

技術系大学におけるGPU教育の一試行

出川智啓（長岡技術科学大学）

発表内容

- ▶ はじめに
 - ▶ 本発表の目的(背景)
- ▶ 講義カリキュラム
 - ▶ 以前のカリキュラムと現在のカリキュラム
- ▶ 実施の結果
 - ▶ 講義の実施結果と所感
- ▶ 今後の展開

本発表の目的

- ▶ 教育プロジェクトの紹介と成果発表
- ▶ GPUについての情報公開の促進
- ▶ GPU教育についての情報公開の促進
- ▶ 公開後の情報の保守管理についてのノウハウの修得

教育プロジェクトの紹介

- ▶ 長岡技術科学大学教育プロジェクト
「ハードウェア指向型IT技術者育成」
- ▶ 高度情報化社会を支えるIT技術者の育成と、世界的に通用する技術者の輩出
- ▶ ソフトウェアだけでなくハードウェアも含めた知識を有した「ハードウェア指向型」の人材育成
- ▶ 産業界が注目しているGPUを教材として、ソフトウェア、ハードウェアに関する知識を教授

教育プロジェクトの紹介

平成23年度	実施検討 GPUクラスタ導入 (M2050 64台)
平成24年度	開講科目, シラバスの確定 模擬授業 (9月, 12月, 3月) 前任者採用
平成25年度	4月 講義開始 8月 <u>GPGPUシンポジウムin長岡技科大</u> 3月 前任者転出
平成26年度	8月 <u>集中講義 (学部4年生対象講義)</u> 10月 <u>発表者着任</u>
平成27年度	

担当して驚いた事

- ▶ GPUに関する興味が全体的に薄い
 - ▶ 知ってはいるが難しそう, 取り組む時間がない
 - ▶ MATLABが高速化されないなら不要とも
- ▶ プログラムが苦手
 - ▶ Linuxに触れたことがない学生も
 - ▶ 学生の知識・技能レベルが不均一
- ▶ 姿勢(技術系大学と称した理由)
 - ▶ ScientistではなくEngineer
 - ▶ ツールは問題解決の手段

授業設計で困った事

- ▶ 適した教科書がない
 - ▶ 「1から10まで」ではなく「0から」
 - ▶ 一時期のGPU熱が冷め、和書の出版が激減
 - ▶ インターネットでも記事の更新が停滞
- ▶ GPUを取り扱う講義はいくつか存在
 - ▶ 参考にしたいが、資料が公開されていない
 - ▶ 対象とする学生像、授業のレベルが異なる
- ▶ もっと情報を公開・交換しませんか？

情報公開と保守管理

▶ 言い出しっぺの法則

▶ 授業の録画, pdf, プログラムソースを公開

▶ CUDA Fortran版も作成中(ソースは完成)

▶ <http://streaming.nagaokaut.ac.jp/gpgpu/index.html>

GPGPU実践プログラミング第8回 総和計算 (高度な最適化)

共有メモリの要素数の決定

- ▶ 配列要素数が64以下(スレッド数が32以下)
 - ▶ 配列外参照を回避するため、共有メモリを大きめに確保
- ▶ 1Warp8スレッド, N=16を想定
- ▶ スレッド数8 (1Warp)
- ▶ 共有メモリの要素数8
 - ▶ Ver. 6までと同じように宣言すると、配列外参照が発生

グローバルメモリから共有メモリへコピー+加算

503 GPGPU実践プログラミング 2015/06/03

0:46:30 / 1:17:32 CC

講義資料(pdf)

[インデックスへ戻る](#)

http://streaming.nagaokaut.ac.jp/gpgpu/gpgpu_programming/2015/gpgpu_programming08.html

情報公開と保守管理

- ▶ 情報公開後の保守管理をどうするか
 - ▶ 動画を撮り直すことはできないが…
 - ▶ 内容の正誤
 - ▶ ソースファイルの公開とライセンス
 - ▶ 教授法や教授内容設計のノウハウ
 - ▶ 教える順序や効果的な例題などの蓄積
- ▶ オープンCAE学会なら情報が交換できるのでは

プロジェクトでの開講科目

▶ GPGPU実践基礎工学(学部3年)

- ▶ GPUのアーキテクチャの基礎を理解
- ▶ CPUの高速化技術をハードウェアの側面から修得
- ▶ GPGPUの基礎知識を体得

▶ GPGPU実践プログラミング(学部4年)

- ▶ CPUとは異なる最適化方法を修得
- ▶ 分野横断的な問題に活用できる汎用的プログラミング能力を体得

プロジェクトでの開講科目

- ▶ 先端GPGPUシミュレーション工学特論(大学院)
 - ▶ 実践的な実装方法を修得
 - ▶ 異分野にわたる高度な専門知識を涵養

- ▶ GPGPU講習会(学内の全教員・学生対象)
 - ▶ 月1回の頻度で開催
 - ▶ 1回で完結
 - ▶ 今年度はCUBLAS, CUSPARSE, Thrustなどライブラリの利用方法や, PyCUDAなど開発環境を重点化

