XcalableACC: OpenACCを用いたアクセラレータクラスタのためのPGAS言語XcalableMPの拡張

中尾 昌広,[1] 村井 均,[1] 下坂 健則 [1]
田渕 晶大,[2] 塙 敏博,[3] 児玉 祐悦 [2,4]
朴 泰祐,[2,4] 佐藤 三久 [1,2,4]

1. 理化学研究所 計算科学研究機構
2. 筑波大学大学院 システム情報工学研究科
3. 東京大学 情報基盤センター
4. 筑波大学 計算科学研究センター

第146回HPC研究会@沖縄産業支援センター，2014年10月

研究背景

- アクセラレータクラスタ上のプログラミング
  - MPI and (**OpenACC** or OpenCL or CUDA)
    - **OpenACC**は**指示文ベース**であるため,
      コードの変更量が少なく，わかりやすい
    - アクセラレータプログラミングの生産性が高い
  - しかしMPIは，各プロセスに対するデータの分散や転送，ループ文のインデックス計算などを
    手動で行う必要があるため，その生産性は低い

筑波大学HA-PACS


東工大 TSUBAME2.5

本発表の目的

- アクセラレータクラスタにおける新しいプログラミングモデル
  XcalableACC（XACC）の提案
  - （3月の和倉温泉での提案内容をブラッシュアップ）
  - XcalableACC = XcalableMP（XMP） + OpenACC + α
- 開発動機：アクセラレータクラスタ用アプリケーションを簡易に作成したい
- この手法を取った理由：
  - XMPはOpenACCと同様に**指示文ベース**の並列言語
  - 逐次コードにXMP指示文とOpenACC指示文を挿入することで，
    アクセラレータクラスタ用アプリケーションを簡易に作成できるはず
  - αの中の1つはXMPに対する拡張で，アクセラレータ上のデータ通信を
    サポート
- 本発表では，XACCの紹介とその有効性を検証するための初期評価を行う

発表の流れ

- XMPとOpenACCの概要
- XACCについて
- XACCコンパイラの実装
- XACCの初期評価

  姫野ベンチマークの実装

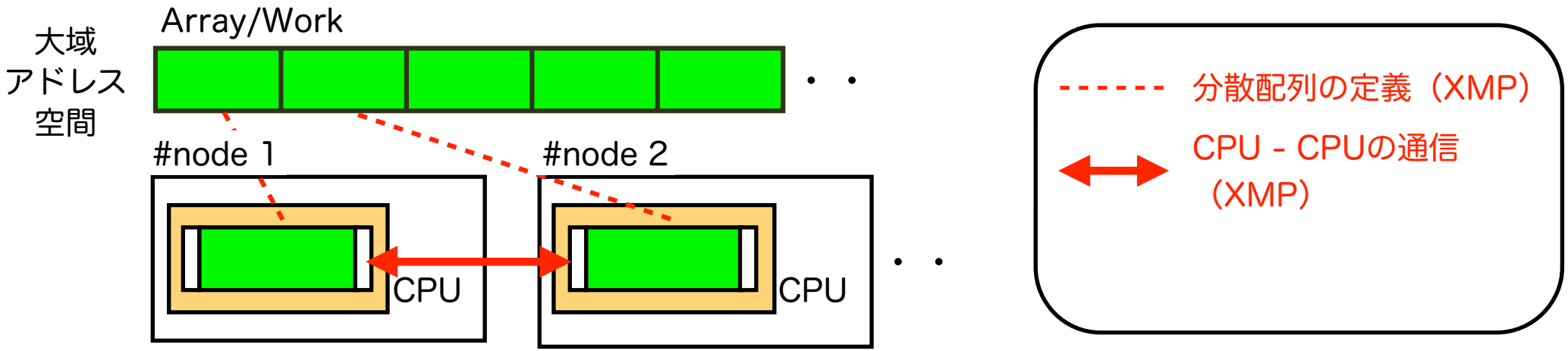
発表の流れ

- XMPとOpenACCの概要
- XACCについて
- XACCコンパイラの実装
- XACCの初期評価

  姫野ベンチマークの実装

XcalableMPのメモリモデル

- 題目：XcalableACC: OpenACCを用いたアクセラレータクラスタのための**PGAS言語**XcalableMPの拡張
  - Partitioned Global Address Space Language
  - 異なるプロセスからアクセスできるメモリ領域（大域アドレス空間）を持つようなプログラミングモデル
  - XMPのメモリモデルは下図

