PGI® COMPILERS & TOOLS

INSTALLATION AND RELEASE NOTES FOR OPENPOWER CPUS AND TESLA GPUS

Version 2018

NVIDIA

TABLE OF CONTENTS

LIST OF TABLES

Chapter 1. WHAT'S NEW IN PGI 2018

Welcome to Release 2018 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 18.5

The PGI 18.5 release contains all features and fixes found in PGI 18.4 and a few key updates for important user-reported problems.

Added full support for CUDA 9.2; use the `cuda9.2` sub-option with the `-ta=tesla` or `-Mcuda` compiler options to compile and link with the integrated CUDA 9.2 toolkit components.

Added support for Xcode 9.3 on macOS.

Improved function offset information in runtime tracebacks in optimized (non-debug) modes.

1.2. What's New in 18.4

The PGI 18.4 release contains all features and fixes found in PGI 18.3 and a few key updates for important user-reported problems.

Added support for CUDA 9.2 if one directs the compiler to a valid installation location of CUDA 9.2 using CUDA_HOME.

Added support for internal procedures, assigned procedure pointers and internal procedures passed as actual arguments to procedure dummy arguments.

Added support for intrinsics SCALE, MAXVAL, MINVAL, MAXLOC and MINLOC in initializers.

1.3. What's New in 18.3

The PGI 18.3 release contains all the new features found in PGI 18.1 and a few key updates for important user-reported problems.

C/C++

Implemented the `__builtin_return_address` and `__builtin_frame_address` functions.

1.4. What's New in 18.1

Key Features

Added support for IBM POWER9 processors.

Added full support for OpenACC 2.6.

Enhanced support for OpenMP 4.5 for multicore CPUs, including SIMD directives as tuning hints.

Added support for the CUDA 9.1 toolkit, including on the latest NVIDIA Volta V100 GPUs.

OpenACC and CUDA Fortran

Changed the default CUDA Toolkit used by the compilers to CUDA Toolkit 8.0.

Changed the default compute capability chosen by the compilers to cc35,cc60.

Added support for CUDA Toolkit 9.1.

Added full support for the OpenACC 2.6 specification including:

- serial construct
- if and if_present clauses on host_data construct
- no_create clause on the compute and data constructs
- attach clause on compute, data, and enter data directives
- detach clause on exit data directives
- Fortran optional arguments
- acc_get_property, acc_attach, and acc_detach routines
- profiler interface

Added support for asterisk ('*') syntax to CUDA Fortran launch configuration. Providing an asterisk as the first execution configuration parameter leaves the compiler free to calculate the number of thread blocks in the launch configuration.

Added two new CUDA Fortran interfaces, cudaOccupancyMaxActiveBlocksPerMultiprocessor and cudaOccupancyMaxActiveBlocksPerMultprocessorWithFlags. These provide hooks into the CUDA Runtime for manually obtaining the maximum number of thread blocks which can be used in grid-synchronous launches, same as provided by the asterisk syntax above.

OpenMP

Changed the default initial value of `OMP_MAX_ACTIVE_LEVELS` from 1 to 16.

Added support for the **`taskloop`** construct's **`firstprivate`** and **`lastprivate`** clauses.

Added support for the OpenMP Performance Tools (OMPT) interface. Available with the LLVM code generator compilers on Linux.

C++

Added support for GNU interoperability through GCC 7.2.

Added partial support for C++17 including constexpr if, fold expressions, structured bindings, and several other C++17 features. See C++17 for a complete list of supported features.

Fortran

Changed how the PGI compiler runtime handles Fortran array descriptor initialization; this change means any program using Fortran 2003 should be recompiled with PGI 18.1.

Improved Fortran debugging support.

Libraries

Reorganized the Fortran cuBLAS and cuSolver modules to allow use of the two together in any Fortran program unit. As a result of this reorganization, any codes which use cuBLAS or cuSolver modules must be recompiled to be compatible with this release.

Added a new PGI math library, libpgm. Moved math routines from libpgc, libpgftnrtl, and libpgf90rtl to libpgm. This change should be transparent unless you have been explicitly adding libpgc, libpgftnrtl, or libpgf90rtl to your link line.

Added new fastmath routines for single precision scalar/vector sin/cos/tan for AVX2 and AVX512F processors.

Added support for C99 scalar complex intrinsic functions.

Added support for vector complex intrinsic functions.

Added environment variables to control runtime behavior of intrinsic functions:

MTH_I_ARCH={em64t,sse4,avx,avxfma4,avx2,avx512knl,avx512}
Override the architecture/platform determined at runtime.

