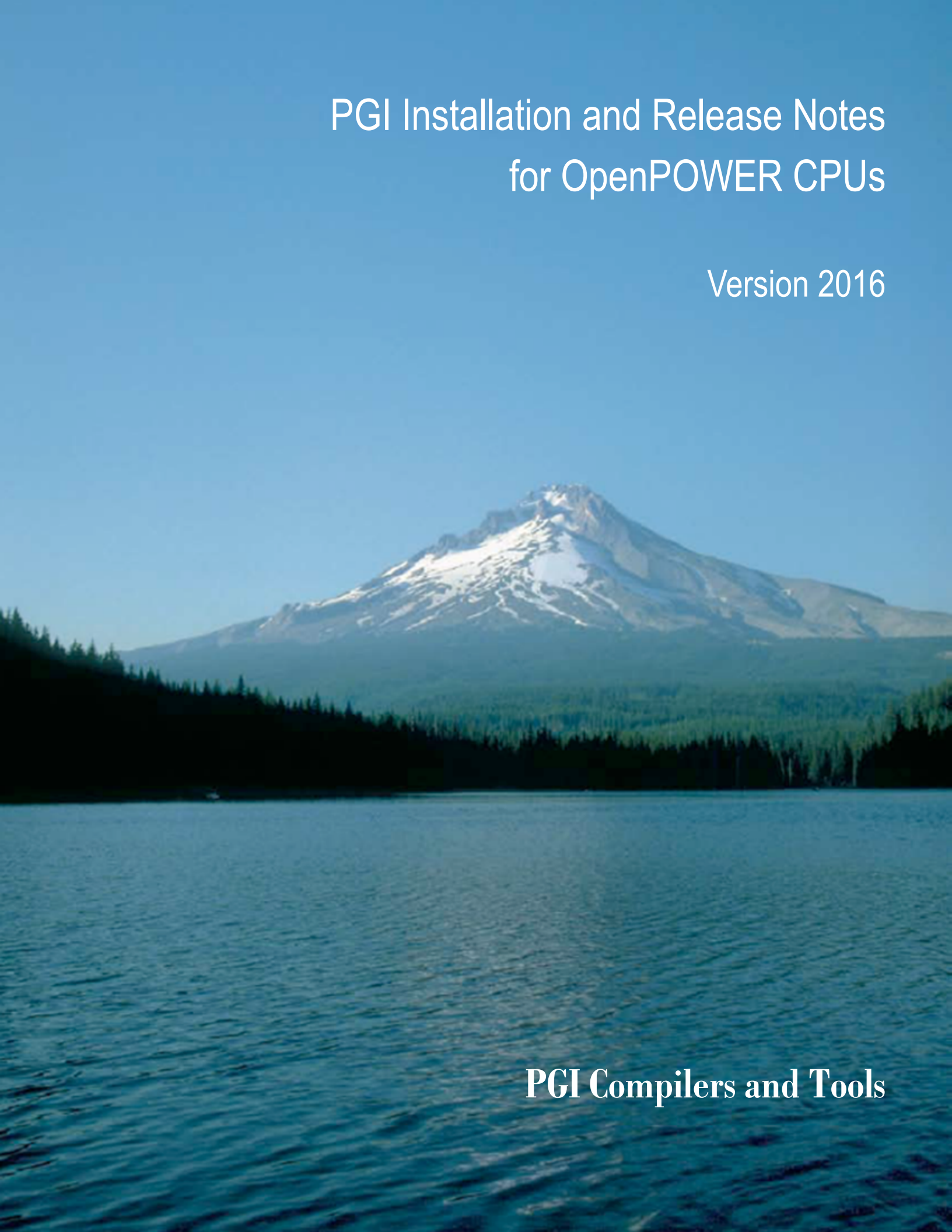
PGI Installation and Release Notes for OpenPOWER CPUs

Version 2016



PGI Compilers and Tools

TABLE OF CONTENTS

LIST OF TABLES

Chapter 1. RELEASE OVERVIEW

Welcome to Release 2016 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.

1.1. 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 Linux/x86.