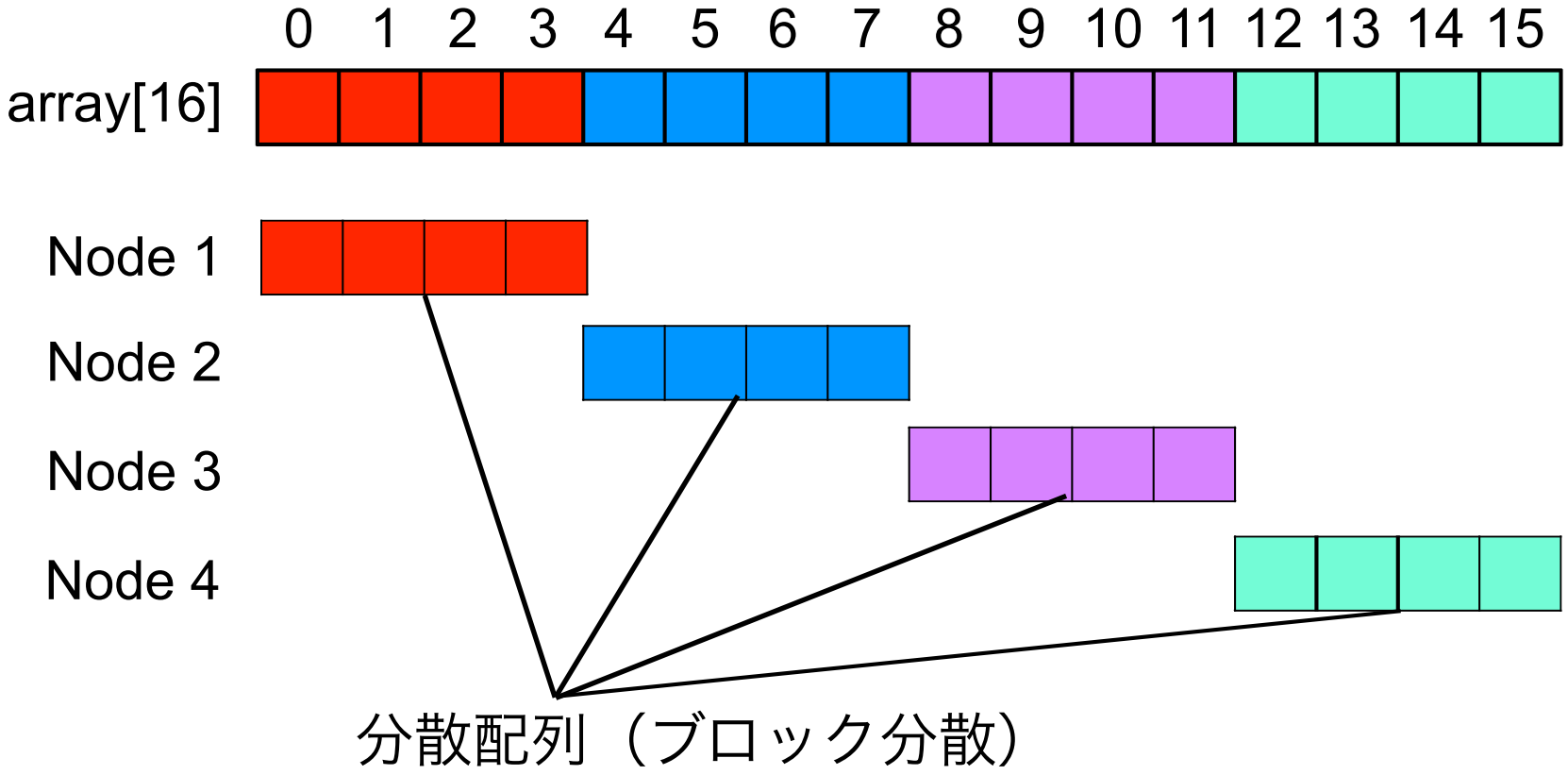


XcalableMP（1/3）

- 分散配列の定義

  通常の配列に指示文で
  分散を指定

```
int array[16];
#pragma xmp nodes p(4)
#pragma xmp template t(0:15)
#pragma xmp distribute t(block) on p
#pragma xmp align array[i] with t(i)
```

