担当
- 大島聡史（助教） ohshima@cc.u-tokyo.ac.jp
- 星野哲也（助教） hoshino@cc.u-tokyo.ac.jp

質問やサンプルプログラムの提供についてはメールでお問い合わせください

第67回お試しアカウント付き並列プログラミング講習会「GPUプログラミング入門」

2016年10月17日（月）

東京大学情報基盤センター

スケジュール

- 09：30 - 10：00　受付
- 10：00 - 12：00　HA-PACSログイン、GPU入門
- 13：30 - 15：00　OpenACC入門
- 15：15 - 16：45　OpenACC最適化入門と演習
- 17：00 - 18：00　OpenACCの活用（CUDA連携とライブラリの活用）

はじめに

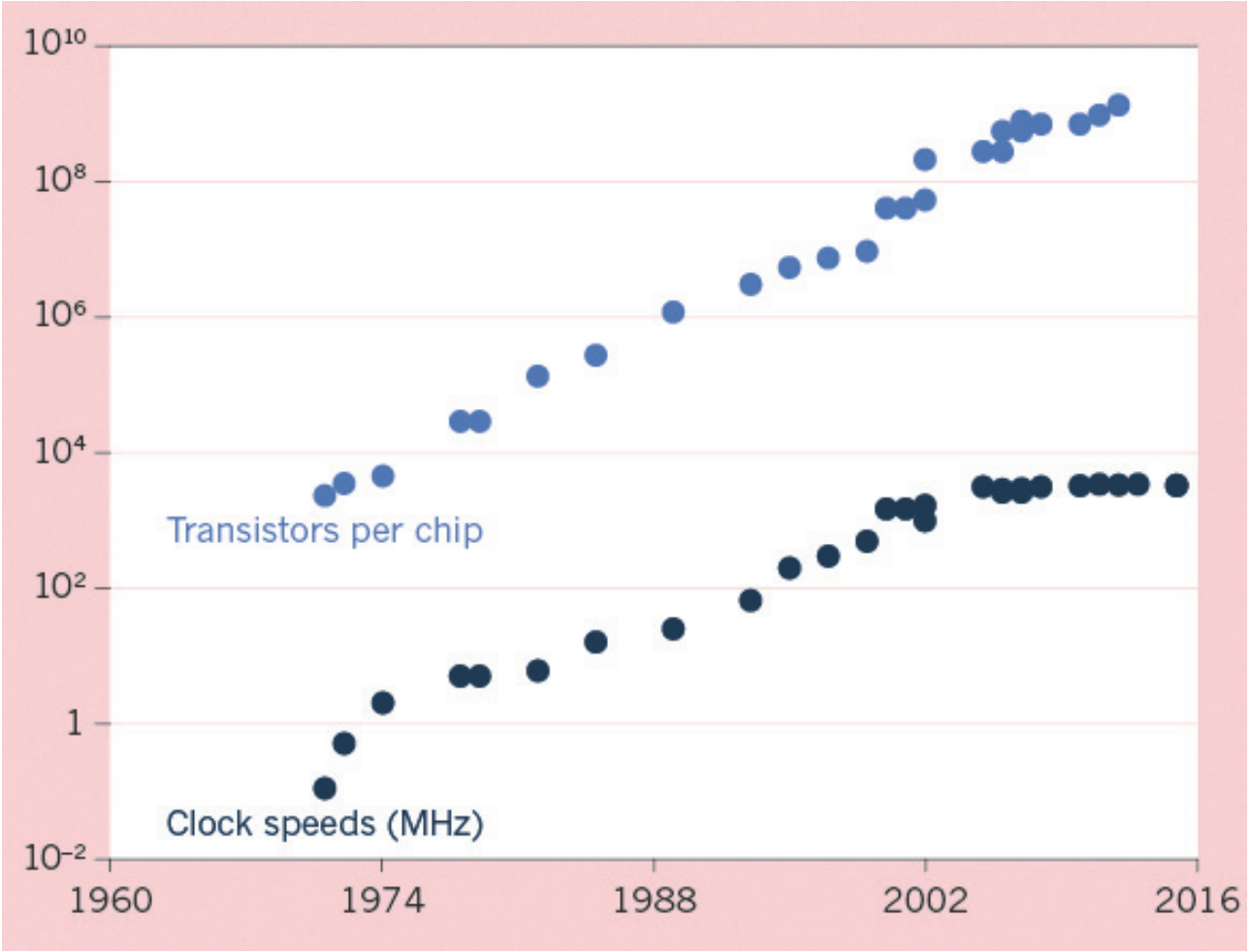
GPUについて
GPUスパコン事情
Reedbushシステムの紹介

多様化する並列計算環境

- 現在のHPC・計算機科学・計算科学分野では様々な並列計算ハードウェアが利用されている
  - マルチコアCPU：複数の計算コアを1つのチップにまとめたCPU
    - 代表例：Intel Xeon / Core, AMD Opteron/FX, IBM POWER, FUJITSU SPARC64, ARM Cortex
      - サーバ向けでは1999年POWER4、PC向けでは2005用Dual-Core Opteron/AthlonX2が初出と思われる
  - メニーコアプロセッサ：マルチコアCPUよりも多数の計算コアを搭載
    - 代表例：Intel Xeon Phi, Sun Niagara, PEZY PEZY-1/SC
      - 明確に何コア以上がメニーコアという定義が有るわけではない
  - GPU：画像処理用HWに端を発するメニーコアプロセッサ
    - 代表例：NVIDIA Tesla/GeForce, AMD FirePro/Radeon
  - FPGA：プログラミングにより回路構成を変更可能なプロセッサ
    - 代表例：Xilinx Virtex, Altera Stratix

背景：計算ハードウェアの性能向上

- ムーアの法則に支えられたCPUの性能向上が終わりつつある
  - 微細化によるチップあたりトランジスタ数の向上
  - クロック周波数の向上
    ↓
  - 消費電力や発熱が
    問題となり頭打ち
    ↓
  - マルチコア化、
    メニーコア化による
    並列演算性能の向上へ



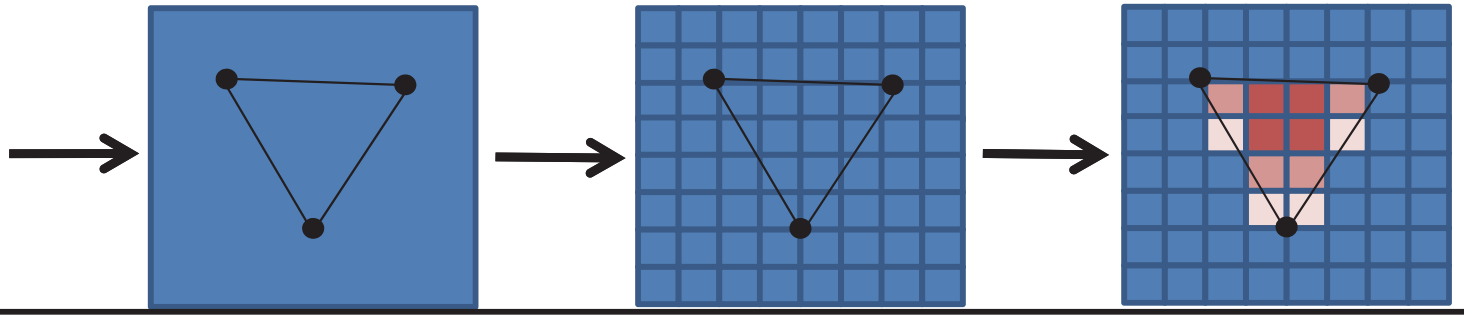
出展 : The chips are down for Moore's law : Nature News & Comment
http://www.nature.com/news/the-chips-are-down-for-moore-s-law-1.19338

GPU (Graphics Processing Unit)

- 画像処理用のハードウェア
  - 高速・高解像度描画、3D描画処理（透視変換、陰影・照明）、画面出力
- CPUやマザーボードに組み込まれたチップとして、また拡張スロットに搭載するビデオカードとして広く利用される
- GPUに求められる処理が並列計算に適した処理であったため、CPUに先んじて並列化による高性能化が進んだ
- 性能・機能の向上に伴い2000年代後半から汎用演算への活用が進み、GPGPUやGPUコンピューティングと呼ばれる
  (General-Purpose computation on GPUs)

※参考：3次元画像描画の手順

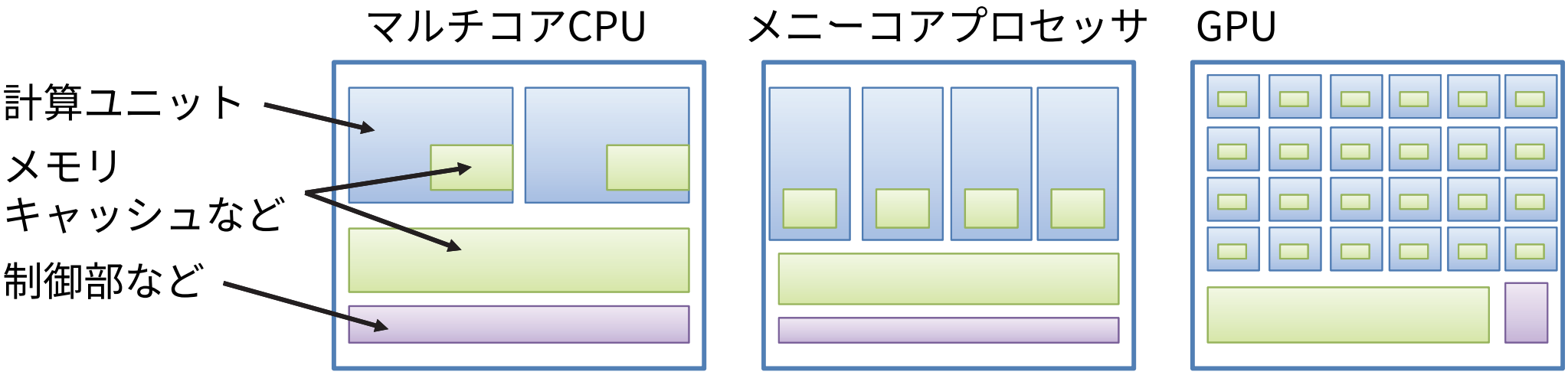
① (2, 2)
② (8, 3)
③ (5, 7)



