

PGI[®] Server 2010
PGI[®] Workstation 2010
リリースノート

Release 2010

2009年11月

2010年3月更新 (PGI 10.3)
2010年4月更新 (PGI 10.4)
2010年5月更新 (PGI 10.5)
2010年6月更新 (PGI 10.6)
2010年8月更新 (PGI 10.8)
2010年9月更新 (PGI 10.9)

株式会社 ソフテック

目次

1	サポートするプラットフォーム	1
1.1	PGI Workstation/Server Release 2010 のソフトウェア内容	2
1.2	サポートするプロセッサ・システムについて	2
1.3	サポートする OS の追加	3
2	PGI 2010 の新機能、変更点等	5
2.1	新しい機能の概要	5
2.2	PGI 2010 の新コンパイラ・オプションと従来との変更部分	9
2.3	C++コンパイル 留意点	11
2.4	Fortran の強化	11
2.4.1	Fortran と C の相互運用性の強化	11
2.4.2	新しい Fortran 組込関数 (intrinsic)	12
2.4.4	新しい組込みモジュール	12
2.4.5	その他の Fortran 強化	14
2.5	新しい Runtime (ランタイム) ライブラリ・ルーチン	14
2.6	PGI Tools の変更点 (新機能、修正機能)	14
2.6.1	PGPROF	14
2.8	ライブラリ・インタフェース	15
3	PGI Workstation 2010	15
3.1	PGI Workstation 2010 for Linux	15
3.1.1	Java Runtime Environment (JRE)	15
3.2	PGI Workstation 2010 for Windows	15
3.3	PGI Workstation 2010 for Mac OS X	15
4	PGI アクセラレータ	15
4.1	コンポーネント	16
4.2	可用性	16
4.3	ユーザ指示のアクセラレータ・プログラミング	16
4.4	カバーしていない、あるいは実装していない機能	16
4.5	必要とされるシステム仕様	16
4.5.1	サポートするプロセッサと GPU	17
4.6	インストールとライセンス	17
4.7	アクセラレータ・プログラムの実行	17
4.7.1	PGI アクセラレータ・コンパイラのランタイム・ライブラリ	17
4.7.2	環境変数	18
4.7.3	コンパイラ・コマンド・オプション	19
4.8	アクセラレータ用の PGI Unified Binary	20
4.9	複数のプロセッサターゲット	21
4.10	複数のアクセラレータ	21
4.11	アクセラレータ・カーネルのプロファイリング	21
4.12	サポートする組込関数	22
5	配布と配置	22
5.1	アプリケーションの配置と再配布	22
5.1.1	PGI 再配布用ランタイム・ライブラリ	23

5.1.2 Linux 再配布用ランタイム・ライブラリ	23
5.1.3 Microsoft® 再配布用ランタイム・ライブラリ	23
6. 既知の制限事項	23
6.1 一般的な制限事項	23
6.2 プラットフォーム特有の制限事項	24
6.2.1 Linux	24
6.2.2 Apple Mac OS X	24
6.2.3 Windows	25
6.3 PGDBG に関する制限事項	25
6.4 PGPROF に関する制限事項	26
6.4 修正事項	26

本資料の全ての情報は、現状のまま提供されます。株式会社ソフテックは、本資料に記述あるいは表現されている情報及びその中に非明示的に記載されていると解釈されうる情報に対して一切の保証をいたしません。また、本資料に含まれる情報の誤りや、それによって生じるいかなるトラブルに対しても一切の責任と補償義務を負いません。また、本資料に掲載されている内容は、予告なく変更されることがあります。

本資料で使用されている社名、製品名などは、一般に各社の商標または登録商標です。

株式会社ソフテック

〒 154-0004 東京都世田谷区太子堂 1-12-39

<http://www.softek.co.jp/>

**Copyright © 2010, SofTek Systems, Inc.
All rights reserved.**

米国 PGI 社の PGI Workstation/Server コンパイラのバージョン 2010 のリリースノートを以下に記します。本リリースにおける製品の主な新機能、特徴に関して纏めたものです。

1 サポートするプラットフォーム

32 ビット Linux

32 ビット x86 互換あるいは 64 ビット x64 互換のプロセッサ上で動作する 32 ビット Linux オペレーティングシステム上で利用可能です。

64 ビット/32 ビット Linux

32 ビット Linux 用ソフトウェアの全ての機能も包括し、64 ビット x64 互換のプロセッサ上で動作する 64 ビット Linux オペレーティングシステム上で利用可能です。

32 ビット Windows

32 ビット x86 互換あるいは 64 ビット x64 互換のプロセッサ上で動作する 32 ビット Windows® オペレーティングシステム上で利用可能です。

64 ビット/32 ビット Windows

32 ビット Windows 用ソフトウェアの全ての機能も包括し、64 ビット x64 互換のプロセッサ上で動作する 64 ビット Windows® オペレーティングシステム上で利用可能です。

32 ビット Apple Mac OS X

32 ビットあるいは 64 ビットの Intel® ベースのプロセッサを搭載したシステムで、32 ビット Apple Mac オペレーティングシステム上で利用可能です。

64 ビット Apple Mac OS X

64 ビットの Intel® ベースのプロセッサを搭載したシステムで、64 ビット Apple Mac オペレーティングシステム上で利用可能です。

64 ビット/32 ビット SUA

32 ビット SUA の全ての機能を包括し、64 ビット x64 互換のプロセッサ上で動作する 64 ビット Windows オペレーティングシステム上の SUA(the Subsystem for Unix-base Applications) で利用可能です。(Windows 製品のライセンスキーが必要です)

32 ビット SFU

32 ビット x86 互換あるいは 64 ビット x64 互換のプロセッサ上で動作する 32 ビット Windows オペレーティングシステム上の SFU(Windows Services for Unix) で利用可能です。(Windows® 製品のライセンスキーが必要です)

32 ビット SUA

32 ビット x86 互換あるいは 64 ビット x64 互換のプロセッサ上で動作する 32 ビット Windows オペレーティングシステム上の SUA(the Subsystem for Unix-base Applications) で利用可能です。(Windows® 製品のライセンスキーが必要です)

64 ビット/32 ビット SUA

32 ビット SUA の全ての機能を包括し、64 ビット x64 互換のプロセッサ上で動作する 64 ビット Windows オペレーティングシステム上の SUA(the Subsystem for Unix-base Applications) で利用可能です。(Windows 製品のライセンスキーが必要です)

1.1 PGI Workstation/Server Release 2010 のソフトウェア内容

- PGF95 ネイティブ OpenMP/自動並列 Fortran 95/90 コンパイラ
- PGF77 ネイティブ OpenMP/自動並列 FORTRAN 77 コンパイラ
- PGHPF データ並列 High Performance Fortran コンパイラ
(Note: PGHPF は Linux のみサポートします)
- PGCC ネイティブ OpenMP/自動並列 ANSI C99 and K&R C コンパイラ
- PGC++ ネイティブ OpenMP/自動並列 ANSI C++ コンパイラ
- PGPROF マルチスレッド/OpenMP/MPI 並列対応グラフィカル・プロファイラ
- PGDBG マルチスレッド/OpenMP/MPI 並列対応グラフィカル・デバッガ
 - (Linux 用製品のみ) MPICH MPI ライブラリ, version 1.2.7 (32-bit and 64-bit)をバンドル
 - PDF、HTML によるオンラインドキュメントと UNIX man ページ
 - (Windows 用製品のみ) Win32 and Win64 環境上で、UNIX ライクの bash シェル・コマンド環境(Cygwin)

(注意) ご購入ライセンス製品によって、使用できるコンパイラ言語の種類が決まります。

1.2 サポートするプロセッサ・システムについて

- 32 ビット x86 系のプロセッサ並びに 64 ビットの AMD64、Intel® 64 のプロセッサに対応します。PGI 2010 において、サポートする CPU は以下の表のとおりです。

Processors supported by PGI 2010										
Brand	CPU	cores	-tp <target>	Mem Addressing	Floating point HW					
					SSE1	SSE2	SSE3	SSSE3	SSE4	ABM SSE4a
AMD	Istanbul	6	istanbul-64	64-bit	Yes	Yes	Yes	No	Yes	Yes
AMD	Istanbul	6	istanbul	32-bit	Yes	Yes	Yes	No	Yes	Yes
AMD	Opteron/Quad-Core	4	shanghai-64	64-bit	Yes	Yes	Yes	No	No	Yes
AMD	Opteron/Quad-Core	4	shanghai	32-bit	Yes	Yes	Yes	No	No	Yes
AMD	Opteron/Quad-Core	4	Barcelona-64	64-bit	Yes	Yes	Yes	No	No	Yes
AMD	Opteron/Quad-Core	4	Barcelona	32-bit	Yes	Yes	Yes	No	No	Yes
AMD	Opteron/Athlon64	2	k8-64	64-bit	Yes	Yes	Yes	No	No	No
AMD	Opteron/Athlon64	2	k8-32	32-bit	Yes	Yes	Yes	No	No	No
AMD	Opteron Rev E/F Turion/Athlon64	2	k8-64e	64-bit	Yes	Yes	Yes	No	No	No
AMD	Opteron Rev E/F	2	k8-32	32-bit	Yes	Yes	Yes	No	No	No
AMD	Turion64 Turion/Athlon64	1	k8-64e	64-bit	Yes	Yes	Yes	No	No	No
AMD	Turion64	1	k8-32	32-bit	Yes	Yes	No	No	No	No
Intel	Core i7/Core i5/i3 (Nehalem)	4	nehalem-64	64-bit	Yes	Yes	Yes	Yes	Yes	Yes
Intel	Core i7/Core i5/i3 (Nehalem)	4	nehalem	32-bit	Yes	Yes	Yes	Yes	Yes	Yes