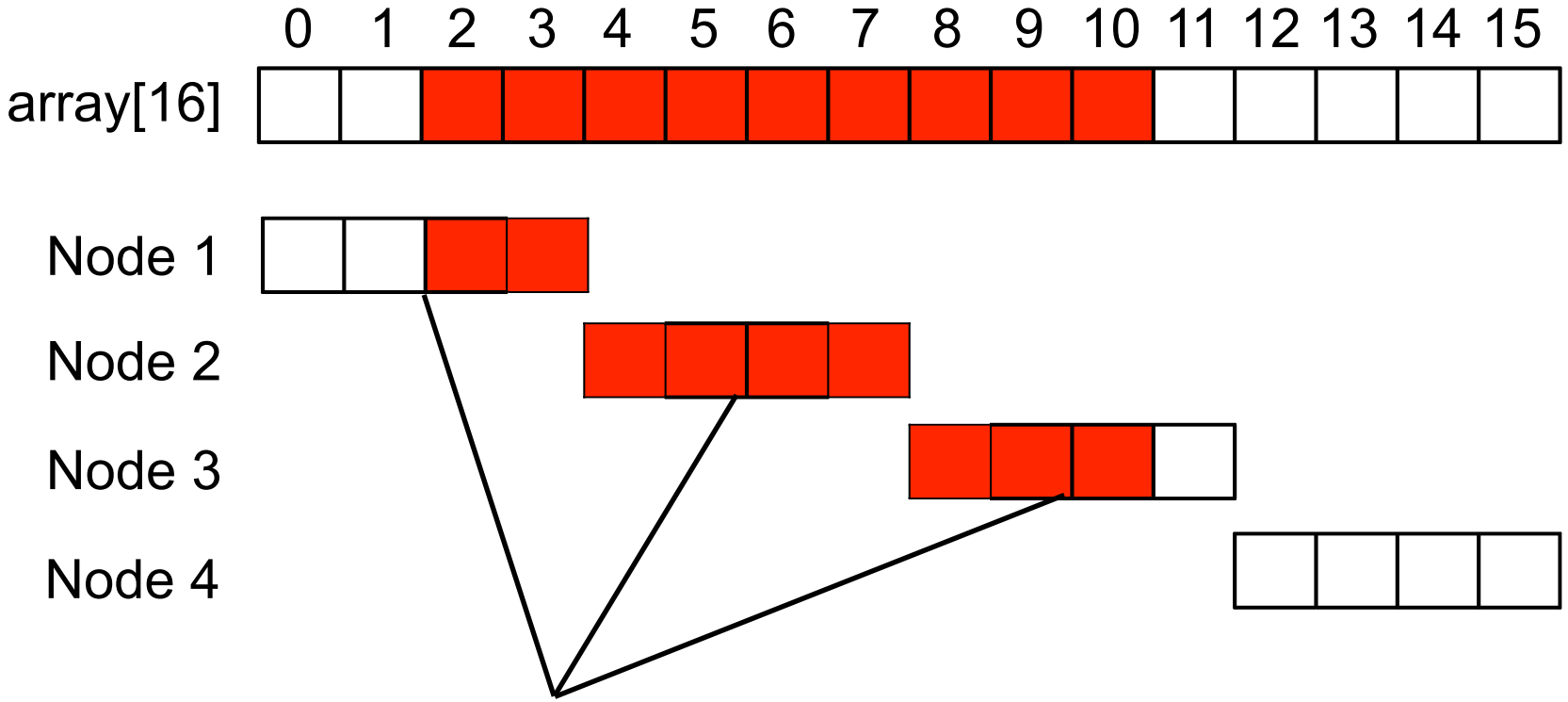


XcalableMP（2/3）

- ループの並列化

```
#pragma xmp loop on t(i)
for(i=2;i<=10;i++){...}
```

```
int array[16];
#pragma xmp nodes p(4)
#pragma xmp template t(0:15)
#pragma xmp distribute t(block) on p
#pragma xmp align array[i] with t(i)
```

