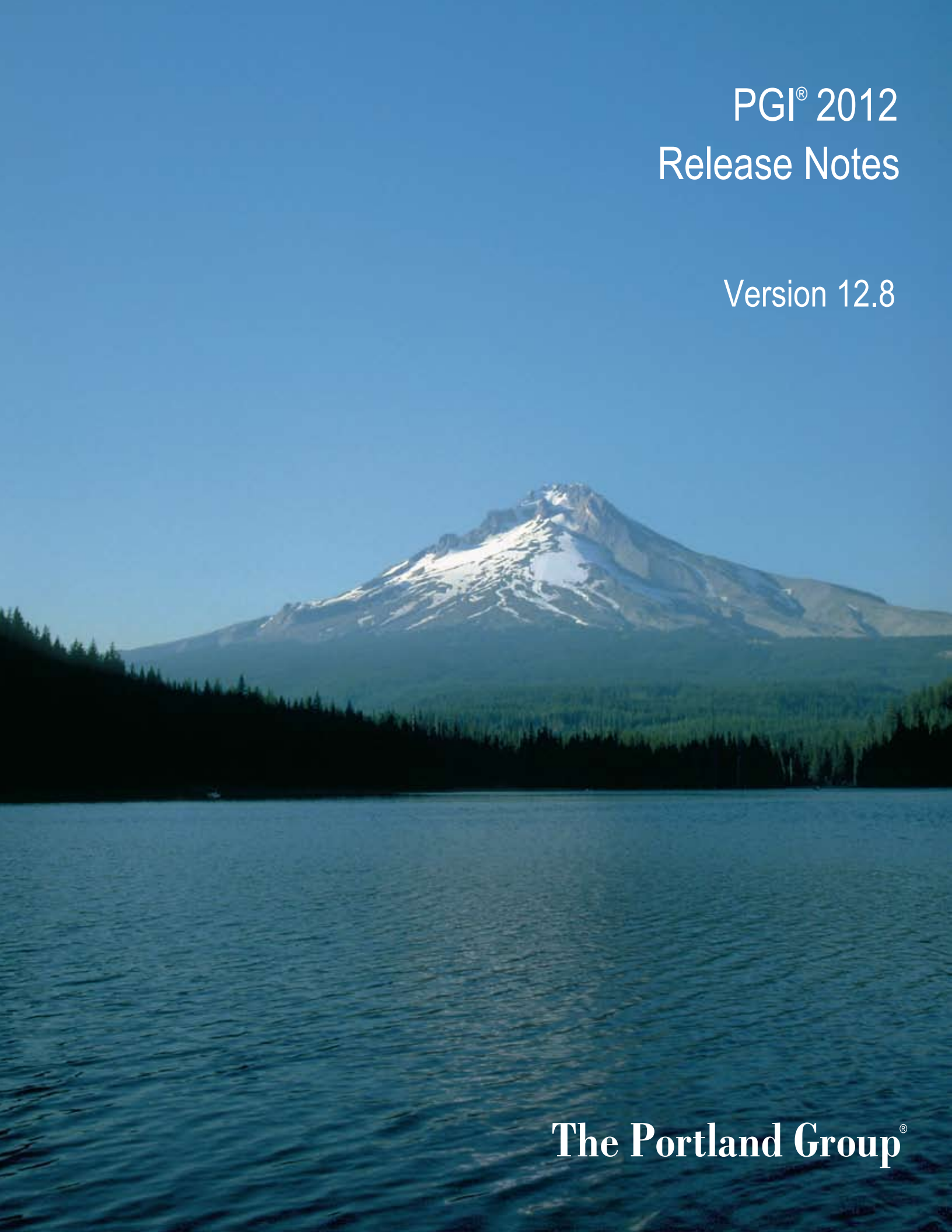
PGI® 2012 Release Notes

Version 12.8



The Portland Group®

PGI® 2012 Release Notes

Printed in the United States of America

First Printing: Release 2012, version 12.1, January 2012
Second Printing: Release 2012, version 12.2, February 2012
Third Printing: Release 2012, version 12.3, March 2012
Fourth Printing: Release 2012, version 12.4, April 2012
Fifth Printing: Release 2012, version 12.5, May 2012
Sixth Printing: Release 2012, version 12.6, June 2012
Seventh Printing: Release 2012, version 12.8, August 2012

