GPUのしくみ、RICCでの使い方およびベンチマーク

理化学研究所 情報基盤センター

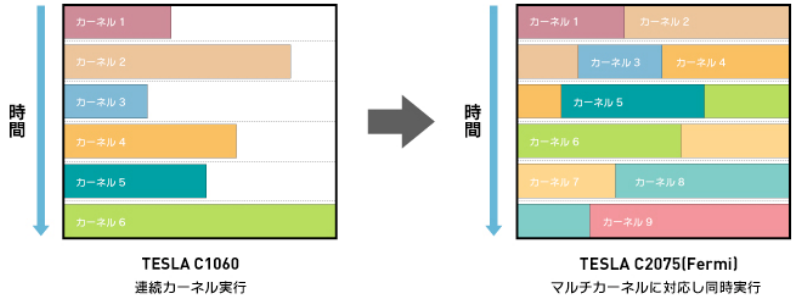
2013/6/27 17:00-17:30

中田真秀

RICCのGPUが高速に! (旧C1060比約6.6倍高速)

- RICCのGPUがC2075になりました!
  - C1060比6.6倍高速
    - 倍精度515GFlops
  - UPCに100枚導入:合計51.5TFlops
  - うまく行くと5倍程度高速化

- RICCユーザーは今お使いいただけます

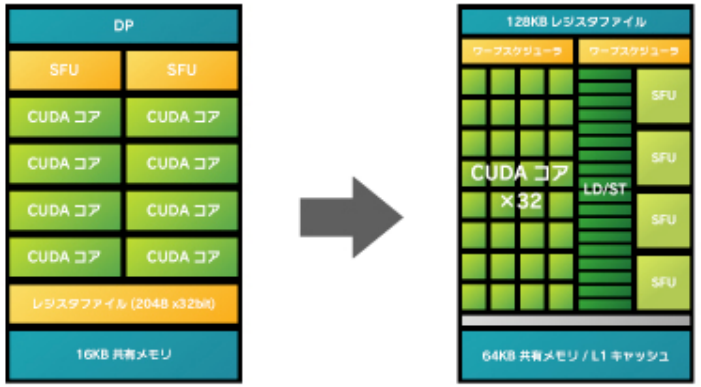


TESLA C1060
連続カーネル実行

TESLA C2075(Fermi)
マルチカーネルに対応し同時実行

TESLA C1060のSM(Streaming Multi-processor)との比較

TESLA C1060のSM(Streaming Multi-processor)との比較



TESLA C1060 (T10)

TESLA C2075(Fermi)

NVIDIA C2075仕様

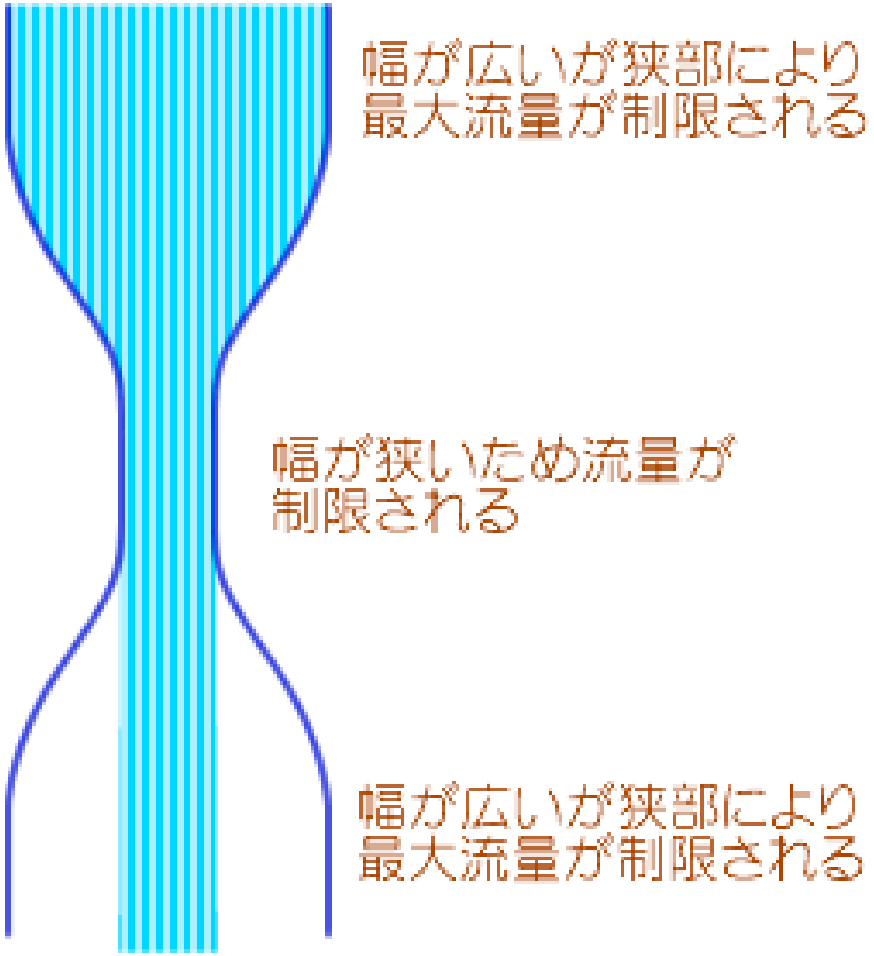
	NVIDIA C2075	NVIDA C1060
搭載GPU	1	1
CUDAコア数	448基	240基
プロセッサ周波数	1.15GHz	1.296GHz
単精度演算性能	1.03TFlops	993GFlops
倍精度演算性能	**515GFlops**	78GFlops
搭載メモリ容量	**6GB**	4GB
メモリインターフェース	GDDR5 SDRAM 384bit	512bit GDDR3
メモリクロック	1.5GHz	800MHz
メモリ転送帯域	**144GB/s**	102GB/s

この講演の対象者と目的

- 対象者
  - GPU、GPGPU、CUDAに興味はあるが、よくは知らない人。
  - RICCでGPU対応アプリケーションを使いたい人
    - 特にAMBER、分子動力学 (Molecular Dynamics; MD)系
    - 数倍程度高速になることもあります。
- 話題
  - コンピュータの簡単な仕組みとNVIDIA C2075 GPUについて
    - GPUはなぜ速いのか?
  - ベンチマーク
    - 行列-行列積
    - 姫野ベンチ
    - 高速フーリエ変換
    - 分子動力学
  - RICCでのジョブの流し方