オブジェクト単位、頂点単位、ピクセル単位で並列処理が可能→並列化により高速化しやすい

GPUの特徴（1/2）

- ハードウェアの構成バランスの違い（イメージ）
  - 限られたトランジスタを主に何に用いるか



  - 多数の計算ユニットを搭載し全体として高性能を得ることを重視
  - （この図ではわからないが）総メモリ転送性能も重視している

GPUの特徴（2/2）

- CPUとは異なる特徴を持つ
  - 非常に多くの（1000以上）の計算ユニットを搭載、計算ユニット単体の性能は低い
    - CPUよりも低い・貧弱な動作周波数、キャッシュ、分岐、……
  - 計算コアが完全に個別には動けない
    - 32個などの単位でスケジューリング、SIMD演算器が大量に搭載されたイメージ
  - 浅めのキャッシュ階層、複数階層のメモリ
- 特定のアプリケーションでは非常に高い性能
  - 最近はビッグデータや機械学習の分野で有用なため注目が高まっている
- CPUとは異なるプログラミング・最適化の知識と技術が必要
  - 本講習会がその手助け・入り口となることを期待します

GPUスパコン

- GPU等のアクセラレータを搭載したスパコンの普及
  - TOP500リスト 2016年6月版
    - TOP20中4システム、TOP500中67システムがGPUスパコン

順位	名称 設置機関	開発ベンダー 国	Rmax Rpeak	構成	コア数 アクセラレータコア数
1	Sunway TaihuLight National Supercomputing Center in Wuxi	NRCPC China	93014594 125435904	Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway	10649600 0
2	Tianhe-2 (MilkyWay-2) National Super Computer Center in Guangzhou	NUDT China	33862700 54902400	TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P	3120000 2736000
3	Titan DOE/SC/Oak Ridge National Laboratory	Cray Inc. US	17590000 27112550	Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x	560640 261632
4	Sequoia DOE/NNSA/LLNL	IBM US	17173224 20132659.2	BlueGene/Q, Power BQC 16C 1.60 GHz, Custom	1572864 0
5	K computer RIKEN Advanced Institute for Computational Science (AICS)	Fujitsu Japan	10510000 11280384	SPARC64 VIIIfx 2.0GHz, Tofu interconnect	705024 0
6	Mira DOE/SC/Argonne National Laboratory	IBM US	8586612 10066330	BlueGene/Q, Power BQC 16C 1.60GHz, Custom	786432 0
7	Trinity DOE/NNSA/LANL/SNL	Cray Inc. US	8100900 11078861	Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect	301056 0
8	Piz Daint Swiss National Supercomputing Centre (CSCS)	Cray Inc. Switzerland	6271000 7788852.8	Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x	115984 73808
9	Hazel Hen HLRS - Höchstleistungsrechenzentrum Stuttgart	Cray Inc. Germany	5640170 7403520	Cray XC40, Xeon E5-2680v3 12C 2.5GHz, Aries interconnect	185088 0
10	Shaheen II King Abdullah University of Science and Technology	Cray Inc. Saudi Arabia	5536990 7235174	Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Aries interconnect	1966080 0

東大情報基盤センター
2016年4月時点で運用中のスパコン

Oakleaf-FX (通常ジョブ用)
(Fujitsu PRIMEHPC FX10)

Total Peak performance	: 1.13 PFLOPS
Total number of nodes	: 4800
Total memory	: 150 TB
Peak performance / node	: 236.5 GFLOPS
Main memory per node	: 32 GB
Disk capacity	: 1.1 PB + 2.1 PB
SPARC64 Ixfx 1.84GHz	

Oakbridge-FX (長時間ジョブ用)
（Fujitsu PRIMEHPC FX10 ）

Total Peak performance	: 136.2 TFLOPS
Total number of nodes	: 576
Total memory	: 18 TB
Peak performance / node	: 236.5 GFLOPS
Main memory per node	: 32 GB
Disk capacity	: 147TB + 295TB
SPARC64 Ixfx 1.84GHz	

Yayoi
(Hitachi SR16000/M1)

Total Peak performance	: 54.9 TFLOPS
Total number of nodes	: 56
Total memory	: 11200 GB
Peak performance / node	: 980.48 GFLOPS
Main memory per node	: 200 GB
Disk capacity	: 556 TB
IBM POWER 7 3.83GHz	

Total Users > 2,000

東大情報基盤センター
2016年度内に稼働を開始したスパコン

Reedbush
(データ解析・シミュレーション融合スーパーコンピュータシステム)

- Reedbush-U (CPU only) と Reedbush-H (with GPU) からなる
- Reedbush-U
  - 508.03 TFlops
  - 2016/7/1 試験運用開始
  - 2016/9/1 正式運用開始
- Reedbush-H
  - 1297.15-1417.15 TFlops
  - 2017/3/1 試験運用開始

Oakforest-PACS

- 最先端共同HPC基盤施設(JCAHPC)により導入
  - JCAHPCは東大-筑波大の共同組織
- ピーク性能:25PFFLOPS
  - 8,208 Intel Xeon Phi (KNL)
  - 日本最速になる予定
- 2016/10/1 試験運用開始

JCAHPC

Reedbush (1/2)

- システム構成・運用：SGI
- Reedbush-U (CPU only)
  - Intel Xeon E5-2695v4 (Broadwell-EP, 2.1GHz 18core) x 2ソケット (合計1.210 TF), 256 GiB (153.6GB/sec)
  - InfiniBand EDR, Full bisection BW Fat-tree
  - システム全系: 420 ノード, 508.0 TF
- Reedbush-H (with GPU)
  - CPU・メモリ：Reedbush-U と同様
  - NVIDIA Tesla P100 (Pascal世代 GPU)
    - (4.8-5.3TF, 720GB/sec, 16GiB) x 2 / ノード
  - InfiniBand FDR x 2ch, Full bisection BW Fat-tree
  - 120 ノード, 145.2 TF(CPU)+ 1.15~1.27 PF(GPU)= 1.30~1.42 PF

“Reedbush”って何？

Blaise Pascal
(1623–1662)

- L'homme est un roseau pensant.
- Man is a thinking reed.
- 人間は考える葦である

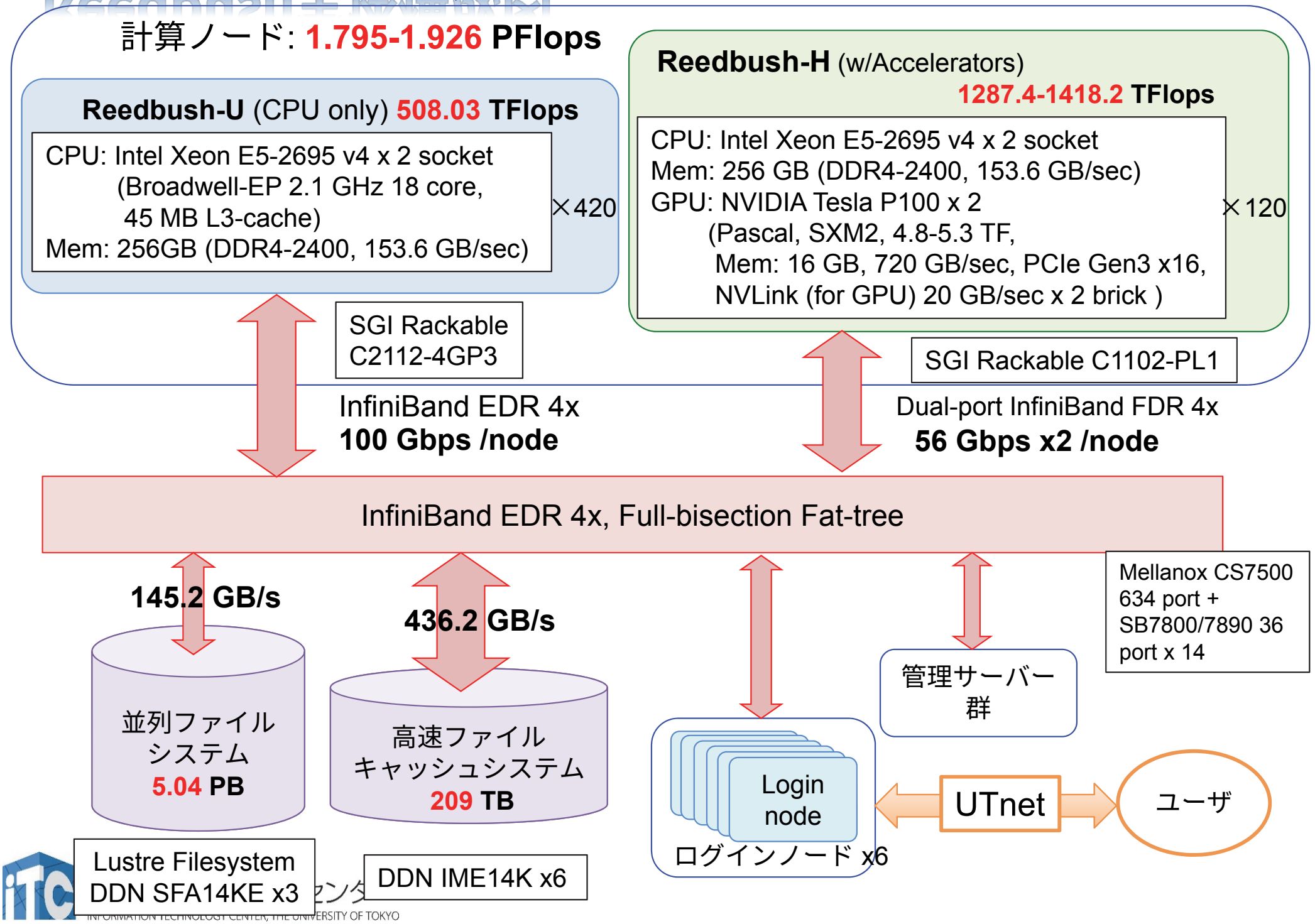
Pensées (Blaise Pascal)

Reedbush (2/2)