平成25年度のカリキュラム

授業回	実践基礎工学(B3)	実践プログラミング(B4)	先端シミュレーション(M)
1	シミュレーション技術の歴史	計算の高速化手法	GPUプログラム構造の詳細
2	HPCとGPGPUの歴史	MIMDとSIMD	GPUメモリ階層の詳細
3	GPGPUでの応用例(1)	GPUアーキテクチャとプログラム構造	並列計算の効率化
4	GPGPUでの応用例(2)	GPUメモリ階層	並列計算の非効率化への対応
5	CPUでの高速化技術	GPGPUプログラミング環境	GPUでの非効率化の原因
6	プロセッサのアーキテクチャ	開発ツール	GPUとCPUの協調
7	シングルコアとマルチコア	GPU最適化ライブラリ	計算ネックな数値計算
8	プロセッサの高速化技術	パフォーマンス解析ツール	画像処理
9	並列計算の概念	数値計算の特徴	計算機合成ホログラム
10	プロセスとスレッド	行列計算	CFDへの応用(粒子法)
11	GPUのアーキテクチャ	画像処理	メモリネックな数値計算
12	GPGPUのプログラム構造	拡散方程式	FDTD法
13	GPGPUのメモリ階層	N体問題	CFDへの応用(差分法)
14	開発環境(CUDA)	遺伝的アルゴリズム	モンテカルロ法
15	開発環境(OpenCL)	モンテカルロ法	演習

平成26年度のカリキュラム（発表者が担当）

授業回	実践基礎工学(B3)	実践プログラミング(B4)	先端シミュレーション(M)
1	シミュレーション技術の歴史	計算の高速化手法	GPUプログラム構造の詳細
2	HPCとGPGPUの歴史	MIMDとSIMD	GPUメモリ階層の詳細
3	GPGPUでの応用例(1)	GPUアーキテクチャとプログラム構造	並列計算の効率化
4	GPGPUでの応用例(2)	GPUメモリ階層	並列計算の非効率化への対応
5	CPUでの高速化技術	GPGPUプログラミング環境	GPUでの非効率化の原因
6	プロセッサのアーキテクチャ	開発ツール	GPUとCPUの協調
7	シングルコアとマルチコア	GPU最適化ライブラリ	計算ネックな数値計算
8	プロセッサの高速化技術	パフォーマンス解析ツール	画像処理
9	並列計算の概念	数値計算の特徴	計算機合成ホログラム
10	プロセスとスレッド	行列計算	CFDへの応用(粒子法)
11	GPUのアーキテクチャ	画像処理	メモリネックな数値計算
12	GPGPUのプログラム構造	拡散方程式	FDTD法
13	GPGPUのメモリ階層	N体問題	CFDへの応用(差分法)
14	開発環境(CUDA)	遺伝的アルゴリズム	モンテカルロ法
15	開発環境(OpenCL)	モンテカルロ法	演習

↑C言語
↓復習

受講生数の推移（平成25, 26年）

▶ 受講生数

科目名	受講生数	
	平成25年度	平成26年度
GPGPU実践基礎工学	34	31
GPGPU実践プログラミング	17	4(4*)
先端GPGPUシミュレーション工学特論	17	0

*平成25年度にGPU関連講義を受講・単位取得した学生数

▶ 単位取得学生数

科目名	単位取得学生数	
	平成25年度	平成26年度
GPGPU実践基礎工学	31	28
GPGPU実践プログラミング	15	3(3*)
先端GPGPUシミュレーション工学特論	10	0

カリキュラム改訂時に考慮したこと

- ▶ 受講した結果、何かができるようになっている
 - ▶ 苦労が報われ、その先に明確なゴールがある
 - ▶ 応用的な問題が解け、視覚的に確認できる
 - ➡ 非圧縮性流れの差分計算をGPUで実行できる
- ▶ 入門から応用まで一貫している
 - ▶ 授業で紹介するGPUの機能を利用した結果、何がどう変わるかを把握できる
 - ➡ ベクトル和の計算を複数GPUで実行できる
- ▶ 単年で完結する

現在のカリキュラム（平成27年）

授業回	実践基礎工学(B3)	実践プログラミング(B4)	先端シミュレーション(M)
1	ガイダンス	ガイダンス	ガイダンス
2	GPGPUの歴史と応用例	GPUアーキテクチャとプログラム構造	GPUでの並列プログラミング
3	GPUクラスタの利用	GPUクラスタの利用	ThreadとWarp
4	CPUのアーキテクチャ	GPUでの並列プログラミング	GPUのメモリ階層
5	CPUの高速化技術(ハードウェア)	GPUのメモリ階層	様々なメモリの利用
6	CPUの高速化技術(ソフトウェア)	パフォーマンス解析ツール	プログラムの性能評価指針
7	シングルコアとマルチコア	総和計算	総和計算(atomic演算)
8	プロセスとスレッド	総和計算(高度な最適化)	拡散方程式
9	GPUのアーキテクチャ	行列計算	移流方程式
0	GPUプログラムの構造	行列計算(高度な最適化)	Poisson方程式
11	GPUでの並列プログラミング	画像処理	CFDへの応用(流れ関数-渦度法)
12	画像処理	拡散方程式	CFDへの応用(GPUへの移植)
13	GPUのメモリ階層	多粒子の運動	CFDへの応用(高度な最適化)
14	組込開発環境(Jetson TK1)	N体問題	複数GPUの利用
15	開発環境(OpenCL)	GPU最適化ライブラリ	CPUとGPUの協調

↑C言語入門・復習↓

単年での完結性（実践基礎工学）

実践基礎工学(B3)

目標:画像処理の高速化

ガイダンス

GPGPUの歴史と応用例

GPUクラスタの利用

CPUのアーキテクチャ

CPUの高速化技術(ハードウェア)

CPUの高速化技術(ソフトウェア)

シングルコアとマルチコア

プロセスとスレッド

GPUのアーキテクチャ

GPUプログラムの構造

GPUでの並列プログラミング

ゴール→

画像処理

GPUのメモリ階層

組込開発環境(Jetson TK1)

開発環境(OpenCL)

- GPUの応用例について, 画像, 映像で紹介する.
- 話題がCPUにうつる前に, GPUに触れる機会を設ける.