Technical support: trs@pgroup.com
Sales: sales@pgroup.com
Web: www.pgroup.com

ID: 12222144

Contents

Chapter 1. Release Overview

Welcome to Release 2012 of *PGI Workstation*™ , *PGI Server*™, and the *PGI CDK*® *Cluster Development Kit*®, a set of compilers and development tools for 32-bit and 64-bit x86-compatible processor-based workstations, servers, and clusters running versions of the Linux and Microsoft Windows operating systems. *PGI Workstation* and *PGI Server* are also available for the Apple Mac OS X operating system.

This document describes changes between previous versions of the PGI 2012 release as well as late-breaking information not included in the current printing of the *PGI Compiler User's Guide*.

Product Overview

PGI Workstation, PGI Server, and the *PGI CDK* include exactly the same PGI compiler and tools software. The difference is the manner in which the license keys enable the software.

Licensing Terminology

The PGI compilers and tools are license-managed. It is useful to have common terminology. These two terms are often confused, so they are clarified here:

- **License** – a legal agreement between ST and PGI end-users, to which users assent upon installation of any PGI product. The terms of the License are kept up-to-date in documents on pgroup.com and in the $PGI/<platform>/<rel_number> 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. License keys are generated by each PGI end-user on pgroup.com 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.

License Options

PGI offers licenses for either x64+GPU or x64 only platforms. *PGI Accelerator*™ products, the x64+GPU platform products, include support for the directive-based PGI Accelerator programming model, CUDA Fortran and PGI CUDA-x86. PGI Accelerator compilers are supported on all Intel and AMD x64 processor-based systems with CUDA-enabled NVIDIA GPUs running Linux, Mac OS X, or Windows.

PGI Workstation and PGI Server Comparison

- All *PGI Workstation* products include a node-locked single-user license, meaning one user at a time can compile on the one system on which the *PGI Workstation* compilers and tools are installed. The product and *license server* are on the same local machine.
- *PGI Server* products are offered in configurations identical to *PGI Workstation*, but include network-floating multi-user licenses. This means that two 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 Server* license keys are installed. There can be multiple installations of the *PGI Server* compilers and tools on machines connected to the license server; and the users can use the product concurrently, provided they are issued a license key by the license server.

PGI CDK Cluster Development Kit

A cluster is a collection of compatible computers connected by a network. The *PGI CDK* supports parallel computation on clusters of 32-bit and 64-bit x86-compatible AMD and Intel processor-based Linux and Windows workstations or servers interconnected by a TCP/IP-based network, such as Ethernet.

Support for cluster programming does not extend to clusters combining 64-bit processor-based systems with 32-bit processor-based systems, unless all are running 32-bit applications built for a common set of working x86 instructions.

Note

Compilers and libraries can be installed on other platforms not in the user's cluster, including another cluster, as long as all platforms use a common floating license server.

Release Components

Release 2012 includes the following components:

- PGFORTRAN™ native OpenMP and auto-parallelizing Fortran 2003 compiler.
- PGCC® native OpenMP and auto-parallelizing ANSI C99 and K&R C compiler.
- PGC++® native OpenMP and auto-parallelizing ANSI C++ compiler.
- *PGPROF*® MPI, OpenMP, and multi-thread graphical profiler.
- *PGDBG*® MPI, OpenMP, and multi-thread graphical debugger.
- MPICH MPI libraries, version 1.2.7, for both 32-bit and 64-bit development environments (Linux only).

Note

64-bit linux86-64 MPI messages are limited to <2GB size each.

- Precompiled OpenMPI library for both 32-bit and 64-bit MacOS development environments.
- A UNIX-like shell environment for 32-bit and 64-bit Windows platforms.
- FlexNet license utilities.

- Documentation in PDF and man page formats.