Intel	Penryn	4	penryn	32-bit	Yes	Yes	Yes	Yes	Yes	No
Intel	Penryn	4	penryn-64	64-bit	Yes	Yes	Yes	Yes	Yes	No
Intel	Core 2	2	Core2	32-bit	Yes	Yes	Yes	Yes	Yes	No
Intel	Core 2	2	Core2-64	64-bit	Yes	Yes	Yes	Yes	Yes	No
Intel	P4/Xeon EM64T	2	p7-64	64-bit	Yes	Yes	Yes	Yes	No	No
Intel	P4/Xeon EM64T	2	p7	32-bit	Yes	Yes	Yes	Yes	No	No
Intel	Xeon/Pentium4	1	p7	32-bit	Yes	Yes	No	No	No	No
AMD	Athlon XP/MP	1	athlonxp	32-bit	Yes	No	No	No	No	No
Intel	Pentium III	1	piii	32-bit	Yes	No	No	No	No	No
AMD	Athlon	1	athlon	32-bit	No	No	No	No	No	No
AMD	K6	1	k6	32-bit	No	No	No	No	No	No
Intel	Pentium II	1	p6	32-bit	No	No	No	No	No	No
Generic	Generic x86	1	p5 or px	32-bit	No	No	No	No	No	No

1.3 サポートする OS の追加

- PGI 2010 において、サポートする OS は以下のとおりです。なお、PGI 7.0 以前でサポートしておりました、古い Linux Distribution である RedHat 8.0/7.3、SuSE 8.1/8.0、SLES8 SP2 は、PGI 7.2 以降サポートしていませんのでご注意ください。

Linux Operating Systems and Features Supported in PGI 2010									
Distribution	Type	64-bit	HT	pgC++	pgdbg	NPTL	NUMA	glibc	GCC
RHEL 5.5	Linux	Yes	Yes	Yes	Yes	Yes	No	2.5	4.1.2
RHEL 5.4	Linux	Yes	Yes	Yes	Yes	Yes	No	2.5	4.1.2
RHEL 5.3	Linux	Yes	Yes	Yes	Yes	Yes	No	2.5	4.1.2
RHEL 5.0	Linux	Yes	Yes	Yes	Yes	Yes	No	2.5	4.1.2
RHEL 4.0	Linux	Yes	Yes	Yes	Yes	Yes	No	2.3.4	3.4.3
RHEL 3.0	Linux	Yes	Yes	Yes	Yes	Yes	No	2.3.2	3.2.3
Fedora C-12	Linux	Yes	Yes	Yes	Yes	Yes	Yes	2.11	4.4.2
Fedora C-11	Linux	Yes	Yes	Yes	Yes	Yes	Yes	2.9	4.3.3
Fedora C-10	Linux	Yes	Yes	Yes	Yes	Yes	Yes	2.9	4.3.2
Fedora C-9	Linux	Yes	Yes	Yes	Yes	Yes	Yes	2.8	4.3.0
Fedora C-8	Linux	Yes	Yes	Yes	Yes	Yes	Yes	2.7	4.1.2
Fedora C-7	Linux	Yes	Yes	Yes	Yes	Yes	Yes	2.6	4.1.2
Fedora C-6	Linux	Yes	Yes	Yes	Yes	Yes	Yes	2.5	4.1.1
Fedora C-5	Linux	Yes	Yes	Yes	Yes	Yes	Yes	2.4	4.1.0
Fedora C-4	Linux	Yes	Yes	Yes	Yes	Yes	No	2.3.5	4.0.0
Fedora C-3	Linux	Yes	Yes	Yes	Yes	Yes	No	2.3.3	3.4.2
Fedora C-2	Linux	Yes	Yes	Yes	Yes	Yes	No	2.3.3	3.3.3
SUSE 11.1	Linux	Yes	Yes	Yes	Yes	Yes	Yes	2.9	4.3.3
SUSE 11.0	Linux	Yes	Yes	Yes	Yes	Yes	Yes	2.8	4.3.0
SuSE 10.3	Linux	Yes	Yes	Yes	Yes	Yes	Yes	2.6.1	4.2.1
SuSE 10.2	Linux	Yes	Yes	Yes	Yes	Yes	Yes	2.5	4.1.0
SuSE 10.1	Linux	Yes	Yes	Yes	Yes	Yes	Yes	2.4	4.1.0

SuSE 10.0	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.3.5	4.0.2
SuSE 9.3	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.3.4	3.3.5
SuSE 9.2	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.3.3	3.3.4
SLES 11	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.9	4.3.3
SLES 10	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.4	4.1.0
SLES 9	<i>Linux</i>	Yes	Yes	Yes	Yes	No	Yes	2.3.3	3.3.3
SuSE 9.1	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	No	2.3.3	3.3.3
SuSE 9.0	<i>Linux</i>	Yes	Yes	Yes	Yes	No	No	2.3.2	3.3.1
SuSE 8.2	<i>Linux</i>	Yes	Yes	Yes	Yes	No	No	2.3.2	3.3
Red Hat 9.0	<i>Linux</i>	No	No	Yes	Yes	Yes	No	2.3.2	3.2.2
Ubuntu 10.04	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.11.1	4.4.3
Ubuntu 9.10	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.10.1	4.4.1
Ubuntu 9.04	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.9	4.3.3
Ubuntu 8.10	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.8	4.3.2
Ubuntu 8.04	<i>Linux</i>	Yes	Yes	Yes	Yes	Yes	Yes	2.7	4.2.1

HT = *hyper-threading*, NPTL = *Native POSIX Threads Library*, NUMA = *Non-Uniform Memory Access*.

- Windows 上で使用可能なコマンドベースのコンパイラ (PGI Workstation & Server) が対応する OS は、以下のとおりです。PGI 2010 は、32 ビットの **Windows 2000** はサポートしません。

Windows® Operating Systems and Features Supported in PGI 2010									
OS	Type	64-bit	HT	pgC++	pgdbg	NPTL	NUMA	glibc	GCC
Microsoft® Windows® (32ビット)	7	No	Yes	Yes	Yes	NA	Yes	NA	NA
	Vista	No	Yes	Yes	Yes	NA	Yes	NA	NA
	XP	No	Yes	Yes	Yes	NA	Yes	NA	NA
	2003	No	No	Yes	Yes	NA	Yes	NA	NA
	2008	No	Yes	Yes	Yes	NA	Yes	NA	NA
	SFU	No	Yes	Yes	Yes	NA	Yes	SFU	3.3
	SUA x86	No	Yes	Yes	Yes	NA	Yes	SUA	3.3
Microsoft® Windows® (64ビット)	7	Yes	Yes	Yes	Yes	NA	Yes	NA	NA
	Vista	Yes	Yes	Yes	Yes	NA	Yes	NA	NA
	XP 64	Yes	Yes	Yes	Yes	NA	Yes	NA	NA
	2003 x64	Yes	Yes	Yes	Yes	NA	Yes	NA	NA
	SUA x64	Yes	Yes	Yes	Yes	NA	Yes	SUA	3.3
	2008 x64	Yes	Yes	Yes	Yes	NA	Yes	NA	NA

- Intel プロセッサベースの Mac OS X は、以下のとおりです。

Apple Operating Systems and Features Supported in PGI 2010									
OS	Type	64-bit	HT	pgC++	pgdbg	NPTL	NUMA	glibc	GCC

Apple Mac OS X	Snow Leopard	Yes	No	Yes	Yes	NA	NA	NA	4.0.1
	Leopard	Yes	No	Yes	Yes	NA	NA	NA	4.0.1

2. PGI 2010 の新機能、変更点等

2.1 新しい機能の概要

[PGI 10.9 \(2010年9月\)リビジョンでの機能追加](#)

■ PGI CDK にバンドルされた MPICH-2 のバージョン変更

PGI CDK パッケージに含まれる MPICH2 ライブラリのバージョンを 1.2.1p1 に更新しました。

■ PGI Visual Fortran

PVF 10.9 では、新しい「プロパティメニュー」を追加しました。この機能は、Intel(R) Math Kernel Library、AMD Core Math Library や IMSL ライブラリ等の特定のライブラリを明示して、ビルド&リンクできるようにするためのものです。

[PGI 10.8 \(2010年8月\)リビジョンでの機能追加](#)

■ PGI 10.8 で追加された Fortran 2003 新機能

- **Accociate 構文** - その構文の実行中に、「名前」を変数または式の値との間に関連付けを作成します。Accociate 構文が実行されると、その associate-name に対応する値や式の値の関連付けが作成されます。関連付ける実体の変数、式の値によって、その宣言型と型パラメータ等が決まります。
- **Sourced Allocation** - 多様な型に対するアロケーション。(但し、無制限に多様な型のオペレーションを許容する訳ではない)。明示的に型や型パラメータを指定してアロケーションする代わりに、Allocate 文の中の source= というクローズを使用して、型や型パラメータ、他の変数あるいは、式からその値を取り込むことができます。アロケートされた変数は、同じ動的な型、パラメータ、ソース変数と同じ値を持つこととなります。

■ PGI Accelerator x64+GPU native Fortran 95/03 and C99 コンパイラと PGI CUDA Fortran

CUDA 3.1 Toolkit をサポートしました。この CUDA 3.1 Toolkit は、CUDA 3.0 のアップデート版であるため、CUDA 3.0 Toolkit は PGI コンパイラ製品にバンドルされません。しかし、以前の PGI インストールで実装されている CUDA 3.0 Toolkit のディレクトリは、そのまま残して置いても問題ありません。CUDA 3.1 API への変更により、CUDA Fortran ホスト・プログラムは、再コンパイルする必要があります。PGI 10.8 ランタイム・ライブラリは、以前の CUDA Fortran のリリースとは互換性がありません。

■ PGI Visual Fortran

PVF 10.8 では、「プロパティメニュー」で CUDA 3.1 toolkit を指定することができます。「Enable CUDA Fortran」を yes とセットすると、Fortran | Language | CUDA Fortran Toolkit プロパティにおいて、CUDA 3.1 toolkit 使用することを明示的に指定することができます。また、アクセラレータ用の設定においては、Fortran | Target Accelerators | NVIDIA : CUDA Toolkit プロパティで同じように設定できます。