- CPUのハードウェアや進化の歴史について雑学的な内容を紹介する.
- キーワード
フォン・ノイマン型計算機, ムーアの法則, マルチコアCPU, メモリ, キャッシュ, フリンの分類, アムダールの法則, OpenMP, MPI等々

- GPUのアーキテクチャやCUDAについて学習する.
- ベクトル和を例題としてCUDAでの並列化の基本を学習する.
- unsigned char型の配列を画像とみなし, 処理を行うカーネルを作成する.
- 共有メモリやコンスタントメモリを利用する事で実行速度が変わることを学習する.

- 電気系の学生が多いことを考慮し, 研究で利用できそうな組込開発環境について紹介する.

単年での完結性（実践プログラミング）

実践プログラミング(B4)

ガイダンス

GPUアーキテクチャとプログラム構造

GPUクラスタの利用

GPUでの並列プログラミング

GPUのメモリ階層

パフォーマンス解析ツール

総和計算

総和計算(高度な最適化)

行列計算

行列計算(高度な最適化)

画像処理

拡散方程式

多粒子の運動

N体問題

GPU最適化ライブラリ

目標: 様々な問題に対するプログラミング能力の修得

- GPUを利用する上で必要になる内容を学習する.
 - GPUのアーキテクチャやCUDAについて紹介する.
 - ベクトル和, 移動平均, 差分法, を例題としてCUDAでの並列化の基本を学習する.
 - 共有メモリやコンスタントメモリの利用方法を学習する.
 - 実践基礎工学を受講した学生にとっては, 6割程度が復習, 4割程度が新規の内容となる.

- プロファイラの使い方を学習する. 以降の回で利用する.

- GPUで高い性能が発揮でき, かつGPU特有の最適化方法が必要な問題を体験する.
 - NVIDIAの資料よりも問題を細分化(7→9段階)
 - 共有メモリのバンクコンフリクトなどをプロファイルする.
 - 複数の最適化を行い, 高速化される要因を考察する.

- Euler系の問題を取り扱う.
- 空間微分の離散化の精度についても言及する.

- Lagrange系の問題を取り扱う.
- 時間積分の離散化の精度についても言及する.

- CUDA付属のライブラリの利用方法を学習する.

メイン

単年での完結性（先端シミュレーション）

先端シミュレーション(M)

目標: 流体の数値シミュレーション, 複数GPUの利用

ガイダンス

GPUでの並列プログラミング

ThreadとWarp

GPUのメモリ階層

様々なメモリの利用

プログラムの性能評価指針

総和計算(atomic演算)

拡散方程式

移流方程式

Poisson方程式

CFDへの応用(流れ関数-渦度法)

ゴール→ CFDへの応用(GPUへの移植)

CFDへの応用(高度な最適化)

ゴール→ 複数GPUの利用

CPUとGPUの協調

- GPUを利用する上で必要になる内容を学習する.
 - GPUのアーキテクチャやCUDAについて紹介する.
 - ベクトル和を例題としてCUDAでの並列化の基本とメモリ階層の基本を学習する.

- 共有メモリ, コンスタントメモリ, ページロックホストメモリの利用方法を学習する.

- Flop/byteについて学習し, ベクトル和の実効性能を評価する.

- 総和計算の方法を学習する.(共役勾配法で使用)
- 数値計算の基礎的なプログラムをCUDAで作成する.
 - 1次元2階微分および2次元拡散方程式
 - 2次元移流方程式および2次元移流拡散方程式
 - Poisson方程式(共役勾配法を利用)

- 移流拡散方程式, Poisson方程式を基にして, 流れ関数-渦度法を用いて流体のシミュレーションができることを学習し, 実際にプログラムを作成する.

- 複数GPUを使ってベクトル和を実行する方法を学習する.
- ストリームについて学習する.