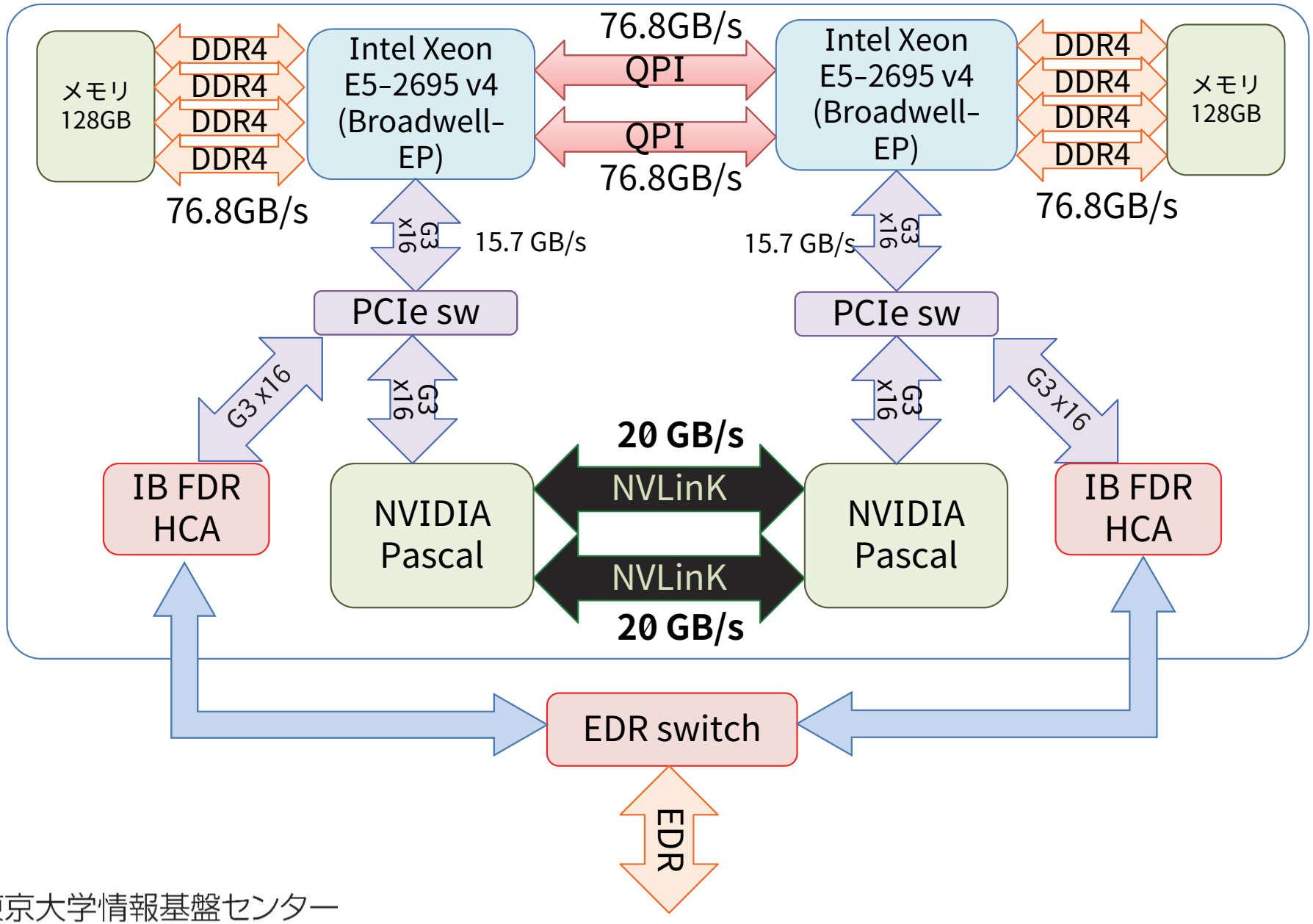
- ストレージ/ファイルシステム
  - 並列ファイルシステム (Lustre)
    - 5.04 PB, 145.2 GB/sec
  - 高速ファイルキャッシュシステム: Burst Buffer (DDN IME (Infinite Memory Engine))
    - SSD: 209.5 TB, 450 GB/sec
- 電力, 冷却, 設置面積
  - 空冷, 378 kVA(冷却除く)、< 90 m^2
- データ解析、ディープラーニング向けソフトウェア・ツールキット
  - OpenCV, Theano, Anaconda, ROOT, TensorFlow, Torch, Caffe, Chainer, GEANT4

詳しくはWebをご参照ください
http://www.cc.u-tokyo.ac.jp/system/reedbush/

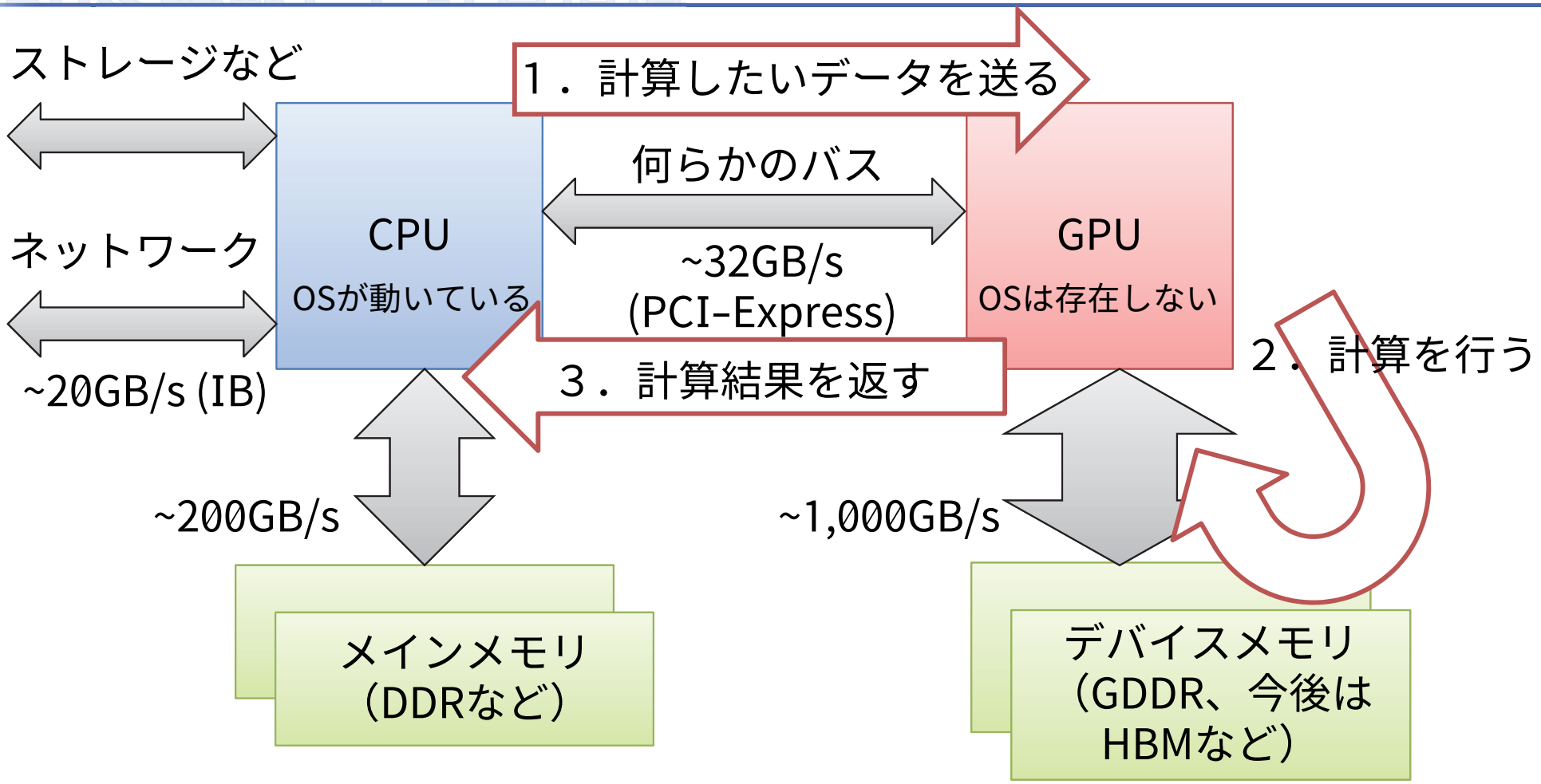
Reedbush全体構成図



Reedbush-Hノードのブロック図



GPUを搭載した計算環境



- CPUやGPUが複数搭載されていることもあるが、ここでは割愛
- GPUを使う為には１．２．３．を考える（実装する）必要がある
- デバイス内外のデータ転送速度差が大きいことから、対象とする
  プロセッサ内で計算が完結していることが望ましいことがわかる

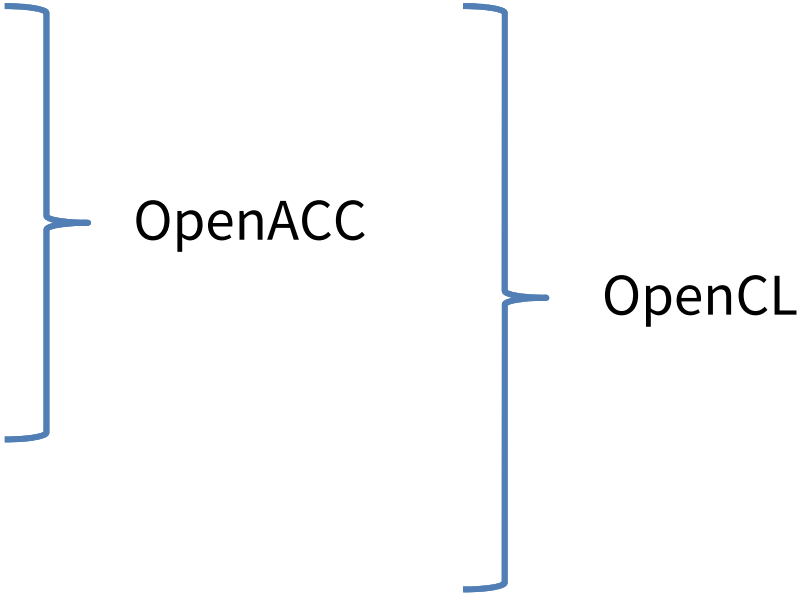
GPUを利用する方法

1. GPUに対応したソフトウェア（アプリケーション）を使う
   - GPU上で行われる計算自体は実装しない、基本的にGPUの知識は不要
   - 存在するものしか使えない、手持ちのプログラムには適用不能
2. （GPUに対応していないプログラムから）GPUに対応したライブラリやフレームワークを使う
   - GPU上で行われる計算自体は実装しない、基本的にGPUの知識は不要
   - 対象分野における共通のAPIが存在しGPU化されていれば恩恵は大
     - BLASなどの数値計算ライブラリ、ビッグデータ・機械学習系のライブラリ・フレームワークなど
3. GPU上で行われる計算そのものを実装する
   - １や２で用いるソフトウェア・ライブラリ等そのものを作る
   - GPUに関する知識が必要
   - 手持ちのプログラム・独自のプログラムをGPU化できる