Additional components for PGI CDK

PGI CDK for Linux also includes these components:

- MPICH2 MPI libraries, version 1.0.5p3, for both 32-bit and 64-bit development environments.
- MVAPICH MPI libraries, version 1.1, for both 32-bit and 64-bit development environments.
- ScaLAPACK linear algebra math library for distributed-memory systems, including BLACS version 1.1 – the Basic Linear Algebra Communication Subroutines) and ScaLAPACK version 1.7 for use with MPICH or MPICH2 and the PGI compilers on Linux systems with a kernel revision of 2.4.20 or higher. This is provided in both linux86 and linux86-64 versions for AMD64 or Intel 64 CPU-based installations.

Depending on the product configuration you purchased, you may not have licensed all of the above components.

You can use PGI products to develop, debug, and profile MPI applications. The MPI profiler and debugger included with *PGI Workstation* are limited to eight local processes. The MPI profiler and debugger included with *PGI Server* are limited to 16 local processes. The MPI profiler and debugger included with *PGI CDK* supports up to 256 remote processes.

Terms and Definitions

These release notes contain a number of terms and definitions with which you may or may not be familiar. If you encounter an unfamiliar term in these notes, please refer to the online glossary at

www.pgroup.com/support/definitions.htm

These two terms are used throughout the documentation to reflect groups of processors:

- AMD64 – a 64-bit processor from AMD designed to be binary compatible with 32-bit x86 processors, and incorporating new features such as additional registers and 64-bit addressing support for improved performance and greatly increased memory range. This term includes the AMD Athlon64, AMD Opteron, AMD Turion, AMD Barcelona, AMD Shanghai, AMD Istanbul, and AMD Bulldozer processors.
- Intel 64 – a 64-bit IA32 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.

Supported Platforms

There are six platforms supported by the *PGI Workstation* and *PGI Server* compilers and tools. Currently, *PGI CDK* supports only the first four of these.

- *32-bit Linux* - supported on *32-bit Linux operating systems* running on either a 32-bit *x86* compatible or an *x64* compatible processor.
- *64-bit/32-bit Linux* – includes all features and capabilities of the 32-bit Linux version, and is also supported on *64-bit Linux operating systems* running an *x64* compatible processor.

- *32-bit Windows* – supported on *32-bit Windows operating systems* running on either a 32-bit *x86* compatible or an *x64* compatible processor.
- *64-bit/32-bit Windows* – includes all features and capabilities of the 32-bit Windows version, and is also supported on *64-bit Windows operating systems* running an *x64* compatible processor.
- *32-bit Mac OS X* – supported on 32-bit Apple Mac operating systems running on either a 32-bit or 64-bit Intel-based Mac system.
- *64-bit Mac OS X* – supported on 64-bit Apple Mac operating systems running on a 64-bit Intel-based Mac system.

Supported Operating Systems

This section describes updates and changes to PGI 2012 that are specific to Linux, Mac OS X, and Windows.

Linux

Java Runtime Environment (JRE)

Although the PGI installation on Linux includes a 32-bit version of the Java Runtime Environment (JRE), sufficient 32-bit X Windows support must be available on the system for the JRE and the PGI software that depends on it to function properly. On some systems, notably recent releases of Fedora Core, these libraries are not part of the standard installation.

The required X Windows support generally includes these libraries:

```
libXau          libXdmcp        libxcb
libX11          libXext
```

Mac OS X

PGI 2012 for Mac OS X supports most of the features of the 32-bit and 64-bit versions for linux86 and linux86-64 environments. Except where noted in these release notes or the user manuals, the PGI compilers and tools on Mac OS X function identically to their Linux counterparts.

Windows

PGI 2012 for Windows supports most of the features of the 32-bit and 64-bit versions for linux86 and linux86-64 environments.

Getting Started

By default, the PGI 2012 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 `-fast` or `-fastsse`.

These 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 for 64-bit targets. They enable vectorization with SSE instructions, cache alignment, and flushz.

Note

The contents of the `-fast` and `-fastsse` options are host-dependent.