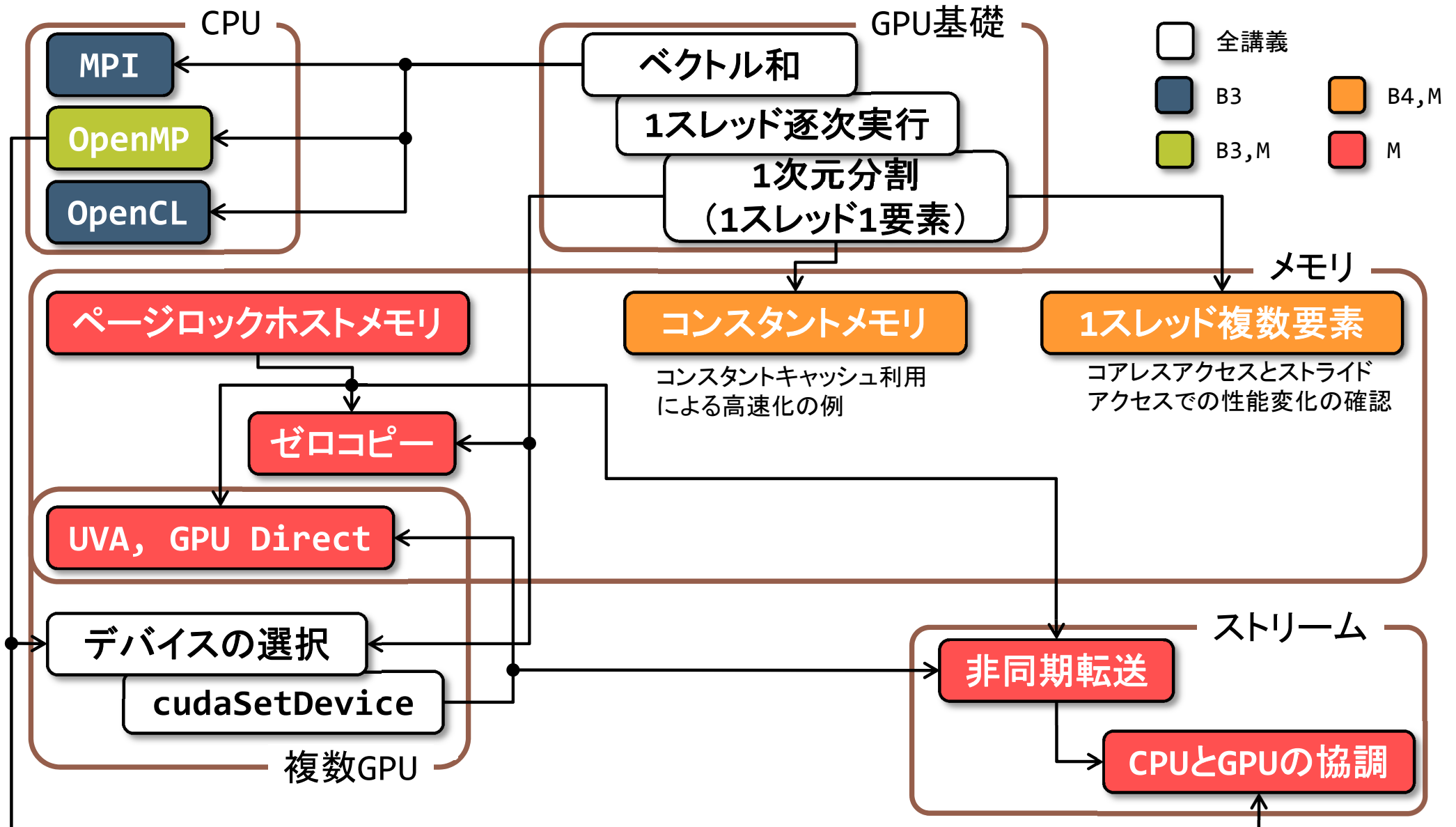
各講義の連携

授業回	実践基礎工学(B3)	実践プログラミング(B4)	先端シミュレーション(M)
1	ガイダンス	→ガイダンス	→ガイダンス
2	GPGPUの歴史と応用例	→GPUアーキテクチャとプログラム構造	→GPUでの並列プログラミング
3	GPUクラスタの利用	→GPUクラスタの利用	ThreadとWarp
4	CPUのアーキテクチャ	→GPUでの並列プログラミング	→GPUのメモリ階層
5	CPUの高速化技術(ハードウェア)	→GPUのメモリ階層	→様々なメモリの利用
6	CPUの高速化技術(ソフトウェア)	パフォーマンス解析ツール	プログラムの性能評価指針
7	シングルコアとマルチコア	総和計算	→総和計算(atomic演算)
8	プロセスとスレッド	総和計算(高度な最適化)	→拡散方程式
9	GPUのアーキテクチャ	行列計算	移流方程式
0	GPUプログラムの構造	行列計算(高度な最適化)	Poisson方程式
11	GPUでの並列プログラミング	→画像処理	CFDへの応用(流れ関数-渦度法)
12	画像処理	拡散方程式	CFDへの応用(GPUへの移植)
13	GPUのメモリ階層	多粒子の運動	CFDへの応用(高度な最適化)
14	組込開発環境(Jetson TK1)	N体問題	複数GPUの利用
15	開発環境(OpenCL)	GPU最適化ライブラリ	CPUとGPUの協調