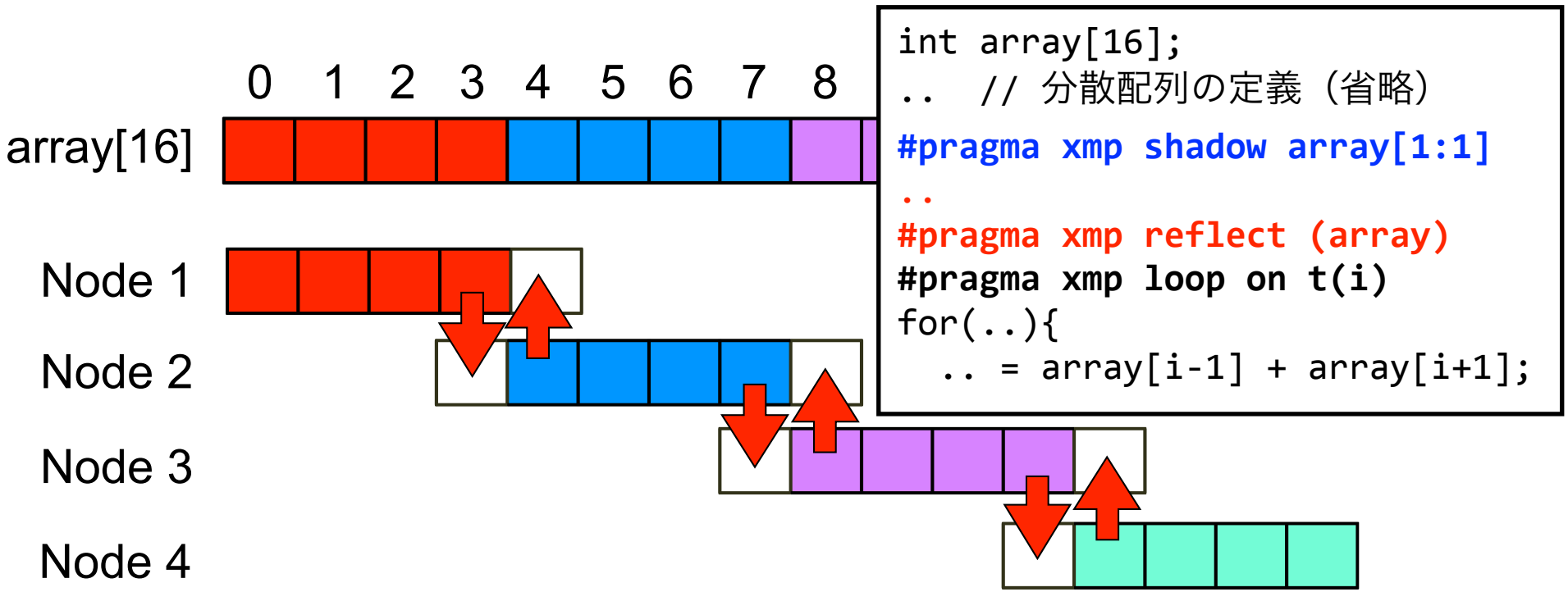


分散配列を持っているノードのみが、赤の要素を処理する

XcalableMP（3/3）

通信指示文：broadcast, reduction, gmove, reflect

reflect指示文: ステンシル計算のための袖交換を行う



1. 袖を定義する（shadow）． 2. 袖を交換する（reflect）

OpenACCとは

- 指示文ベースのプログラミングモデル
- GPU以外のアクセラレータにも対応できるように規格化されている
- 機能
  - ホストメモリとアクセラレータメモリとのデータ転送
  - アクセラレータに対する計算処理
- コンパイラ
  - 商用：PGI，Cray，HMPP
  - 無償：Omni，accULL，OpenUH

OpenACC
Directives for Accelerators

```
double a[N], b[N], c[N];
#pragma acc data copy(a,b,c)
{
#pragma acc parallel loop
  for(i=0; i<N; i++){
    c[i] = a[i] + b[i];
    ...
```