本講習会の対象

並列計算環境を利用するためのソフトウェア

- 主な開発環境（プログラミング言語など、特に並列化に用いるもの）
  - CPU/MIC
    - MPI, OpenMP
    - (pthread, Cilk+, TBB, …)
  - GPU
    - CUDA, DirectCompute
  - FPGA
    - Verilog HDL



  - 従来は個別のものが使われていたが、近年では共通化も進みつつある
    - 習得が大変、移植が大変という利用者の声が反映されている

本講習会で用いるハードウェアと開発環境

- 対象とするGPU：NVIDIA Tesla M2090
  - Tesla：NVIDIA社が開発しているGPUシリーズの1つ、HPC向け
    - コンシューマ向けのGeForceシリーズと比べて、倍精度演算が高速、ECC対応メモリを搭載、などの違いがある
    - M2090は2011年に発売されたGPU、アーキテクチャ名はFermi。やや古いGPUではあるが、GPUプログラミングの基礎を学ぶには十分。
- 対象とするGPUプログラミング開発環境：主にOpenACC
  - OpenACC：指示文を用いて並列化を行うプログラミング環境。C言語とFortranの両方の仕様が定められている。PGIコンパイラなど幾つかのコンパイラが対応。（GPUが主なターゲットだが）GPU専用言語ではない。
  - CUDA (Compute Unified Device Architecture)：NVIDIAのGPU向け開発環境。C言語版はCUDA CとしてNVIDIAから、Fortran版はCUDA FortranとしてPGI(現在はNVIDIAの子会社)から提供されている。
  - （特に単純なプログラムにおいては）OpenACCでもCUDAでも同様の性能が出ることもあるが、一般的にはCUDAの方が高速

参考：GPUプログラミングの歴史

- 2004年頃：GPU上である程度プログラミングが可能となった
  - 「プログラマブルシェーダ」が登場
  - それ以前は機能の切替程度しかできなかった
  - 主に画像処理のためのプログラミングであり、様々な汎用計算アルゴリズムを実装するのに十分なものとは言えなかった
- 2006年頃：CUDAが登場、様々な制限はありつつも「普通のプログラム」が利用可能に
  - 様々なアルゴリズムが実装された、科学技術計算への応用も活発化
  - GPUスパコンの誕生
  - バージョンアップ（最新は7.5）により高機能化、制限の撤廃
- 2011年頃：OpenACCが提案される
  - CUDAより容易で汎用性のある（NVIDIA GPUに縛られない）プログラミング環境に対する要求の高まり
  - 最新仕様は2.5、実装されているのは2.1程度まで

NVIDIA社のGPUラインナップについて（1/3）

- GeForce
  - コンシューマ向けグラフィックスカード
  - 主にゲーミングPCで使われる（＋最近は機械学習、VR等も）
  - 単精度演算性能を重視（倍精度演算用のHWをあまり搭載していない）、クロック周波数が高めの傾向
  - 安価
- Quadro
  - ワークステーション用グラフィックスカード
- Tesla
  - HPC（科学技術計算、スパコン）向けGPU
  - 画面出力できないモデルも多い（“Graphics” Processing Unit……？）
  - 倍精度演算性能も重視、クロック周波数が低めの傾向、ECCメモリ対応
  - 安価とは言い難い

NVIDIA社のGPUラインナップについて（2/3）

- アーキテクチャ（世代）と特徴・新機能
  - Tesla：最初のHPC向けGPU
  - Fermi：本講習会で用いるGPU
    - ECCメモリ、FMA演算、atomic演算
  - Kepler：現行のHPC向けGPU
    - コア群を構成するコア数の増加、動的な並列処理（GPUカーネルからGPUカーネルの起動）、Hyper-Q（複数CPUコアによるGPU共有）、シャッフル命令、読み込み専用データキャッシュ、Unifiedメモリ、PCI-Express 3.0
  - Maxwell：コンシューマ向けGPU
    - 電力あたり性能の向上、Teslaとしての製品は存在しない
  - Pascal：Reedbush-Hに搭載予定、単体Teslaとしての普及はこれから
    - HBM2（三次元積層タイプの高速メモリ）、NVLink（高速バス）
  - Volta：次世代GPU

NVIDIA社のGPUラインナップについて（3/3）

- 「現行GPUではできるが、講習会で使うGPUではできないこと」もあるが、最適化を行ううえで基本となる点は共通している
  →Reedbushでも活用できる
- 世代毎に色々な制限等に違いがあるため、細かい最適化パラメタについては都度考える必要がある
  - 最大並列度、レジスタ数、共有メモリ容量、命令実行サイクル数、etc.

GPU入門

- GPUの構造と特徴について学ぶ
- 最適化を行ううえで考えるべきことの概要を学ぶ

- 本講習会ではCUDAを用いたプログラム最適化については扱わないが、その方法（概要）を知っているとOpenACCの最適化も行いやすくなるため、簡単に説明する

ハードウェアスペックの確認

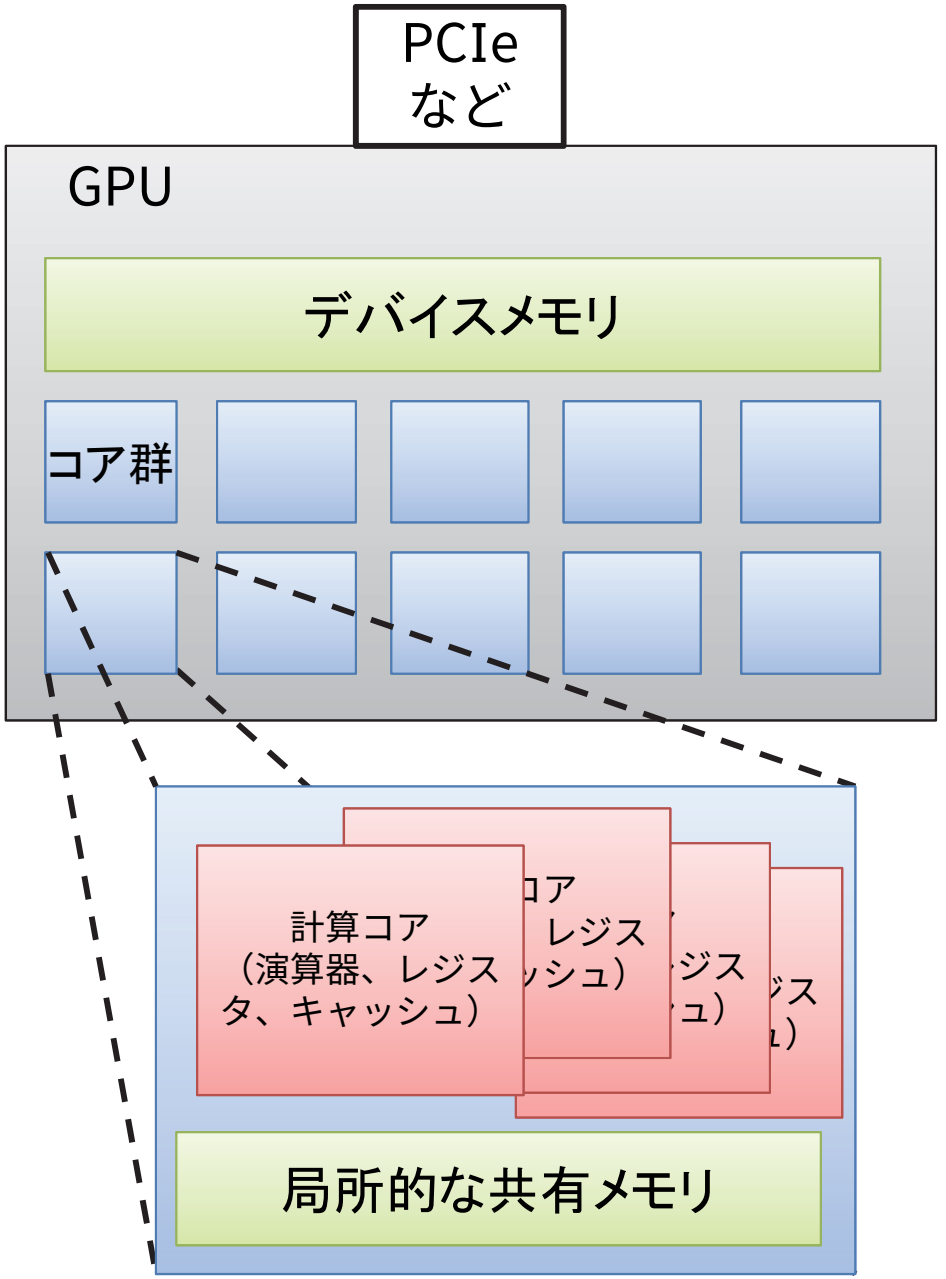
今回利用するGPU

現行のHPC向けGPU

	SPARC64 IXfx	Xeon E5-2670 (Sandy Bridge-EP) HA-PACS ホストCPU	Tesla M2090 (Fermi) HA-PACS GPU	Tesla K40 (Kepler)
コア数	16	8 (HT 16)	512 (32*16)	2880 (192*15)
クロック周波数	1.848 GHz	2.60 GHz	1.3 GHz	745 MHz
搭載メモリ種別	DDR3 32GB	DDR3 最大384GB (HA-PACS 64GB/socket)	GDDR5 6GB	GDDR5 12GB
Peak FLOPS [GFLOPS] (SP/DP)	236.5	332.8/166.4	1330/665	4295/1430
Peak B/W [GB/s]	85	51.4	178 (ECC off)	288
TDP [W]	110	115	225	235