ベクトル和の計算を複数GPUで実行するまでの課程

授業回	実践基礎工学(B3)	実践プログラミング(B4)	先端シミュレーション(M)
1	ガイダンス	ガイダンス	ガイダンス
2	GPGPUの歴史と応用例	GPUアーキテクチャとプログラム構造	GPUでの並列計算
3	GPUクラスタの利用	GPUクラスタの利用	ThreadとWarp
4	CPUのアーキテクチャ	GPUでの並列プログラミング	GPUのメモリ階層
5	CPUの高速化技術(ハードウェア)	GPUのメモリ階層	様々なメモリの利用
6	CPUの高速化技術(ソフトウェア)	パフォーマンス解析ツール	プログラムの性能評価指針
7	シングルコアとマルチコア	総和計算	総和計算(atomic演算)
8	プロセスとスレッド	総和計算(高度な最適化)	拡散方程式
9	GPUのアーキテクチャ	行列計算	移流方程式
10	GPUプログラムの構造	行列計算(高度な最適化)	Poisson方程式
11	GPUでの並列プログラミング	画像処理	CFDへの応用(流れ関数-渦度法)
12	画像処理	拡散方程式	CFDへの応用(GPUへの移植)
13	GPUのメモリ階層	多粒子の運動	CFDへの応用(高度な最適化)
14	組込開発環境(Jetson TK1)	N体問題	複数GPUの利用
15	開発環境(OpenCL)	GPU最適化ライブラリ	CPUとGPUの協調