`-fast` and `-fastsse` typically include these options:

`-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` for 64-bit targets and `-fastsse` for both 32- and 64-bit targets also typically include:

`-Mvect=sse`	Generates SSE instructions.
`-Mscalarsse`	Generates scalar SSE code with xmm registers; implies `-Mflushz`.
`-Mcache_align`	Aligns long objects on cache-line boundaries **Note.** On 32-bit systems, if one file is compiled with the `-Mcache_align` option, all files should be compiled with it. This is not true on 64-bit systems.
`-Mflushz`	Sets SSE to flush-to-zero mode.
`-M[no]vect`	Controls automatic vector pipelining.

Note

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

In addition to `-fast` and `-fastsse`, the `-Mipa=fast` option for inter-procedural 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 or Modified Features

This chapter provides information about the new or modified features of Release 2012 of the PGI compilers and tools.

What's New in Release 2012

12.8 Updates and Additions

- PGI 12.8 provides an initial implementation of the texture attribute qualifier to the CUDA Fortran language. The `texture` attribute may be added to a CUDA Fortran array pointer. This attribute is only supported for `integer` and `real` types. For more information, refer to "Using the `texture` Attribute in CUDA Fortran," on page 10.

12.6 Updates and Additions

- OpenACC now supports the `cache` construct and the entire OpenACC API library.
- PGI supports ACML version 5.1.0. For 32-bit operating systems, the ACML version is 4.4.0.
- PGI Accelerator x64+GPU native Fortran 2003 and C99 compilers and CUDA Fortran now support the CUDA 4.1 Toolkit as the default toolkit. PGI compilers and tools also support CUDA 4.2 Toolkit.

12.5 Updates and Additions

- OpenACC has initial support for the `acc parallel` construct.
- Type parameters for derived types, also known as parameterized derived types, are now supported.
- PGI 12.5 supports the `__m256`, `__m256d`, and `__m256i` data types in C and C++. Through use of the `immintrin.h` header file, over 100 intrinsics are defined and available which make use of `ymm` registers and other new hardware features in AVX-enabled processors.
- PGDBG has enhanced auto-scrolling in the PGDBG I/O Tab.

12.4 Updates and Additions

- PGDBG now supports AVX registers on Windows.

12.3 Updates and Additions

- OpenACC Open Beta (linux only)

  The OpenACC Application Program Interface is a collection of compiler directives and runtime routines that allow you, the programmer, to specify loops and regions of code in standard C and Fortran that you want offloaded from a host CPU to an attached accelerator, such as a GPU.

  To prepare your system for using the PGI OpenACC implementation, and to see examples of how to write, build and run programs using the OpenACC directive, refer to the *OpenACC Getting Started Guide*. More information about PGI's OpenACC implementation is available at http://www.pgroup.com/openacc.

Updates and Additions Prior to 12.3

- Release 1.0 of CUDA-x86, including optimized code generator and texture support.
- *PGDBG* supports local and remote debugging. For more information, refer to "New or Modified Tools Functionality," on page 8.
- Eclipse integration

  Eclipse is a free, open source, integrated software development environment. It can be obtained from the Eclipse Foundation at eclipse.org.

  On Linux systems, the PGI C and C++ compilers can be used with the Eclipse integrated development environment. To enable this feature, run the Eclipse plug-in installer. For more information on how to install Eclipse, refer to Chapter 8, "Eclipse," in the *PGI Compiler User's Guide*.
- `dflib` now contains the routine `makedirqq` that allows users to create a new directory.
- OpenMP nested parallelism support
- PGI Accelerator x64+GPU native Fortran 2003 and C99 compilers and CUDA Fortran now support the CUDA 4.0 Toolkit as the default toolkit.

New or Modified Compiler Options

Unknown options are treated as errors instead of warnings. This feature means it is a compiler error to pass switches that are not known to the compiler; however, you can use the switch `-noswitcherror` to issue warnings instead of errors for unknown switches.

New or Modified Tools Functionality

PGDBG is licensed software available from The Portland Group. *PGDBG* supports debugging programs running on local and remote systems. The PGI license keys that enable *PGDBG* to debug must be located on the same system where the program you want to debug is running.