Documentation includes the `pgcc`, `pgc++` and `pgfortran` man pages and the `-help` option. In addition, you can find online versions of these installation and release notes, the *PGI Compiler User's Guide* and *PGI Compiler Reference Manual* for OpenPOWER. Online documentation is available at http://www.pgroup.com/resources/docs.htm.

1.2. Release Components

Release 2016 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.
- PGPROF® OpenACC, CUDA, OpenMP, and multi-thread profiler.
- Open MPI version 1.10.2 including support for NVIDIA GPUDirect. GPUDirect requires CUDA 7.0 or later. 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 PDFs.

1.3. Supported Platforms

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

- CPUs: POWER8, POWER8E, POWER8NVL
- Linux distributions:
  - Ubuntu 14.04, 14.10, 16.04
  - RHEL 7.3 Beta
- GCC versions: 4.8.4, 4.8.5, 4.9.1, 5.3.1
- CUDA Toolkit versions:
  - 7.5 driver versions 352.39, 352.79
  - 8.0 driver version 361.93.02

1.4. Supported Operating Systems

The PGI 16.10 compilers were built on a Linux/OpenPOWER system running Ubuntu 14.04 OS with a GCC 4.8.2 toolchain. They have been tested on that platform, on Ubuntu 14.10 with GCC 4.9.1 and on RHEL 7.3 Beta with GCC 4.8.5.

1.5. Supported CUDA Software

Select components of the CUDA Toolkit 7.0, 7.5 and 8 are included under the PGI installation tree in `/opt/pgi`.

1.5.1. Targeting a CUDA Toolkit Version

- The CUDA 7.0 Toolkit is set as the default in PGI 16.10. To use the CUDA 7.0 Toolkit, first download the CUDA 7.0 driver from NVIDIA at http://www.nvidia.com/cuda.
- You can compile with the CUDA 7.5 Toolkit either by adding the option `-ta=tesla:cuda7.5` to the command line or by adding `set DEFCUDAVERSION=7.5` to the `siterc` file.
- You can compile with the CUDA 8.0 Toolkit either by adding the option `-ta=tesla:cuda8.0` to the command line or by adding `set DEFCUDAVERSION=8.0` to the `siterc` file.
- `pgaccelinfo` prints the driver version as the first line of output. For a 7.0 driver, it prints:

```
CUDA Driver Version 7000
```

1.6. Precompiled Open-Source Packages

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

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

- OpenBLAS 0.2.14 – customized BLAS and LAPACK libraries based on the OpenBLAS project source.
- Open MPI 1.10.2 – 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 1.10.2.

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:

- MPICH 3.2 – open-source MPI implementation.
- NetCDF 4.3.3.1 – 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.
- NetCDF-C++ 4.2.1 – C++ interfaces to NetCDF libraries.
- NetCDF-Fortran 4.4.2 – Fortran interfaces to NetCDF libraries.
- CURL 7.46.0 – tool and a library (usable from many languages) for client-side URL transfers; included with NetCDF.
- SZIP 2.1 – extended-Rice lossless compression algorithm; included with NetCDF.
- ZLIB 1.2.8 – file compression library; included with NetCDF.

In additon, 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.

- FFTW 2.1.5 – version 2 of the Fast Fourier Transform library, includes MPI bindings built with Open MPI 1.10.2.
- FFTW 3.3.4 – version 3 of the Fast Fourier Transform library, includes MPI bindings built with Open MPI 1.10.2.

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

1.7. Getting Started

By default, the PGI 2016 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 contents 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 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 2. COMPILER FEATURES

Following is a summary of features and limitations in this release.

16.10 Updates and Additions

- Integrated support for the production release of CUDA Toolkit 8. The 16.10 PGI compilers use CUDA 7 by default. CUDA 7.5 or CUDA 8 can be used instead by adding a sub-option (`cuda7.5` or `cuda8.0`, respectively) to the `-ta=tesla` or `-Mcuda` compile- and link-time options.
- Implemented initial support for -ta=multicore for all optimization levels except in the presence of `-g`.
- Improved support for the OpenACC cache directive.
- Improved support for structs, unions, derived types in accelerator regions.
- Extended data types supported by atomic operations in accelerator regions.