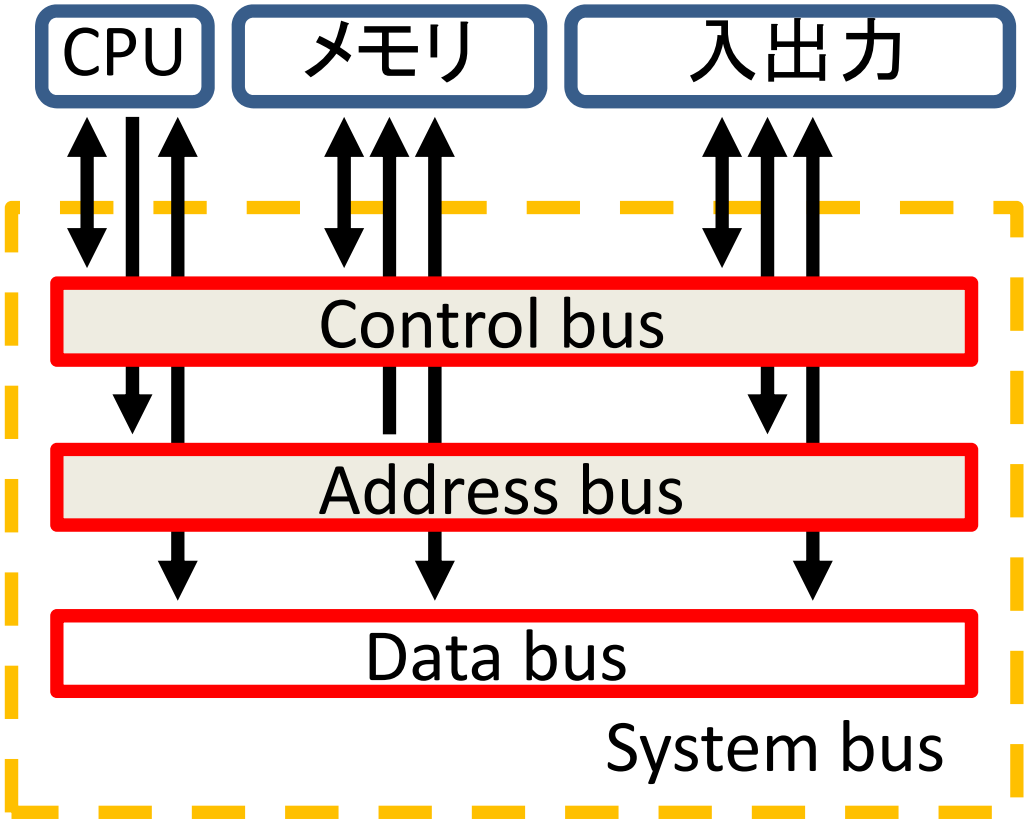
コンピュータの簡単な仕組みについて

ボトルネックとは


幅が広いが狭部により
最大流量が制限される
幅が狭いため流量が
制限される
幅が広いが狭部により
最大流量が制限される

コンピュータの簡単な仕組み

- コンピュータを一番簡単に
  - 入出力はハードディスクやネットワークなど
- バス=CPU, メモリ,入出力とデータをやり取りする道のようなもの
- コンピュータが高速とは?
  - CPUが高速
  - メモリが高速
  - 入出力が高速
- だけではダメで、
  - バスのスピードも高速じゃないとダメ
  - フォン・ノイマンボトルネックという

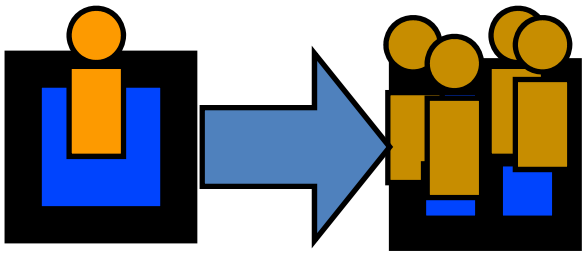


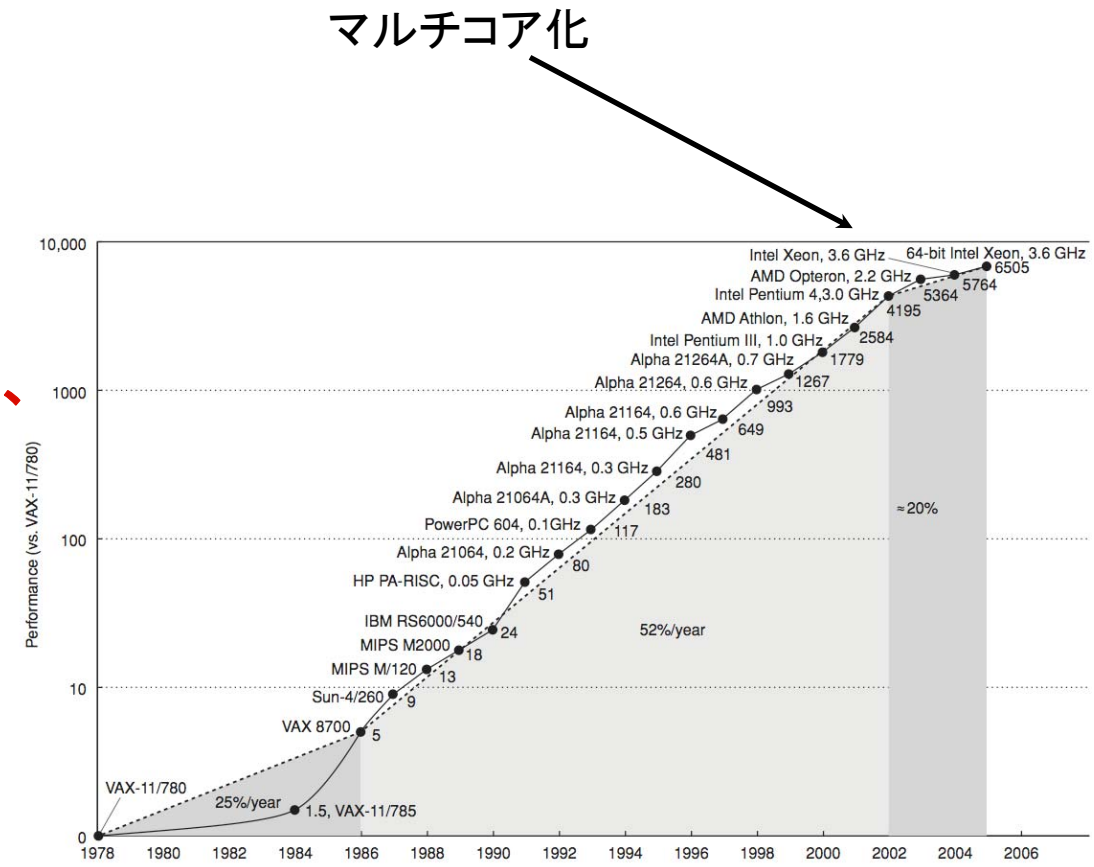
フォン・ノイマン型コンピュータ

CPU, メモリについて

CPUのスピードについて

- コンピュータは年々高速になってきている。
- ただ、コア一個単位処力は落ちてきている
  - 様々な物理的な限界
- マルチコア化
- いくつもコアを用意し、処理能力をあげる
  - CPUはマルチコア
  - NVIDIA, AMDのGPU
  - IntelのXeon Phi