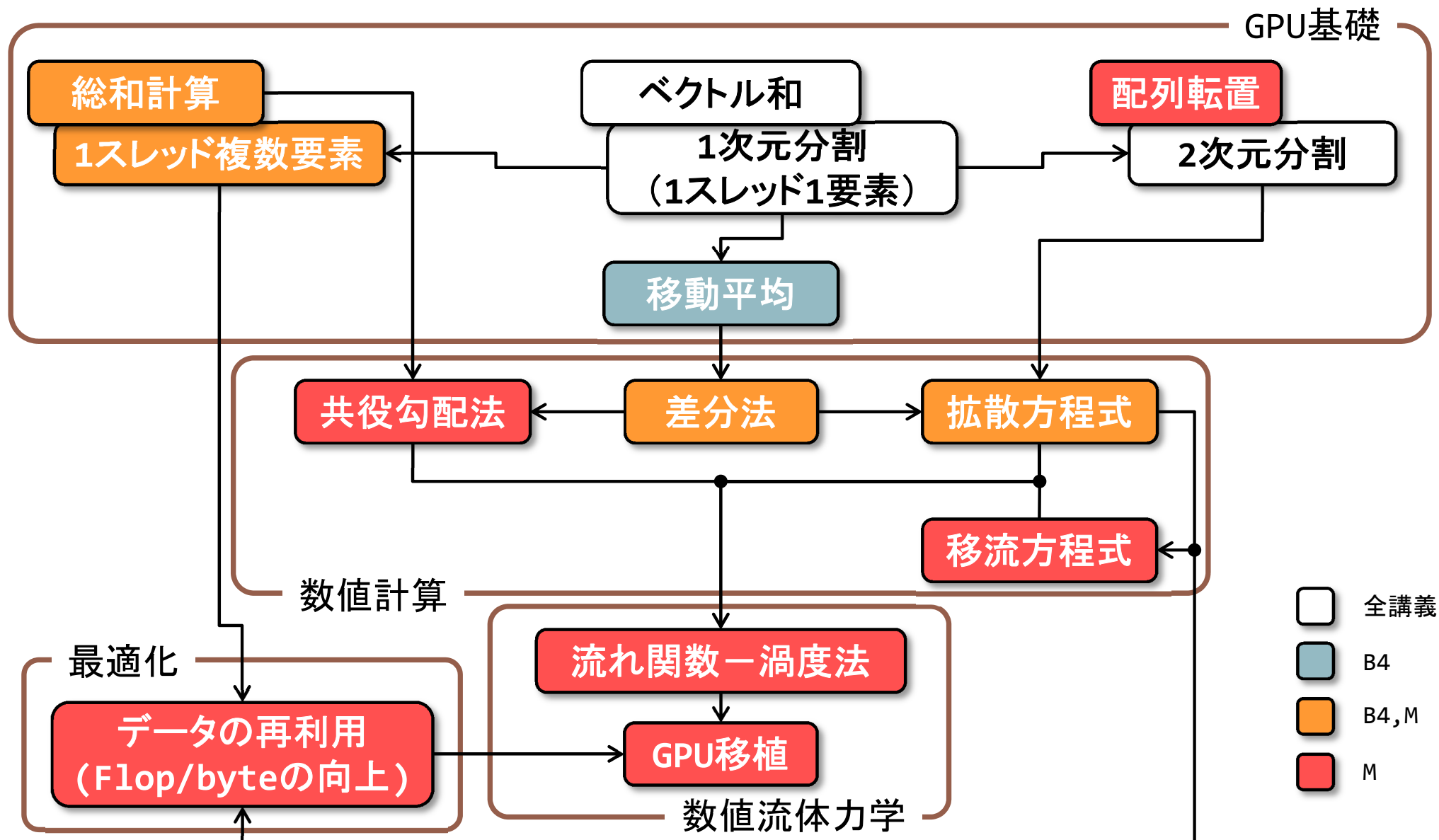
ベクトル和の計算を複数GPUで実行するまでの課程



非圧縮性流れの計算をGPUで実行するまでの課程

授業回	実践基礎工学(B3)	実践プログラミング(B4)	先端シミュレーション(M)
1	ガイダンス	ガイダンス	ガイダンス
2	GPGPUの歴史と応用例	GPUアーキテクチャとプログラム構造	GPUでの並列計算
3	GPUクラスタの利用	GPUクラスタの利用	ThreadとWarp
4	CPUのアーキテクチャ	GPUでの並列プログラミング	GPUのメモリ階層
5	CPUの高速化技術(ハードウェア)	GPUのメモリ階層	様々なメモリの利用
6	CPUの高速化技術(ソフトウェア)	パフォーマンス解析ツール	プログラムの性能評価指針
7	シングルコアとマルチコア	総和計算	総和計算(atomic演算)
8	プロセスとスレッド	総和計算(高度な最適化)	拡散方程式
9	GPUのアーキテクチャ	行列計算	移流方程式
10	GPUプログラムの構造	行列計算(高度な最適化)	Poisson方程式
11	GPUでの並列プログラミング	画像処理	CFDへの応用(流れ関数-渦度法)
12	画像処理	拡散方程式	CFDへの応用(GPUへの移植)
13	GPUのメモリ階層	多粒子の運動	CFDへの応用(高度な最適化)
14	組込開発環境(Jetson TK1)	N体問題	複数GPUの利用
15	開発環境(OpenCL)	GPU最適化ライブラリ	CPUとGPUの協調

非圧縮性流れの計算をGPUで実行するまでの課程



受講生数の推移（平成25~27年）

▶ 受講生数

科目名	受講生数		
	平成25年度	平成26年度	平成27年度
GPGPU実践基礎工学	34	31	100 ←!?
GPGPU実践プログラミング	17	4(4*)	9(8*)
先端GPGPUシミュレーション工学特論	17	0	16(2**+9*)

*以前にGPU関連講義を受講・単位取得した学生数

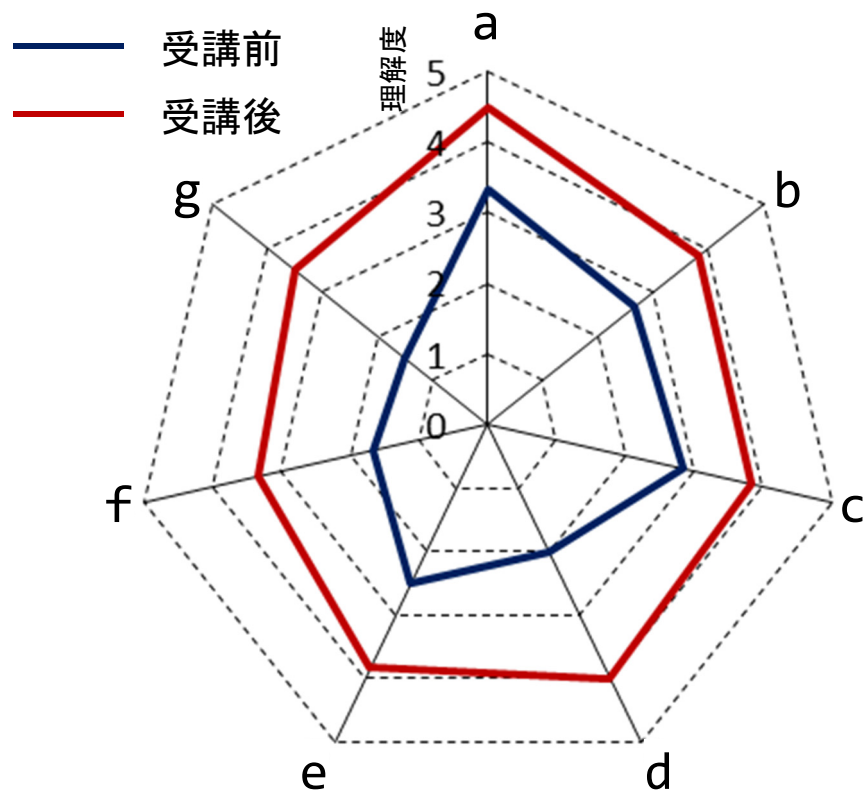
**全てのGPU関連講義を受講・単位取得した学生数

▶ 単位取得学生数

科目名	単位取得学生数		
	平成25年度	平成26年度	平成27年度
GPGPU実践基礎工学	31	28	開講中
GPGPU実践プログラミング	15	3(3*)	8(7*)
先端GPGPUシミュレーション工学特論	10	0	14(2**+9*)

GPGPU実践プログラミングの実施結果

- ▶ 受講学生数 9人 8人が平成26年度の講義を受講, 単位取得
- ▶ 単位取得学生数 8人
- ▶ 達成度自己評価(平均値)



項目	目標
a	GPUのアーキテクチャを理解し, CPUのアーキテクチャとの違いを説明できる.
b	GPGPUのプログラム構造を理解し, 並列化の概念をCPUの並列化と対応付けて説明できる.
c	GPGPUのメモリ階層を理解し, その特性を説明できる.
d	GPGPUでの高速化手法を理解し, 適切なアルゴリズムを選択し, その理由をアーキテクチャと関連付けて説明できる.
e	数値計算をGPUへ実装し, その性能を評価できる.
f	各種ライブラリの使用方法を取得し, 実際にプログラムを作成し, その性能を評価できる.
g	パフォーマンス解析ツールの使用方法を習得し, プログラムの性能を評価し, その情報を基に性能を改善できる.