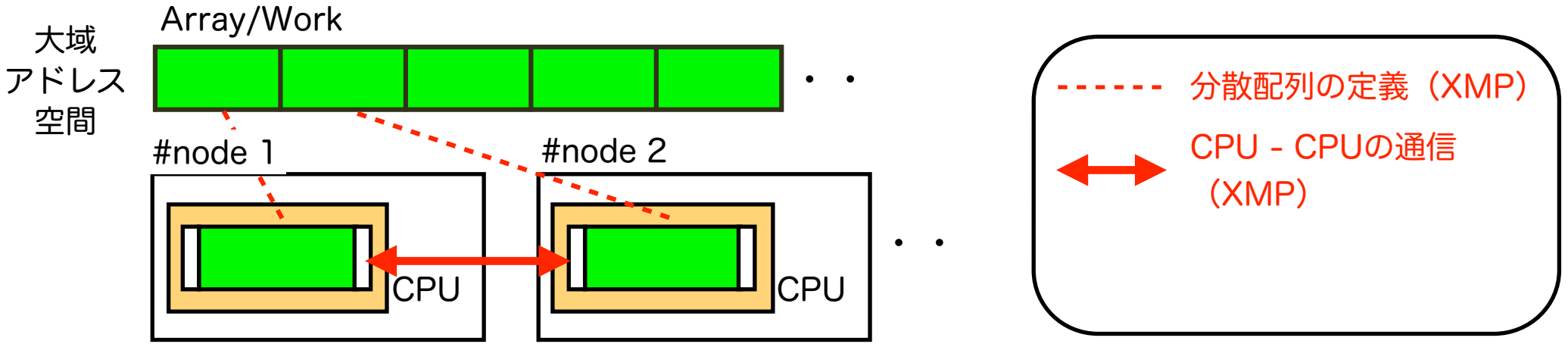
発表の流れ

- XMPとOpenACCの概要
- XACCについて
- XACCコンパイラの実装
- XACCの初期評価

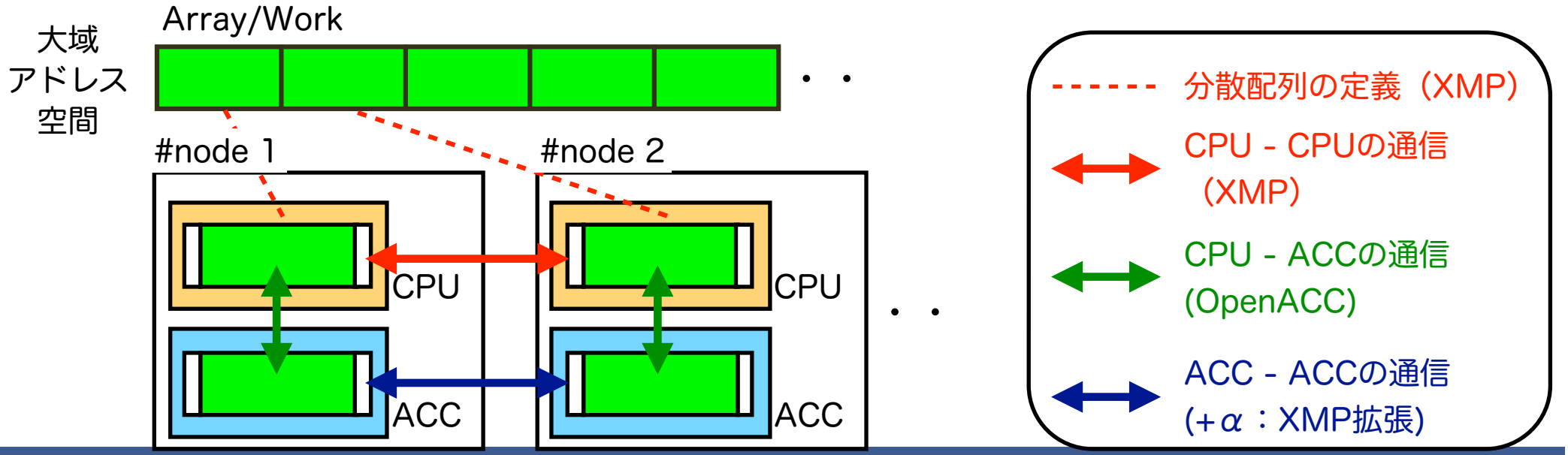
  姫野ベンチマークの実装

XcalableACCのメモリモデル

- XMPのメモリモデル（再掲）



- XACC（XMP+OpenACC+α）のメモリモデル



XcalableACCのプログラミング例

```
double u[XSIZE][YSIZE], uu[XSIZE][YSIZE];

…

  for(k=0; k<MAX_ITER; k++){

   for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        uu[x][y] = u[x][y];

    for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        u[x][y] = (uu[x-1][y]+uu[x+1][y]+
                   uu[x][y-1]+uu[x][y+1])/4.0;
  } // end k
```

2次元ラプラス方程式

XcalableACCのプログラミング例

```
double u[XSIZE][YSIZE], uu[XSIZE][YSIZE];
#pragma xmp nodes p(x, y)
#pragma xmp template t(0:YSIZE-1, 0:XSIZE-1)
#pragma xmp distribute t(block, block) onto p
#pragma xmp align [j][i] with t(i,j) :: u, uu
#pragma xmp shadow uu[1:1][1:1]
…


  for(k=0; k<MAX_ITER; k++){
#pragma xmp loop (y,x) on t(y,x)

   for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        uu[x][y] = u[x][y];

#pragma xmp reflect (uu)

#pragma xmp loop (y,x) on t(y,x)

    for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        u[x][y] = (uu[x-1][y]+uu[x+1][y]+
                   uu[x][y-1]+uu[x][y+1])/4.0;
  } // end k
```

2次元ラプラス方程式

2次元分散配列と袖の定義

配列uuの袖交換

XcalableACCのプログラミング例

```
double u[XSIZE][YSIZE], uu[XSIZE][YSIZE];
#pragma xmp nodes p(x, y)
#pragma xmp template t(0:YSIZE-1, 0:XSIZE-1)
#pragma xmp distribute t(block, block) onto p
#pragma xmp align [j][i] with t(i,j) :: u, uu
#pragma xmp shadow uu[1:1][1:1]
…
#pragma acc data copy(u) copyin(uu)
{
  for(k=0; k<MAX_ITER; k++){
#pragma xmp loop (y,x) on t(y,x)
#pragma acc parallel loop collapse(2)
   for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        uu[x][y] = u[x][y];

#pragma xmp reflect (uu) acc

#pragma xmp loop (y,x) on t(y,x)
#pragma acc parallel loop collapse(2)
    for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        u[x][y] = (uu[x-1][y]+uu[x+1][y]+
                   uu[x][y-1]+uu[x][y+1])/4.0;
  } // end k
} // end data
```

2次元ラプラス方程式

2次元分散配列と袖の定義

分散配列をアクセラレータのメモリに転送

XMP指示文で分散したループをOpenACC指示文が分散して処理

配列uuの袖交換

acc節を指定すると，アクセラレータ上のデータが送信される

XcalableACCのプログラミング例

```
double u[XSIZE][YSIZE], uu[XSIZE][YSIZE];




…
#pragma acc data copy(u) copyin(uu)
{
  for(k=0; k<MAX_ITER; k++){

#pragma acc parallel loop collapse(2)
   for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        uu[x][y] = u[x][y];




#pragma acc parallel loop collapse(2)
    for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        u[x][y] = (uu[x-1][y]+uu[x+1][y]+
                   uu[x][y-1]+uu[x][y+1])/4.0;
  } // end k
} // end data
```