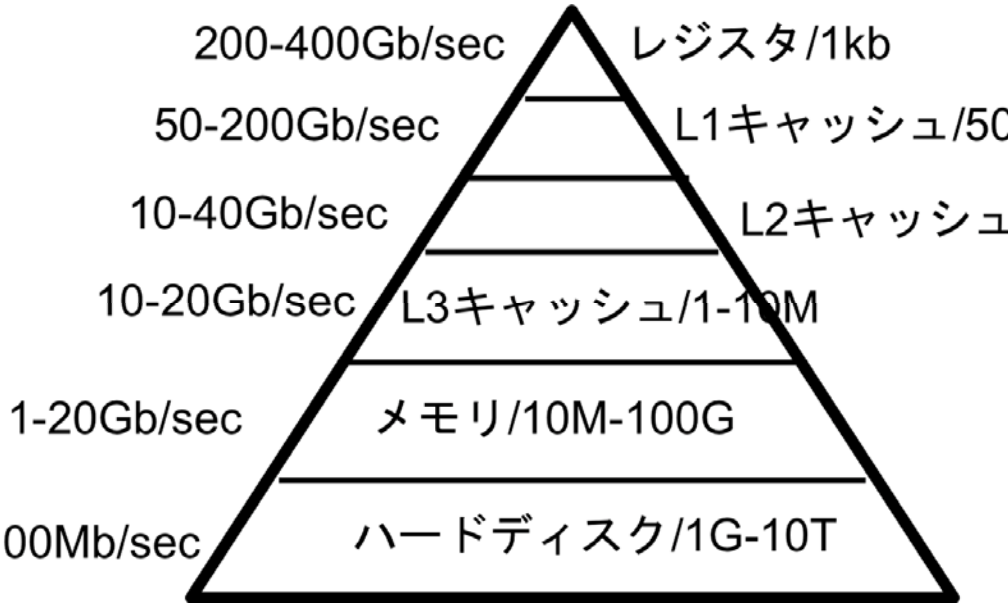
CPUの理論演算処理性能値

- CPUのスピードのよくある定義
  - CPUに入っている演算器が全て動いたとき
  - 一秒間に何回浮動小数点演算ができるか:FLOPs (Floting point operation per second)
  - 必ずしも「役に立つ」計算ではない。
  - 理論演算処理性能値、ピーク性能値、カタログ性能値などということもある。
  - クロック周波数、コア数、SIMD (一つの命令で複数の計算実行)などで決まる。
- Intel Core i7 920での理論性能値の計算方法(推定、TurboBoost off)
  - 2.66 (GHz) x 4 (コア) x 4 (演算器/コア) = 42.56 Gflops
  - Turbo Boostが入るとよくわからなくなる。

- 京コンピュータの理論演算処理性能値
  - 2 (GHz) x 8 (コア) x 8 (演算器/コア) = 128GFlops (1CPUあたり)
  - 128GFlops x 864 ラック x 102 ノード = 11,280,384GFlops = 11.3PFlops
- C2075の理論演算処理性能値(推定)
- 448(CUDAコア) / 2 (per clock) x 2 (FMA) x 1.15GHz = 515.20GFlops

メモリ(記憶装置)のスピードについて

- メモリの種類がある。
  アクセススピードが速い=コスト高、容量小
  アクセススピードが遅い=コスト安、容量大
  一桁容量が大きくなると、一桁遅くなる
  一桁容量が小さくなると、一桁速くなる
- メモリとCPU間のデータ通信のスピード
  - メモリバンド幅が大きい:速い
  - メモリバンド幅が小さい:遅い
- レイテンシ
  - データを一個取ってくるまでにかかる時間。短い方がいい。
- 高速化するには
  - アクセススピードを意識しよう。
  - データの移動を少なくしよう。
  - 一度にデータを転送し、転送している間に計算をしよう(=レイテンシを隠す)



メモリ(記憶装置)のスピードの理論性能値

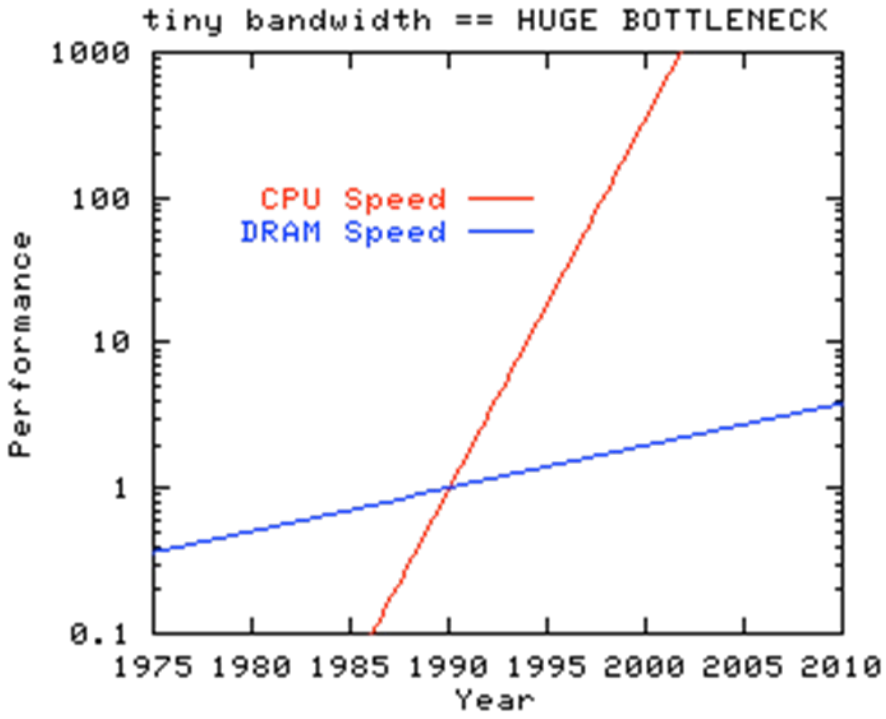
- メモリのスピードというと若干定義しづらい。
  - メモリだけのスピードではなく、メモリ <-> CPUのバスのバンド幅(太さ)にもよる。
  - レジスタ、L1, L2, L3 キャッシュなどのスピードは考えてない。

- DDR3-1066 は、8.5GB/sec
  - 133MHz x 4 (外部クロック) x 8 (I/Oバッファ) x 2 (8bit per 0.5 clock) / 8 (1byte=8) x 8 (interface データ幅) = 8.53GB/sec
  - チップ規格名
  - PC3-8500:モジュール規格名
- Intel Core i7だとトリプルチャネル (3つのDDR3メモリを同時に扱える)
  - 8.53GB/sec x 3 = 25.6GB/sec
- 京コンピュータの場合
  - 64GB/sec (SPARC64 VIIIfx諸元による)
  - DDR3 の4チャネル

- NVIDIA GPU
  - Tesla K20X : 250GB/sec (GDDR5) ,
  - **Tesla C2075 150GB/sec (GDDR5, on RICC)**

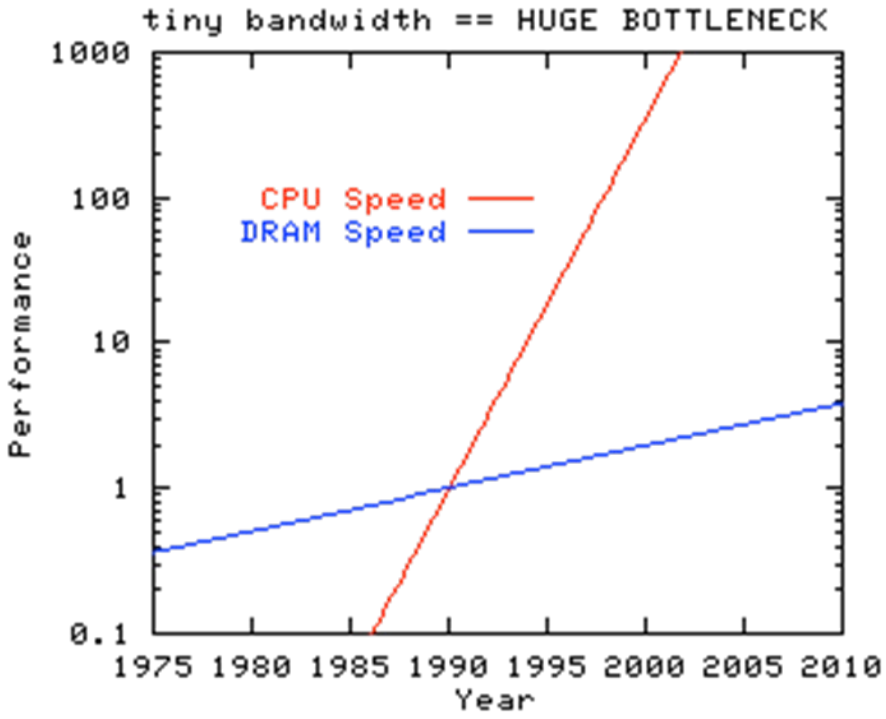
CPUとメモリのスピード比の変化

- CPUとメモリのパフォーマンス(=スピード)を年によってプロットしてみる。
- 1990年まで:メモリ>CPU
  - メモリのほうがCPUより高速
  - CPUになるべく計算させないほう高速
- 1990年以降:メモリ<CPU
  - メモリに保存するより、無駄でも毎回計算させた方が高速。
- このトレンドは変わらないといわれている。
  - デバイスの物理的制限
  - 革命的技術を待つ状況



