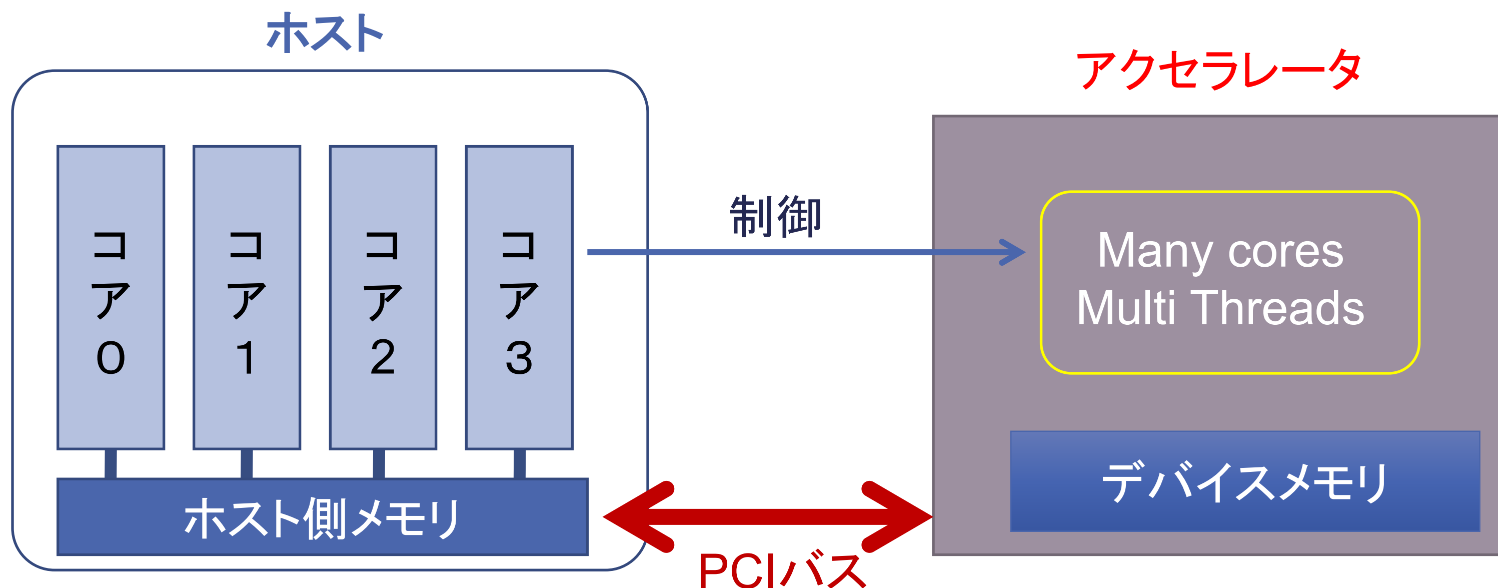
KOBE HPC サマースクール 2018
講義資料　2018/8/10

兵庫県立大学シミュレーション学研究科
安田　修悟

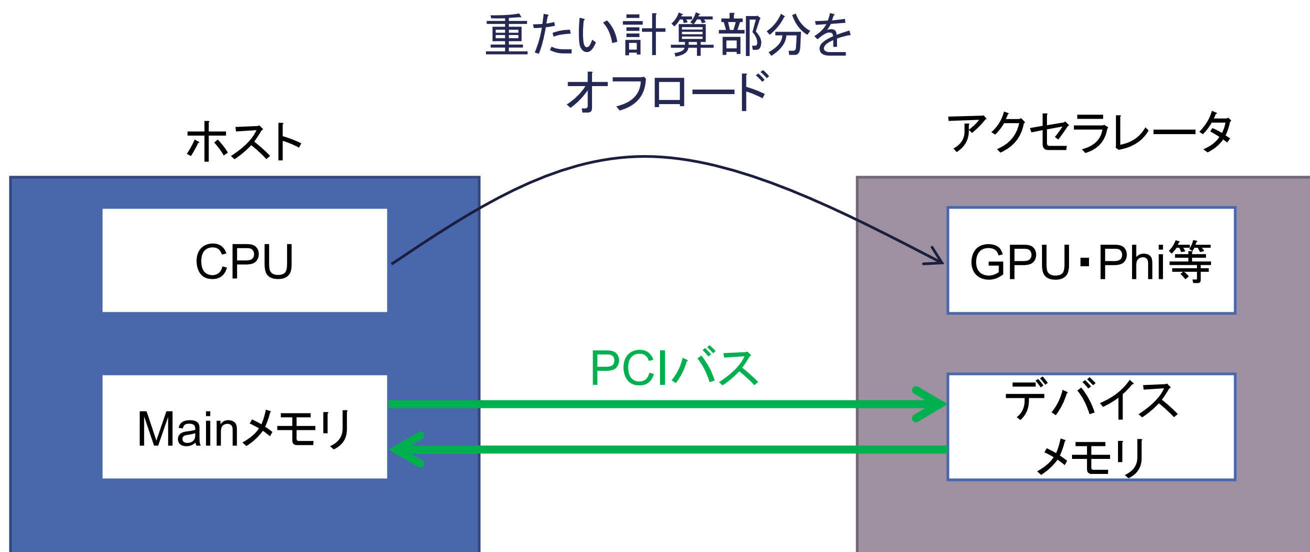
アクセラレータとは？

- CPUの処理を代替して処理の効率を向上させる装置（ディバイス）。
  - GPGPU (NVIDIA), Xeon Phi (Intel), APU (AMD)など



アクセラレータの実行イメージ

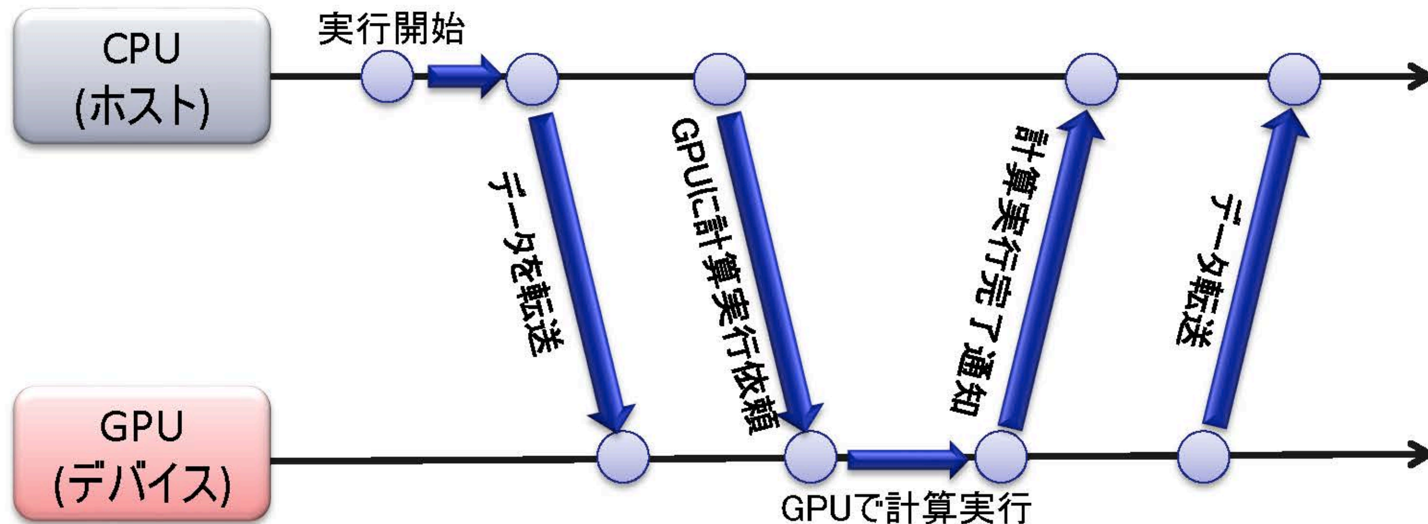
ハイブリット構成 （CPU＋アクセラレータ）



※データのオフロード通信がボトルネックとなる

- CPUのメモリ空間とディバイスのメモリ空間が異なる
- 一般的なメモリ帯域幅に比べてPCIバスでの低速な転送

GPUでのプログラム実行イメージ



- GPU(デバイス)ではCPU(ホスト)から制御をします。
- GPU(デバイス)の計算で必要なデータはCPU(ホスト)から転送します。
- GPU(デバイス)で計算した結果はCPU(ホスト)に転送します。

UNIVERSITY OF HYOGO
MMIV

OpenACC

- Accelerators用のAPIの一つ。
- CPUからディバイスに処理をオフロードする際に使用する。
- Fortran/C/C++言語でディレクティブを指定（OpenMPと同様）
- CUDA等の専用プログラムに比べて、導入が容易。