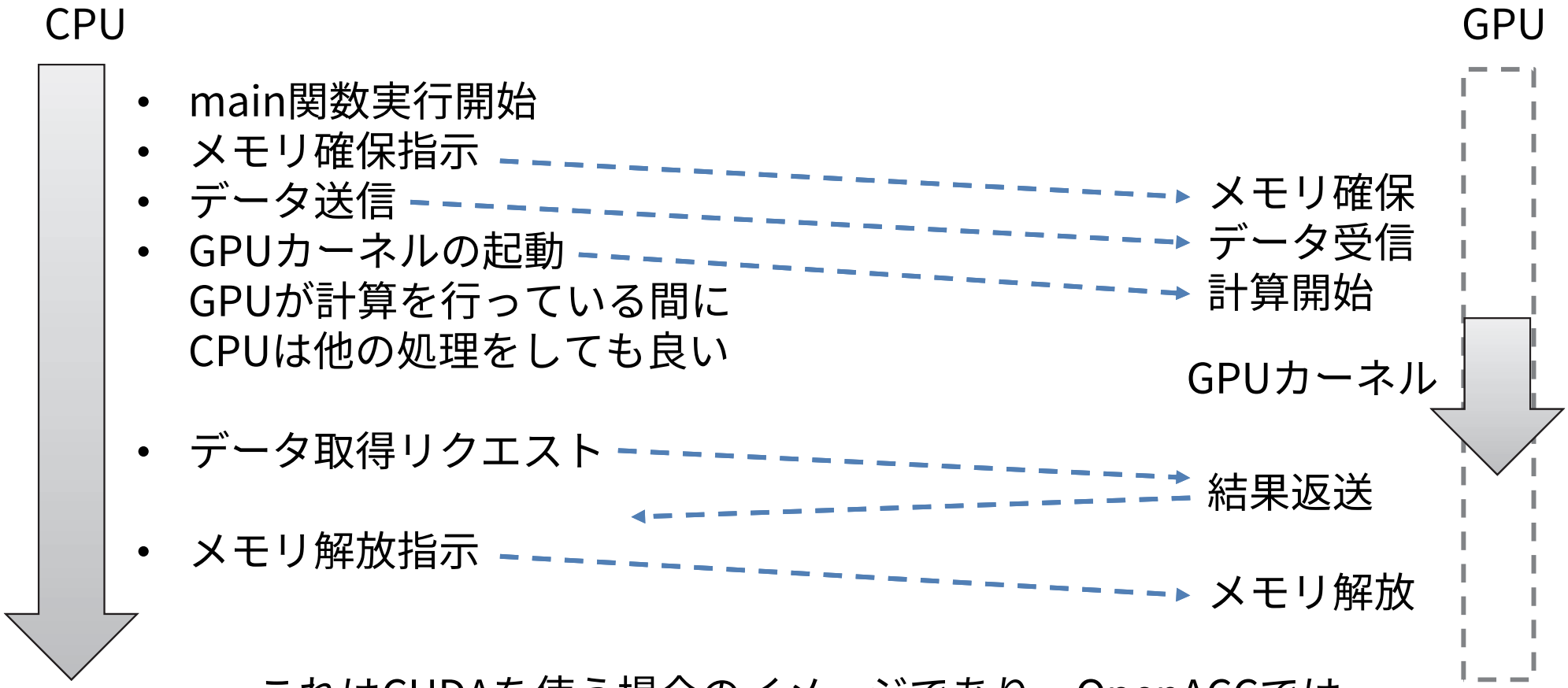
GPUハードウェア：「超」概要

- ホスト(CPU)とデバイス(GPU)はPCI-Expressなどで接続されている
- GPU上にはいくつかのコア群とデバイスメモリが搭載されている
- コア群にはいくつかの計算コアと局所的な共有メモリが搭載されている
- 局所的な共有メモリはデバイスメモリと比べて高速だが小容量



GPUプログラムの動作イメージ

- CPUからの指示に従ってGPUが動作する
  - 現状、CPUとGPUは主従の関係にあると言える

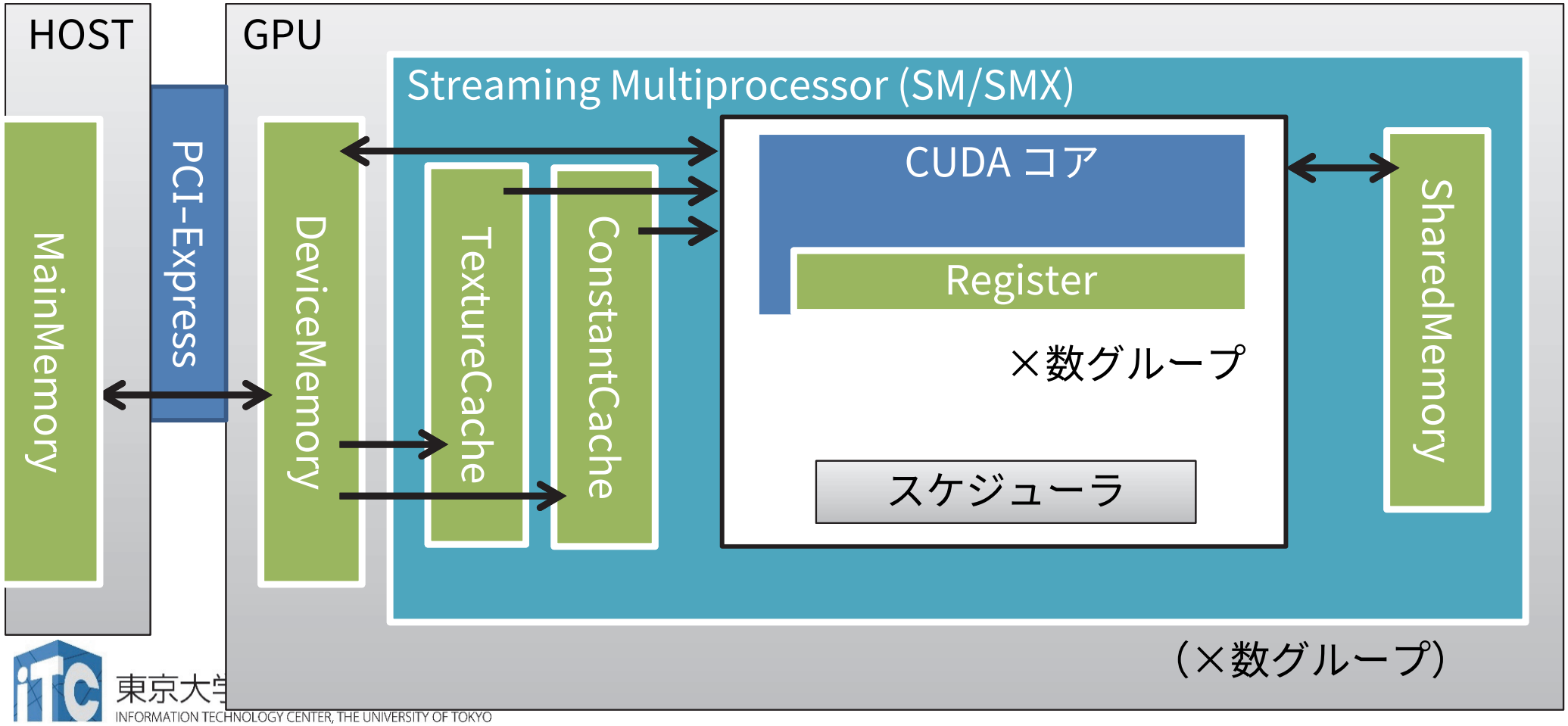


これはCUDAを使う場合のイメージであり、OpenACCでは
もう少し単純なイメージで扱う（→後述）

CUDAアーキテクチャ（物理的な構成）

※以下、SM/SMXをSMxと表記する

- 物理的な構成の概要
  - SM/SMXはGPUあたり1～30（GPUのグレードに依存）
  - CUDAコアはSM/SMXあたり8~192（GPU世代に依存）