CPUとメモリのスピード比の変化

- CPUは遊んでいることが多い(?)
- Intel Core i7 920+PC3-8500
  - CPU : 42.56 Gflops
  - DDR3 : 25.6GB/s
- 一演算あたり一回メモリアクセスが有るアプリは…
  - 25.6GB / 8 = 3.19GFlopsしかでない。
  - CPUは7.5%しか使ってない
- メモリに保存するより、無駄でも毎回計算させた方が高速。



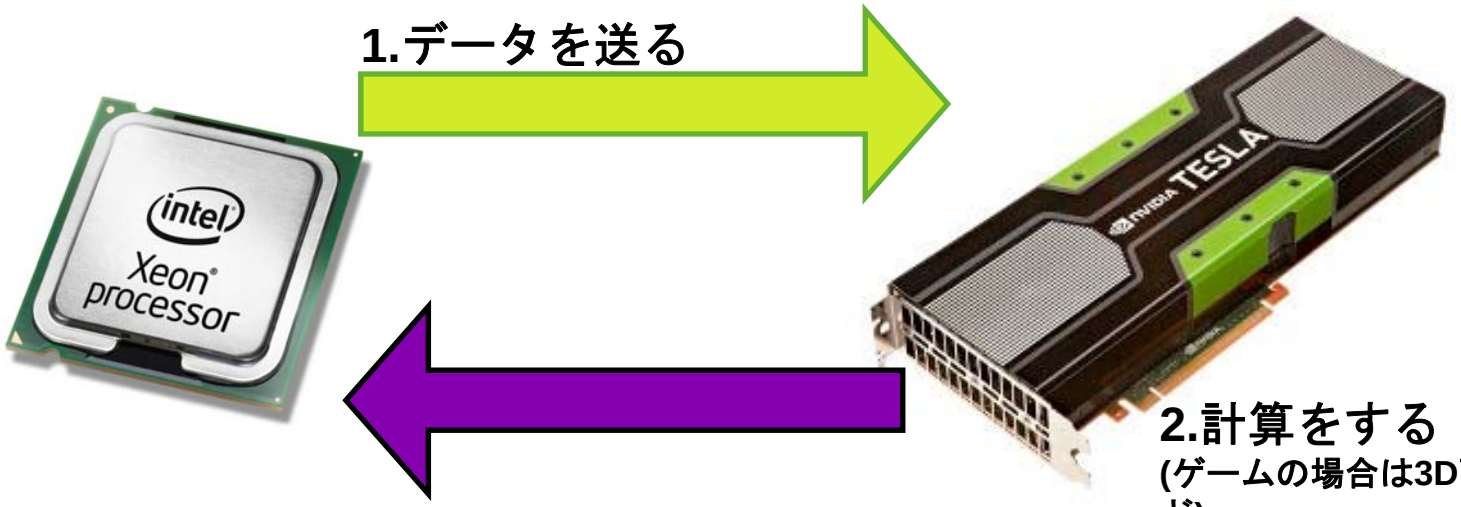
GPUについての紹介

GPUとは？GPGPUとは？

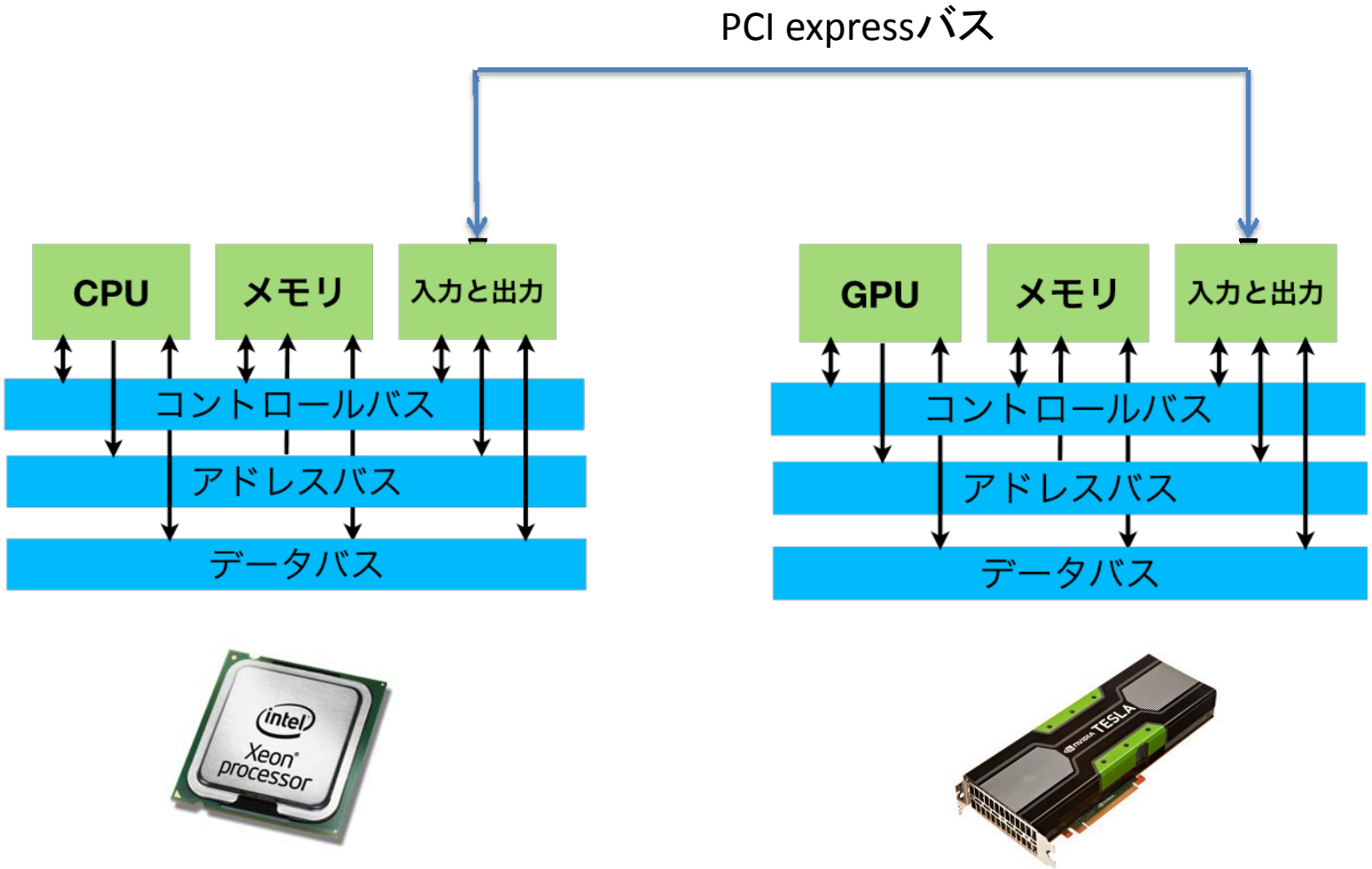
- GPUとは?
  - Graphics Processing Unit (グラフィックス処理器)のこと。
  - 本来、画像処理を担当する主要な部品
    - 例:3Dゲーム、ムービー、GUIなどの処理を高速に行える
    - 2006年からは科学計算にも使われるようになってきた。
- GPGPUとは?
  - General-Purpose computing on Graphics Processing Units
  - GPUによる、汎用目的計算
    - 画像処理でなくて科学技術計算することは
    - GPGPUといえる。
- 現在はPCI expressにつなげる形で存在。
  - バスがボトルネック
  - 将来はCPU/GPUが統合されるはず