Local debugging
: If you want to debug a program running on the system where you have launched *PGDBG*, you are doing local debugging and you need license keys on that local system.

Remote debugging

If you want to debug a program running on a system other than the one on which *PGDBG* is launched, then you are doing remote debugging and you need license keys on the remote system. The remote system also needs an installed copy of PGI Workstation, PGI Server, or PGI CDK.

Using MPICH-2 on Linux

PGI CDK for Linux includes MPICH-2 libraries, tools, and licenses required to compile, execute, profile, and debug MPI programs.

If you want to build your MPI application using the instance of MPICH-2 installed with the PGI compilers, you need to append the location of `libmpl.so.1` to the `LD_LIBRARY_PATH` environment variable.

For 32-bit:

```
%setenv LD_LIBRARY_PATH "$LD_LIBRARY_PATH":$PGI/linux86/2012/mpi2/mpich/lib:
/$PGI/linux86/12.8/libso
```

For 64-bit:

```
%setenv LD_LIBRARY_PATH "$LD_LIBRARY_PATH":$PGI/linux86-64/2012/mpi2/mpich/lib:
$PGI/linux86-64/12.8/libso
```

You may need to put in `.cshrc/.bashrc` when running a program on slave nodes.

Then add the `-Mmpi=mpich2` option to the compilation and link steps, or you can use the `-Mprof=mpich2` option to instrument for MPICH-2 profiling. The `-Mmpi=mpich2` option automatically sets up the include and library paths to use the MPICH-2 headers and libraries. For example, you can use the following command to compile for profiling with MPICH-2:

```
% pgfortran -fast -Mprof=mpich2,time my_mpi_app.f90
```

To use a different instance of MPICH-2, set the `MPIDIR` environment variable before invoking and linking with `-Mmpi=mpich2`. `MPIDIR` specifies the location of the instance of MPI to use. For example, set `MPIDIR` to the root of the MPICH-2 installation directory that you want to use, that is, the directory that contains `bin`, `include`, `lib`, and so on.

PGI Accelerator and CUDA Fortran Enhancements

PGI Accelerator x64+GPU native Fortran 95/03 and C99 compilers and CUDA Fortran now support the CUDA 4.1 Toolkit as the default toolkit. PGI compilers and tools also support CUDA 4.2 Toolkit and continue to support previous versions of CUDA provided these toolkits exist on your system from a previous installation.

Since the default toolkit for PGI 2012 version 12.8 is CUDA 4.1, CUDA Fortran host programs should be recompiled. The PGI 12.8 CUDA Fortran runtime libraries are not compatible with previous CUDA Fortran releases.

To specify the version of the CUDA Toolkit that is targeted by the compilers, use one of the following options:

In PGI Accelerator:

For CUDA Toolkit 4.1

`-ta=nvidia:cuda4.1` or `-ta=nvidia:4.1`

For CUDA Fortran:

For CUDA Toolkit 4.1

```
-Mcuda=cuda4.1  or  -Mcuda=4.1
```

You can also specify a specific version by adding a line to the `siterc` file in the installation `bin/` directory or to a file named `.mypgirc` in your home directory. For example, to specify CUDA Toolkit 4.1, add the following line to one of these files:

```
set DEFCUDAVERSION=4.1;
```

PGI Accelerator Runtime Routines

For complete description of the PGI Accelerator model runtime routines available in version 12.8, refer to Chapter 4, "PGI Accelerator Compilers Reference" of the *PGI Compiler Reference Manual*.

Memory Management in CUDA

A new memory management routine, `cudaMemGetInfo`, returns the amount of free and total memory available (in bytes) for allocation on the device.

The syntax for `cudaMemGetInfo` is:

```
integer function cudaMemGetInfo( free, total )
     integer(kind=cuda_count_kind) :: free, total
```

Declaring Interfaces to CUDA Device Built-in Routines

A Fortran module is available to declare interfaces to many of the CUDA device built-in routines.

