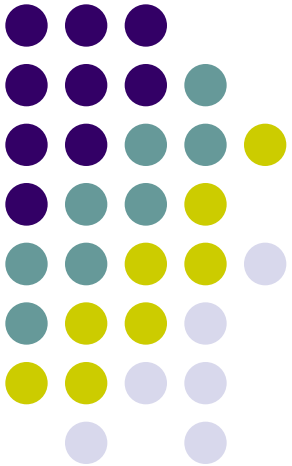
2013年度
実践的並列コンピューティング
第12回

GPUプログラミング (3)

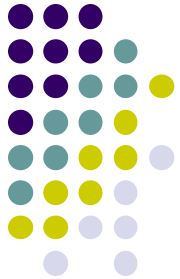


遠藤 敏夫

endo@is.titech.ac.jp

2013年7月1日

行列積演算 (1)



- 行列積演算サンプルプログラム

  行列A, B, Cがあるとき、C=A×Bを計算する

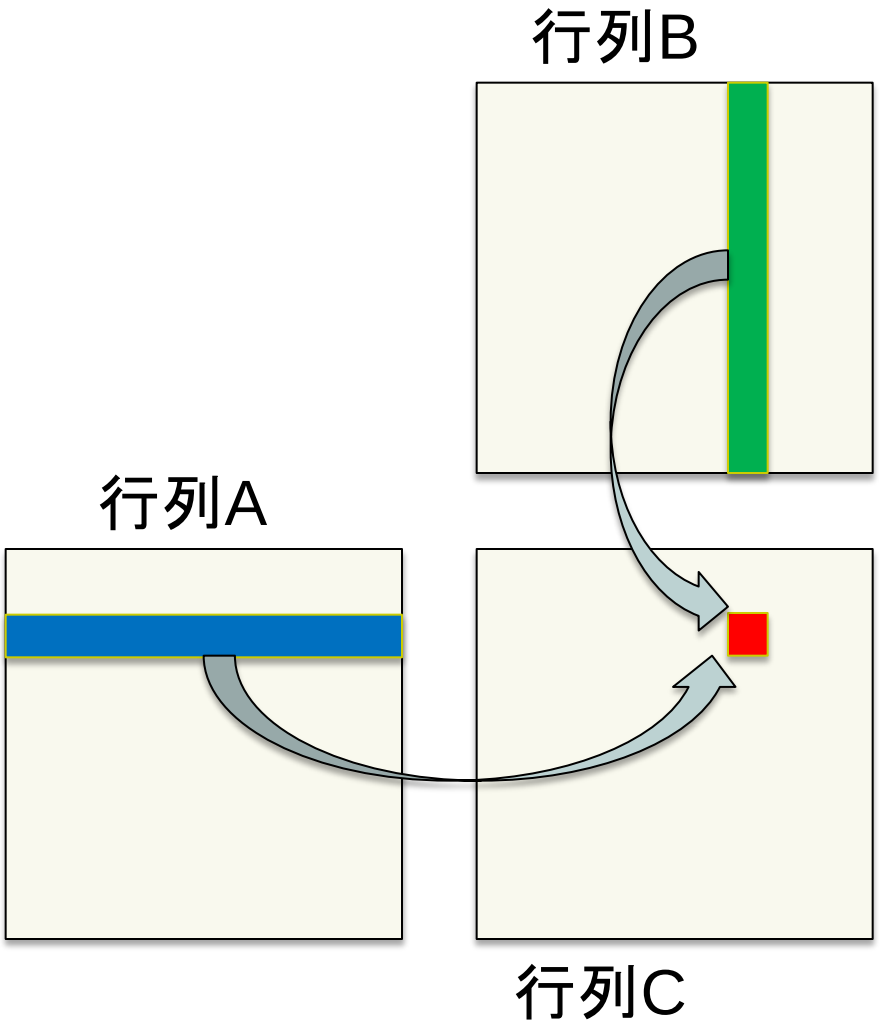
  全行列とも1024x1024のとき、いくつかのバージョンを比較:

  - mm
    - CPU上の1スレッドで計算 → **約1.1秒**
  - mm-omp
    - CPU上の複数スレッドで計算 → **約0.14秒** (12スレッド)
  - mm-cuda1t
    - GPUの1スレッドで計算 → **約400秒**