[PGI 10.6 \(2010年6月\)リビジョンでの機能追加](#)

■ PGI 10.6 で追加された Fortran 2003 新機能

- I/O ENCODING, I/O DECIMAL 指定子, ASYNCHRONOUS 属性と文, CHARACTER MIN/MAX/MINLOC/MAXLOC/MINVAL/MAXVAL, ALLOCATE/DEALLOCATE ERRMSG 指定子, non-polymorphic 型に対する ALLOCATE SOURCE 指定子, and IEEE_features モジュールの追加. New object-oriented features include deferred type-bound procedures, TYPE PUBLIC and PRIVATE attributes, and TYPE ABSTRACT attributes.
- TYPE PUBLIC and PRIVATE 属性, and TYPE ABSTRACT 属性や deferred type-bound 手続きを含むオブジェクト指向機能の追加。

■ PGI Accelerator x64+GPU native Fortran 95/03 and C99 コンパイラと PGI CUDA Fortran

CUDA ビルトイン関数 `syncthread_count`, `syncthread_and`, `syncthread_all`, `threadfence`, `threadfence_block`, `threadfence_system` and `ballot` の追加。さらに、CUDA Fortran デバイスコードは、`popcnt()`, `poppar()`, `leadz()` 関数をサポートしました。
PGI アクセラレータ領域内の自動アンロール機能を実装しました。

■ PGPROF Profiler

PGI Accelerator モデルや PGI CUDA Fortran を含むコードのプロファイリングをサポートした。

■ PGI Visual Fortran

Microsoft (R) Visual Studio 2010 をサポートした。

[PGI 10.5 \(2010年5月\)リビジョンでの機能追加](#)

■ PGI 10.5 で追加された Fortran 2003 新機能

- Type-bound procedure 機能の追加
- PASS / NOPASS 属性追加：手続きに対して、どの引数が呼び出されるオブジェクトに渡されるかを指定する属性。例えば、NOPASS 属性は、オブジェクトを引数として渡すことを阻止します。

■ PGI Accelerator x64+GPU native Fortran 95/03 and C99 コンパイラと PGI CUDA Fortran

`acc_get_device_num` というランタイム・ライブラリをサポートしました。この関数は、アクセラレータ領域を実行するために使用されているデバイスの数を返します。

■ Accelerator Profiling

PGI Accelerator モデルや PGI CUDA Fortran を使用したプログラムのプロファイリングを行う際に、`pgcollect` ユーティリティは自動的にアクセラレータ性能に関する情報を収集して、プロファイル出力として表示できるようにします。

(注意) プログラムプロファイル出力の中のアクセラレータ性能情報の取り込みは、Linux においては、時間ベースのサンプリング手法でも、イベントベースのサンプリング手法のどちらでも可能です。

ある Linux システム上では、パワーセーブ状態になっているアクセラレータハードウェアの CUDA ドライバの初期化は、非常に多くの時間を費やします。この遅延を避けるために、以下のいずれかの方法で対処することができます。

- バックグラウンドで、`pgcudainit` プログラムを実行させる。GPU への通電の状態を保ち、後続のプログラムの初期化の時間を削減できる。この詳細は、PGI User's Guide の 7 章を参照下さい。
- `pgcollect` のオプションに `-accinit` を付ける。これは、初期化オーバーヘッドの多くを削減し、より精度の高いプロファイリングを提供できます。また、ユーザが定義した型メンバーや配列要素に

対する Fortran に特化した機能が向上しました。

(例) `pgcollect -time -accinit myaccelprog`

- PGI Visual Fortran は、Enhanced Variable Rollover や Watch and Quick Watch 機能を強化しました。

PGI 10.4 (2010年4月)リビジョンでの機能追加

- PGI Accelerator x64+GPU native Fortran 95/03 and C99 コンパイラと PGI CUDA Fortran は、CUDA 3.0 Toolkit and compute capability 2.0 をサポートしました。

CUDA compute capability 2.0 を指定するためには、次のオプションの一つを使用する。

PGI Accelerator には、`-ta=nvidia:cc20`

PGI CUDA Fortran には、`-Mcuda=cc20`

- コンパイラがターゲットとする CUDA toolkit のバージョンを指定するためには、次のオプションの一つを使用する。

PGI Accelerator には、

CUDA toolkit 3.0: `-ta=nvidia:cuda3.0` or `-ta=nvidia:3.0`

CUDA toolkit 2.3: `-ta=nvidia:cuda2.3` or `-ta=nvidia:2.3`

PGI CUDA Fortran には、

CUDA toolkit 3.0: `-Mcuda=cuda3.0` or `-Mcuda=3.0`

CUDA toolkit 2.3: `-Mcuda=cuda2.3` or `-Mcuda=2.3`

- CUDA Kernel のための `wait` オプションの新設: `-ta=nvidia:[no]wait` オプションが、NVIDIA アクセラレータをターゲットとしたときに有効となります。ホスト・プログラム上で、kernel の実行が終了するまで待つか、あるいは、待たずに実行するかを指定することができます。デフォルトは `wait` です。
- CUDA Fortran において、`fused multiply-add` 命令をサポートしました。ユーザは、PGI アクセラレータ機能ならびに CUDA Fortran の両方において、`fused multiply-add` 命令を使用したコード生成を行うかどうかの制御が可能となりました。アクセラレータ機能においては、既に `-ta=nvidia:nofma` オプションでサポートされておりますが、PGI 10.4 以降では、CUDA Fortran においても、`-Mcuda=nofma` オプションにより、この制御機能をサポートしました。
- CUDA Fortran において、`fast math library` が使用できるようになりました。ユーザは、PGI アクセラレータ機能ならびに CUDA Fortran の両方において `fast math library` を使用できます。アクセラレータ機能においては、既に `-ta=nvidia:fastmath` オプションでサポートされておりますが、PGI 10.4 以降では、CUDA Fortran においても、`-Mcuda=fastmath` オプションにより、この制御機能をサポートしました。
- グローバル・サブルーチンを含む Fortran Module 内の `allocatable device arrays` を使用した CUDA Fortran のサポートがされました。これによって、`module` を使用するホスト側コードから、また、`module` 内に含まれるデバイスコードの両方からこの主の配列へのアクセスが可能となりました。

(注意) CUDA 3.0 toolkit を使用してコンパイルした場合 (`-ta=nvidia:cuda3.0`)、あるいは、コンパイラの初期化ファイル `siterc` ファイルの中に `set CUDAVERSION=2.0` を設定した場合は、2.3 CUDA ドライバ上では

動作しませんのでご注意ください。CUDA ドライバも CUDA 3.0 用に変更する必要があります。現在、システムに実装されている CUDA ドライバを確認する方法は、pgacclinfo を実行して下さい。

```
$ pgacclinfo
```

```
CUDA Driver Version      2030
```

```
(以下省略)
```

ここで、

2.3 driver の場合: CUDA Driver Version 2030

3.0 driver の場合: CUDA Driver Version 3000 と表示されます

PGI 10.3 (2010年3月)リビジョンでの機能追加

■ CUDA Compute capability 1.0~1.3 ハンドリングの強化

- デフォルトの Compute capability を 1.0 と 1.3 の両方にするようにした。PGI 10.3 以降、ユーザはターゲットとされる Compute capability を複数のリビジョンの指定が可能となった。例えば、4 つの compute capability 1.0, 1.1, 1.2, 1.3 の全てをターゲットとするには、コマンド・オプションで次のように指定する。

```
-Mcuda=cc10, -Mcuda=cc11, -Mcuda=cc12, -Mcuda=cc13
```

あるいは、

```
-Mcuda=cc10,cc11,cc12,cc13
```

- -Mcuda=keepgpu フラグの追加

■ さらに、次の Fortran 2003 フィーチャを追加した。

- Abstract インタフェース
- IS_IOSTAT_END, IS_IOSTAT_EOR, and NEW_LINE の組込関数追加
- オブジェクト指向の機能の追加: classes, type extensions (non-polymorphic), polymorphic entities, typed allocation, inheritance association, EXTENDS_TYPE_OF や SAME_TYPE_AS 組込関数
- 新構文と修正構文: WAIT 文; READ 文のための blank, pad, and pos 指定子; WRITE 文のための delim and pos 指定子; INQUIRE 文のための pending と pos 指定子

PGI 10.0 (2010) メジャーリリースの機能 (PGI 10.0~10.2)

- **PGI アクセラレータ x64+GPU Fortran95/03 and C99 コンパイラサポート** — NVIDIA の GPGPU 搭載のシステム上で、コンパイラ・ディレクティブ (指示行) ベースでプログラミング並びに最適化可能な PGI Accelerator Programming Model V1.0 スタandard に完全準拠しました。
 - Linux、Windows、Mac OS X プラットフォームをサポート
 - MIRROR, REFLECTED, UPDATE ディレクティブを使用した GPU デバイス(メモリ)に常駐するデータのサポート
 - COMPLEX、DOUBLE COMPLEX データ、Fortran 派生タイプ、C structs のサポート
 - 自動 GPU サイドのループ・アンローリングと UNROLL ディレクティブ節のサポート
 - OpenMP 並列リージョン内におけるアクセラレータ・リージョンのサポート
- **PGI CUDA Fortran extensions 機能** — PGI 2010 の Fortran 95/03 コンパイラにて、明示的な CUDA GPU プログラミングをサポートしました。
 - 定数、あるいは共有メモリの CUDA GPU device 内における変数の宣言