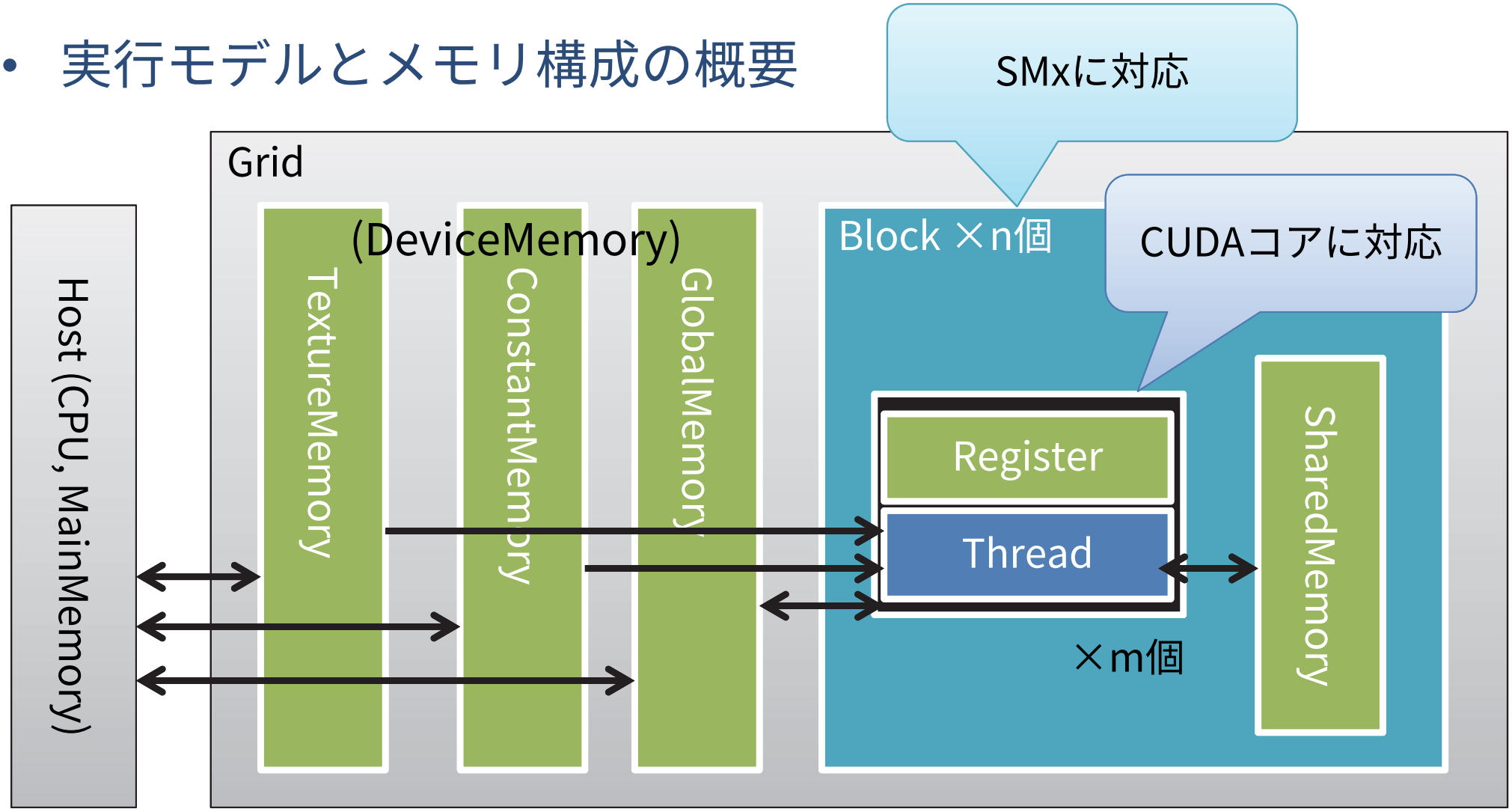


ハードウェアの特徴

- 階層性のあるハードウェア構成
  - 演算器の構成
    - 階層性のある演算器配置（CUDAコア*m × SMx*n）
    - 幾つかの計算コアがグループを構成、同一グループ内のコアは同時に同じ演算のみ可能（SIMD的な構成）
      - CPUのコアのように独立して動作できず、分岐方向が違う場合にはマスク処理される
      - NVIDIAはSIMT(Single Instruction Multiple Threads)と呼んでいる
  - メモリの構成
    - 階層性と局所性のあるメモリ配置
      - 全体的な共有メモリ ＋ 部分的な共有メモリ ＋ ローカルメモリ
      - GPU上に搭載された大容量でグローバルなメモリ：DeviceMemory
      - 局所的に共有される小容量高速共有メモリ：SharedMemory
      - コア毎に持つレジスタ

CUDAアーキテクチャ（実行モデルとメモリ）

- 実行モデルとメモリ構成の概要



- CPUのプロセスやスレッド同様に、Block・Threadは物理的な数以上に生成可能、CUDAではGPUカーネル起動時に<<<,>>>という記号を用いて指定する
- 特にThreadは物理的な数を超えて作成した方が良い（後述）

詳細なメモリ構成

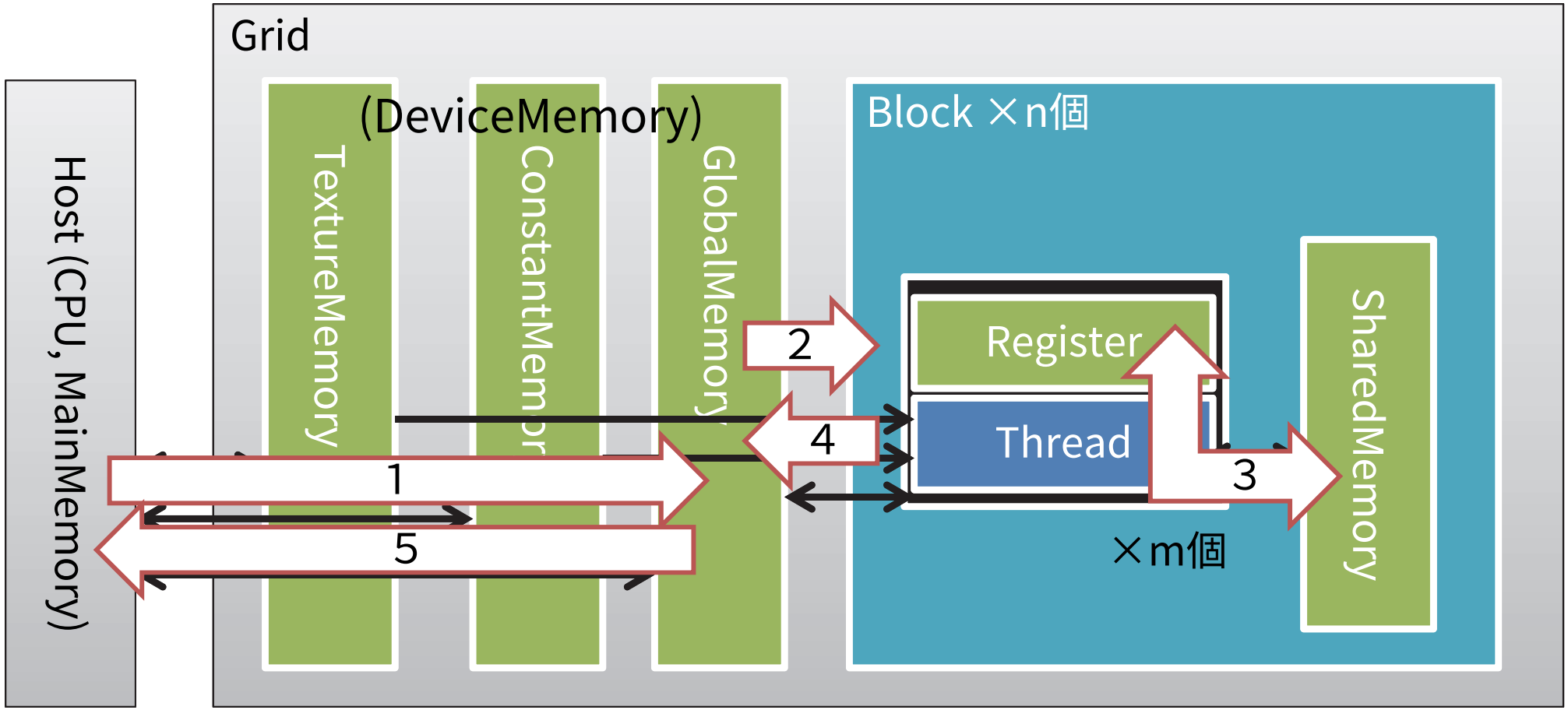
- 特徴の異なる複数種類のメモリ
  - 必ずしも全てのメモリを使う必要はない

名称	Lifetime	共有範囲	速度	容量
GlobalMemory	プログラム	GPU全体	高速・高レイテンシ	～GB
ConstantMemory	プログラム	GPU全体	高速・高レイテンシ＋キャッシュ	64KB
TextureMemory	プログラム	GPU全体	高速・高レイテンシ＋キャッシュ	GlobalMemoryと共用
SharedMemory	Grid	SMx単位	超高速・低レイテンシ	~112KB/SMx *
Register	Grid	非共有	超高速・低レイテンシ	~64KB/SMx *
LocalMemory **	Grid	非共有	高速・高レイテンシ	-