To access this module, do one of the following:

- Add this line to your Fortran program:

  ```
  use cudadevice
  ```

- Add this line to your C program:

  ```
  #include <cudadevice.h>
  ```

You can also use these routines in CUDA Fortran global and device subprograms, in CUF kernels, and in PGI Accelerator compute regions both in Fortran and in C. Further, the PGI compilers come with implementations of these routines for host code, though these implementations are not specifically optimized for the host.

For a complete list of the CUDA built-in routines that are available, refer to the *PGI CUDA Fortran Programming and Reference*.

Using the `texture` Attribute in CUDA Fortran

To use the texture attribute as supported in this release, do the following:

1. Add a declaration similar to the following one to a module declaration section that is used in both the host and device code:

```
          real, texture, pointer :: t(:)
```

2. In your host code, add the target attribute to the device data that you wish to put into texture memory:

   Change:
   ```
          real, device :: a(n)
   ```
   To:
   ```
          real, target, device :: a(n)
   ```

   The target attribute is standard F90/F2003 syntax to denote an array or other data structure that may be "pointed to" by another entity.

3. Tie the global (by module use-association in both the host program and device subroutine) texture declaration to the device array by using the F90 pointer assignment operator, so a simple expression like the following one performs all the underlying CUDA texture binding operations.

   ```
          t =>  a
   ```

   Your CUDA Fortran device code contained in the module that declares `t`, or uses a module that contains the declaration of `t`, can now access `t` without any other declaration. For example:

   ```
   ! Vector add, s through device memory, t is through texture memory
   i = threadIdx%x + (blockIdx%x-1)*blockDim%x
   s(i) = s(i) + t(i)
   ```

   Accesses of `t`, targeting `a`, go through the texture cache.

C++ Compatibility

PGI 2012 C++ object code is incompatible with prior releases.

All C++ source files and libraries that were built with prior releases must be recompiled to link with PGI 2012 or higher object files.

New or Modified Runtime Library Routines

PGI 2012 supports new runtime library routines associated with the PGI Accelerator compilers. For more information, refer to the ”Using an Accelerator” chapter of the *PGI Compiler User’s Guide*.

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 Chapter 8 of the *PGI Compiler User’s Guide*.

Environment Modules

Note

This section is only applicable to *PGI CDK*

On Linux, if you use the Environment Modules package (e.g., the **module load** command), then PGI 2012 includes a script to set up the appropriate module files.

Mac OS X Lion Support

If you upgraded to Mac OS X Lion, it is best to update Xcode to 4.0 or later before installing the PGI compilers and tools. To update, follow these steps:

1. Go to Apple App store. Apple menu | App Store...
2. Search for "Xcode"
3. Click the "Install" button.
4. Once step 3 is complete, double click the "Install Xcode" icon in the Application folder and follow the directions on the screen.

PGI CUDA C++ Compilers for x86

Developers can utilize the PGI C++ compiler to compile CUDA C/C++ code and then run it on an x86 target.

Compiler Options

Certain options may be useful when targeting your CUDA build for x86.

`-Mcudax86`
: Add this option on the PGI C++ command line. If the file extension is `.cu`, then this option may not be required. In addition to enabling recognition of CUDA syntax, this option pulls the required libraries into the link process.

`--no_using_std`
: Use this option to disable implicit use of the standard namespace in C++. This option is important for consistent behavior between pgCC and g++.

Sample Code

Here is an example of building and running the CUDA SDK MonteCarlo example on Linux:

```
% pgcpp --no_using_std -Mcudax86 -DUNIX -O2 \
        -I. -I../../common/inc -I../../shared/inc \
        MonteCarlo.cpp MonteCarlo_gold.cpp MonteCarlo_SM10.cu MonteCarlo_SM13.cu \
        -L../../lib -L../../common/lib/linux -L../../../shared/lib \
        -lcutil_x86_64 -lshrutil_x86_64
```

```
% ./a.out
[Monte Carlo]
Generating input data...
Allocating memory...
Generating normally distributed samples...
Running GPU Monte Carlo...
Options         : 256
Simulation paths: 262144
Time (ms.)      : 524.278015
GPU options per sec.: 488.290549
GPU Monte Carlo vs. Black-Scholes statistics
L1 norm        : 2.971674E-06
Average reserve: 387.263539
CPU Monte Carlo vs. Black-Scholes statistics...
```

```
L1 norm: 2.970427E-06
Average reserve: 386.847322
CPU vs. GPU Monte Carlo statistics...
L1 norm: 3.964267E-08
[Monte Carlo] - Test summary
PASSED
```

Debugging CUDA-x86 Applications with PGDBG

Developers can use *PGDBG* to debug their CUDA device code. When setting breakpoints in device code, `OMP_NUM_THREADS` threads, each running one task in emulation of a CUDA thread, hit the breakpoint in parallel.