2次元ラプラス方程式

分散配列をアクセラレータのメモリに転送

XMP指示文で分散したループをOpenACC指示文が分散して処理

発表の流れ

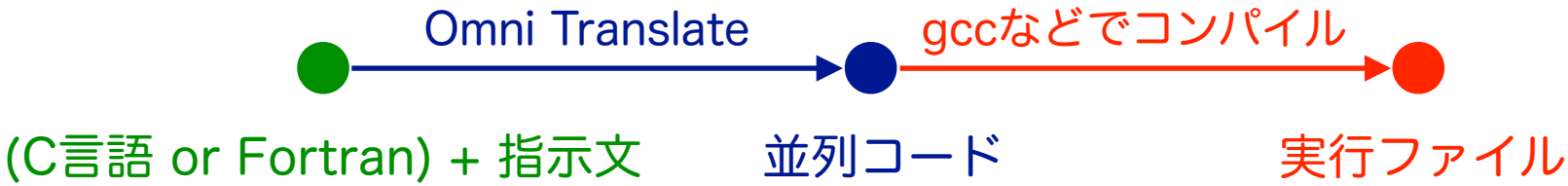
- XMPとOpenACCの概要
- XACCについて
- XACCコンパイラの実装
- XACCの初期評価

  姫野ベンチマークの実装

実装概要

- Omniコンパイラ（http://omni-compiler.org）を拡張する

  XMP，OpenACC，OpenMPを処理できるsource-to-sourceコンパイラ



- 今回の拡張項目（時間の都合上，説明は3.のみ）

  1. XMPの分散配列をOpenACCのdata指示文に指定可能にする
  2. XMPのloop指示文とOpenACCのloop指示文との同時利用
  3. アクセラレータ上の分散配列に対する袖交換（reflect指示文）

実装詳細（1/4）

3. アクセラレータ上の分散配列に対する袖交換（reflect指示文）

```
#pragma acc data copy(u) copyin(uu)
{
  ..
#pragma xmp reflect (uu) acc

#pragma xmp loop (y,x) on t(y,x)
#pragma acc parallel loop collapse(2)
    for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        u[x][y] = (uu[x-1][y]+uu[x+1][y]+
                   uu[x][y-1]+uu[x][y+1])/4.0;
```

配列uuの袖交換

acc節を指定すると，アクセラレータ上のデータが送信される

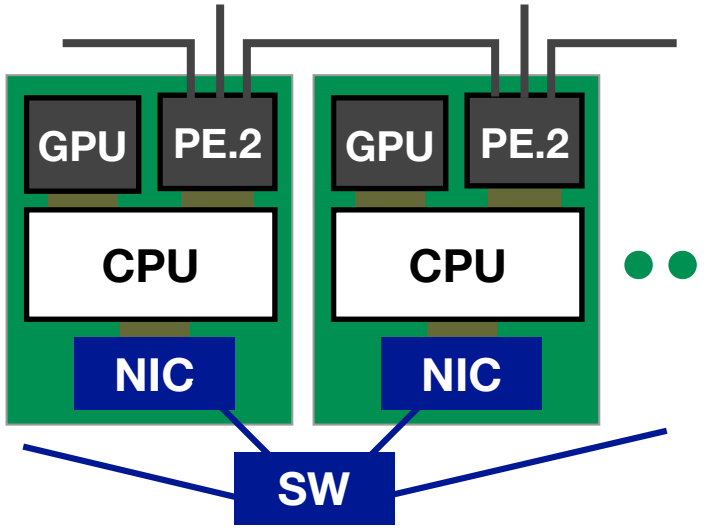
- 3つの実装方法
  1. Tightly Coupled Accelerators (TCA)（PEACH2）
  2. MVAPICH2-GDR（GPUDirect RDMAを利用）
  3. CUDAでホストメモリにコピーした後，MPIで転送

小さい転送量の場合，上ほど高速になると考えられる

実装詳細（2/3）：TCAとは

- アクセラレータ間の直接通信
- PEACH2（PCIe Adaptive Communication Hub ver. 2）
  - TCAインタフェースボード
  - Altera社Stratix IV GX（FPGA）