MTH_I_STATS=1
Provide basic runtime statistics (number of calls, number of elements, percentage of total) of elemental functions.

MTH_I_STATS=2
Provide detailed call count by element size (single/double-precision scalar, single/double-precision vector size).

MTH_I_FAST={relaxed,precise}
Override compile time selection of fast intrinsics (the default) and replace with either the relaxed or precise versions.

MTH_I_RELAXED={fast,precise}
Override compile time selection of relaxed intrinsics (the default with `-Mfprelaxed=intrinsic`) and replace with either the fast or precise versions.

MTH_I_PRECISE={fast,relaxed}
Override compile time selection of precise intrinsics (the default with `-Kieee`) and replace with either the fast or relaxed versions.

Profiler

Improved the CPU Details View to include the breakdown of time spent per thread.

Added an option to let one select the PC sampling frequency.

Enhanced the NVLink topology to include the NVLink version.

Enhanced profiling data to include correlation ID when exporting in CSV format.

Operating Systems and Processors

LLVM Code Generator

Upgraded the LLVM code generator to version 5.0.

Deprecations and Eliminations

Stopped including components from CUDA Toolkit version 7.5 in the PGI packages. CUDA 7.5 can still be targeted if one directs the compiler to a valid installation location of CUDA 7.5 using CUDA_HOME.

Deprecated legacy PGI accelerator directives. When the compiler detects a deprecated PGI accelerator directive, it will print a warning. This warning will include the OpenACC directive corresponding to the deprecated directive if one exists. Warnings about deprecated directives can be suppressed using the new `legacy` sub-option to the `-acc` compiler option. The following library routines have been deprecated: acc_set_device, acc_get_device, and acc_async_wait; they have been replaced by

acc_set_device_type, acc_get_device_type, and acc_wait, respectively. The following environment variables have been deprecated: ACC_NOTIFY and ACC_DEVICE; they have been replaced by PGI_ACC_NOTIFY and PGI_ACC_DEVICE_TYPE, respectively. Support for legacy PGI accelerator directives may be removed in a future release.

Dropped support for CUDA Fortran emulation mode. The `-Mcuda=emu` compiler option is no longer supported.

1.5. 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.

Current limitations include:

- The **`simd`** construct can be used to provide tuning hints; the **`simd`** construct's **`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 **`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.

1.6. C++ Compiler

1.6.1. C++17

The PGI 18.1 C++ compiler introduces partial support for the C++17 language standard; access this support by compiling with `--c++17` or `-std=c++17`.

Supported C++17 core language features are available on Linux (requires GCC 7 or later) and OS X.

This PGI compiler release supports the following C++17 language features:

- Structured bindings

- Selection statements with initializers
- Compile-time conditional statements, a.k.a. **`constexpr if`**
- Fold expressions
- Inline variables
- Constexpr lambdas
- Lambda capture of *this by value

The following C++17 language features are not supported in this release:

- Class template deduction
- Auto non-type template parameters
- Guaranteed copy elision

The PGI products do not include a C++ standard library, so support for C++17 additions to the standard library depends on the C++ library provided on your system. On Linux, GCC 7 is the first GCC release with significant C++17 support. On OS X, there is no support for any of the C++17 library changes with one exception: `std::string_view` is available on OS X High Sierra.

The following C++ library changes are supported when building against GCC 7:

- std::string_view
- std::optional
- std::variant
- std::any
- Variable templates for metafunctions

The following C++ library changes are not available on any system that this PGI release supports:

- Parallel algorithms
- Filesystem support
- Polymorphic allocators and memory resources

1.6.2. C++ and OpenACC

There are limitations to the data that can appear in OpenACC 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.
- Member variables are not fully supported in the use_device clause of a host_data construct; this placement may result in an error at runtime.

Conflicts may arise between the version of GCC required to enable C++ language feature support (GCC 5 or newer for C++14, GCC 7 or newer for C++17) and use of the `nollvm` sub-option to `-ta=tesla`. The `nollvm` sub-option uses components of the CUDA

Toolkit which check for compatibility with the version of GCC installed on a system and will not work with a version newer than their specified maximum.