- ページロックされる pinned ホストメモリや、CUDA デバイスメモリ、定数メモリ、共有メモリの動的なアロケーション、
 - Fortran 割り当て文による、ホスト側と GPU 側のデータ転送
 - GPU 計算カーネルを投入するための明示的な CUDA のグリッド/スレッド・ブロックの宣言
 - CUDA ランタイム API 関数と機能をサポート
 - 簡単に CUDA Fortran をデバッグするために、ホスト・サイドでのエミュレーション
- Fortran 2003 追加機能** — IMPORT, pointer reshaping, procedure pointers and statement, abstract interface, iso_c_binding intrinsic module, c_associated, c_f_pointer, c_associated, enum, move_alloc(), iso_fortran_env module, optional kind to intrinsics, allocatable scalars, volatile Getting Started 6 attribute and statement, pass and nopass attributes, bind (c), value, command_argument_count, get_command, get_command_argument, get_environment_variable, ieee_exceptions module, ieee_arithmetic module
- PGC++/PGCC (2010 C++)** — 新しい機能とその強化がなされました。
 - 最新の EDG リリース 4.1 を採用し、さらに、GNU 並びに Microsoft との互換性を高めました。
 - デフォルトで、extern インラインをサポート、実行モジュール内にインライン関数の複数のコピーは、この度削除されました
 - BOOST のサポートを強化し、さらに大きなコードをより良くサポートするために、内部のテーブルを拡張しました
 - C++ -mp オプションで、スレッド・セーフ例外のハンドリング
- OS のサポート強化** — RHEL 5, Fedora 11, SLES 11, SuSE 11.1, Ubuntu 9, Windows 7 と Mac OS X Snow Leopard のサポート
- コンパイラの最適化と機能強化**
 - OpenMP は、最大 256 スレッドまで拡張しました。(以前は、64)
 - AVX (Advanced Vector Extensions) コードの生成
 - 部分的な冗長部削除機能
 - 実行モジュールサイズの最適化、向上
- PGI Visual Fortran 強化点**
 - Visual Studio 内から Windows クラスタ上の MSMPI プログラムのデバッグとジョブ投入のサポート
 - PGI Accelerator Programming model のフル・サポート
 - CUDA Fortran のサポート
 - CCFF のサポートを備えた PGPROF 性能プロファイラーをスタンドアロンで提供
- ドキュメンテーションの更新** — the PGI Users Guide、PGI Tools、Guide、PGI Fortran Reference のドキュメントを更新しました。

2.2 PGI 2010 の新コンパイラ・オプションと従来との変更部分

未知のコンパイラ・オプションを指定した場合、今までは「警告」レベルの扱いでしたが、PGI 7.1 より、「エラー」として扱われます。この変更によって、コンパイラが認識できないオプション（スイッチ）が指定された場合、コンパイルエラーとなり停止することになります。(以前のバージョン PGI 7.1 から適用されたデフォルトですのでご注意ください)

一例 : pgf95-Error-Unknown switch: -Mconcu

しかし、`-noswitcherror` を指定すると、従来のように、未知のオプション（スイッチ）が指定された場合、エラーで停止する代わりに「警告」を出すように変更できます。

以下のオプションは、PGI 2010 から追加あるいは、修正されたものです。

- **-m32** — デフォルトのプロセッサタイプとして、32 ビットコンパイラを使用することをコンパイラに指示する。(PGI 10.3 新設)
- **-m64** — デフォルトのプロセッサタイプとして、64 ビットコンパイラを使用することをコンパイラに指示する。(PGI 10.3 新設)
- **-ta=vidia (,vidia_suboptions) , host** — PGI アクセラレータ・コンパイラに伴う新しいオプションです。**-ta** は、ターゲット・アーキテクチャを意味します。Fortran における **!\$ACC** ディレクティブ、C における **#pragma acc** ディレクティブをコンパイラに認識させるために、このオプションを使用します。これは、Fortran 並びに C コンパイラのみで有効です。このオプションは、以下のサブオプションを有します。
 - PGI 2010 で `vidia_suboptions` に以下の機能が強化するために設けられた。

C10	compute capability 1.0 のコード生成。
C11	compute capability 1.1 のコード生成。
C12	compute capability 1.2 のコード生成。
C13	compute capability 1.3 のコード生成。
C20	compute capability 2.0 のコード生成。(PGI 10.4 以降)
cuda2.3 or 2.3 CUDA toolkit 2.3	バージョンを使用 (PGI 10.4 以降)
cuda3.0 or 3.0 CUDA toolkit 3.0	バージョンを使用 (PGI 10.4 以降)
fastmath	fast math ライブラリを使用。
keepgpu	kernel ソースファイルを保持する。
keepptx	GPU コードのための portable assembly(.ptx) ファイルを保持する
maxregcount:n	GPU 上で使用するレジスタの最大数を指定。ブランクの場合は、制約が無いと解釈する。
mul24	添字計算のために 24 ビット乗算を使用。
- **-Mautoinline** — 新しいサブオプションが追加された。

Levels:n	インラインを行うレベル階層を最大 n まで行うことを指示。デフォルトは 10 です。
Maxsize:n	n サイズを超える関数のインラインを行わない。デフォルトは 100。
totalsize:n	インライン h 対象が、n サイズ時にインラインを止めることを指示。デフォルトは 800。
- **-Mpre** と **-Mnpre** — 新オプション。部分的な冗長性削除の最適化を行う/抑止する。
- **-Meh_frame** と **-Mnoeh_frame** — 新オプション。リンカーに、executable 内の `eh_frame` の call frame を保持/非保持することを指示する。 **(注意)** このオプションは、システム `unwind` ライブラリを持つ、最新の Linux、Windows システムでのみ有効です。
- **--gnu_version <num>** — 新オプション。コンパイル時に使用する GNU C++ 互換性をセットする。デフォルトは、最新のバージョン番号がセットされる。使用例： `--gnu_version 4.3.4`。 [C ++ コンパイラのみ]
- **--microsoft_version <num>** — 新しいオプション。コンパイル時に使用する Microsoft C++ 互換

性をセットする。デフォルトは、最新のバージョン番号がセットされる。使用例: `--microsoft_version 1.5`。[C++コンパイラのみ]

- **-Mcuda** — 新しいオプション。-Mcuda オプションは、コンパイラに CUDA Fortran を使用できるように指示する。以下のサブオプションを有する。

C10	compute capability 1.0 のコード生成。
C11	compute capability 1.1 のコード生成。
C12	compute capability 1.2 のコード生成。
C13	compute capability 1.3 のコード生成。
C20	compute capability 2.0 のコード生成。(PGI 10.4 以降)
cuda2.3 or 2.3	CUDA toolkit 2.3 バージョンを使用 (PGI 10.4 以降)
cuda3.0 or 3.0	CUDA toolkit 3.0 バージョンを使用 (PGI 10.4 以降)
fastmath	fast math library を使用する (PGI 10.4 以降)
nofma	fused multiply-add 命令を使用しない (PGI 10.4 以降)
emu	CUDA Fortran エミュレーションモード
keepptx	GPU コードのための portable assembly(.ptx) ファイルを保持する
keepgpu	kernel ソースファイルを保持し、ファイル(.gpu)として出力する(PGI 10.3)
keepbin	kernel バイナリファイルを保持し、ファイル(.bin)として出力する
maxregcount:n	GPU 上で使用するレジスタの最大数を指定。ブランクの場合は、制約が無いと解釈する。

(更新情報: PGI 10.3 以降) GPU デバイスの Compute capability の指定オプション -Mcuda=cc?? を複数指定することができます。PGI 10.2 以前のデフォルトの GPU デバイス Compute capability は、1.3(cc13) でした。PGI 10.3 以降、デフォルトのターゲットは、1.0(cc10) 並びに 1.3(cc13) となりました。さらに、複数の GPU compute capability のターゲットとするようなコード生成したい場合は、-Mcuda=cc10 -Mcuda=cc11 -Mcuda=cc12 -Mcuda=cc13 という風に、コマンドラインで複数指定することができます。

2.3 C++コンパイル 留意点

このリリースにおいて、C++の以前のバージョンとのオブジェクトの互換性が無くなりました。全ての C++ソースファイルとライブラリは、10.x オブジェクト・ファイルとリンクする際、再コンパイルが必要です。

2.4 Fortran の強化

このセクションは、C 言語との相互運用性、文、割り当て、組込関数、モジュール、配列アロケーション、I/O 処理に係わる Fortran の強化ポイントについて述べたものです。

2.4.1 Fortran と C の相互運用性の強化

Fortran 2003 は、C 言語との相互運用性のメカニズムを提供しています。この相互運用に係わる任意の entity は、C と Fortran の両方で等価な宣言を行わなければなりません。PGI は、これらのコンポーネントを追加することにより、C との相互運用性を拡張します。

- Enumerators (ENUM 構文)
整数定数のセットです。種別型 kind は、C 言語の同じ定数で選択した整数型に対応します。
- C_f_pointer — C ポインタ・ターゲット cptr を Fortran ポインタ fptr へ割り当てるサブルーチンです。また、そのオプションとして、配列の形状 shape を指定することができます。構文は次の通りです。

c_f_pointer (cptr, fptr [,shape])

これらコンポーネントの詳細は、PGI Fortran Reference の章である「C 言語との相互運用」をご参照下さい。

2.4.2 新しい Fortran 組込関数 (intrinsic)

固有の関数 (intrinsic) は、指定された言語上のコンパイラによって実装された (組込まれた) 関数です。従って、コンパイラは、親和性の高い固有関数の知識を有するため、より質の高い統合や最適化を行うことが可能となります。このリリースでは、PGI は以下の intrinsic を強化しました。詳細は、PGI Fortran Reference Guide の Intrinsic chapter をご覧下さい。

GET_COMMAND_ARGUMENT(NUMBER [, VALUE, LENGTH, STATUS])

プログラムを起動する際のコマンドのコマンドラインの引数を返す。

GET_COMMAND([COMMAND, LENGTH, STATUS])

プログラムを起動する際のコマンドライン全体を返す。

GET_ENVIRONMENT_VARIABLE (NAME [,VALUE, LENGTH, STATUS, TRIM_NAME])

指定されている環境変数の値を返す。

2.4.4 新しい組込みモジュール

PGI 2010 は、IEEE 算術演算および IEEE 例外を処理できるようにするために、Fortran 組込みモジュール `ieee_arithmetic` と `ieee_exceptions` をサポートしました。