  - mm-cuda
    - GPUの複数スレッドで計算

    → **約0.055秒 (cudaMemcpy除くと0.044秒)**

行列積演算(2): cpu版



行列Cの要素$C_{i,j}$を求めるには

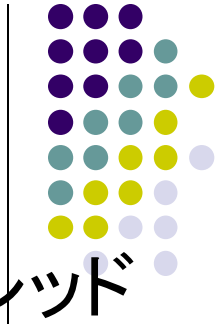
- Aの第i行全体
- Bの第j列全体

の内積計算を行う

→ このためにforループ

C全体を計算するためには、
三重のforループ

行列積演算 (3): GPU並列版



- mm-cudaでは、m x n個のスレッドを用い、1スレッドがCの1要素を計算

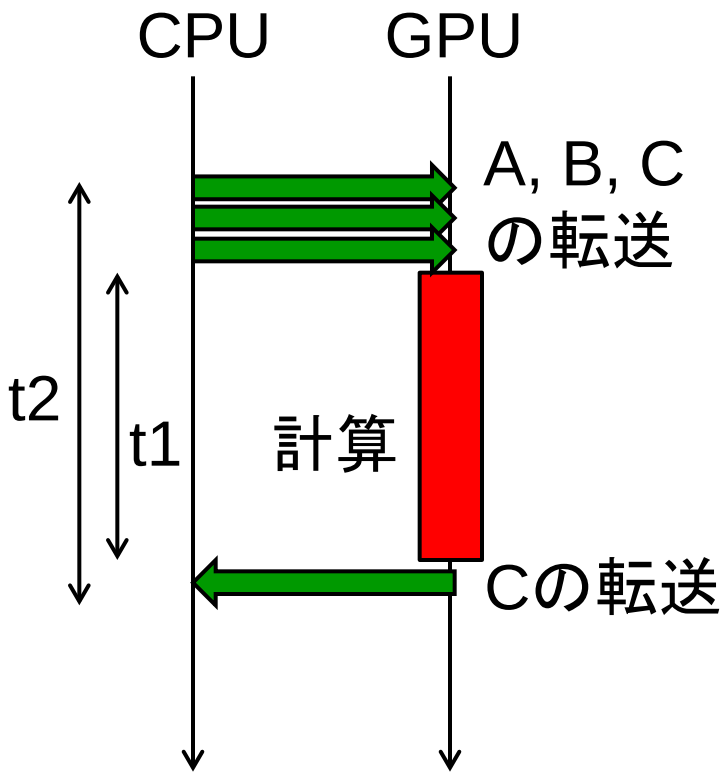
```
matmul_kernel<<<dim3(m / BS, n / BS), dim3(BS, BS)>>>
        (DA, DB, DC, m, n, k);
```

BSは前もって適当に決めた数(16)

- カーネル関数は内積のための一重forループ
- グリッドサイズ・ブロックサイズとも二次元で指定

ちなみに、更なる並列化のために、Cの1要素の計算を複数スレッドで行うのは容易ではない (内積のreduction時にスレッド間の同期が必要)

データ転送時間を性能測定に含めるべきか否か？



- 一概にはどちらが正しいとは言えない。実用的なプログラムでは、前後の文脈によるため
- サンプルプログラムでは、t1とt2両方表示

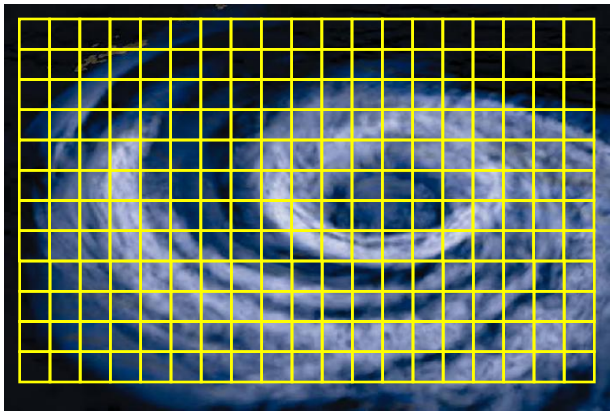
- t1 ≒ cmnk
- t2 ≒ t1 + d(mk+kn+2mn)
  - c,dはアーキテクチャから決まる定数