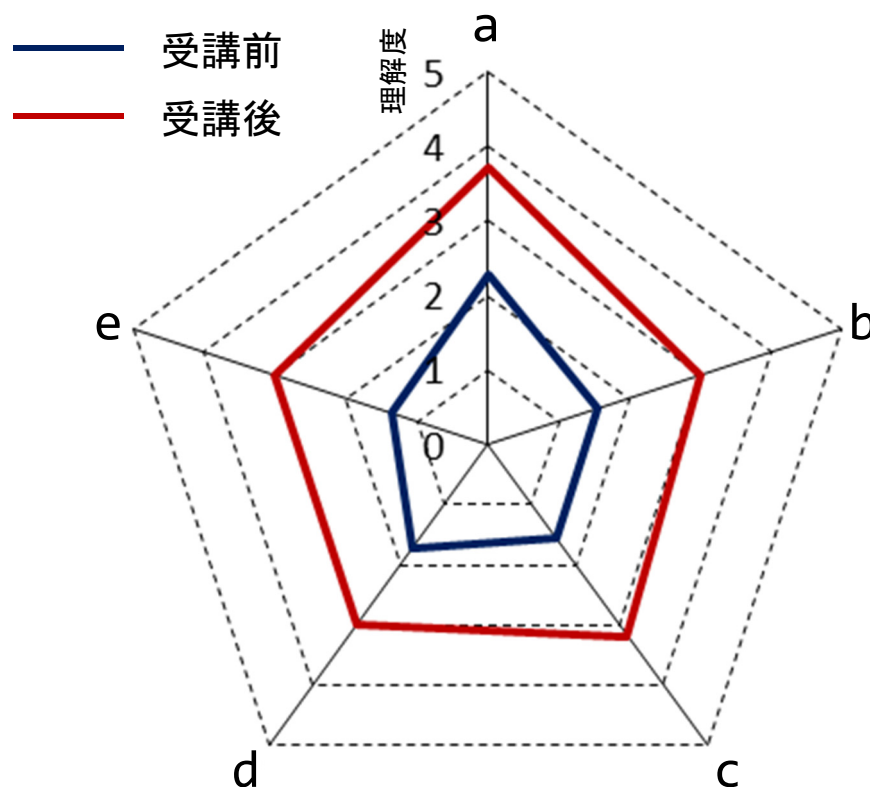
GPGPU実践プログラミングの実施結果

▶ 問題点

- ▶ レポートの量が少ない(2題のみ)
 - ▶ 「GPUの基本的な使い方」と「総和計算」
 - ▶ 高度なチューニングを課題として出すには難易度が高いと判断し、講義での説明に重点を置いた
 - ▶ 実践なのだからもっと課題をやりたかったという意見があった
- ▶ 問題をいくつも紹介するのではなく、問題を限定した方が効果的かもしれない

先端GPGPUシミュレーション工学特論の実施結果

- ▶ 受講学生数 16人 2人が平成25,26年度の講義を受講, 単位取得, 9人が平成25年度の講義を受講, 単位取得
- ▶ 単位取得学生数 14人
- ▶ 達成度自己評価(平均値)



項目	目標
a	産業界におけるGPUの役割を説明できる.
b	GPUプログラミングでのメモリの最適な使用方法を理解し, 適切なメモリを選択できる.
c	分岐条件におけるパフォーマンス低下の理由を理解し, 回避方法や適切な実装方法を説明できる.
d	数値計算の高速化手法を修得し, プログラムを作成して実行し, その性能を評価できる.
e	CPUとGPUの協調プログラミングを修得し, プログラムを作成することができる.

先端GPGPUシミュレーション工学特論の実施結果

▶ 問題点

- ▶ レポートの量が多い(5題)
 - ▶ ベクトル和の計算
 - ▶ 内積の計算
 - ▶ 移流拡散方程式の計算
 - ▶ Poisson方程式の求解(共役勾配法)
 - ▶ キャビティ流れの計算
- ▶ 一つの項目が理解できないと、その後連鎖的にできなくなってしまう

GPGPU実践基礎工学の経過

- ▶ 受講学生数 100人
 - ▶ うち2割がC言語未経験者
 - ▶ 授業毎に20人程度の欠席者
- ▶ 予想単位取得者数 70人程度

GPGPU実践基礎工学の経過

- ▶ 問題点はまだ不透明
 - ▶ 課題として4題を課す予定
 - ▶ 授業で言及した内容+少し発展的な内容
 - ▶ 発展的な内容には, その次の授業で言及
 - ▶ CUDAやGPUの制約に抵触する条件での実行も指示
- ▶ 変更できるなら
 - ▶ CPUに関係する5回分を削り, GPUに充てたい
 - ▶ 受講生の人数を制限する, もしくはC言語の事前履修を必須にする

所感

- ▶ 興味のある学生が集まる
 - ▶ 授業態度は良好
- ▶ 課題の出来はそこそこ
 - ▶ 課題を完璧にこなす学生から、エラーを解決できなかった学生まで様々
- ▶ わからなくても質問には来ない
 - ▶ 授業前後に課題の様子を尋ねると効果的

所感

- ▶ とにかくデバッグができない
 - ▶ 授業中にデバッグ方法の説明ができなかった
 - ▶ 応用問題になるほど原因の特定が困難
 - ▶ 言語, アルゴリズム, 並列化, ハードウェア等々
 - 計算の安定性や収束性はGPUとは別問題
 - ▶ 自力でバグを特定・修正できるようにするには
- ▶ CUDA Debug Drill(仮称)があると有益？
 - ▶ 学生はドリルでデバッグの技術を修得
 - ▶ 教員はデバッグ対応に時間を割かなくてすむ