IEEE_ARITHMETIC

`ieee_arithmetic` 組込モジュールは、二つの派生タイプへのアクセスを提供する他に、名前付きコンスタント、総称名による手続きの集まりを提供します。

このモジュールは、`ieee_exceptions` モジュールのための `USE` ステートメントを含む化のように振る舞います。従って、`ieee_exceptions` の全ての機能が含まれております。

Note

詳細に関しては、PGI Fortran Reference Guide をご覧下さい。

Defined Elemental Operators(要素間のオペレーション)

- ==
派生タイプのものに対する二つの値間の処理：同じ値であれば、`true` を返し、他は `false` を返す
- /=
派生タイプのものに対する二つの値間の処理：異なる値であれば、`true` を返し、他は `false` を返す

派生タイプ

- `ieee_class_type` – 浮動小数点の値のクラスを定義
- `ieee_round_type` – 特定の丸めモードを定義する

次の表は、各クラスの型が取ることができる値の一覧を示します。

This derived type...	Takes these values...
<code>ieee_class_type</code>	<code>ieee_signaling_nan</code> <code>ieee_quiet_nan</code> <code>ieee_negative_inf</code> <code>ieee_negative_normal</code>

	ieee_negative_denormal ieee_negative_zero ieee_postive_zero ieee_postive_denormal ieee_postive_normal ieee_postive_inf ieee_other_value (Fortran 2003 only)
ieee_round_type	ieee_nearest ieee_to_zero ieee_up ieee_down

Inquiry Functions

- ieee_support_datatype([x])
- ieee_support_denormal([x])
- ieee_support_divide([x])
- ieee_support_inf([x])
- ieee_support_nan([x])
- ieee_support_rounding (round_value[,x])
- ieee_support_sqrt([x])
- ieee_support_standard ([x])
- ieee_support_underflow_control ([x]) Fortran 2003 only

Elemental Functions

- ieee_class(x)
- ieee_copy_sign(x,y)
- ieee_is_finite(x)
- ieee_is_nan(x)
- ieee_is_negative(x)
- ieee_is_normal(x)
- ieee_is_logb(x)
- ieee_next_after(x,y)
- ieee_rem(x,y)
- ieee_rint(x,y)
- ieee_scaln(x,i)
- ieee_unordered(x,y)
- ieee_value(x,class)
- ieee_support_datatype

Non-Elemental Subroutines

- ieee_get_rounding_mode(round_value)
- ieee_get_underflow_mode(gradual)
- ieee_set_rounding_mode(round_value)
- ieee_gst_underflow_mode(gradual)

Transformational Function

- ieee_selected_real_kind([p] [,r])

詳細は、PGI Fortran Reference Guide の Ininsics Modules をご覧下さい。

IEEE_EXCEPTIONS

組込モジュールは、二つの派生タイプへのアクセスを提供する他に、名前付きコンスタント、総称名による手続きの集まりを提供します。

Note

詳細に関しては、PGI Fortran Reference Guide をご覧下さい。

Derived Types(派生タイプ)

- `ieee_flag_type` - Identifies a particular exception flag.
- `ieee_status_type` - Saves the current floating-point status.

Inquiry Functions(問い合わせ関数)

- `ieee_support_flag(flag [,x])`
- `ieee_support_halting(flag)`

Subroutines for Flags and Halting Modes (ホールドモードとフラグ)

- `ieee_get_flag(flag, flag_value)`
- `ieee_get_halting_mode(flag, halting)`
- `ieee_set_flag(flag, flag_value)`
- `ieee_set_halting_mode(flag, halting)`

Subroutines for Floating-Point Status(浮動小数点状況のサブルーチン)

- `ieee_get_status(status_value)`
- `ieee_set_status(status_value)`

2.4.5 その他の Fortran 強化

- Fortran 2003 Asynchronous Input/Output : Fortran 2003 非同期 I/O は、一部、PGF77 と PGF95 コンパイラに実装されております。
 - OPEN 文で、`ASYNCHRONOUS='YES'` で指定された外部ファイルは、非同期 I/O が使用できます。
 - 非同期 I/O 処理は、`READ/WRITE` 文で `ASYNCHRONOUS='YES'`を指定することで可能です。
 - コンパイラは、`ASYNCHRONOUS` 属性あるいは、`ASYNCHRONOUS` 文を実装していません。

2.5 新しい Runtime (ランタイム) ライブラリ・ルーチン

PGI 2010 は、PGI アクセラレータ・コンパイラのための新たな run-time ライブラリを導入しました。詳細は、後述します。

2.6 PGI Tools の変更点 (新機能、修正機能)

このセクションで述べる、新しい機能については、PGI Tools Guide に詳細に述べられています。

2.6.1 PGPROF

PGPROF は、グラフィカルな MPI/OpenMP/マルチスレッド用性能解析、チューニング・プロファイラです。このリリースにおいての強化された機能は、以下の通りです。

- `pgcollect` ユーティリティを使用した新しいデータ集約メカニズムを採用することにより、再コンパイルなしでのプロファイリングや、`Oprofile` 用の特別のソフトウェアの導入無しでのプロファイリングが可能となっております。また、PGI ソフトウェアだけを使用した `time-base` のサンプリングを行った場合、`pgcollect` を単独使用モードで使用できます。これは、Linux と Mac OS X 10.5(Leopard) で可能です。
- Linux 上では、共有オブジェクト・ファイルの中のコードのプロファイルをサポートします。Mac OS X

上では、まだ、ダイナミック・ライブラリはサポートされていません。

- 複数のソースファイルへ「タブ」にてアクセスするための GUI を更新しました。アセンブリコードへのドリル・ダウン（下位＝アセンブラに移動する）が向上しました。
- PGI コンパイラ以外によってコンパイルされたバイナリのプロファイルをサポートしました。
- Linux、Windows、Mac OS 上で、再コンパイルなし、あるいは、特別なソフトウェア権限なしで、クロスプラットフォーム対応の性能プロファイリングが可能です。
- PGI Accelerator と CUDA Fortran の GPU サイドの性能統計の提供
- GUI の更新

2.8 ライブラリ・インタフェース

PGI は、Fortran モジュールを使用することによって C インタフェースをエクスポートするライブラリへのアクセスを提供します。これらのライブラリと関数は、PGI User's Guide の 8 章の述べられております。

3 PGI Workstation 2010

この章では、Linux、Windows、and Mac OS X の各 OS に関わる PGI Workstation 2010 のアップデート並びに変更について述べています。

3.1 PGI Workstation 2010 for Linux

3.1.1 Java Runtime Environment (JRE)

Linux 上での PGI のインストールでは、32 ビットバージョンの JRE を含みますが、必要十分な 32 ビット X Window をサポートするためには、適正な JRE と PGI ソフトウェア等が有効である必要があります。

いくつかのシステムでは、たとえば、最近の Fedora Core では、これらの関数の一部が標準のインストールにおいて行われなくなりました。必要とされる X Window では、一般的には、以下のライブラリを含みます。

libXau	libXdmcp	libxcb
libX11	libXext	

3.2 PGI Workstation 2010 for Windows

PGI Workstation 2010 for Windows は、linux86(32 ビット) 並びに linux86-64(64 ビット) 環境と同様なほとんどの機能をサポートします。

PGI Workstation 2010 for Windows のインストールにおいて、インストール時に www.pgroup.com から自動ライセンスファイル生成手続きができます。

3.3 PGI Workstation 2010 for Mac OS X

PGI Workstation 2010 for Mac OS X は、linux86(32 ビット) 並びに linux86-64(64 ビット) 環境と同様なほとんどの機能をサポートします。このリリースノートあるいはユーザマニュアルで指摘していること以外は、基本的に Linux 版の機能と同一です。

4. PGI アクセラレータ

「アクセラレータ」とは、特別の目的で CPU にアタッチして使用する協調プロセッサであり、時間の掛かる計算部分を CPU の演算機構からデータと実行部分のカーネルをオフロードするために使用されます。この章では、新しい PGI アクセラレータ・コンパイラについて説明します。この内容には、ホスト CPU からアタッチされているアクセラレータへオフロード可能な Fortran、C におけるコードの領域(region)を指定する

ために使われるディレクティブに関する事項も含まれます。

PGI アクセラレータ、プログラミングモデル、ディレクティブに関する詳細な情報は、PGI User's Guide の 7 章「Using an Accelerator」と 18 章「PGI Accelerator Compilers Reference」をご参照下さい。

4.1 コンポーネント

PGI アクセラレータ・コンパイラのテクノロジーは、次のコンポーネントを含みます。

- PGF95 自動並列化アクセラレータ付き Fortran 90/95 コンパイラ
- PGCC 自動並列化アクセラレータ付き ANSI C99 and K&R C コンパイラ
- NVIDIA CUDA ツールキット・コンポーネント
- システムが適切な GPU あるいは、アクセラレータカードを有するかどうかを検出するための単純なコマンドライン・ツール

アクセラレータ対応のデバッガあるいはプロファイラーは、このリリースでは含まれておりません。

4.2 可用性

PGI 2010 (PGI 10.0) Fortran & C アクセラレータ・コンパイラは、x86 プロセッサベースのシステム上のワークステーションやサーバでのみ使用できます。もちろん、このシステム上には NVIDIA CUDA-enable GPU あるいは、TESLA カードが実装されている必要があります。これらのコンパイラは PGI がサポートする、Mac OS X 64bit を除く全てのプラットフォームで利用できます。

4.3 ユーザ指示のアクセラレータ・プログラミング

ユーザが指示するタイプのアクセラレータ・プログラミングにおいては、ユーザはアクセラレータ・デバイスへオフロードするために、その対象とすべきホスト・プログラムの領域を指定します。また、ターゲットとなるアクセラレータでサポートされていないコンストラクトを含む領域やユーザ・プログラムの大半は、ホスト上で実行されます。ここでの説明は、アクセラレータにオフロードされるコードの領域やループの指定方法に関してのみ、説明しています。