GPUの使い方

- CPUからデータを送り、GPUで計算させて、計算結果を回収
  - メモリは共有されない。

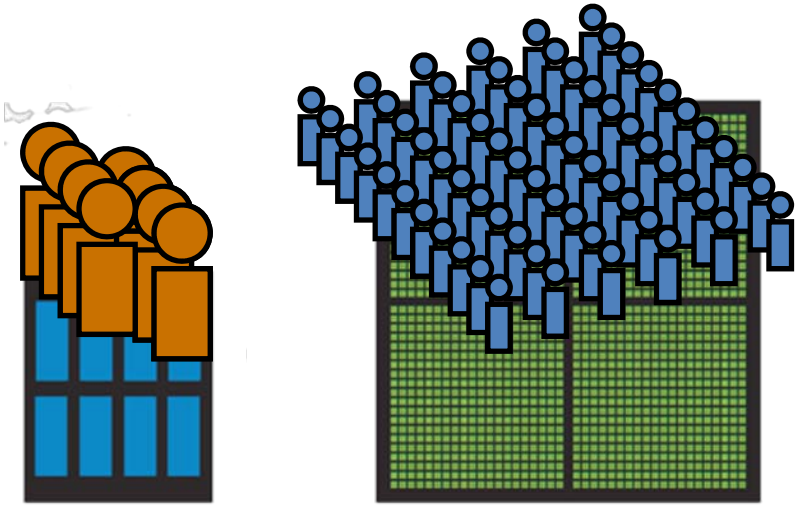


CPUとGPUの関係:フォン・ノイマン図的に



GPUはどうして高速か? Part I

- CPUと比べると1コ1コの処理能力は低いが、ものすごい数のコアがあって、似たような処理を同時に沢山行えるので高速。

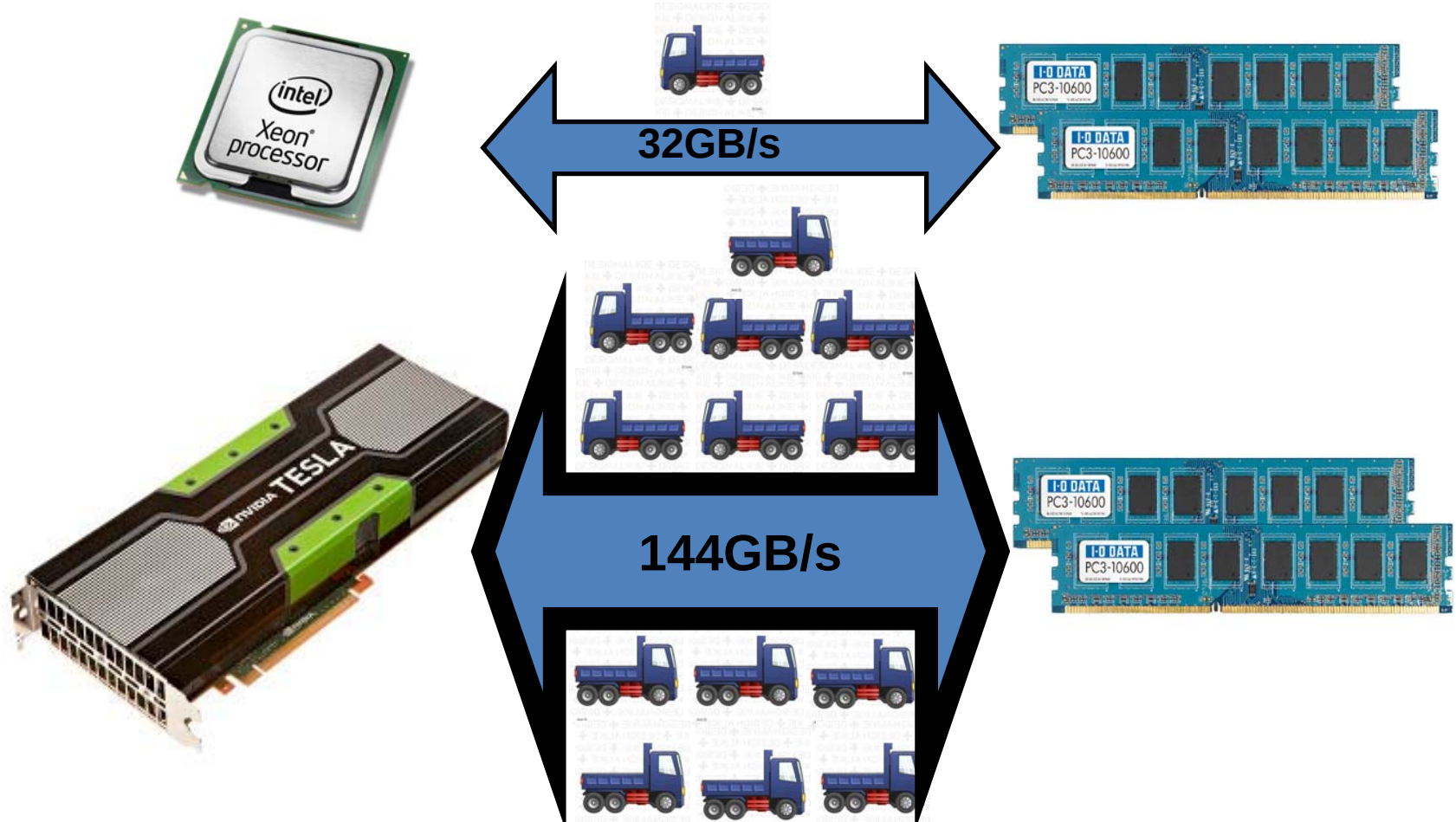


CPU　　GPU

- 画像処理だと沢山独立した点に対して似たような処理をする
- CPUみたいには複雑な処理はできないが、工夫次第で色々可能

GPUはどうして高速か? Part II

メモリバンド幅がGPUのほうが大きい



ベンチマーク&ジョブサブミット編

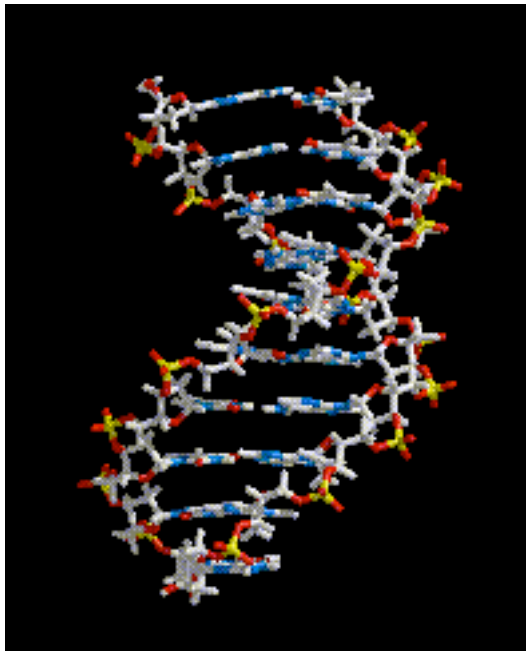
RICCにインストールされているGPU対応アプリについて

- RICCにはすでにいくつかのGPUに対応したアプリケーションやライブラリが用意されている。
  - AMBER11
  - AMBER12
  - GROMACS 4.5.3, 4.5.5
  - NAMD (近日中利用可能)
  - GAMESS (テスト中;利用は可能)
  - Qchem (近日予定)
  - cuFFT
  - cuBLAS
  - etc...