OpenACCプログラミング

- 基本フォーマット

```
#pragma␣acc␣ディレクティブ␣節1␣節2␣･･･
{
        構造化ブロック（途中で抜け出したり、終了しない）
}

例
#paragma ␣acc ␣kernels␣copy(a)
for(i=0;i<imax;i++){
        a[i]=a[i]*a[i];
        ････
}
```

OpenACCプログラミング

- 基本フォーマット (Fortranの場合)

```
!$acc␣ディレクティブ␣節1␣節2␣・・・
          構造化ブロック
!$acc end ディレクティブ

!$acc ␣kernels␣copy(a)
do i=0, imax
          a[i]=a[i]*a[i];
          ・・・・
end do
!$acc end kernels
```

基本的なディレクティブ構文

① Accelerator compute構文
- オフロードするループ対象部分を指定する。
- kernels構文やparallel構文がある

② DATA構文
- ホストとアクセラレータのデータ転送を明示的に指示する。
- 高速化の鍵を握る

③ Loop構文
- ①のオフロード領域のループのベクトル長や並列分割の詳細を明示的に指示する。

PGIコンパイラー

- GPUノードでインタラクティブ実行
  qsub␣-I␣-q␣G
- Module環境の設定
  module␣load␣pgi
  - module␣listでintelが設定されている場合
    module␣unload␣intel
    でIntel環境を先にunloadする。
- OpenACCのコンパイル
  pgcc␣-acc␣<prog.c>
  pgfortran –acc <prog.f90>
  - “-acc”はOpenACCディレクティブを有効にするためのオプション。

PGIコンパイラー

- OpenMPのコンパイル
  pgcc␣-mp␣program.c
  - “-mp”はOpenMPディレクティブを有効にするためのオプション。
- 最適化オプション
  - “-fastsse”：一般的な最適化オプション
  - “-O3”, “-O4”: より高度な最適化オプション
    (”-fastsse”より後に入れる
    例 pgcc␣-fastsse␣-O3␣program.c)

PGIコンパイラー

- 環境変数（実行前に設定）
  - PGI_ACC_TIME
    - 実行後に簡易プロファイル情報を標準出力に出力。
    - 0（ゼロ）を指定すると機能を抑制（デフォルト）、ゼロ以外の整数値で機能する。
  - PGI_ACC_NOTIFY
    - ディバイスの実行イベントを表示する
    - 整数値１：Kernel launch のイベントを出力、整数値２：データ転送のイベントの出力、など
    - 0（ゼロ）を指定すると機能を抑制（デフォルト）

OpenACCの演習

- GPUインタラクティブキューを使ってOpenMPプログラム(sumseq_omp.c)をPGIコンパイラーでコンパイルし20スレッドで実行.
  export OMP_NUM_THREADS=20
- OpenMPディレクティブをOpenACCのもに書換えたプログラム(sumseq_acc.c)を、PGIコンパイラーでコンパイルし実行.
- 計算時間を比較せよ.

OpenACCの演習

- ラプラス方程式のOpenACCプログラム

1. OpenMPのディレクティブを単純にOpenACCのディレクティブに書換えてみる.
   (laplace_omp.c)

   #pragma␣omp␣.....
   => #pragma␣acc␣kernels

2. OpenACCで計算時間は短縮したか？

OpenACCプログラミング

- 現状のラプラス方程式のOpenACCプログラムでは、毎ループごとに配列T[nmax+1][nmax+1]とTold[nmax+1][nmax+1]のデータがホストからアクセラレータにデータコピーが行われている.
- これを解消するためにデータ構文を使って、データコピーを抑制するプログラムに変更する。

DATA構文

- #pragma␣acc␣data␣節
- 節には以下のようなものがある.（Kernel構文の節にも使える.）
  - copyin(a,b): 配列a, bをホストからアクセラレータにコピーする.
  - copyout(c): 配列cをアクセラレータからホストにコピーする.
  - copy(A) : copyinとcpyoutの双方を行う.
  - create(B): アクセラレータでローカルに使用する配列Bの領域を作成.
  - present(C): 対象となる処理に入ったときにすでにデバイスメモリにある配列Cのデータを使用する.

演習

- ラプラス方程式のプログラム（laplace.c) をデータ構文を使って，ループ開始後のデータコピー回数を減らすようにプログラムせよ.
- OpenMP (laplace_omp.c) とOpenACCの計算時間を比較して高速な計算が実行されているか確認せよ.