Table 1 GCC Version Compatibility with -ta=tesla:nollvm

CUDA Toolkit Version	Maximum GNU Version Supported
CUDA 7.5	GCC 4.9
CUDA 8.0	GCC 5.x
CUDA 9.0	GCC 6.x
CUDA 9.1	GCC 6.x
CUDA 9.2	GCC 7.x

Chapter 2. RELEASE OVERVIEW

This chapter provides an overview of Release 2018 of the PGI Accelerator™ C11, C++14 and Fortran 2003 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 2018 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 profiler.
- Open MPI version 2.1.2 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/18.5/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
  - Ubuntu 14.04, 16.04
- GCC versions: 4.8.4, 4.8.5, 5.3.1, 5.4.0, 7.2.1
- CUDA Toolkit versions:
  - 7.5 driver version 352.39
  - 9.0 driver versions 384.59, 384.81
  - 9.1 driver versions 387.26, 390.31 (POWER9)
  - 9.2 driver version 396.11

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 packages puts the required CUDA Toolkit components into a PGI installation directory called `2018/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. Use it if you are unsure which version of the CUDA Driver is installed on your system.

PGI 18.5 contains the following versions of the CUDA Toolkits:

- CUDA 8.0 (default)
- CUDA 9.0
- CUDA 9.1
- CUDA 9.2

By default, the PGI compilers in this release use the CUDA 8.0 Toolkit from the PGI installation directory. You can compile with a different version of the CUDA Toolkit using one of the following methods:

- Use a compiler option. 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.

By default, the PGI compilers use the CUDA Toolkit components installed with the PGI compilers and in fact most users do not need to use any other CUDA Toolkit installation than those provided with PGI. 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 7.5 with PGI 18.5, there are three options:

- Use the rcfile variable DEFAULT_CUDA_HOME to override the base default

```
set DEFAULT_CUDA_HOME = /opt/cuda-7.5;
```

- Set the environment variable CUDA_HOME

```
export CUDA_HOME=/opt/cuda-7.5
```

- Use the compiler compilation line assignment CUDA_HOME=

```
pgfortran CUDA_HOME=/opt/cuda-7.5
```

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

- In the absence of any other specification, the CUDA Toolkit located in the PGI installation directory 2018/cuda will be used.
- The rcfile variable DEFAULT_CUDA_HOME will override the base default.
- The environment variable CUDA_HOME will override all of the above defaults.
- 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 2018/cuda will be used.
- The compiler compilation line assignment CUDA_HOME= will override all of the above defaults (including the cudaX.Y sub-option).
- The environment variable PGI_CUDA_HOME overrides all of the above; reserve PGI_CUDA_HOME for advanced use.

2.6. 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 0.2.19 – 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 OpenPOWER targets using the PGI compilers and are available to download from the PGI website at pgicompilers.com/downloads.

- MPICH 3.2 – open-source MPI implementation.
- NetCDF 4.5.0 – 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 1.10.1 – 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 2.1.1 – extended-Rice lossless compression algorithm.
  - ZLIB 1.2.11 – file compression library.
- Parallel NetCDF 1.9.0 for MPICH
- Parallel NetCDF 1.9.0 for Open MPI

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 2.1.2.
- FFTW 3.3.7 – 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 pgicompilers.com/tips.

2.7. Getting Started

By default, the PGI 2018 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 that enable use of SIMD instructions .

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

The following table shows the typical -fast options.

Table 2 Typical -fast Options

Use this option...	To do this...
-O2	Specifies a code optimization level of 2 and -Mvect=SIMD.
-Munroll=c:1	Unrolls loops, executing multiple instances of the original loop during each iteration.
-Mlre	Indicates loop-carried redundancy elimination.
-Mautoinline	Enables automatic function inlining in C & C++.

For best performance on processors that support SIMD instructions, use the PGFORTRAN compiler, even for FORTRAN 77 code, and the -fast option.

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, 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 3. INSTALLATION AND CONFIGURATION

Follow these steps to install PGI 18.5 compilers on an OpenPOWER system. The default installation directory is `/opt/pgi`, but it can be any directory:

```
% tar zxpf pgilinux-2018-185-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 2018 directory.

   This is the directory where CUDA is installed, along with example programs; links are created to the subdirectories of `/opt/pgi/linuxpower/18.5`.
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/2018/man
% set path=($PGI/linuxpower/2018/bin $path)
% which pgc++
/opt/pgi/linuxpower/2018/bin/pgc++
%
```

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

```
% export PGI=/opt/pgi
% export MANPATH=$MANPATH:$PGI/linuxpower/2018/man
% export PATH=$PGI/linuxpower/2018/bin:$PATH
% which pgc++
/opt/pgi/linuxpower/2018/bin/pgc++
%
```

The PGI directory structure is designed to accommodate co-installation of multiple PGI versions. When 18.5 is installed, it will be installed by default in the directory `/opt/pgi/linuxpower/18.5` and links can optionally be created to its sub-directories to make `18.5` 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/18.5/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-2018-185-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 18.5 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.

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 PGI 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.