# CUDAプログラミング入門（2011年4月1日版とそれ以前の版）の重要な誤り

2012年6月1日

「CUDAプログラミング入門」(2011年4月1日版とそれ以前の版)の、記述の誤りや追加項目を本資料に示します。修正した箇所のうち、特に重要と思われる項目を、本資料のP1,P2に示します。
また、本資料の内容を反映した版を、「CUDAプログラミング入門」(2012年6月1日版)として、本サイト(http://accc.riken.jp/HPC/training.html)に添付します。

■(重要) 以下の説明が抜けていました。

P3-4、P4-5、6－3節，6－4節に出てくるcudaMemcpy～Async型のCUDA関数の場合、以下の②で使用するホスト側の配列Aは、①,③のようにPage-Locked(またはpinned)ホストメモリ(6－2節参照)として確保/解放する必要があります。なお、P6-18の図6-3-9の例はそのようになっています。

```
float *A;
size_t size = N*sizeof(float);
cudaHostAlloc((void**)&A,size,cudaHostAllocDefault); ①
 :
cudaMemcpyAsync(A,dA,size,cudaMemcpyDeviceToHost,0); ②
  :
cudaFreeHost(A);                                     ③
```

■(重要) P6-16の記述に誤りがありました。

●P6-16の4～7行目の3つの規則のうち、1つ目の規則で、例えば現在「コピーのタスク」が実行中の場合、
【誤】待ち行列内の一番上のタスクのみ、選択が可能です。
【正】待ち行列内の、(本例では)「カーネル関数のタスク」のうち、一番上のタスクのみ、選択が可能です。例えば以下で「ｶｰﾈﾙ関数 ①」は選択が可能です。

```
      ｺﾋﾟｰ ②  実行中
      ｺﾋﾟｰ ③
ｶｰﾈﾙ関数 ①  選択可能
ｶｰﾈﾙ関数 ②
```

●P6-16の4～7行目以外に、下記の規則が追加されます。
ストリームID「0」のタスクは、他のタスクと同時に実行することはできません。

図6-16の図6-3-7のように、待ち行列内に多くのタスクが存在する場合、スケジューラーは上記の条件以外にもいくつかの条件を考慮し、実行するタスクを選択します。説明が長くなりますので、詳細は「CUDAプログラミング入門」(2012年6月1日版)のP3-4,P4-5,6－3節,6－4節(特に6－3－3節)を参照して下さい。

■(重要) P3-21の5行目で、「一方__device__修飾子で宣言した配列については、(グローバルメモリ上で何バイト境界から開始するか)マニュアルに記述がないようです。」と書きましたが、「CUDA C Programming Guide」の5.3.2.1.1節によると、「グローバルメモリ内に存在する変数((筆者注)スカラー変数と配列を意味すると思われます)は、グローバルメモリ上の少なくとも256バイト境界から開始する。」と記載されています。従って、__device__修飾子で宣言した変数/配列も、cudaMemcpyを使用して変数/配列を確保したP3-5の図3-2-6(3)と同様に、少なくとも256バイト境界から開始すると思われます。

■(重要) P2-15の図2-5-5(スレッド数は32)で、要素数が0の場合(通常はありえませんが)、(1)の方法ではブロック数が0(正しい)になりますが、(2)の方法ではブロック数が1(誤り)になるので、(1)の方法を使用した方が無難だと思われます。

■(重要) P4-3の「■__syncthreads()に関する補足(1)」の追加です。下記の図1では、同一ブロック内の各スレッドが、①と②のいずれかの__syncthreads()で同期を取っていますが、このような使用方法は誤りです。図2の③のように、同一ブロック内の各スレッドが、同一の__syncthreads()で同期を取るようにして下さい。

```
   :
if(~){
   :
  __syncthreads(); ①
}else{
   :
  __syncthreads(); ②
}
   :
```

