グラフィックスプロセッサ を用いた数値計算

nVIDIA GeForce GPUとCUDA編

1

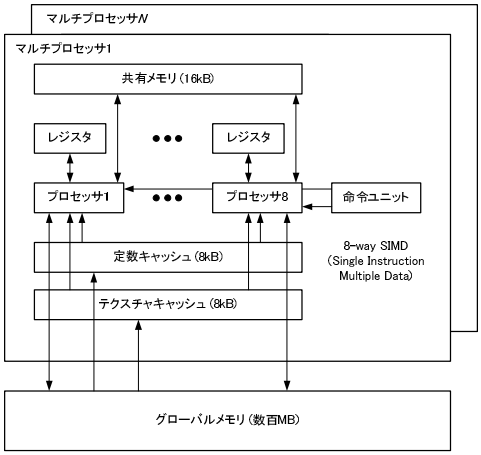
グラフィックスプロセッサ (GPU)

- 画像処理用のハードウェア
  - 3次元グラフィックス
- 高速化、高機能化
  - 高並列
  - 32bit浮動小数点演算
- 汎用数値計算指向
  - コンパイラの提供

2

nVIDIA GeForce 8000/9000 GPU

- 高並列
  - 8-way SIMD
  - 4〜16組
  - 32〜128プロセッサ
- 複数のメモリ
  - 共有メモリ
  - キャッシュ×2:読出専用
  - グローバルメモリ



3

nVIDIA CUDA
(Compute Unified Device Architecture)

- GPUで数値計算を行う開発環境
- Cコンパイラやライブラリを提供
  - BLAS (Basic Linear Algebra Subprograms)
  - FFT
  - Matlab Interface
- C言語を拡張
  - 並列処理
  - CPUとGPU
  - 複数のメモリ

4

並列処理

- スレッド (Thread)
  - 最小単位
- スレッドブロック (Thread Block)
  - 複数のスレッド
- グリッド (Grid of Thread Block)
  - 複数のスレッドブロック
  - 単一の「カーネル」
- ワープ (Warp)
  - 同一命令を並列実行する単位
  - 32スレッド (8PU×4Thread)

5

グリッド

- 単一の「カーネル」（GPUで実行するジョブ）
- 複数のスレッドブロックに分割
  - 2次元以内のブロックID⇒二重ループを展開
  - 最大で65536×65536
  - ブロック間の同期は無い
  - 実行順序は不定
  - 複数の「マルチプロセッサ」で並列処理
- 複数のグリッドを実行可能

6

スレッドブロック

- 複数のスレッド
  - 3次元以内のスレッドID⇒三重ループを展開
  - 最大512×512×64、合計512スレッド
  - 実行順序は不定
- 単一の「マルチプロセッサ」で実行
- 共有メモリ
  - スレッドブロック生成時に確保、終了時に開放
- スレッド間の同期

7

ワープ

- 同一命令を並列実行する単位
- 1ワープ＝32スレッド
  - 8個のPUが同一命令を実行
  - 同じ命令を4サイクル連続で実行
    - ループアンローリング(後述)と関連
- ワープサイズ分の並列度を持たせるべき

8

並列処理の考え方

- ループを自動並列化
- 五重ループまで対応

Cプログラム

```
for (i=0;i<I;i++)
 for (j=0;j<J;j++)
  for (k=0;k<K;k++)
   for (l=0;l<L;k++)
    for(m=0;m<M;m++)
     Func(i,j,k,l,m);
```

CUDAで実行

```
Dg={J,I,0};
Db={M,L,K};
Func <<Dg,Db>>();

Func() {
  j=blockIdx.x; …;
  m=threadIdx.x; …;
```

9

CUDAのプログラミング

- 処理の流れ
  1. データをCPUからGPUに転送
  2. GPUでプログラムを実行
  3. データをGPUからCPUに転送
- ループを並列処理に変更

10

プログラムの最適化

- スレッド、ブロック、グリッドの分割
- レイテンシの隠蔽
  - 同時に実行できるスレッドを増やす
- メモリの使い分け
  - 共有メモリ
  - 読み出し専用キャッシュ
- アルゴリズムの見直し

11

レイテンシ

- 演算開始から終了までの時間
  - 単精度浮動小数点加減算、乗算は4クロック
  - 特殊関数は16～32クロック
  - 共有メモリアクセスは4クロック
  - グローバルメモリアクセスは400～600クロック
- 高性能化
  - 低レイテンシ処理を選ぶ
  - レイテンシを隠蔽する
    - ループアンローリング
    - 同時に実行可能なスレッド数を増やす

12

ループアンローリング(1)

レイテンシを隠蔽する手法の一つ

例: スループット=1, レイテンシ=4

ループアンローリング前

1: データ1 処理1
2: 待
3: 待
4: 待
5: データ1 処理2
6: 待
7: 待
8: 待
9: データ1 処理3
10: 待

ループアンローリング後

1: データ1 処理1
2: データ2 処理1
3: データ3 処理1
4: データ4 処理1
5: データ1 処理2
6: データ2 処理2
7: データ3 処理2
8: データ4 処理2
9: データ1 処理3
10: データ2 処理3

13

ループアンローリング(2)

- パイプラインの有効利用
- 大量のレジスタが必要
- GeForce GPUはレジスタが多い
  - レジスタ8192本/マルチプロセッサ
  - 平均レジスタ1024本/プロセッサ
  - 数百クロックのレイテンシを隠蔽できる
- レイテンシに見合った反復回数を確保すべし
  - グローバルメモリアクセスは400～600

14

同時実行可能なスレッド数を増やす

- ループアンローリング
  - パイプラインを埋める
  - 単一スレッドの高速化
- GPUは多数のスレッドを同時に実行可能
  - 同時実行でパイプラインを埋める
  - 同時実行できるスレッドを単一スレッドブロックに

15

メモリの使い分け

- 共有メモリ
  - 高速、読み書き可能
  - スレッドブロック内で頻繁に読み書きするデータ
- キャッシュ
  - 高速、読み出し専用
  - スレッドブロック内で頻繁に読み出すデータ
- グローバルメモリ
  - レイテンシに注意

16

GeForce + CUDAサーバ

- 仕様
  - Core 2 Duo E6750 (2.66GHz, L2 4MB)
  - 4GB RAM
  - GeForce 8800GTS×2
    - 128 Shader Processors
    - 1.652GHz
    - 512MB RAM

18

参考資料

- NVIDIA CUDA Programming Guide
- Accelerating MATLAB with DUDA Using MEX Files

19