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 P 1，P 2 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。

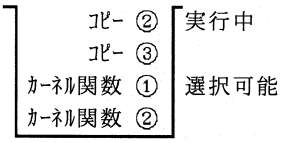
■（重要）以下の説明が抜けていました。

P 44、P 89、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、P 151 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）P 149 の記述に誤りがありました。

- P 149 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。



- P 149 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 149 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 P 44，P 89，6-3 節，6-4 節 (特に 6-3-3 節) を参照して下さい。

■（重要）P 61 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 P 45 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）P 27 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）P 87 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 P 49，P 50 を参照して下さい。

■ P 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。
■ P 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。
■ P 13 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。」
■ P 17 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);
```

■ P 24 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。
■ P 26 の図 2-5-1 (2) と図 2-5-1 (1) の下線部を逆にして下さい。
■ P 42 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」
■ P 44 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。
■ P 44 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。
■ P 60 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。
■ P 66 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。
■ P 71 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。
■ P 71 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。
■ P 77 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。
■ P 81 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。
■ P 82 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」
■ P 92 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。

■ P 117 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。
■ P 119 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。
■ P 133 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。
■ P 141 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。
■ P 147 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。
■ P 150 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、P 151 の図 6-3-10 も削除して下さい。
■ P 150 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。
■ P 150 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。
■ P 153 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。
■ P 155 の図 6-5-2 で、「`else`」を「`}else{`」に変更して下さい。
■ P 179 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。
■ P 180 の下から 3 行目、P 181 の 8 行目、P 182 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。
■ P 186 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 199 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 200 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 200 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 200 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 200 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 200 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 201 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 203 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