4.4 カバーしていない、あるいは実装していない機能

現在、PGI アクセラレータ・コンパイラは、コンパイラや他のツール等による、アクセラレータにオフロード化できるコード領域の自動検出やそのオフロード化のための機能は含んでおりません。PGI コンパイラの将来のバージョンでは、自動的なオフロード化や同一ホスト上のマルチ・アクセラレータの使用、あるいは、異なるタイプのマルチ・アクセラレータの使用もできるように計画していますが、現在のバージョンでは、これらはサポートされておりません。

4.5 必要とされるシステム仕様

PGI アクセラレータの機能を使用するためには、各システム上で NVIDIA ドライバーコンポーネントをインストールしておく必要があります。

- NVIDIA Driver
- CUDA Toolkit
- CUDA SDK

これらは、NVIDIA の Web サイト(www.nvidia.com/cuda/) からダウンロード可能です。これらは、PGI コンポーネントではありませんので、NVIDIA 社によってライセンス並びにサポートされます。

さらに、最新の PGI リリースと CUDA ソフトウェア、ドライバーの両方をサポートするシステムが必要とさ

れます。

4.5.1 サポートするプロセッサと GPU

PGI アクセラレータ・コンパイラのリリースは、PGI コンパイラ&ツールの 2010 リリース以降がサポートする全ての AMD64 並びに Intel 64 のホストプロセッサ上で利用できます。また、コンパイラのオプション・フラグである `-tp <target>` フラグを使用することができます。

NVIDIA GPU を対象としたアクセラレータのディレクティブの認識を有効にするための `-ta=nvidia` フラグが使用できます。そして、CUDA-enable な GeForce、Quadro、Tesla カードを有した CUDA がインストールされた任意のシステム上で、生成されたコードを使用することができます。

アクセラレータのテクノロジーに関するこれらのフラグの詳細な情報は、PGI User's Guide をご参照下さい。また、サポートされる GPU の詳細なリストは、NVIDIA のウェブサイトでご覧下さい。
www.nvidia.com/object/cuda_learn_products.html

4.6 インストールとライセンス

PGI アクセラレータ・コンパイラは、標準的な PGI Workstation、Server、CDK のライセンスキーの他に、別のライセンスキーが必要となります。

PGI アクセラレータ・コンパイラは、別のコンパイラ・エディション製品として発売しますので、アクセラレータ機能を使用したいお客様は、新規のご購入、あるいは、現製品のアップグレードを行っていただく必要があります。

4.7 アクセラレータ・プログラムの実行

アクセラレータのディレクティブを有し、`-ta=nvidia` フラグを使ってコンパイル・リンクしたプログラムを実行させる方法は、`-ta=nvidia` フラグを付けないでコンパイル・リンクした場合と同じである。プログラムは CUDA ライブラリを探し、動的にロードします。もし、ライブラリがない場合、あるいは、プログラムをコンパイルした際に存在した場所とは異なるディレクトリ上にある場合は、`LD_LIBRARY_PATH` 環境変数に CUDA ライブラリの所在を追加する必要があります。

プログラムが最初のアクセラレータ領域に到達した際、約 0.5~1.5 秒ほどのポーズ時間が存在します。これは、GPU への操作権を得て、静的なリソースを割り付けるための時間オーバーヘッドです。すなわち、GPU がパワーオフの状態からウォームアップするまでの間のオーバーヘッドです。この遅延を避けたい場合は、GPU をオンにするために `pgcudainit` プログラム (関数) をバックグラウンドで実行しておく必要があります。

アクセラレータ・プログラムを CUDA-enable の NVIDIA GPU を有しないシステムで動作させた場合、あるいは、ランタイム・ライブラリの検索範囲のディレクトリの場所に CUDA ライブラリが存在しない場合、プログラムは、その実行時にエラーメッセージ無しに終了してしまいます。

環境変数 `ACC_NOTIFY` にゼロ以外の整数をセットした場合、ランタイム・ライブラリは GPU のカーネル (kernel) が開始される度に、その開始したことを標準エラーにプリントします。

4.7.1 PGI アクセラレータ・コンパイラのランタイム・ライブラリ

PGI アクセラレータ・コンパイラは、アクセラレータの機能に関する問い合わせや実行時にアクセラレータ用のプログラムの挙動を制御するような、ユーザによる呼び出し可能な関数やライブラリ・ルーチンを提供します。Fortran において、PGI アクセラレータ・コンパイラのランタイム・ルーチンは、PURE あるいは ELEMENTAL 手続きからコールされません。

アクセラレータ・ライブラリにアクセスするためには、コンパイル時と同じ`-ta` フラグ (コンパイル・オプション) をリンクする必要があります。

C と Fortran では、各々、別のランタイム・ライブラリが存在します。

- C ランタイム・ライブラリ・ファイル — C では、ランタイム・ライブラリ・ルーチンのプロトタイプが `"accel.h"` という名前のヘッダーファイルで用意されています。全てのランタイム・ライブラリ・ルーチンは、`"C"` リンケージを有する外部関数です。このファイルは、以下を定義します。
 1. このセクションの述べられている全てのルーチンのプロトタイプ
 2. アクセラレータの型を記述する `enumulation` 型を含むこれらのプロトタイプで使用されている任意のデータ型
- Fortran ランタイム・ライブラリ・ファイル — Fortran では、`interface` の宣言は、`accel_lib.h` という名前の Fortran インクルード・ファイルの中、あるいは、`accel_lib` という Fortran モジュールの中で提供されます。これらのファイルでは、以下を定義します。
 1. このセクションの述べられている全てのルーチンのインタフェース
 2. これらのルーチンへの引数のための整数 `kind` 値を定義するための整数パラメータ
 3. アクセラレータの型を記述するための整数パラメータ。
`yyyymm` という値を持った整数パラメータ `accel_version` は、サポートしているアクセラレータ・プログラミングモデルのバージョンを示すもので、`yyyy` は年、`mm` は月を意味します。この値は、プリプロセッサ変数 `_ACCEL` の値に合致します。

次のリストは、PGI が現時点でサポートしている PGI アクセラレータ・コンパイラのランタイム・ライブラリを簡単に述べたものです。これらのルーチンの完全な説明は、PGI User's Guide の PGI Accelerator Runtime Routines の章をご覧ください。

- `acc_get_device` : 使用中のアクセラレータ・デバイスのタイプを返す。
- `acc_get_num_devices` : ホスト側にアタッチしているアクセラレータ・デバイスの数を返す。
- `acc_init` : アクセラレータ・デバイスに接続し、初期化を行い、アクセラレータ・ライブラリの中の制御構造を割り付ける。
- `acc_on_device` : 特定のデバイス上で実行しているかどうかをプログラムに伝える。
- `acc_set_device` : アクセラレータ計算領域を実行するとき使用するデバイスのタイプをランタイムへ伝える。
- `acc_set_device_num` : アタッチされているデバイス間で使用するタイプのデバイスをランタイムに伝える。
- `acc_shutdown` : アクセラレータ・デバイスへの接続を切るようにランタイムに伝える。そして、任意のランタイムリソースを自由にする。

4.7.2 環境変数

PGI は、アクセラレータ領域の挙動を修正するための環境変数を提供します。このセクションでは、実行時のアクセラレータを使用するプログラムの挙動を制御するために使用する、ユーザが指定可能な環境変数について説明します。これらの環境変数は、以下のルールに準拠する必要があります。

- 環境変数の名前は、大文字でなければなりません。
- 環境変数にセットする値は、大文字・小文字の区別はありません。また、最初と最後の余白指定も可能です。
- プログラムが開始した後で、環境変数の値が変更された場合、その挙動は実装依存です。また、例えばプログラム自身が値を変えたとしてもこれは同様です。

次のリストは、PGI がサポートしている PGI アクセラレータ環境変数を簡単に述べたものです。これらのルーチンの完全な説明は、PGI User's Guide の PGI Accelerator Runtime Routines の章をご覧ください。

- ACC_DEVICE : アクセラレータのための PGI Unified Binary を実行する時に使用するアクセラレータのデフォルト・デバイスを制御する。この環境変数は "NVIDIA" あるいは、"HOST" のどちらかとなる。
- ACC_DEVICE_NUM : アクセラレータ領域を実行する際に使用するデフォルトのデバイス番号を制御する。この値は、負ではない整数で 0 からホストにアタッチしているデバイスの数までの値となる。
- ACC_NOTIFY: 負ではない整数値をセットしたとき、デバイス上のカーネル(kernel) が開始される度に、その開始したことを標準エラーにプリントします。

4.7.3 コンパイラ・コマンド・オプション

アクセラレータを動作させるために特に使用するコマンド・オプションは以下の通りです。

- -tp : 対象となるホストプロセッサのアーキテクチャを指定するオプション
- -Minfo or -Minfo=accel : このオプションを指定すると、コンパイラがアクセラレータ領域を GPU カーネルに翻訳できたかどうかについて、コンパイラのメッセージとして出力します。
- -ta=nvidia(,nvidia_suboptions), host :
PGI アクセラレータ・コンパイラに伴う新しいオプションです。-ta は、ターゲット・アーキテクチャを意味します。Fortran における !\$ACC ディレクティブ、C における #pragma acc ディレクティブをコンパイラに認識させるために、このオプションを使用します。これは、Fortran 並びに C コンパイラのみで有効です。このオプションは、以下のサブオプションを有します。

- nvidia – NVIDIA アクセラレータをターゲットとして選択します。さらに、以下の nvidia 用のサブオプションがあります。

analysis	ループの解析のみ行い、コードの生成を行いません。
cc10	compute capability 1.0 のコードを生成
cc11	compute capability 1.1 のコードを生成
cc12	compute capability 1.2 のコードを生成
cc13	compute capability 1.3 のコードを生成
fastmath	fast math ライブラリを使用。
keepgpu	kernel ソースファイルを保持する。
keepptx	GPU コードのための portable assembly(.ptx) ファイルを保持する
maxregcount:n	GPU 上で使用するレジスタの最大数を指定。ブランクの場合は、制約が無いと解釈する。
mul24	添字計算のために 24 ビット乗算を使用。
nofma	fused-multiply-add 命令を生成しない
time	アクセラレータ領域の単純な時間情報を集積するために プロファイル・ライブラリをリンクする