GPU対応AMBER11(12)の走らせ方

AMBERってなに？

- 分子動力学のプログラムパッケージ
- 力場を用いて分子の動きをシミュレーションする。



GPU対応AMBER11(12)の走らせ方

qsubスクリプト例

```
[username@ricc:AMBER] vi run_amber11_accel.sh
#!/bin/sh
#--- qsub option ---#
#MJS: -accel  -amber11
#MJS: -proc 1
#MJS: -time 1:00:00
#MJS: -eo
#MJS: -cwd

#--- FTL command  ---#
#FTLDIR: $MJS_CWD

#--- Program execution ---#
srun pmemd.cuda -O -i mdin ¥
-c inpcrd.quil -o bench.out -x mdcrd -gpu -1
```

ハード、ソフトウェアリソース名を指定

プロセス数は、1 のみ指定可能

経過時間を指定

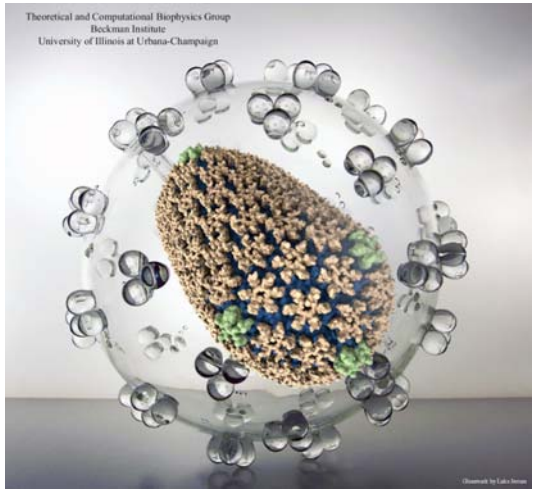
標準エラー出力と標準出力のマージ

入力ファイルを指定 (FTL)

GPU対応NAMD

NAMDとは?

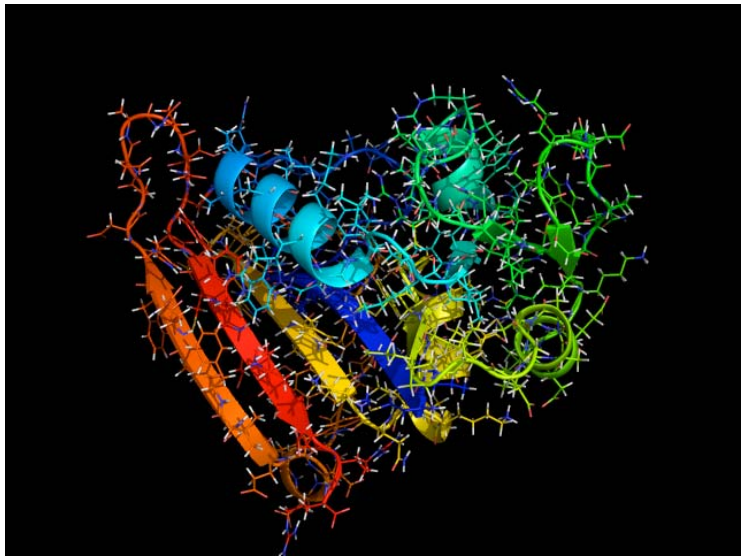
- 分子動力学計算プログラム
  - AMBERとシェアを競っている
  - **RICC導入予定**



GPU対応GROMACS

GROMACSとは?

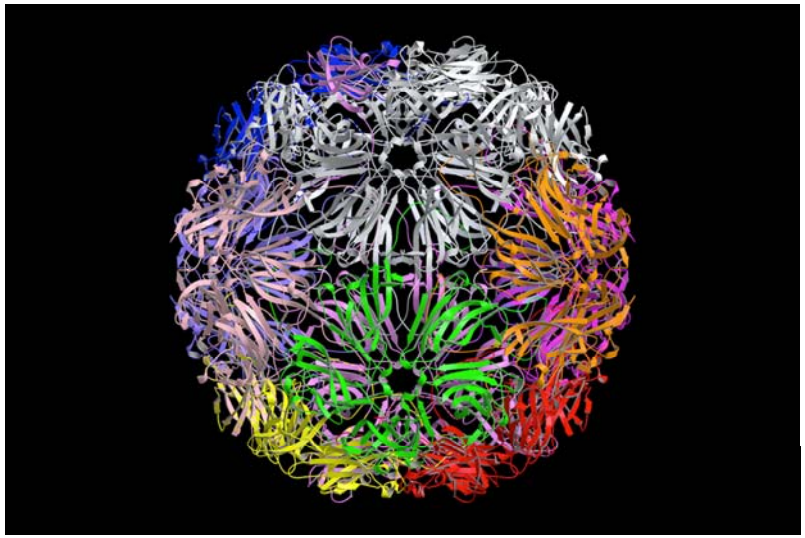
- 分子動力学計算プログラム
- これもよく使われている
- フリーソフトウェア(GPL)



GPU対応NAMDベンチマーク

Satellite Tobacco Mosaic Virus 100万原子

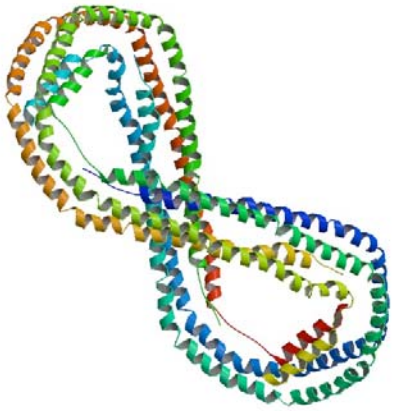
4cpu w. GPU　　：0.575935 s/step
K computer 80 core (10 node)：0.454294 s/step
ただしKでのNAMDの最適化はあまりやってない。



感覚的にはK computer 8node = 4CPU+C2075程度

GPU対応NAMDベンチマーク

apolipoprotein A-I （10万原子)