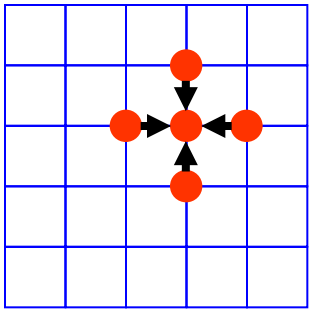
再掲:
サンプルプログラム: Advection

二次元格子空間の，流体の簡単な流れをシミュレート

詳細は http://sacsis.hpcc.jp/2010/gpu/

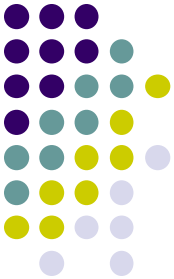
- f(x,y)がその点の「インク濃度」
- 空間の端のfと，流速u, vが与えられたときの，時間経過の様子
- きめられた時間ステップ計算したら終了
- ダブルバッファリング





隣の点の情報を用いて値更新
(正確には最大2つ隣り)

サンプルプログラム: Advection (続き)

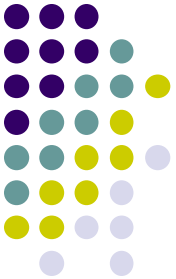


- 実行オプション: ./main [pno]
  - Pno: 問題番号 = 1～3, 11～15
    あとの問題のほうが時間がかかる. gpucapi.c参照
    注: 問題サイズ11~15では、回答チェック機能が働きません
- 計算量: O(nx × ny × nt)
  - Nx, ny: 空間サイズ. nt: 時間ステップ数

これをGPUで並列化するには??

- 空間ループを並列化するのは、CPU版と同じ。本サンプルでは、1スレッドが1点を計算。
- GPU版: ~endo-t-ac/ppcomp/13/advection-cuda/