- host – ターゲットとして、host を選択する。nvidia オプションとの組み合わせで使用されます。アクセラレータ領域をホスト側で実行するようにコンパイルする。このオプションは、PGI Unified Binary コードを生成します。

コンパイラは、自動的に必要とする CUDA ソフトウェアのツールを発動し、GPU カーネルコードを生成し、オブジェクト・ファイルの中にカーネルを埋め込みます。

コマンドライン上でリンク時に、アクセラレータのライブラリにアクセスするためには、必ず、コンパイル時に指定したのと同じ、-ta フラグ・オプションを指定することが必要です。

4.8 アクセラレータ用の PGI Unified Binary

PGI コンパイラは、異なるホストプロセッサ用に最適化がなされた関数を備えた、単一の executable 形態として実行モジュールを生成するための機能である PGI Unified Binary をサポートします。このリリースは、PGI Unified Binary をアクセラレータ用まで拡張します。特に、以下のような関数の二つのバージョンを含む単一バイナリを生成できます。

- 一つは、アクセラレータ用に最適化したバージョン
- 他は、アクセラレータが有効ではないとき、あるいは、アクセラレータ上での実行とホスト上での実行を比較したいときに、ホストプロセッサ上で実行するバージョン

この機能を有効にするには、拡張された **-ta** フラグ・オプション **-ta=nvidia,host** を使用します。このフラグは、コンパイラに対して、有効なアクセラレータ領域を有するプログラムに対して、以下のような二つのバージョンの関数を生成するように指示する。

- アクセラレータを対象としたコンパイル・バージョン
- アクセラレータ・ディレクティブを無視し、ホストプロセッサを対象としたコンパイル・バージョン

-Minfo フラグをコンパイル時に指定したときは、以下のコンパイル・メッセージと同じようなメッセージを得るでしょう。

```
s1: (host 上の executable バージョン)
    12, PGI Unified Binary version for -tp=barcelona-64 -ta=host
    18, Generated an alternate loop for the inner loop
        Generated vector sse code for inner loop
        Generated 1 prefetch instructions for this loop
s1: (nvidia 上の executable バージョン)
    12, PGI Unified Binary version for -tp=barcelona-64 -ta=nvidia
    15, Generating copy(b(:,2:90))
        Generating copyin(a(:,2:90))
    16, Loop is parallelizable
    18, Loop is parallelizable
        Parallelization requires privatization of array t(2:90)
    Accelerator kernel generated
    16, !$acc do parallel
    18, !$acc do parallel, vector(256)
        Using register for t
```

上記では、PGI Unified Binary のメッセージが、サブルーチン s1 に関して、二つのバージョンを生成したことを示しています。

- 一つは、アクセラレータのないバージョン (-ta=host)
- もう一方は、NVIDIA GPU のためのバージョン (-ta=nvidia)

実行時に、プログラムは NVIDIA CUDA 動的ライブラリをロードしようとします。そして、GPU の存在を確かめます。もし、ライブラリが有効でないか、あるいは、GPU が見つからない場合、プログラムはホストバージョンで実行されます。

また、NVIDIA GPU 上で実行するようにプログラムに指示するために、環境変数をセットすることができます。これを行うためには、ACC_DEVICE を NVIDIA あるいは nvidia セットします。一方、この環境変数にこれ以外の任意の値をセットすると、ホストバージョンを使うようになります。

なお、今回のリリースにおける **-ta** ターゲットは、“nvidia” と “host” の二つに限られます。

4.9 複数のプロセッサターゲット

-ta フラグと共に、複数のプロセッサターゲットを指定する形で、-tp フラグ・オプションも使用できます。この場合、次に述べるような挙動となります。

- 一つの -tp ターゲット値の指定と、一つの -ta 値を指定した場合
その指定された「プロセッサターゲット」と「アクセラレータターゲット」向けに生成された各サブプログラムを有する「一つの実行モジュール・バージョン」が生成される。
- 一つの -tp ターゲット値の指定と、複数の -ta ターゲット値を指定した場合
コンパイラは、その指定された「プロセッサターゲット」と各「アクセラレータターゲット」向けのアクセラレータ領域を含むサブプログラムを有する「二つの実行モジュール・バージョン」が生成される。
- 複数の -tp ターゲット値の指定と、一つの -ta の指定をした場合
二つあるいはそれ以上の「プロセッサターゲット」向けのそれぞれのサブプログラムのバージョンを一つの実行モジュール内に生成する。また、その各バージョンは、選択したアクセラレータ用のバイナリコードを含む。
- 複数の -tp ターゲット値の指定と、複数の -ta ターゲット値を指定した場合
例えば、N 個の -tp の値と二つの -ta ターゲット値を指定すると、コンパイラは、N+1 個のサブプログラムのバージョンを生成する。それは、始めに各 -tp ターゲット用に N 個のバージョンを -ta=host の指定と等価なアクセラレータ領域を無視したバイナリで生成します。そしてさらに、アクセラレータ向けの一つのバージョンを生成します。

4.10 複数のアクセラレータ

並列の MPI あるいは、OpenMP プログラムを書くことにより、複数枚の NVIDIA GPU を使用することができます。

MPI

同じノード上で並列に各 MPI ランクが実行出来る MPI プログラムにおいては、acc_device_num 手続きを使用して異なる GPU を選択するために MPI ランクの値を使用できます。

OpenMP

OpenMP スレッド並列プログラムにおいては、omp_num_thread_num 関数を使用して、各スレッド用に異なる GPU を選択することができます。

OpenMP 並列領域において、各スレッドがイテレーションの異なるサブセットを計算するようなループを持つアクセラレータ・リージョンを含むことができます。

4.11 アクセラレータ・カーネルのプロファイリング

本リリースでは、以下のコマンドライン・オプションをサポートします。

-ta=nvidia,time

Time サブオプションは、タイマーライブラリをリンクします。このサブオプションは、アクセラレータ領域と生成されたカーネルに関するタイミング情報を集積し、印字します。

Sample Accelerator Kernel Timing Data

Accelerator Kernel Timing data

bb04.f90

s1

```
15: region entered 1 times
    time(us): total=1490738
           init=1489138 region=1600
           kernels=155 data=1445
    w/o init: total=1600 max=1600
           min=1600 avg=1600
18: kernel launched 1 times
    time(us): total=155 max=155 min=155 avg=155
```

上記の例において、いくつかの事象を見て取れます。

- 各アクセラレータ領域において、ファイル名 **bb04.f90** とサブルーチンあるいは関数名 **s1** とそのアクセラレータ領域の行番号が印字されています。上記例では、15 行目という風に。
- ライブラリは何回、領域に入ったかをカウントしています（上記例では、1）。そして、その領域内で消費した時間をマイクロ秒単位で表示します（上記例では、1490738）。この内訳は、初期化に係わる時間（例では、1489138）と実行時間（例では、1600）の二つに分けて表示します。
- 実行時間は、「カーネル実行時間」と「ホストと GPU 間のデータ転送時間」に分離して表示されます。
- 各カーネルは、行番号が表示されます（上記例では 18）。カーネルの開始数に沿って、カーネルで消費した total、max、min、平均の各時間が表示されます。この例では、1 回のカーネル実行ですので、全て 155 です。

4.12 サポートする組込関数

PGI アクセラレータ・コンパイラは、Fortran と C の組込関数、サブプログラムをサポートします。PGI 組込関数に関する詳細な情報は、PGI User's Guide における 7 章「Using an Accelerator」の「Supported Intrinsics」をご参照下さい。さらに、組込関数については、今後のリリースで追加される予定です。

5. 配布と配置

この章では、コンパイラを使用する上で、関連するトピックスについて述べます。特に、PGI Unified Binary 技術を通しての最適化や、Windows 上のリンクオプションについて、さらに、siterc ファイルあるいは rc ファイルによるコンパイラのカスタマイズについて説明します。

- PGI Unified Binaries の生成、そのコマンドラインのスイッチ、ディレクティブ、プラグマに関しては、PGI User's Guide の 9 章の Distributing Files - Deployment にて詳しく説明しております。
- スタティックリンクやダイナミック・リンクを選択する際のコンパイラ・オプションの使用例やスタティック、ダイナミック・リンク・ライブラリの生成、使用法等に関する詳細は、PGI User's Guide の 8 章の Creating and Using Libraries に説明しております。
- 特別の用途で、コンパイラの初期設定ファイルである、siterc ファイルやユーザ rc ファイルをカスタマイズする方法は、PGI User's Guide の 1 章の Examples of Using siterc and User rc Files を参照してください。

5.1 アプリケーションの配置と再配布

PGI コンパイラで構築されたプログラムは、ランタイム・ライブラリ・ファイルを必要とする場合があります。PGI コンパイラがインストールされていないシステム上でこのようなプログラムを実行するような場合は、プログラムと共にランタイム・ライブラリ・ファイルも配布する必要があります。このために、

全ての OS プラットフォーム用の再配布用ファイルを提供しています。Windows 版では、PGI は、Microsoft® の再配布ファイルも提供しています。

5.1.1 PGI 再配布用ランタイム・ライブラリ

PGI 2010 リリースでは、ランタイム・ライブラリを含むディレクトリは以下のとおりです。

- \$PGI/linux86/10.0/REDIST
- \$PGI/linux86/10.0/REDIST-RLR
- \$PGI/linux86-64/10.0/REDIST
- \$PGI/linux86-64/10.0/REDIST-RLR
- \$PGI/win32/10.0/REDIST
- \$PGI/win32/10.0/REDIST-RLR
- \$PGI/osx86/10.0/REDIST-RLR