図1 ✖ 誤り

```
   :
if(~){
   :
}else{
   :
}
 __syncthreads(); ③
   :
```

図2 ○ 正しい

■(重要) 構造体を用いる場合、グローバルメモリのデータのロード/ストアがコアレスアクセスにならず、速度が低下する可能性があります。これについては、この資料に記載すると説明が長くなりますので、「CUDAプログラミング入門」(2012年6月1日版)のP3-9,P3-10を参照して下さい。

■ P1-1の16行目で、「NVIDIA社が無償で提供し」の下線部を追加して下さい。

■ P1-6の2行目の「(変数iとは異なる)レジスター(実際は複数)」を、「スレッドごとに異なるレジスター」に変更して下さい。

■ P2-1の下から12行目を以下のように修正して下さい。
(修正前)「関数内で、C言語の～使用できません。」
(修正後)「関数内で、printf関数やmalloc関数などを使用することはできません。」

■ P2-5の図2-3-2は、下記のように書くこともできます。

```
dim3 NBLOCKS,NTHREADS;
NBLOCKS  = dim3(2,1,1);
NTHREADS = dim3(4,1,1);
kernel<<<NBLOCKS,NTHREADS>>>(dA);
```

↓ 下線部を省略

```
dim3 NBLOCKS,NTHREADS;
NBLOCKS  = 2;
NTHREADS = 4;
kernel<<<NBLOCKS,NTHREADS>>>(dA);
```

```
dim3 NBLOCKS  = dim3(2,1,1);
dim3 NTHREADS = dim3(4,1,1);
kernel<<<NBLOCKS,NTHREADS>>>(dA);
```

↓ 下線部を省略

```
dim3 NBLOCKS  = 2;
dim3 NTHREADS = 4;
kernel<<<NBLOCKS,NTHREADS>>>(dA);
```

■ P2-12の下から4行目の「SMで次の命令の実行を実行します。」の下線部を削除して下さい。

■ P2-14の図2-5-1(2)と図2-5-1(1)の下線部を逆にして下さい。

■ P3-2の図3-1-1の、コンスタントキャッシュ内を以下のように修正して下さい。
「dC[2]」→「dC[2]の一部」

■ P3-4の下から3行目の「10000回コールされ、」の下線部を削除して下さい。

■ P3-4の下から6行目の「その時点で稼働中のCUDA関数、またはカーネル関数の全スレッドが」を「その時点より前にコールされた全てのCUDA関数、およびカーネル関数が」に変更して下さい。

■ P3-20の下から3行目の「一方、通常の(CUDA化していないない)」の下線部を削除して下さい。

■ P3-26の上から8行目の「(つまり、当該ブロック内の全スレッドが～」の下線部を追加して下さい。

■ P3-31の上から7行目の「シェアードメモリ上に配列dS[2]を確保します(配列dSはブロックごとに確保され、ブロック内のスレッド数が2なので～」の下線部を追加して下さい。

■ P3-31の下から8行目の「ただし本例の場合は～⑰はなくても構いません。」を削除して下さい。

■ P3-37の図3-6-30(2)の一番右の列の各要素がすべてS[0][16]になっていますが、上から順にS[0][16] ･･･ S[1][16], S[15][16]に修正して下さい。

■ P3-40の8行目の最後の「スレッド関数」を「カーネル関数」に変更して下さい。

■ P3-41の以下の部分を修正して下さい。

16行目「図3-8-3の④に示すように」→「図3-8-3の③に示すように」
18行目「⑤に示すように、デバイスメモリ上の」→「④に示すように、デバイスメモリ上の」

■ P4-8の図4-2-4内の「memory size ～ 32 bit」(2箇所) を「out of memory.」に変更して下さい。

■ P5-5の図5-2-9(1)と図5-2-10(1)で、左側が「A[0] = A[1] + B[0];」,
右側が「A[1] = A[2] + B[1];」に修正して下さい。