CUDA C++ for GPUs

Some features of CUDA C++ for GPUs are not supported in this release.

- Warp-synchronous programming.

  There is no current plan to support this feature. Many CUDA SDK examples use warp-synchronous programming techniques, for example in reductions; and these examples need to be rewritten for CUDA x86. For example, the reduction in the MonteCarlo code should be rewritten like this:

  ```
  template<class T, unsigned int blockSize>
  __device__ void sumReduceSharedMem(volatile T *sum, volatile T *sum2, int tid)
  {
      // do reduction in portable, non-warp-synchronous manner
      for(unsigned int s=256; s>0; s>>=1)
      {
          if (blockSize >= (s+s)) {
              if (tid < s) {
                  sum[tid] += sum[tid+s]; sum2[tid] += sum2[tid+s];
              }
              __syncthreads();
          }
      }
  }
  ```

- The CUDA driver-level API. This feature may be supported in a future release.
- OpenGL interoperability. This feature may be supported in a future release.

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 chapter addresses how to effectively distribute applications built using PGI compilers and tools.

Application Deployment and Redistributables

Programs built with PGI compilers may depend on runtime library files. These library files must be distributed with such programs to enable them to execute on systems where the PGI compilers are not installed. There are PGI redistributable files for all platforms. On Windows, PGI also supplies Microsoft redistributable files.

PGI Redistributables

The PGI 2012 release includes these directories:

$PGI/linux86/12.8/REDIST
$PGI/linux86-64/12.8/REDIST
$PGI/osx86/12.8/REDIST
$PGI/win32/12.8/REDIST
$PGI/win64/12.8/REDIST

These directories contain all of the PGI Linux runtime library shared object files, Mac OS dynamic libraries, or Windows dynamically linked libraries that can be re-distributed by PGI 2012 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 2012 directory.

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.

Microsoft Redistributables

The PGI products on Windows include Microsoft Open Tools. The Microsoft Open Tools directory contains a subdirectory named "redist". PGI 2012 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 chapter contains information about known limitations, documentation errors, and corrections.

For up-to-date information about the state of the current release, visit the frequently asked questions (FAQ) section on pgroup.com at: www.pgroup.com/support/index.htm

General Issues

Most issues in this section are related to specific uses of compiler options and suboptions.

- Object files created with prior releases of PGI compiler are incompatible with object files from PGI 2012 and should be recompiled.
- The `-i8` option can make programs incompatible with the ACML libraries; use of any INTEGER*8 array size argument can cause failures. Visit developer.amd.com to check for compatible libraries.
- Using `-Mipa=vestigial` in combination with `-Mipa=libopt` with PGCC, you may encounter unresolved references at link time. This problem is due to the erroneous removal of functions by the vestigial sub-option to `-Mipa`. You can work around this problem by listing specific sub-options to `-Mipa`, not including `vestigial`.
- OpenMP programs compiled using `-mp` and run on multiple processors of a SuSE 9.0 system can run very slowly. These same executables deliver the expected performance and speed-up on similar hardware running SuSE 9.1 and above.

Platform-specific Issues

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.

Apple Mac OS X

The following are known issues on Mac OS X:

- On MacOS platform, the PGI 2012 compilers do not support static linking of binaries. For compatibility with future Apple updates, the compilers only support dynamic linking of binaries.
- Using `–Mprof=func` or `–Mprof=lines` is not supported.

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.