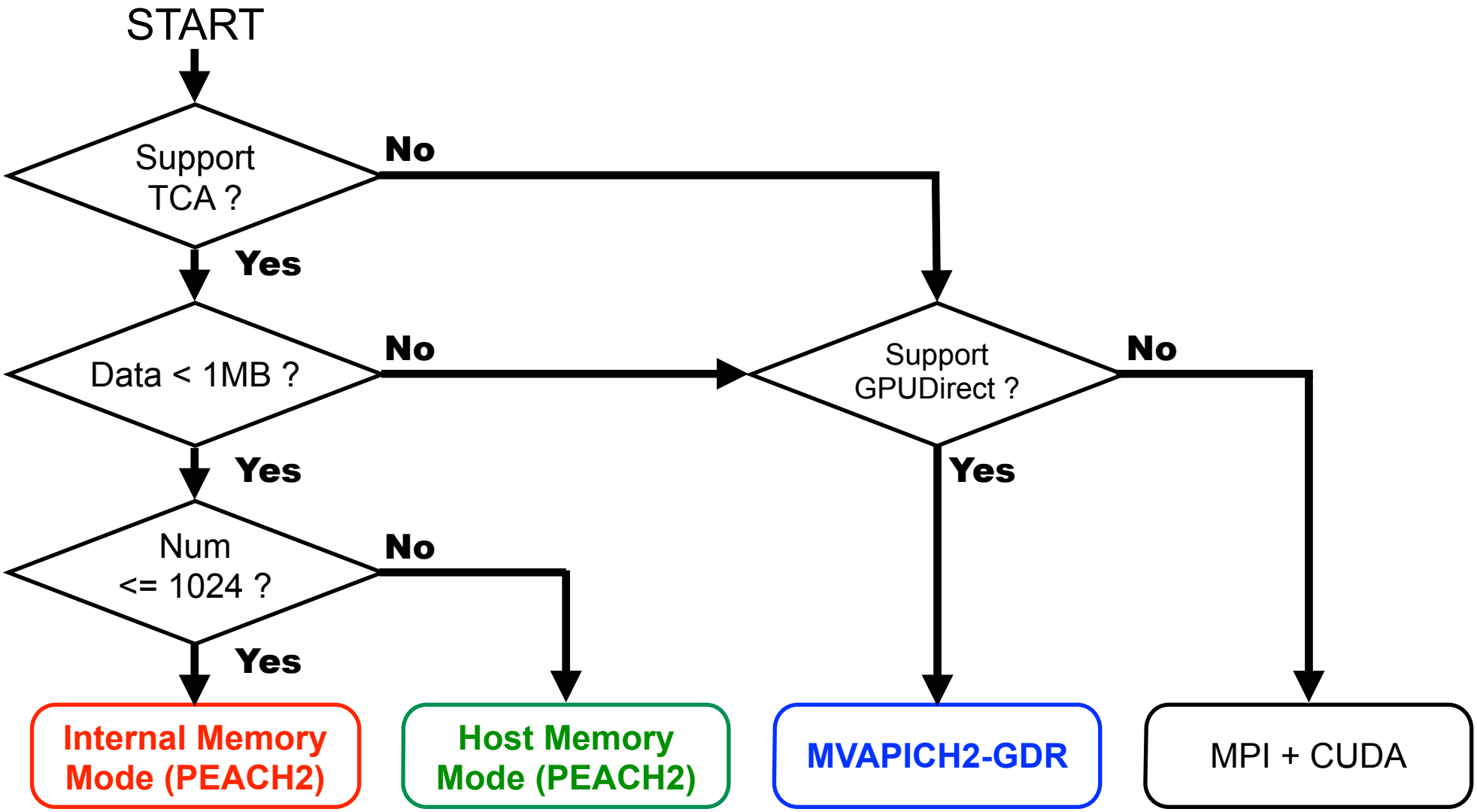
    PCIe Gen2 x8レーン x4ポート
  - PEACH2同士をPCIe外部ケーブルで相互接続
  - Direct Memory Accessで隣接GPUに通信
    - 内蔵メモリモードとホストメモリモード
    - 内蔵メモリモードの方が速い
    - 内蔵メモリモードのディスクリプタの数は1024個まで



PE.2 = PEACH2

実装詳細（3/3）：通信切替のフロー

袖交換時に，どのように通信モードを切り替えるか？



発表の流れ

- XMPとOpenACCの概要
- XACCについて
- XACCコンパイラの実装
- XACCの初期評価
  - 姫野ベンチマークの実装

XcalableACCの評価（1/3）

- 姫野ベンチマークの実装
  - ステンシルアプリケーションベンチマーク
  - ポアッソン方程式をヤコビの反復法で解く場合の処理速度

```
float p[MIMAX][MJMAX][MKMAX];
// XMP指示文を用いて分散配列の定義
#pragma xmp shadow p[1:1][1:1][0]

#pragma acc data copy(p) ..
{
..
#pragma xmp reflect (p) acc
..
#pragma xmp loop (k,j,i) on t(k,j,i)
#pragma acc parallel loop ..
for(i=1 ; i<MIMAX ; ++i)
  for(j=1 ; j<MJMAX ; ++j){
#pragma acc loop vector ..
    for(k=1 ; k<MKMAX ; ++k){
      S0 = p[i+1][j][k] * ..;
```