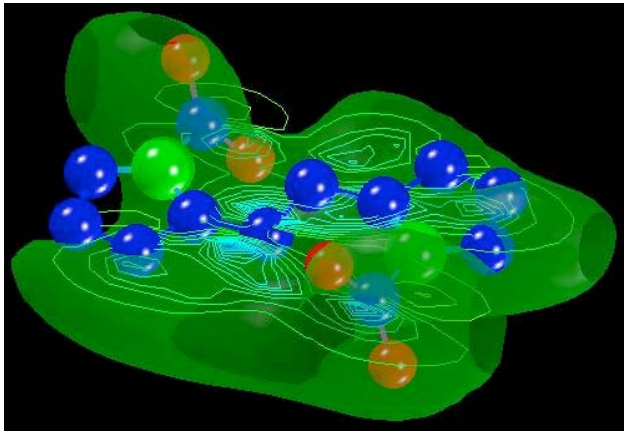


4 cpu w/o GPU　　4cpu w. 1GPU
0.340919 s/step　0.0615918 s /step

GPU版はCPU版と比較して5.5倍高速

GPU対応GAMESSの走らせ方

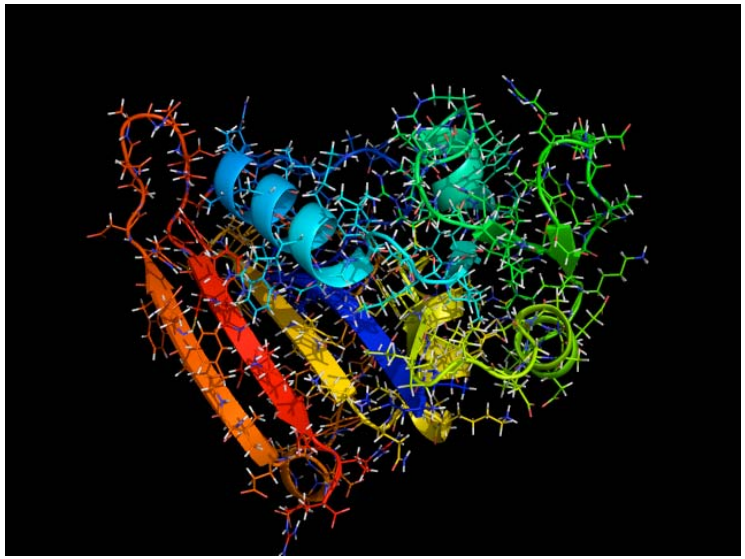
- GAMESSとは？
- 第一原理からの、量子化学パッケージ
- Gaussianについで広く使われている
- RICCでももちろん利用可能



GPU対応GROMACS

GROMACSとは?

- 分子動力学計算プログラム
- これもよく使われている
- フリーソフトウェア(GPL)



GPU対応GROMACS

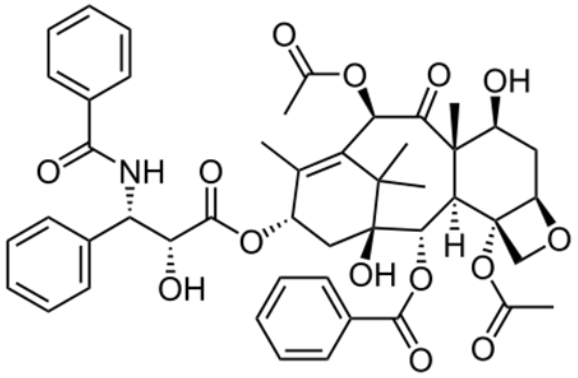
```
#!/bin/sh

#MJS: -accel
#MJS: -time 10:00
#MJS: -proc 1
#MJS: -eo
#MJS: -cwd

source /usr/local/gromacs-4.5.5/bin/GMXRC_gpu.bash
srun mdrun-gpu -maxh 0.1
```

GPU対応GAMESS：ベンチマーク

ベンチマーク例:タキソール分子
113原子、452電子、1032基底、SCF計算
RICCで計算



wGPU 43時間13分

w/oGPU 46時間31分

クロスアビリティ社のモジュールでより高速になるか?

GPU対応GAMESSの走らせ方

qsubスクリプト例

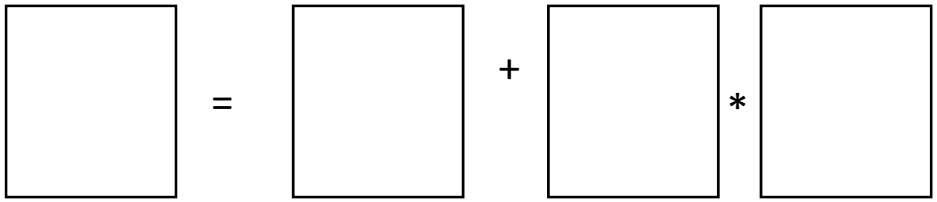
```
#!/bin/sh
#MJS: -upc
#MJS: -proc 8
#MJS: -time 72:00:00
#MJS: -eo
#MJS: -cwd

rm -rf ~/scr
mkdir ~/scr
export LD_LIBRARY_PATH=/usr/local/cuda-4.2/lib64:$LD_LIBRARY_PATH

VERNO=gpu_mpi
/usr/local/gamess_2012R1/rungms_gpu_mpi taxol.inp $VERNO
```

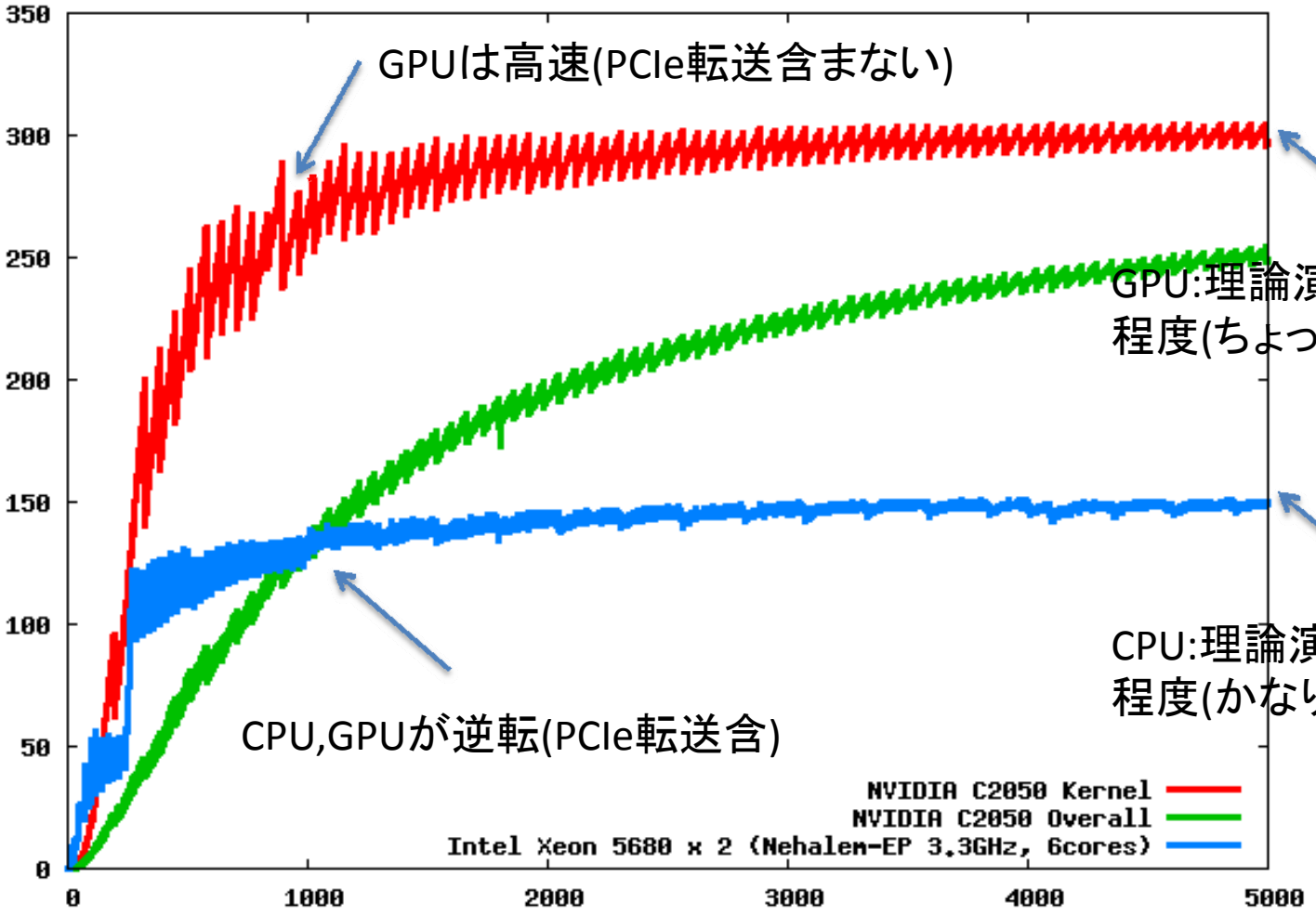
DGEMM 行列-行列積

- マシンのパワーをみるには、DGEMM (行列-行列積)と、DGEMV (行列ベクトル積)をみればよい。
- DGEMM (行列-行列積)
  - CPU/GPUのパワーがどの程度あるかの良い目安。
  - C←αAB+βC