16.9 Updates and Additions

- Enabled the `-ta=tesla:managed` compiler option for Pascal GPUs; this option can now be combined with the `cc60` sub-option.
- Added support for native atomic instructions for Pascal GPUs.
- Changed the default memory model from medium to large.
- Added a BLAS and LAPACK library based on the customized OpenBLAS project source and built with PGI compilers. To link against this library, simply add the `-lblas` and `-llapack` options to the link line.

16.7 Updates and Additions

- Initial support for full OpenMP 3.1 in the `pgc++` compiler.
- Support for CUDA Toolkit 8 RC
- Initial support for Pascal GPUs
- LLVM 3.8 is now default
- Support for POWER8 CPUs with NVLINK

- Support for debugging of pgc++-compiled programs
- XLF-compatible complex return types and BIND(C) attribute
- Numerous bug fixes.

2.1. Programming Models

CUDA Fortran and OpenACC in C/C++/Fortran targeting Tesla are supported in this Release. Initial OpenMP 3.1 support for OpenPOWER multicore CPUs is enabled in the C, C++ and Fortran compilers; it includes support for all OpenMP 3.1 directives, clauses and API calls. Auto-parallelization across multiple OpenPOWER cores is supported using the `-Mconcur` compiler option.

2.2. Command-line Environment

The PGI compilers for Linux/OpenPOWER are command-line compatible with the corresponding PGI products on Linux/x86, 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 should work with little or no modification on Linux/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 /proj/pgi/linuxpower/16.10/bin/.pgccrc
-fast               Common optimizations; includes -O2 -Munroll=c:1 -Mlre -
Mautoinline
                    == -Mvect=simd
-M[no]vect[=[no]simd|[no]assoc]
                    Control automatic vector pipelining
    [no]simd        Generate [don't generate] SIMD instructions
    [no]assoc       Allow [disallow] reassociation
%
```

2.3. Fortran Language

The PGFORTRAN compiler supports Fortran 2003 features as defined in the document ISO/IEC 1539-1 : 2004, *Information technology – Programming Languages – Fortran*, Geneva, 2004 (Fortran 2003).

2.4. C Language

The PGCC compiler supports C99 and many of the important C11 language features as defined in the document ISO/IEC 9899:2013, *Information Technology – Programming Languages – C*, Geneva, 2011 (C11). In particular, thread-local storage, type generics, the `_Noreturn` specifier and `__Alignof`/`__Alignas` are all supported. Certain C11 features including anonymous `structs`/`unions`, `_static_assert`, atomics and unicode are not yet supported. PGCC supports compiler options `-c89`, `-c99` and `-c11` to enable/restrict support for/to C89, C99 and

C11 respectively. Support for C99 is default. The `-c11` compiler option must be used to enable support and processing of the C11 features.

2.5. C++ Language

The PGC++ compiler supports C++14 as defined in the document ISO/IEC 14882:2014, *Information Technology – Programming Languages – C++*, Geneva. The `--c++14` compiler option must be used to enable support and processing of C++14 features, and the `--c++11` compiler option must be used to enable support and processing of C++11 features. C++14 features that are not supported include: i) generalized `constexpr` and `constexpr` member functions and implicit `const`, ii) variable templates, and iii) clarifying memory allocation (merged allocation). PGC++ is substantially compatible with GNU `g++` through GCC 5.1. In particular it uses GNU name-mangling conventions, uses GCC header files and libraries directly from a standard GCC installation, and is designed to be link-compatible with `g++`.

2.6. OpenACC

The `pgcc`, `pgc++` and `pgfortran` compilers implement substantial parts of OpenACC 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
- `bind` clause on the `routine` directive
- `finalize` clause on the `exit` data directive
- `if_present` clause on the `update` directive
- `init`, `shutdown` and `set` directives
- Fortran assumed-shape arguments
- API routines to get/set the default `async_queue` value
- not all clauses are supported after the `device_type` clause
- OpenACC 2.5 routine bind changes
- enforcement of the new `cache` clause restriction that all references to listed variables must lie within the region being cached
- the `declare` directive for `static` or `global` data

With respect to OpenACC and Tesla support, these 16.10 compilers for Linux/OpenPOWER include the same features as the PGI 16.10 production compilers for Linux/x86. As pertains to C/C++/Fortran and NVIDIA Tesla GPUs, the OpenACC features documented in Chapter 7 of the *PGI User's Guide* for OpenPOWER (http://www.pgroup.com/doc/pgi16ug-openpower.pdf) are all functional in the PGI 16.10 compilers on Linux/OpenPOWER. Likewise, as pertains to CUDA Fortran and NVIDIA Tesla GPUs, all features documented in the *PGI CUDA Fortran Programming Guide and Reference* are all functional in the Fortran compiler (http://www.pgroup.com/doc/pgicudaforug.pdf), including Beta support for the "managed" attribute and CUDA Unified Memory.

The PGI 16.10 compiler installation for Linux/OpenPOWER includes by default the PGI OpenACC Unified Memory Evaluation Package, which is a Beta feature in PGI 15.4 and

later production PGI Accelerator OpenACC compilers on Linux/x86. The package includes header files, object files and a library. The compiler option to enable CUDA Unified Memory is `-ta=tesla:managed`. It is a compile- and link-time option which causes all C/C++ `malloc`/`calloc`/`free` dynamic memory allocations, as well as C++ `new`/`delete`, to be performed using CUDA Unified Memory; similarly, all Fortran `ALLOCATE`/`DEALLOCATE` statements are performed using CUDA Unified Memory.

When an OpenACC executable is compiled with `-ta=tesla:managed`, the OpenACC runtime dynamically checks variables that normally would be moved to/from accelerator memory. If they are allocated in CUDA Unified Memory the runtime performs no data movement on these variables, allowing the CUDA driver to manage any necessary data movement between system and device memories. Otherwise, the OpenACC runtime proceeds normally. OpenACC runtime-managed data and dynamically-allocated driver-managed CUDA Unified Memory can be used in the same program. For a more extensive description of the capabilities of this feature, see http://www.pgroup.com/lit/articles/insider/v6n2a4.htm.

Chapter 3. INSTALLATION AND CONFIGURATION

Follow these steps to install PGI 16.10 compilers on a Linux/OpenPOWER system. The default installation directory is `/opt/pgi`, but it can be any directory:

```
% tar zxpf pgilinux-2016-1610-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 2016 directory.

   This is the directory where CUDA is installed, along with example programs; links are created to the subdirectories of `/opt/pgi/linuxpower/16.10`.
6. Install Open MPI.

3.1. License Management

This release includes a PGI Community Edition license key which is stored in a file named license.pgi in the installation directory. This license key allows use of PGI 16.10 through November 30, 2017.

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/2016/man
% set path=($PGI/linuxpower/2016/bin $path)
% which pgc++
/opt/pgi/linuxpower/2016/bin/pgc++
%
```

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

```
% export PGI=/opt/pgi
% export MANPATH=$MANPATH:$PGI/linuxpower/2016/man
% export PATH=$PGI/linuxpower/2016/bin:$PATH
% which pgc++
/opt/pgi/linuxpower/2016/bin/pgc++
%
```

The PGI directory structure is designed to accommodate co-installation of multiple PGI versions. When 16.10 is installed, it will be installed by default in the directory `/opt/pgi/linuxpower/16.10` and links can optionally be created to its sub-directories to make `16.10` default without affecting a previous (e.g., `16.9-beta`) 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/16.10/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-2016-16.10-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, visit the frequently asked questions (FAQ) section on pgroup.com at http://www.pgroup.com/support/faq.htm.

4.1. Release Specific Limitations

The following PGI features are limited or are not implemented in the 16.10 release for Linux/OpenPOWER+Tesla:

- No support for parsing of OpenMP 4.0/4.5 pragmas/directives.
- `-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.
- No debugging support for Fortran.

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: sales@pgroup.com
WWW: http://www.pgroup.com

The PGI User Forum is monitored by members of the PGI engineering and support teams as well as other PGI customers. The forum newsgroups may contain answers to commonly asked questions. Log in to the PGI website to access the forum:

http://www.pgroup.com/userforum/index.php

Many questions and problems can be resolved by following instructions and the information available at our frequently asked questions (FAQ) site:

http://www.pgroup.com/support/faq.htm

Submit technical support requests through the online form at:

https://www.pgroup.com/support/support_request.php

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

PGI®