袖領域の定義

アクセラレータに分散配列を転送

アクセラレータメモリの袖交換

ループの分散処理

逐次版姫野ベンチマークに
対して，指示文を挿入するのみ

XcalableACCの評価（2/3）

- 行数

	トータル	XMP	OpenACC	指示文以外
XACC	204	22	9	173
OpenACC + MPI	325	-	15	310

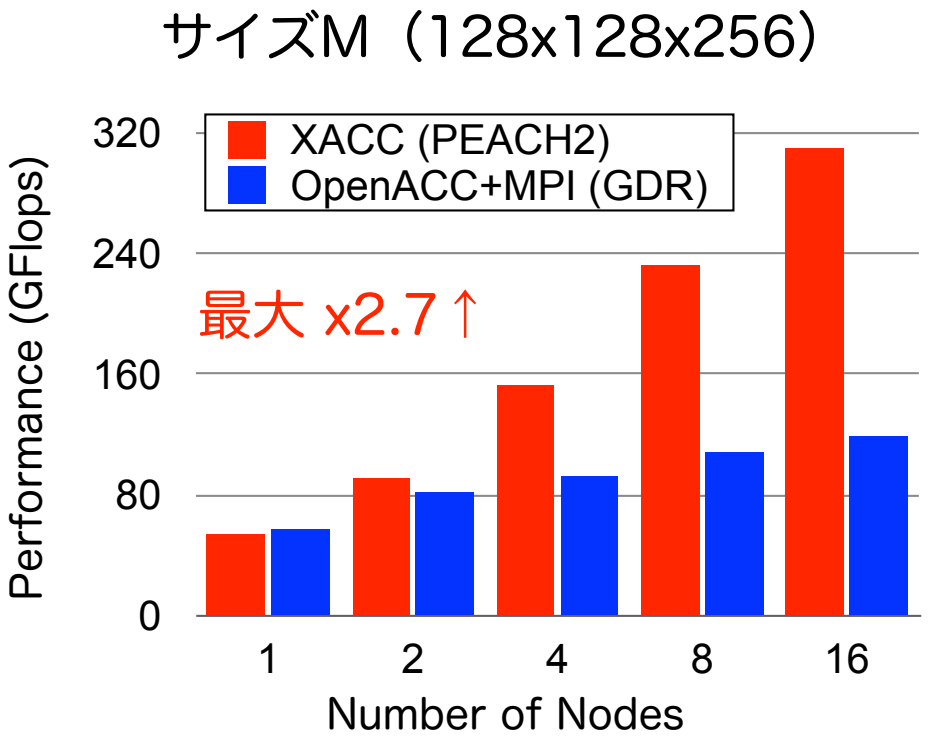
- XACCは逐次版の姫野ベンチマークに指示文を挿入するのみ
  - 逐次のイメージを保ったまま，アクセラレータクラスタで動作可能
- XACCとの比較として，MPI版の姫野ベンチマークにOpenACC指示文を挿入してアクセラレータクラスタ用に変更（典型的な並列化方法）
  - MPVAPICH2-GDRを用いるため，OpenACC指示文をMPI関数の前に挿入
  - OpenACC + MPIにおいて指示文以外の行数が多い理由は、ループ文のインデックスの計算や通信のための座標計算を行う必要があるため

XcalableACCの評価（3/3）

姫野ベンチマークのサイズS，M，Lで評価（時間の都合上サイズMのみ紹介）
※ 強スケーリング問題

筑波大学HA-PACS

- NVIDIA K20X
- InfiniBand QDR 2rails
- MVAPICH-GDR2.0b
- gcc-4.7, CUDA6.0



XMPはPEACH2を簡易に用いることができる.
C言語などからPEACH2を利用することも可能だが,
さらにプログラミングコストが高くなる
※OpenACC + PEACH2の性能はXACCとほぼ同じ

まとめ

- アクセラレータクラスタ用の新しいプログラミングモデルの提案
- 指示文ベースのXMPとOpenACCの混合利用
- XMPを拡張し、通信指示文にacc節を加えることで,
  アクセラレータメモリ上のデータを転送可能に
  - データ転送は，高速と考えられる通信機構を利用
- 姫野ベンチマークを用いた生産性の評価
  - XACCは逐次のイメージを保ったまま，アクセラレータクラスタ用の
    アプリケーションを作成可能

今後の課題

- 袖交換以外のアクセラレータ間の通信も開発する（bcastなど）
- ミニアプリ and/or 実アプリを用いたXACCの評価を大規模な環境で行う
- マルチGPUも使えるようにする（OpenACC拡張）
  - XACC -> XMP + OpenACC + α
  - α -> XMP拡張（アクセラレータ間の通信） + OpenACC拡張
  - 動機：
    - OpenACCのモデルはマルチGPUを扱いにくい
      - acc_set_device_num()を利用してデバイスをstatement毎に指定
      - ループ文のインデックス計算やGPU間のコピーは、もちろん手動

    ➡ XMPの考え方を基にOpenACCを拡張

    （詳しくは次回以降のHPC研究会 or SC14のAICSブースで）