所感

- ▶ GPUの知名度は年々上昇
 - ▶ 高専の先生からやっておいた方がいいと言われた
 - ▶ インターンシップ先で使っているのを見た
 - ▶ 機械学習に興味がある等々
- ▶ プログラムを全く知らない人にも浸透
 - ▶ 独学はまだまだハードルが高い
- ▶ 大学等でカリキュラムに組み込む場合
 - ▶ 授業対象の学生像を明確に
 - ▶ 授業内容に応じた予備知識の選定が不可欠

今後の展開

- ▶ 継続的な情報発信
 - ▶ GPUの世代更新への対応
 - ▶ GPU教育についての情報公開
 - ▶ 間違い, 不正確な内容への対応
 - ▶ ソースコードの保守管理

今後の展開

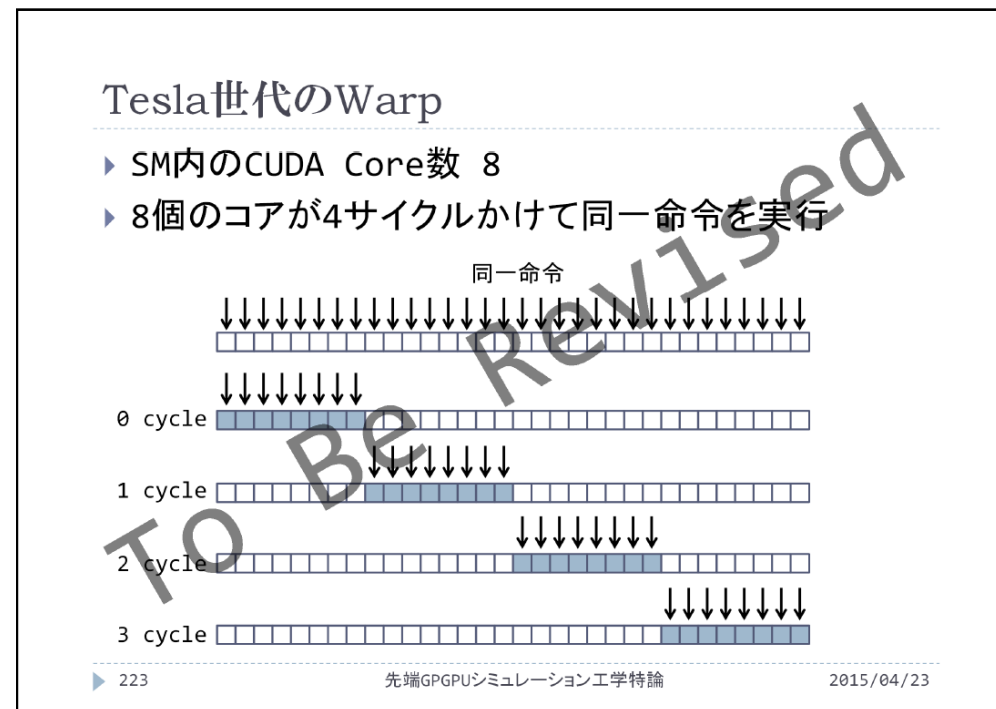
- ▶ GPUの世代更新への対応
 - ▶ 個人で対応するのは半ばあきらめ気味
- ▶ 実際には
 - ▶ 世代更新の度にシステムを更新しない
 - ▶ 主に使う世代を対象にしていればよいか？
- ▶ 最新のGPUを入手した人が情報を公開してくれるよう機運が高まってくれと嬉しい

今後の展開

- ▶ GPU教育についての情報公開
 - ▶ 大学の事情が強く反映されてしまう
- ▶ ある一つの例として公開
 - ▶ 何かが誰かの参考になれば
- ▶ 例) [東京工業大学GPUコンピューティング](#)
 - 長岡技科大の学生には敷居が高い
- ▶ 例) [CFD Python:12 steps to Navier-Stokes](#)
 - CFD(特に差分法)の経験者がPythonを修得するには効果的だが、何も知らない学生には…

今後の展開

- ▶ 間違い, 不正確な内容への対応
 - ▶ 誰が正当性をチェックするか
 - ▶ 指摘を受けた際にどのように対応するべきか
 - ▶ 現状は要改訂として公開中



http://streaming.nagaokaut.ac.jp/gpgpu/advanced_gpgpu/2015/advanced_gpgpu03.pdf

今後の展開

- ▶ ソースコードの保守管理
 - ▶ 授業で用いた全ソースコードをオープンソースで公開する予定
- ▶ どこで公開するか
- ▶ どのライセンスで公開するか
- ▶ 誰に向けて公開するか
 - ▶ コメントの内容, 記述量, 言語, エンコード

まとめ

- ▶ プロジェクトの概要を紹介
- ▶ GPU関連講義のカリキュラムを紹介
 - ▶ 変更した背景
 - ▶ 授業のゴールと各回の連携
 - ▶ 実施結果
- ▶ 今後の展開(情報公開)について
 - ▶ オープンソースに造詣の深い皆様に教えを請いたい