GPU版Advectionの流れ



初期の二次元格子データをCPUからGPUへ

for (jt = 0; jt < nt; jt++) //時間ループ

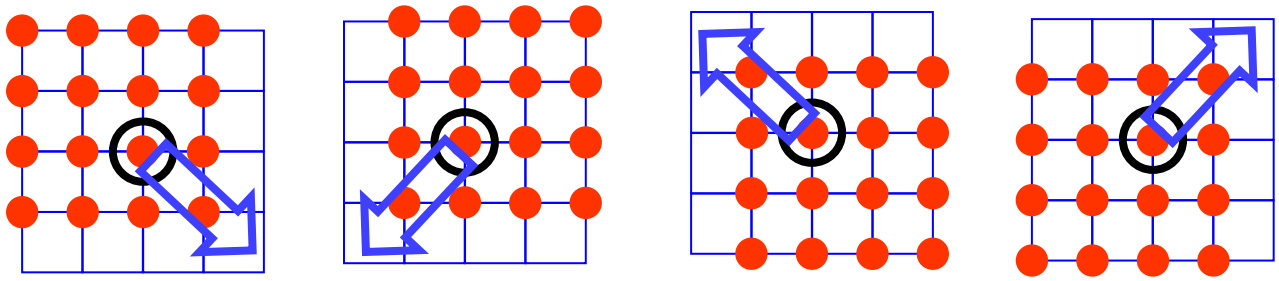
　　全格子点をGPUで計算

　　二つのバッファを交換

結果の二次元格子データをGPUからCPUへ

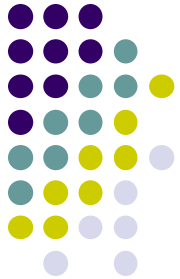
GPU版Advectionの一点の計算

- 「風向き」によってif文で分岐
  - http://sacsis.hpcc.jp/2010/gpu/ ⇒ 規定課題 ⇒ 規定課題マニュアルも参照



Divergent分岐あり⇒性能低下している可能性。この影響の軽減が望ましい
Ifを完全に無くすのは難しい。Ifの中の処理を減らし、ifの外に出せるとよいかも

マルチGPUの利用



- GPU内の多数CUDA coreを用いてプログラムの高速化可能だが、限界がある

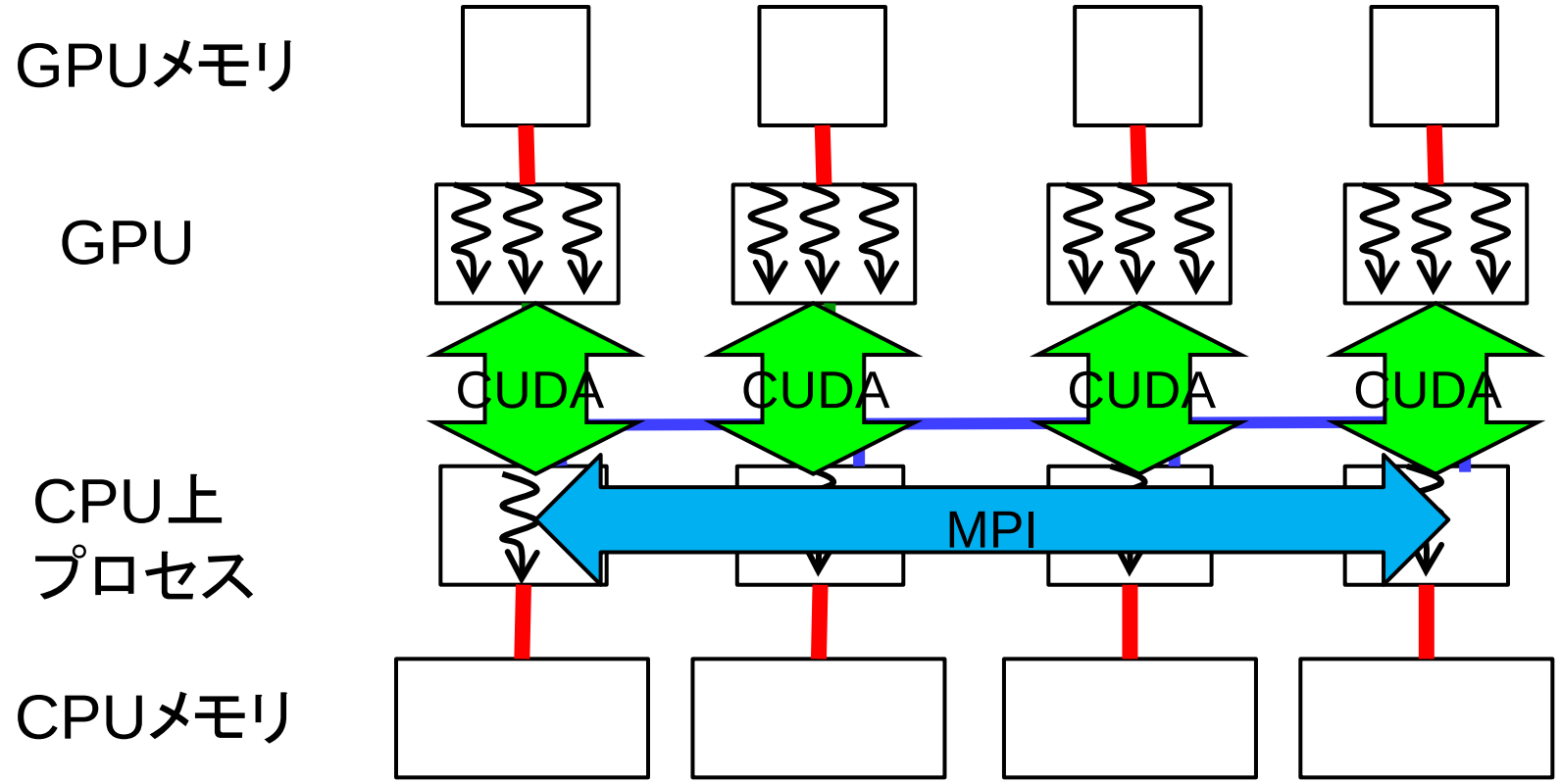
⇒ 多数GPUを用いて限界突破を狙う

  - TSUBAME2.0には1ノードあたり3GPU、システム全体で4200GPU

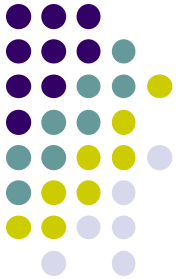
いくつかの基本方針:

- MPI+CUDA: 1プロセスが1GPUを担当 ←これを解説
- OpenMP+CUDA: 1スレッドが1GPUを担当
- 最近のCUDAでは、複数プロセスが1GPU利用したり、1プロセスが複数GPU利用できる(今回は略)。

MPI+CUDAの考え方



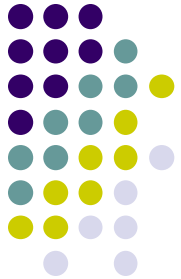
MPI+CUDAの考え方



- CPU-GPU間の通信はcuda(cudaMemcpyなど)で、CPU-CPU間の通信はMPIで
  - GPU-GPU間の直接通信は原則ない

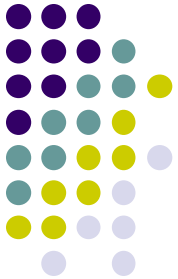
    ⇒ ただし、GPU Directで一部可能な場合も。
    MVAPICH 1.8以降で対応開始

MPI+CUDAの注意



- 1ノードに複数GPUが搭載されている場合、デバイス番号の指定の必要
  - 各プロセスは最初にcudaSetDevice(デバイス番号); を呼ぶ
  - TSUBAME2.0は3つ。デバイス番号は0~2
  - 1ノードに3MPIプロセスずつ起動し、cudaSetDevice(rank % 3); など

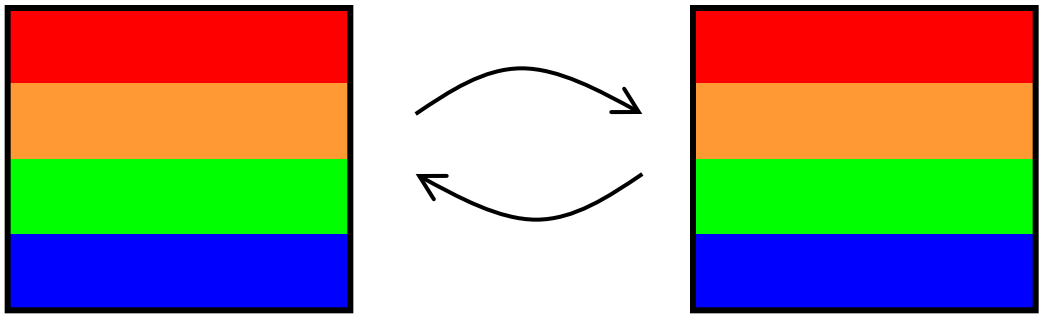
MPI+CUDAの注意(2)



- .cuファイルをnvccでコンパイルした後、MPIライブラリ、CUDAライブラリをリンクする必要
  - ~endo-t-ac/aahpc/12/mpicudatest/Makefile を参照
- OpenMPIでは、mpirunのオプションに以下が必要
  - -mca mpi_leave_pinned 0 または

    -mca btl_openib_flags 1
  - MPIとCUDAの両方がDMAを用いることによる問題を避ける

Advection on MPI+CUDAの方針



- 格子を複数プロセスに空間分割するのは基礎編と同じ
- 毎ステップで境界領域を隣接プロセス間で交換(MPI通信)するのも同じ
  - しかし、計算中の格子データはGPUメモリ上にある。どうするか