* GPUの世代により異なる
** 実体はGlobalMemory、レジスタを使いすぎるとLocalMemoryに配置されてしまう

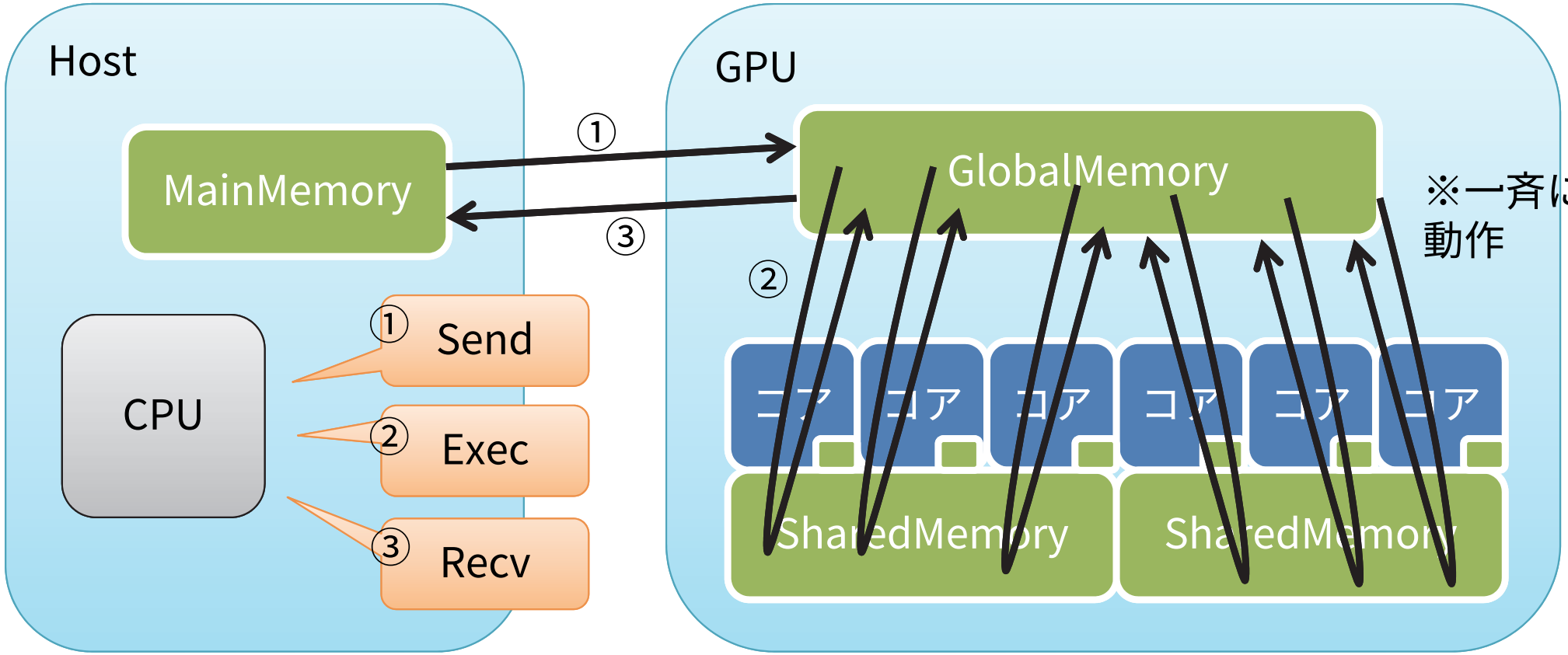
CUDAアーキテクチャ（データの流れ）

- 計算時の データの流れ



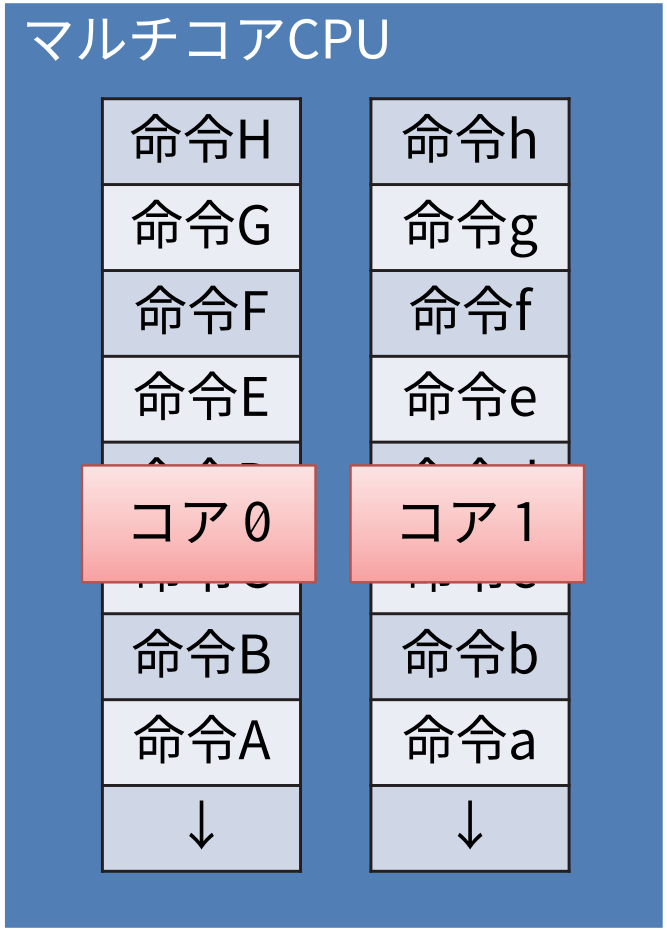
CUDAアーキテクチャ（GPUの制御）

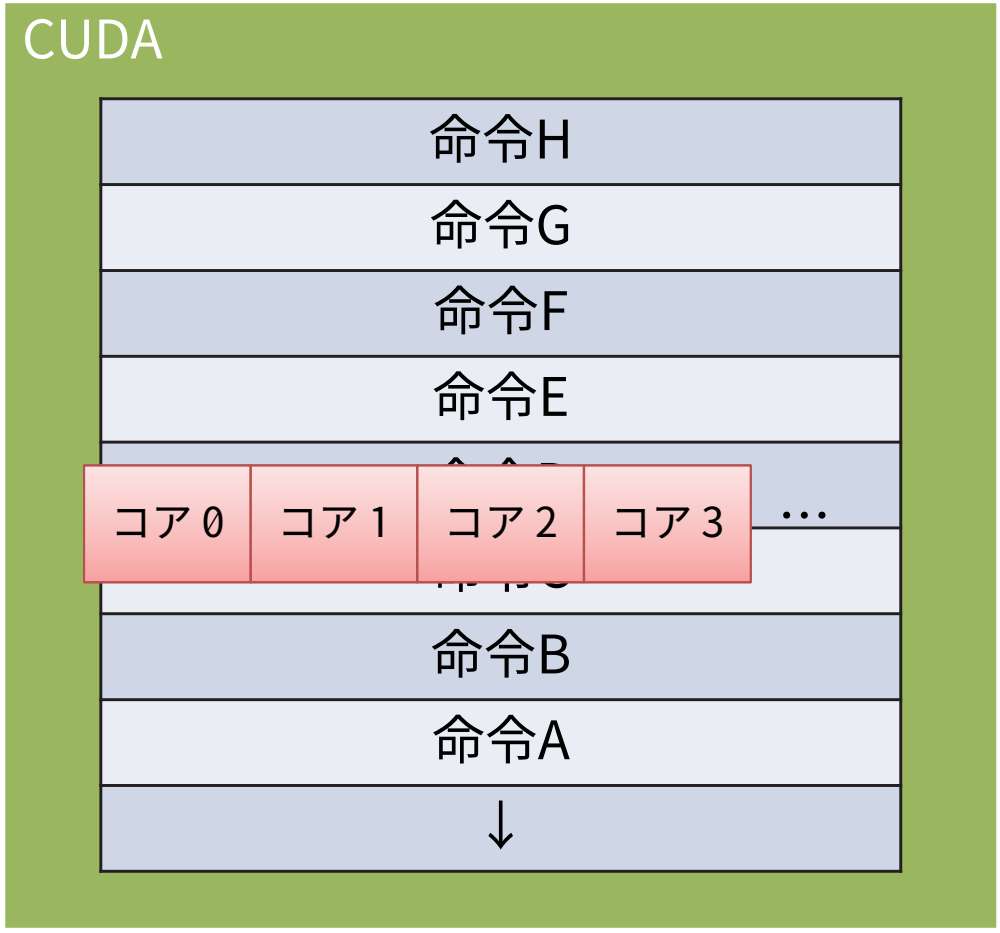
- もう少し詳しい実行モデル解説
  - CPUによるGPU制御、GPU上のコアの一斉動作



CUDAアーキテクチャ（命令の流れ）

- もう少し詳しい実行モデルのイメージ
  - 各コアが流れてくる命令を処理していくようなイメージ
  - GPU上のコア群は同時に同じ命令を実行している（全体で、ではない）





さらに詳しい実行モデルの解説

- 実際のスケジューリングは32スレッド単位（＝WARP単位）で実行される
- 異なるデータに対して同時に同じ演算を行う
  - 実行時に取得できるIDを用いて各自の計算対象（配列インデックス）を算出する
  - この点においてはMPIやOpenMPとあまり変わらない
- WARP内のスレッド毎に分岐方向が異なるプログラムを実行する場合は、分岐方向の異なるスレッドは待たされる
  - divergent WARP、重要な性能低下要因
  - スレッドIDが連続する32個のスレッド毎に分岐方向が揃うようなプログラムを作成すれば、divergent WARPによるペナルティが発生しない

CUDA最適化プログラミングの考え方

- 高性能が得られるプログラムの条件
  - 大量のスレッドを生成する
    - 理想的なBlockあたりスレッド数は64~256程度
    - GPUの世代やプログラムの複雑度などにも影響を受ける
  - GlobalMemoryのコアレスアクセスを行う
    - メモリアクセスをまとめる機能がある
  - SharedMemoryのバンクコンフリクトを回避する
    - SharedMemoryを利用する際に同じメモリバンクにアクセスすると性能が低下する
  - 分岐しない
    - GPUはCPUと比べて分岐処理に弱い
    - ループアンローリングがなどにより改善することもある
- 以下、各手法の概要について説明する
  - 最適化の際には各手法が衝突することもあるので注意が必要

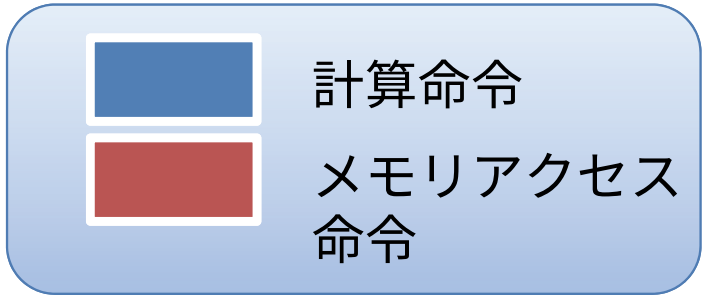
大量のスレッドを生成する

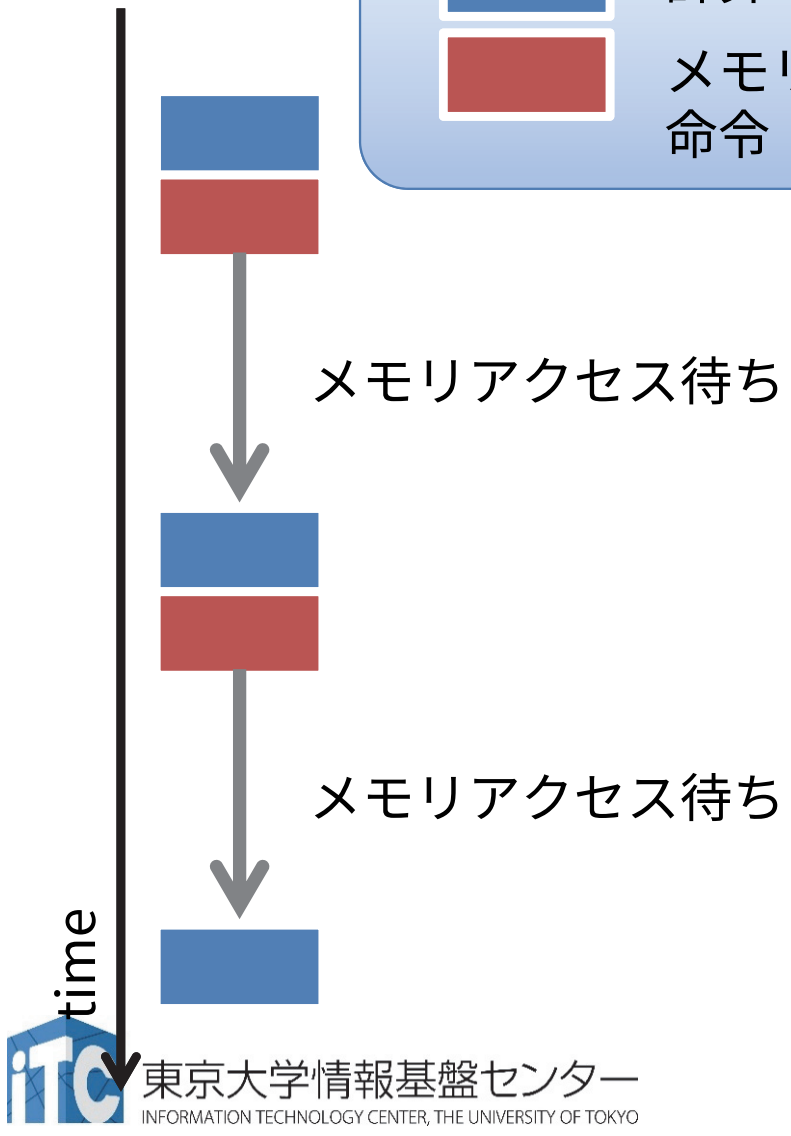
- スレッドのコンテキストを切り替えるのがとても速いため、メモリアクセスを待つよりコンテキストを切り替えて別のスレッドを処理した方が速い
- 逆に言えば大量のスレッドでGlobalMemoryに対するメモリアクセスのレイテンシを隠蔽しないと高い性能が得られない
- ただし、レジスタや共有メモリの使用量が多すぎると多数のスレッドを実行できない
  - 同時に実行できるスレッドやブロックの数は色々な資源の使用量によって決まる

