with a CUDA Toolkit older than the oldest CUDA Toolkit installed with a PGI release. For these users, PGI compilers can interoperate with components from a CUDA Toolkit installed outside of the PGI installation directories.

PGI tests extensively using the co-installed versions of the CUDA Toolkits and fully supports their use. Use of CUDA Toolkit components not included with a PGI install is done with your understanding that functionality differences may exist.

To use a CUDA toolkit that is not installed with a PGI release, such as CUDA 9.1 with PGI 19.7, there are three options:

- Use the rcfile variable `DEFAULT_CUDA_HOME` to override the base default

  ```
  set DEFAULT_CUDA_HOME = /opt/cuda-9.1;
  ```

- Set the environment variable `CUDA_HOME`

  ```
  export CUDA_HOME=/opt/cuda-9.1
  ```

- Use the compiler compilation line assignment CUDA_HOME=

  ```
  pgfortran CUDA_HOME=/opt/cuda-9.1
  ```

The PGI compilers use the following order of precedence when determining which version of the CUDA Toolkit to use.

1. If you do not tell the compiler which CUDA Toolkit version to use, the compiler picks the CUDA Toolkit from the PGI installation directory `2019/cuda` that matches the version of the CUDA Driver installed on your system. If the PGI installation directory does not contain a direct match, the newest version in that directory which is not newer than the CUDA driver version is used. If there is no CUDA driver installed on your system, the compiler falls back on an internal default; in PGI 19.7, this default is CUDA 9.2.
2. The rcfile variable `DEFAULT_CUDA_HOME` will override the base default.
3. The environment variable `CUDA_HOME` will override all of the above defaults.
4. The environment variable `PGI_CUDA_HOME` overrides all of the above; it is available for advanced users in case they need to override an already-defined `CUDA_HOME`.
5. A user-specified `cudaX.Y` sub-option to `-Mcuda` and `-ta=tesla` will override all of the above defaults and the CUDA Toolkit located in the PGI installation directory `2019/cuda` will be used.

6. The compiler compilation line assignment `CUDA_HOME=` will override all of the above defaults (including the `cudaX.Y` sub-option).

2.6. Compute Capability

The compilers can generate code for NVIDIA GPU compute capabilities 3.0 through 7.5. The compilers construct a default list of compute capabilities that matches the compute capabilities supported by the GPUs found on the system used in compilation. If there are no GPUs detected, the compilers select cc35, cc60, and cc70.

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=60 70;
```

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

The generation of device code can be time consuming, so you may notice an increase in compile time as the number of compute capabilities increases.

2.7. Precompiled Open-Source Packages

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

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

- OpenBLAS 0.3.3 – customized BLAS and LAPACK libraries based on the OpenBLAS project source.
- Open MPI 3.1.3 – 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 3.1.3.

The following list of open-source software packages have been precompiled for execution on OpenPOWER targets using the PGI compilers and are available to download from the PGI website at pgicompilers.com/downloads.

- ESMF 7.1.0r for Open MPI 3.1.3 – The Earth System Modeling Framework for building climate, numerical weather prediction, data assimilation, and other Earth science software applications.
- NetCDF 4.6.2 – 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.
- Parallel NetCDF 1.11.0 – for Open MPI 3.1.3.
- HDF5 1.10.4 – data model, library, and file format for storing and managing data.
- SZIP 2.1.1 – extended-Rice lossless compression algorithm.
- ZLIB 1.2.11 – file compression library.

In addition, these software packages have also been ported to PGI on OpenPOWER 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 pgicompilers.com/tips.

- FFTW 2.1.5 – version 2 of the Fast Fourier Transform library, includes MPI bindings built with Open MPI 3.1.3.
- FFTW 3.3.8 – version 3 of the Fast Fourier Transform library, includes MPI bindings built with Open MPI 3.1.3.

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

Chapter 3. INSTALLATION AND CONFIGURATION

Follow these steps to install PGI 19.7 compilers on an OpenPOWER system. The default installation directory is `/opt/pgi`, but it can be any directory:

```
% tar zxpf pgilinux-2019-197-ppc64le.tar.gz
% ./install
<answer installation questions, assent to licenses>
...
```

Typically for this release, you will want to choose the following during the installation:

1. Choose a "Single-system install", not a "Network install".
2. Install the PGI software in the default `/opt/pgi` directory.
3. Install the CUDA toolkit.

   This installs CUDA components in the PGI directory tree, and will not affect a standard CUDA installation on the same system in any way.
4. Install the OpenACC Unified Memory Evaluation package.
5. Create links in the 2019 directory.

   This is the directory where CUDA is installed, along with example programs; links are created to the subdirectories of `/opt/pgi/linuxpower/19.7`.
6. Install Open MPI.

3.1. License Management

Installation may place a temporary license key in a file named `license.pgi` in the PGI installation directory if no such file already exists.

If you purchased a perpetual license and have obtained your new license key, either replace the contents of license.pgi with your new license key, or set the environment variable `LM_LICENSE_FILE` to the full path of the desired license file.

If you have not yet obtained your new license key, please consult your PGI order confirmation email for instructions for obtaining and installing your permanent license key. Contact PGI Sales at sales@pgroup.com if you need assistance.

Usage Logging: This release provides per-user records of most recent use in the `.pgiusage` subdirectory inside the main installation directory. Set the environment variable `PGI_LOG_DIRECTORY` to specify a different directory for usage logging.

3.2. Environment Initialization

Assuming the software is installed in `/opt/pgi`, use these commands in `csh` to initialize your environment for use of the PGI compilers:

```
% setenv PGI /opt/pgi
% setenv MANPATH "$MANPATH":$PGI/linuxpower/2019/man
% set path=($PGI/linuxpower/2019/bin $path)
% which pgc++
/opt/pgi/linuxpower/2019/bin/pgc++
%
```

In `bash`, `sh` or `ksh`, use these commands:

```
% export PGI=/opt/pgi
% export MANPATH=$MANPATH:$PGI/linuxpower/2019/man
% export PATH=$PGI/linuxpower/2019/bin:$PATH
% which pgc++
/opt/pgi/linuxpower/2019/bin/pgc++
%
```

The PGI directory structure is designed to accommodate co-installation of multiple PGI versions. When 19.7 is installed, it will be installed by default in the directory `/opt/pgi/linuxpower/19.7` and links can optionally be created to its sub-directories to make `19.7` default without affecting a previous (e.g., `16.10`) install. Non-default versions of PGI compilers that are installed can be used by specifying the -V<ver> option on the compiler command line.

3.3. Network Installations

PGI compilers for OpenPOWER may be installed locally on each machine on a network or they may be installed once on a shared file system available to every machine. With the shared file system method, after the initial installation you can run a simple script on each machine to add that system to the family of machines using the common compiler installation. Using this approach, you can create a common installation that works on multiple linuxpower systems even though each system may have different versions of *gcc/libc*.

Follow these steps to create a shared file system installation on OpenPOWER systems:

1. Create a commonly-mounted directory accessible to every system using the same directory path (for example, `/opt/pgi`).
2. Define a locally-mounted directory with a pathname that is identical on all systems. That is, each system has a local directory path with the same pathname (for example `/local/pgi/19.7/share_objects`). Runtime libraries which are *libc*-version dependent will be stored here. This will ensure that executable files built on one system will work on other systems on the same network.
3. Run the install script for the first installation:

```
% tar zxpf pgilinux-2019-197-ppc64le.tar.gz
% ./install
<answer installation questions, assent to licenses>
...
```

At the "Please choose install option:" prompt, choose "Network install".

4. Once the initial PGI installation is complete, configure the environment as described in the preceding section.
5. On each subsequent system, follow these steps:
   a. Set up the environment as described in the preceding section.
   b. Run the `add_network_host` script which is now in your $PATH:

      ```
      $ add_network_host
      ```

      and the compilers should now work.

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.

4.1. Release Specific Limitations

The following PGI features are limited or are not implemented in the 19.7 release for OpenPOWER+Tesla:

- -Mipa is not enabled (no PGI inter-procedural analysis/optimization); the command-line option is accepted and silently ignored.
- -Mpfi/-Mpfo are not enabled (no profile-feedback optimization); the command-line options are accepted and silently ignored.
- Passing **`-M[no]llvm`** to MPI wrappers (mpicc, mpifort, etc.) is not supported. Doing so will cause unresolved symbol errors and segmentation faults when compiling.
- Using **`-V<version>`** with the PGI 2019 compilers to target PGI 2018 or earlier releases is not supported. This is a known limitation that is the result of the switch to using LLVM compilers as the default code generator for the PGI 2019 compilers.

4.2. Profiler-related Issues

Some specific issues related to the PGI Profiler:

- Debugging information is not always available on OpenPower which can cause the profiler to crash when attempting to unwind the call-stack. We recommend you use the --cpu-profiling-unwind-stack off option to disable call-stack tracing if you encounter any problem profiling on OpenPower.
- 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 set the environment variable: PAMI_DISABLE_CUDA_HOOK=1, omit the following option: -gpu and add the options: -x PAMI_DISABLE_CUDA_HOOK and -disable_gpu_hooks.

Chapter 5. CONTACT INFORMATION

You can contact NVIDIA's PGI compilers and tools team at:

9030 NE Walker Road, Suite 100
Hillsboro, OR 97006

Or electronically using any of the following means:

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

The PGI User Forum, pgicompilers.com/userforum 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, pgicompilers.com/login 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), pgicompilers.com/faq.

Submit support requests using the PGI Technical Support Request form, pgicompilers.com/support-request.