MPI+CUDA版Advectionの流れの想定

MPI_Init

初期の「部分」二次元格子データをCPUからGPUへ

for (jt = 0; jt < nt; jt++) //時間ループ

　　行B, DをGPUからCPUへコピー

　　B, DをMPIで送信, A, Eを受信

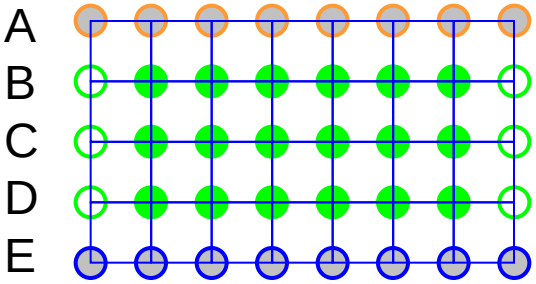
　　行A, EをCPUからGPUへコピー

　　担当の格子点をGPUで計算

　　二つのバッファを交換

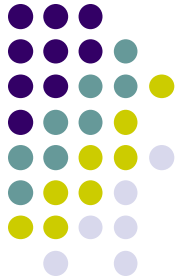
結果の二次元格子データをGPUからCPUへ

MPI_Finalize



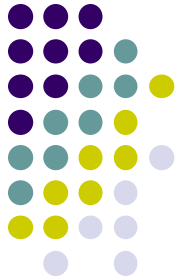
デッドロックに注意。
MPIの回参照

その他、取り上げられなかった話題



- CUDA Stream
  - 複数stream利用により、GPU計算、CPU→GPU通信、GPU→CPU通信をオーバラップ可能に
- CUDA Profiler
  - GPUカーネルの性能解析
- より詳細なアーキテクチャによる影響
  - Shared memory のbank conflict
  - スレッドブロック数とレジスタ数・shared memory容量の関係
- CUDA 5.0やKepler世代GPU特有の機能
- OpenACC言語: CUDAよりも気軽なGPUプログラミング
  - http://tsubame.gsic.titech.ac.jp → 各種利用の手引き → OpenACC利用の手引き

本授業のレポートについて



- 基礎編から一問＋応用編から一問、計二問のレポート提出を必須とします
- 基礎編
  - OpenMP+MPIの、1.～3.の中から一問以上
- 応用編
  - GPUプログラミング
  - Map-Reduce(予定)

  } この中から一問以上
- それぞれの編で二問以上を提出してもよい

応用編(GPU)課題説明 (1)

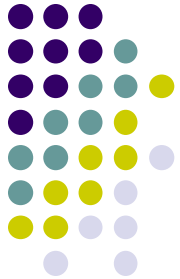
応用編レポートの〆切は8/8(木)

以下のG1, G2, G3、または次回以降のMapReduce編課題の、いずれかについてレポートを提出してください.

[G1] 行列積プログラムの性能を、行列サイズを変化させながら性能評価してください。CPU(OpenMP)版とも比較してください。

- データ転送コストを速度計算に入れる場合・入れない場合それぞれについて測定
- GPU版とCPU版の性能比が近くなるのはどういう場合か
- プログラムを改良してもok

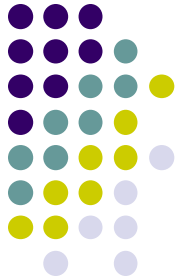
応用編(GPU)課題説明 (2)



[G2] Advectionサンプルプログラムの GPU版を何らかの形で改良してください。

- ~endo-t-ac/ppcomp/13/advection-cuda
- たとえば:
  - Divergent分岐の影響の削減
  - Shared memoryの利用による高速化
  - マルチGPUの利用
  - ほか

応用編(GPU)課題説明 (3)



[G3] 自由課題：任意のプログラムを，CUDA (OpenACCなども可) を用いて並列化し、GPU上で評価してください

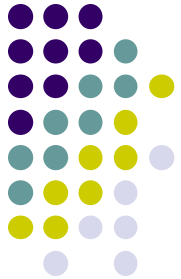
- たとえば，過去のSuperConの本選問題

  http://www.gsic.titech.ac.jp/supercon/

  たんぱく質類似度(2003)，N体問題(2001)・・・

  入力データは自分で作る必要あり
- たとえば，自分が研究している問題

課題の注意



- いずれの課題の場合も，レポートに以下を含むこと
  - 計算・データの割り当て手法の説明
  - TSUBAME2などで実行したときの性能
    - スレッド数、スレッドブロック数、GPU数を様々に変化させたときの変化に触れているとなお良い
    - 問題サイズを様々に変化させたとき(可能な問題なら)
  - 高性能化のための工夫が含まれているとなお良い
    - 「XXXのためにXXXをしてみたが高速にならなかった」のような失敗でも可
  - プログラムについては，zipなどで圧縮して添付
    - 困難な場合，TSUBAME2の自分のホームディレクトリに置き，置き場所を連絡

次回/Next

- 7/8(月)
  - Map-Reduceプログラミング (1)
- 7/15(月・祝)
  - 特別講義
- 7/22(月) 最終回
  - Map-Reduceプログラミング (2)

  - スケジュールについてはOCW pageも参照
    - http://www.el.gsic.titech.ac.jp/~endo/

    → 2013年度前期情報(OCW) → 講義ノート