  The Cygwin team is working to resolve this issue.
- 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 workaround this problem, set SHELL to refer to a shell in the cygwin bin directory, e.g. /bin/bash.
- C++ programs on Win64 that are compiled with the option `-tp x64` fail when using PGI Unified Binaries. The `-tp x64` switch is not yet supported on the Windows platform for C++.
- On Windows, runtime libraries built for debugging (e.g. `msvcrtd` and `libcmtd`) are not included with *PGI Workstation*. 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.

The following are known issues on Windows and *PGDBG*:

- In *PGDBG* on the Windows platform, Windows times out stepi/nexti operations when single stepping over blocked system calls. For more information on the workaround for this issue, refer to the online FAQs at www.pgroup.com/support/tools.htm.

The following are known issues on Windows and *PGPROF*:

- Do not use `-Mprof` with PGI runtime library DLLs. To build an executable for profiling, use the static libraries. When the compiler option `-Bdynamic` is not used, the static libraries are the default.

PGDBG-related Issues

The following are known issues on *PGDBG*:

- Before *PGDBG* can set a breakpoint in code contained in a shared library, `.so` or `.dll`, the shared library must be loaded.

- Breakpoints in processes other than the process with rank 0 may be ignored when debugging MPICH-1 applications when the loading of shared libraries to randomized addresses is enabled.
- Debugging of PGI Unified Binaries, that is, 64-bit programs built with more than one `-tp` option, is not fully supported. The names of some subprograms are modified in the creation, and *PGDBG* does not translate these names back to the names used in the application source code. For detailed information on how to debug a PGI Unified Binary, see www.pgroup.com/support/tools.htm.
- To begin an OpenMPI debugging session with *PGDBG* on Mac OS X Snow Leopard or later, use the following steps:

  1. Invoke the debugger using the full pathname of the executable. For example, you might use a command similar to this one:

     ```
     pgdbg -mpi:mpirun -np 4 /home/user1/a.out
     ```

  2. Set a breakpoint on main.
  3. Continue to the breakpoint.
  4. Begin your debugging session.

PGPROF-related Issues

The following are known issues on *PGPROF*:

- Programs compiled and linked for `gprof`-style performance profiling using `-pg` can result in segmentation faults on system running version 2.6.4 Linux kernels.
- Times reported for multi-threaded sample-based profiles, that is, profiling invoked with options `-pg` or `-Mprof=time`, are for the master thread only. To obtain profile data on individual threads, PGI-style instrumentation profiling with `-Mprof={lines | func}` or **pgcollect** must be used.

CUDA Fortran Toolkit Issues

The CUDA 4.1 Toolkit is set as the default in PGI 2012. To use the CUDA 4.1 Toolkit, first download the CUDA 4.1 driver from NVIDIA at www.nvidia.com/cuda.

You can compile with the CUDA 4.1 Toolkit either by adding the `-ta=nvidia:cuda4.1` option to the command line or by adding `set CUDAVERSION=4.1` to the `siterc` file.

`pgaccelinfo` prints the driver version as the first line of output. For a 4.1 driver, it prints:

```
CUDA Driver Version 4010
```

Corrections

A number of problems have been corrected in the PGI 2012 release. Refer to www.pgroup.com/support/release_tprs.htm for a complete and up-to-date table of technical problem reports, TPRs, fixed in recent releases of the PGI compilers and tools. This table contains a summary description of each problem as well as the version in which it was fixed.

Chapter 5. Contact Information

You can contact The Portland Group at:

The Portland Group
STMicroelectronics, Inc.
Two Centerpointe Drive, Suite 320
Lake Oswego, OR 97035 USA

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:

www.pgroup.com/userforum/index.php

Or contact us electronically using any of the following means:

Fax	+1-503-682-2637
Sales	sales@pgroup.com
Support	trs@pgroup.com
WWW	www.pgroup.com

All technical support is by email or submissions using an online form at www.pgroup.com/support. Phone support is not currently available.

Many questions and problems can be resolved at our frequently asked questions (FAQ) site at www.pgroup.com/support/faq.htm.

PGI documentation is available at www.pgroup.com/resources/docs.htm or in your local copy of the documentation in the release directory doc/index.htm.