これらのディレクトリには、PGI End-user License Agreement(EULA) の条項に基づいた PGI2010 ライセンス契約によって再配布可能な PGI Linux ランタイム・ライブラリの共有オブジェクト・ファイル、あるいは、Windows ダイナミック・リンク・ライブラリ (DLL) が含まれます。PGI End-user License Agreement(EULA)の正式なコピーは、2010 ディレクトリ、あるいはフォルダの中に含まれております。

5.1.2 Linux 再配布用ランタイム・ライブラリ

Linux REDIST ディレクトリの中には、サポートする全ての(CPU)ターゲット用の PGI ランタイム・ライブラリの共有オブジェクト・ファイルが含まれています。これによって、PGI がサポートしている、ほとんどの Linux システム上で実行することが可能な実行モジュールと PGI ランタイム・ライブラリのパッケージを作成することが可能となります。但し、このために必要な事項は、以下のとおりです。

- 実行モジュールを実行するエンドユーザは、ランタイム・ライブラリの場所 (パス) の設定等、そのための適切な環境を構築しておくこと
- Linux においては、PGI 共有オブジェクトの検索場所を指定するために LD_LIBRARY_PATH 環境変数をセットすること。

5.1.3 Microsoft® 再配布用ランタイム・ライブラリ

Windows 上の PGI 製品は、Microsoft Open Tools を含みます。Microsoft Open Tools のディレクトリには、"redist"と言うサブ・ディレクトリが存在します。PGI 2010 ライセンスは、PGI End-user License Agreement を準用し、このディレクトリ内のファイルを再配布することができます。

6. 既知の制限事項

この章は、既知の制約事項や、ドキュメンテーションのエラー、PGI Workstation/Server への修正事項等を含みます。

6.1 一般的な制限事項

ここで述べる問題の多くは、コンパイラ・オプション、サブオプションの特定な使用方法に関連するものです。

- PGI 2010 で生成されたオブジェクトあるいはモジュールは、PGI 5.x 並びにそれ以前のリリースのオブジェクト、モジュールとは互換性がありません。

- PGI 6.1 並びにそれ以前のリリースにおいて `-Mipa` オプションを付加して作成されたオブジェクト・ファイルは、PGI 2010 上で再コンパイルが必要です。
- `-i8` オプションを使用するプログラムにおいて、PGI コンパイラにバンドルしている ACML ライブラリととともに使用する場合、互換性がありません。このオプションの互換ライブラリに関しては、developer.amd.com をご参照ください。
- `-i8` オプションの使用するプログラムにおいて、MPI ライブラリと ACML ライブラリを使用する場合、互換性がありません。これらのライブラリと `Integer*8` 配列 `size` を引数に使用した場合、プログラムは異常終了します。
- PGCC の `-Mipa=libopt` と `-Mipa=vestigial` オプションを同時に使用した場合、リンク時に未解決な参照が生じる場合があります。これは、`-Mipa` にサブオプション `vestigial` を付加することにより、エラーとなる関数が削除されるために起こります。この場合は、`vestigial` を付加せずに、`-Mipa` のみの指定で対処してください。
- `mp` を使用して生成された OpenMP プログラムが、SuSE9.0 システム上の複数プロセッサで並列実行した場合、極端に遅くなります。しかし、これと同じ実行モジュールが SuSE9.1 上で動作させると期待される性能と効果が得られます。

6.2 プラットフォーム特有の制限事項

ここでは、各プラットフォーム別のコンパイラ制限事項について説明します。

6.2.1 Linux

- PGDBG あるいは PGPROF の GUI において極端に性能が低下するような状況が起きた場合、X ライブラリの `libxcb` を最新のバージョンへアップデートしてみてください。このバージョン番号は、`distribution` に依存して変化します。Linux distribution から正式にパッチを得るようにして下さい。
- `-mcmodel=medium` オプションを使用してコンパイルされたオブジェクトを組み込んだプログラムは、静的にリンクされることはできません。これは、linux 86-64 環境における制約事項であり、PGI コンパイラによる制約事項ではありません。

6.2.2 Apple Mac OS X

- **Tiger** は、PGI2010 以降サポートされませんのでご注意ください。
- Apple Mac OS プラットフォーム上では、PGI Workstation 2010 コンパイラは、ユーザバイナリの静的リンク (Static linking) をサポートしません。今後の Apple 社のアップデートとの互換性のために、コンパイラは、ユーザバイナリのダイナミック・リンク方式 (Dynamic linking) をサポートします。
- `Mprof=func` あるいは、`-Mprof=lines` の使用は、サポートしていません。
- Mac OS X 10.5 上の PGDBG による OpenMPI のデバッグセッションを開始するには以下の方法をとります。
 1. デバッガを起動する際に、実行モジュール名を「フルパス名」で指定します。例えば、次のような形態です。


```
pgdbg -mpi:mpirun -np 4 /home/user1/a.out
```
 2. Main 上でブレークポイントを指定します。
 3. ブレークポイントへ `continue` します。
 4. デバッグセッションを開始します。

6.2.3 Windows

- Windows において、cygwin に含まれる vi のバージョンは、SHELL 変数に予期しないものが指定されている場合、問題が起きる可能性があります。このケースの場合、vi が起動される時、次のようなメッセージを出力します。この問題を修復するには、SHELL 環境変数に対して cygwin の bin ディレクトリの中のシェルを指定してください。(例、/bin/bash)

E79: Cannot expand wildcards Hit ENTER or type command to continue

- Windows 上で、デバッグのためにビルドされたランタイム・ライブラリ (例 msvcrt と libcmtd) は、PGI Workstation には含まれていません。デバッグ用途のために、-g オプションを付けてコンパイルするとき、PGI ランタイム・ライブラリと Microsoft ランタイム・ライブラリの両方の標準的な non-debug バージョンが、いつも使用されます。この制約は、アプリケーションコードのデバッグに影響しません。

以下は、Windows と PGDBG に係る問題・制限事項となります。

- PGDBG Windows—一般にディレクトリ・パス名を区切るためにバックスラッシュ文字 (“\”) を使用します。PGDBG は、C 言語の数式表示として、バックスラッシュをエスケープ文字の意味として使用します。しかし、Windows 上の PGDBG においては、ディレクトリ・パス名を区切るためにフォワードスラッシュ文字 (“/”) を使用してください。

(注意) 上記事項は DEBUG コマンド、あるいはコマンドライン上の実行モジュール名には適用されません。(但し、このコンベンション自体は、これらのコマンドにおいても動作します)

- Windows プラットフォーム上の PGDBG において、一つのブロックである「システムコール」を超えてシングル・ステップ実行したとき、Windows は stepi/nexti をタイムアウトします。これを改善する方法に関しては、オンライン www.pgroup.com/support/tools.htm をご参照ください。
- PGDBG SFU/SFA—オペレーティングシステムのサポートがないために、SFU/SUA システム上の PGDBG は、ハードウェア watchpoint (hwatch コマンド) をサポートしていません。
- PGDBG SFU/SFA—オペレーティングシステムの制約により、32bit SUA 上のプログラムだけが PGDBG のマルチスレッドのデバッグが可能です。但し、一つ制約があり、一度ストップすると、プロセスは全てのスレッド、あるいは、シングルスレッドの続行 (スレッドの部分的なセットの続行ではなく) をしなければ、プロセスの続行ができません。スレッドの部分的なセットの続行は、結果として全てのプロセス (スレッド) を続行させることとなります。

以下は、Windows と PGPROF に係る問題・制限事項となります。

- PGI Workstation ランタイム・ライブラリ DLLs と共に、-Mprof を使用しないでください。プロファイリングのための実行モジュールを構築するためには、スタティック・ライブラリを使用してください。基本的に、-Bdynamic を使用しない限り、スタティック・ライブラリがデフォルトとなります。

6.3 PGDBG に関する制限事項

- PGDBG— .so や .dll 等の shared ライブラリは、そのライブラリの中に含まれるコードがブレイクポイントとしてセットされる前にロードされていなければなりません。
- Fedora Core 6 or RHEL5 上でシェアード・ライブラリのロード認識において PGDBG 内の問題により、MPICH-1 のデバッグにおいて rank0 のプロセス以外のプロセスのブレイクポイントが無視されます。これは、ランダムなアドレスを有効にするシェアード・ライブラリのローディングを行っている

る時に生じます。

- PGDBG – Unified Binaries (-tp x64 を付けて構築されたモジュール) のデバッグは、フルにサポートされません。いくつかのサブプログラム名が unified binaries を生成する過程で変更されています。PGDBG は、アプリケーションのソースコード内で使用されているプログラム名とこれら変更された名前との間の変換を行いません。デバッグにおいては、Unified Binaries 生成オプションを付けずに、native なプロセッサターゲットで実行モジュールを生成してください。あるいは、Unified Binaries のデバッグを行う方法に関しては、<http://www.pgroup.com/support/tools.htm> に詳細な情報がありますので、ご参照下さい。

6.4 PGPROF に関する制限事項

- -Mprof=func 並びに -mcmodel=medium あるいは -mp を一緒に使用した場合、生成された実行モジュールはセグメンテーション・フォールトが生じます。これらのオプションを同時に使用することは避けてください。
- gprof スタイルのプロファイリング機能(-pg)を使用してコンパイル・リンクしたプログラムは、2.6.4 Linux カーネルのシステム上でセグメンテーション・フォールトが生じます。
- マルチスレッドのサンプルベースのプロファイル機能 (gprof スタイルのプロファイリング: -pg あるいは、-Mprof=time オプション) でレポートされる時間は、マスタ・スレッドだけのものとなっています。一方、PGI 形式のプロファイリング(-Mprof=lines|func)あるいはハードウェア・カウンタ・ベースプロファイリング(-Mprof=hwcts) あるいは、pgcollect は、個々のスレッドあるいはプロセスの時間データを個別に取得するために使用されます。

6.4 修正事項

多くの問題が、PGI 2010 で修正されました。TPR(Technical Problem Report) やその詳細については、www.pgroup.com/support/release_tprs.htm をご覧下さい。解決した各問題のサマリーを記しております。

以上

※なお、文中で使用されている商品名、名称は、各社の商標あるいは登録商標です