並列度の高いシンプルなGPUカーネルが望ましい

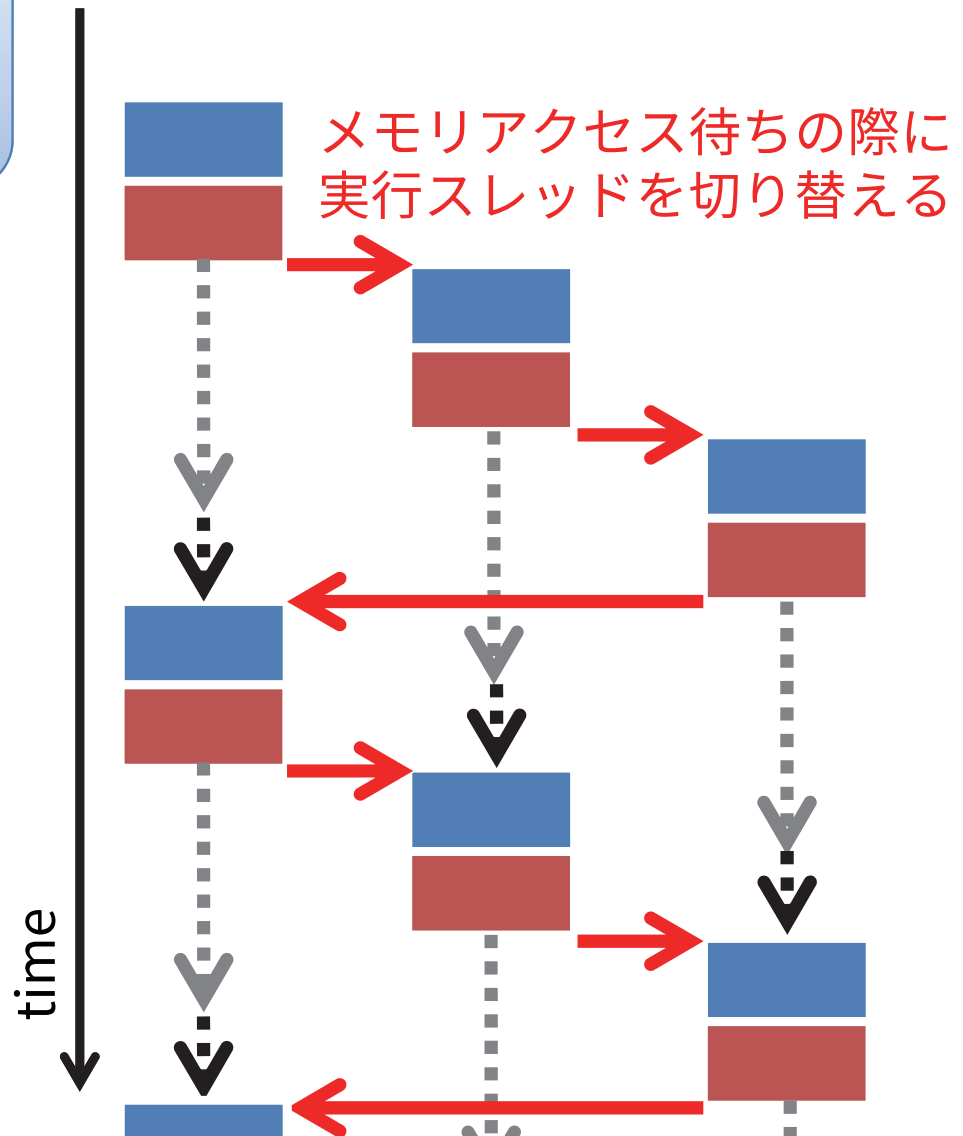
大量スレッドによるメモリアクセス隠蔽のイメージ

- CPU



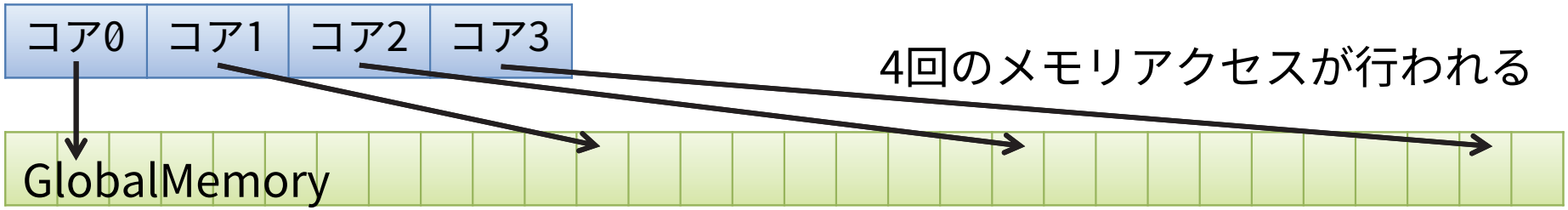


- CUDA

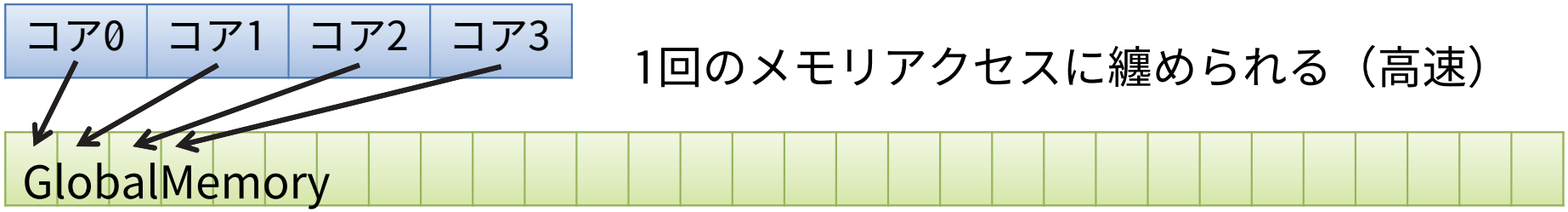


GlobalMemoryのコアレスアクセス

- 同一SMx内の複数CUDAコアによるメモリアクセスが近い場合にまとめてアクセスできる
  - 詳細な条件はGPUの世代によって異なる
    - 最新世代ほど条件が緩い
- アクセスがバラバラな（遠い）場合

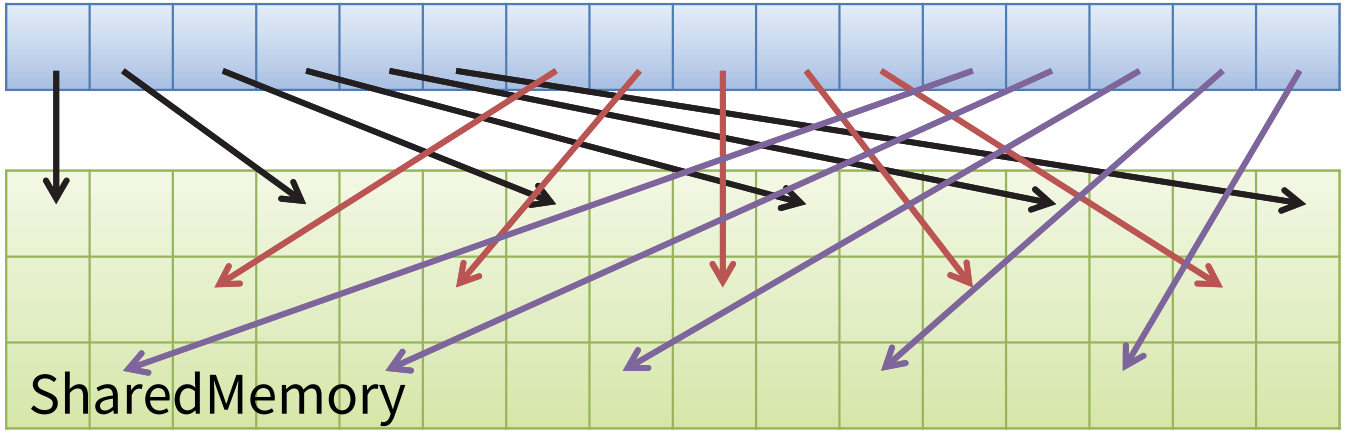


- アクセスが揃っている（近い）場合

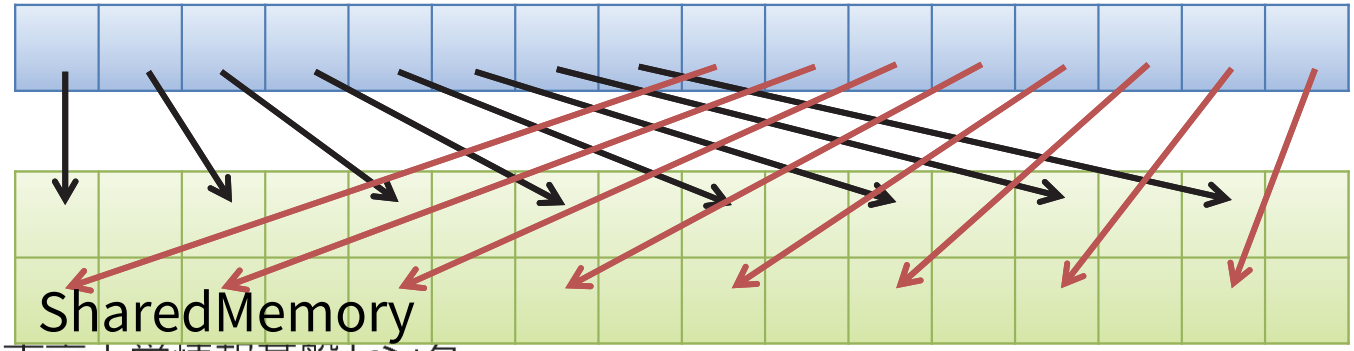


SharedMemoryのバンクコンフリクト回避

- 高速共有メモリは16個or32個ずつのバンクにより構成
  - 同一バンクへのアクセスが集中すると性能低下
- 均等なアクセス＝性能低下しない



- アクセスが集中＝性能低下する



2-way
バンクコンフリクトの例