■ P5-7の5,6行目の「ループの各反復で同じ一時変数を使用するループを並列化した場合」の下線部を「更新」に変えて下さい。

■ P6-1の14行目で、「またGPUでは、スレッドの切り換えを高速なハードウェアで行うため、時間がかかりません。」とありますが、これについて補足します。CPUやコアでは、プロセスやスレッドの切り換えで、レジスターの内容をメモリに保存/復元します。GPUではレジスターが多く、メモリへの保存や復元が必要ないので、切り換えに時間がかかりません。

■ P6-9の下から2行目の「～図6-1-15(1)のように～」の下線部を追加して下さい。

■ P6-14の下から3行目の「両方ともコピー(または両方ともカーネル関数)」の下線部を削除して下さい。

■ P6-17の下から4行目の「●図6-3-9では～方法もあります。」を削除して下さい。これに伴い、P6-18の図6-3-10も削除して下さい。

■ P6-17の上から24行目で、「ストリーム②はdA[12]から始まる要素をカーネル関数に渡します。」の下線部を追加して下さい。

■ P6-17の下から2行目の「や６－４節で紹介するタイマー」を削除して下さい。

■ P6-20の最後の行で「ストリームIDを指定して下さい((8)、および(11),(12),(13)も同様)。」の下線部を追加して下さい。

■ P6-22の図6-5-2で、「else」を「}else{」に変更して下さい。

■ P8-3の上から4行目で、「大きさ2の配列dSをシェアードメモリ上に確保します(配列dSはブロックごとに確保されます)」の下線部を追加して下さい。

■ P8-4の下から3行目、P8-5の8行目、P8-6の8行目で「配列dAの(計算すべき)要素を担当していないスレッド」の下線部を追加して下さい。

■ P8-10の最後の行で、「連続しておらず、(ストライドが偶数で)とびとびになっているためです。」の下線部を追加して下さい。

■ 付録A-1の4行目の CUDAの最新版のマニュアルのURLを、以下に変更して下さい。
http://developer.nvidia.com/cuda-downloads で「GET LATEST CUDA TOOLKIT PRODUCTION RELEASE」をクリック。

■ 付録A-2の「■ NVIDIA社が作成した資料、Webサイト」に下記を追加して下さい。
●「エヌビディア ジャパン チャンネル」
http://www.youtube.com/user/NVIDIAJapan セミナーの動画です。
●「GPUコンピューティング ソリューションファインダー」
http://www.nv-event.jp/solution-finder/index.php

■ 付録A-2の「■ 外部の講習会」の爆発研究所のセミナー(上から4つ目)を以下に変更して下さい。
●「GPUトレーニングコース」 爆発研究所
http://bakuhatsu.jp/computational/gpu-training/

■ 付録A-2の「■ 外部の講習会」に以下を追加して下さい。
●「エヌビディア主催トレーニング」nvidia社が開催するトレーニングコース
http://www.nvidia.co.jp/object/cuda-dev-program-jp.html

■ 付録A-2の「■ 外部の講習会」に以下を追加して下さい。
●「AOCプランニング」
https://sites.google.com/a/aocplan.com/web/gpgpu/business-results

■ 付録A-2の下からの2行で、以下の下線部を削除して下さい。
●「GPGPUセミナー2010」 サードウェーブ
http://gpgpu.dospara.co.jp/gpgpu_seminar2010.html

■ 付録A-3の「■ Webサイト」に下記を追加して下さい。
●「ゼロから始めるGPUコンピューティング」 http://www.gdep.jp/page/view/203
●東京工業大学青木教授の講義資料
http://www.ocw.titech.ac.jp/ で「講義を探す」の欄に「GPUコンピューティング」とキーインし、「検索する」をクリックします。表示された画面で「講義ノート」をクリックします。
●「GPUコードジェネレーター」 http://www.otb-japan.co.jp/gimmick/index.html

■ 付録A-5の「■ 書籍/雑誌」に下記を追加して下さい。
●「先端グラフィックス言語入門」 安福健祐、伊藤拡、大熊建保 著 (フォーラムエイト)
●日経ソフトウエア 2011年8月号 の記事「これから始めるGPUプログラミング」
●「CPU & GPUがわかる本」(工学社)
●(英語) CUDA Application Design and Development