  - GPU, CPUでベンチマーク
    - GPUはC2050 (C2075とほぼ同じ)
    - CPUはXeon 5680 x 2 (RICCよりよい)

行列-行列積のベンチマーク



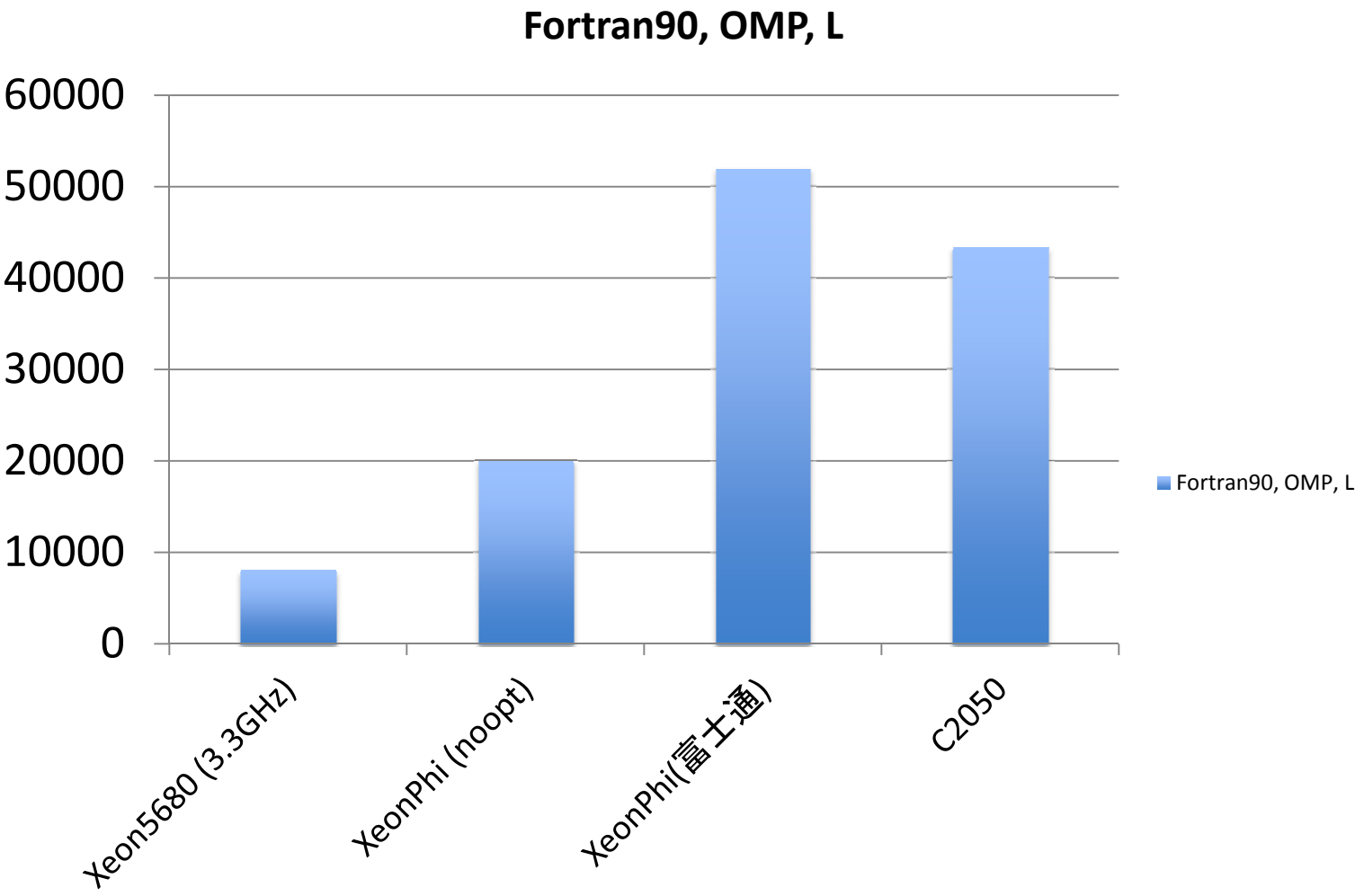
行列-行列積のベンチマークからわかること

- GPUはCPUとくらべて高速
  - ただしGPU-CPUのデータ転送を伴うと遅くなる
- うまく使うとパフォーマンスの大幅向上期待
- GPU, CPUの演算性能を計るベンチ

姫野ベンチマーク

- 姫野ベンチマークとは？？
  - 情報基盤センター・センター長の姫野龍太郎氏が非圧縮流体解析コードの性能評価のために考えたものでポアッソン方程式解法をヤコビの反復法で解く場合に主要なループの処理速度を計るものです。
  - ベンチマークテストは主に計算機のメモリバンド幅の性能を計るもの
  - **GPU, CPU, Intel Xeon Phi(エンジニアサンプル)で測定**
    - **行ったチューニング**
    - **CPUはそのまま**
    - **GPUはhttp://blogs.yahoo.co.jp/natto_heaven/MYBLOG/yblog.html**
    - **Intel Xeon Phiはそのまま and/or 富士通最適化**

姫野ベンチマーク


Fortran90, OMP, L
60000
50000
40000
30000
20000
10000
0
Xeon5680 (3.3GHz)
XeonPhi (noopt)
XeonPhi(富士通)
C2050
Fortran90, OMP, L

姫野ベンチマーク

- 姫野ベンチだとC2050は最新のIntel Xeon Phi+富士通最適化より10%程度遅い程度。
  - Intel Xeon Phiはメモリバンド幅が大きいのにそんなに変わらなかった?
- GPUのメモリバンド幅の大きいので高速!

さらに、K20 32枚導入!
(近日公開予定)

NVIDIAから:さらに32枚はK20を導入

- 近日予定、乞うご期待

	NVIDIA C2075	NVIDIA K20
CUDAコア数	448基	2496基
プロセッサ周波数	1.15GHz	1.296GHz
単精度演算性能	1.03TFlops	**3.52GFlops**
倍精度演算性能	515GFlops	**1.17TFlops**
搭載メモリ容量	**6GB**	5GB
メモリ転送帯域	144GB/s	**208GB/s**
RICC搭載	100枚	32枚

- 対応予定アプリ
  - AMBER, NAMD, GAMESS…

- ご希望、ご相談はお気軽に
  - hpc@riken.jp まで

情報基盤センターは
理研の研究者の研究をサポートします。