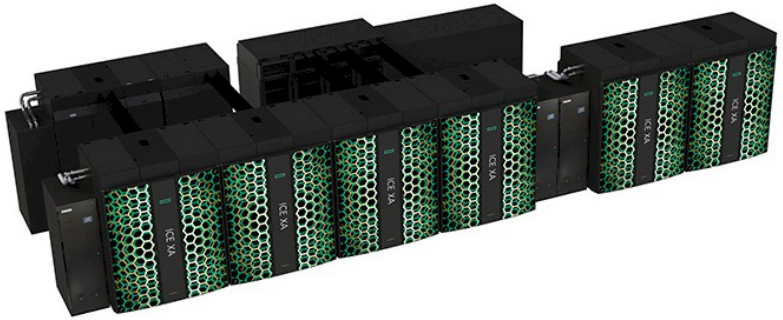
Tightly Coupled Accelerators/InfiniBandハイブリッド通信を用いたアクセラレータクラスタ用並列言語XcalableACCの評価

中尾 昌広[1], 小田嶋 哲哉[1], 村井 均[1], 田渕 晶大[2‡], 藤田 典久[3], 塙 敏博[4], 朴 泰祐[2], 佐藤 三久[1]

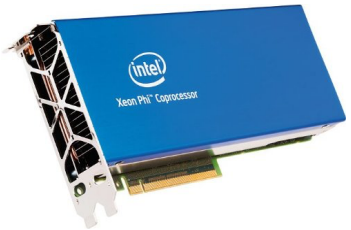
1. 理化学研究所 計算科学研究センター
2. 筑波大学 大学院 システム情報工学研究科
‡（現：富士通研究所）
3. 筑波大学 計算科学研究センター
4. 東京大学 情報基盤センター

研究背景（1/2）

- アクセラレータクラスタシステムの台頭
  - 優れた電力性能比とメモリバンド幅
  - Top500やGreen500の上位のほとんどは
    アクセラレータクラスタ

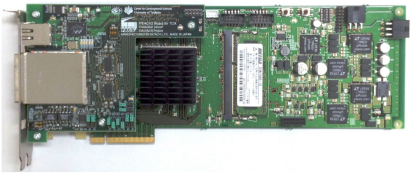


- アクセラレータクラスタにおける課題
  1. ノードを跨ぐアクセラレータ間の通信レイテンシの削減
     - 強スケーリングにおける性能が重要になりつつある
  2. 低いポータビリティと複雑なプログラミングの簡易化
     - GPUクラスタの場合，CUDA+MPIが一般的
     - CUDAはNVIDIAのGPUしかサポートしていない
     - MPIはプリミティブな関数しか提供していない
     - OpenACC+MPIも利用されつつあるが，MPIは難しいまま



研究背景（2/2）

- Tightly Coupled Accelerators（TCA）の提案
  - 密結合並列演算加速機構
  - 低レイテンシでアクセラレータ間の通信を実現する技術
  - PEACH2（PCIe Adaptive Communication Hub ver. 2）
  - PEACH2同士をPCIe外部ケーブルで相互接続



Altera社Stratix IV GX

- 並列言語XcalableACC（XACC）の開発
  - 並列言語XcalableMP（XMP）に対するOpenACCを用いた拡張
  - 分散配列の定義・通信などの機能を持つ指示文を逐次コードに挿入
  - 簡易に並列アプリケーションを作成可能

```
int main(){
  double a[MAX];
#pragma xmp align a[i] with t[i]
#pragma acc enter data copyin(a)

#pragma xmp loop on t[i]
#pragma acc parallel loop
  for(int i=0; i<MAX; i++){
    a[i] = i;
     :
```

研究目的

- TCA/InfiniBand(TCA/IB)ハイブリッド通信を用いたXACCの評価
  - 低レイテンシのTCAと高バンド幅のIBを組合せた通信
    （詳細は後のスライド）
  - TCA/IBハイブリッド通信を利用できる機能をXACCに実装
  - 高性能と高生産性の両立

- 64計算ノードで構成されたGPUクラスタを用いて，XACCと既存のプログラミングモデルであるCUDA+MPIとOpenACC+MPIとの比較を行い，XACCの性能と生産性について考察する

目次

- 研究背景と研究目的
- TCAとTCA/IBハイブリッド通信の性能評価
- XACCプログラミングモデル
- 評価
  - 性能について
  - 生産性について

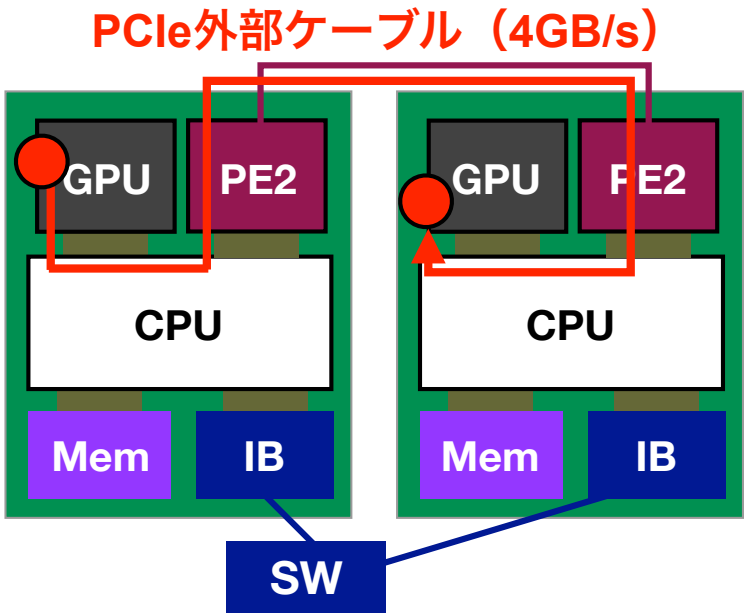
TCAの性能評価（IBとの性能比較）

- HA-PACS/TCA
  - 筑波大学計算科学研究センターで運用
    （2018年3月まで）
  - GPUクラスタ（NVIDIA Tesla K20X）
  - 共用クラスタとして，一般のアプリも
    サポートするため，IB FDRも搭載
- IBの評価にはCUDA-Aware MPIである
  MVAPICH2-GDRを利用
  - MVAPICH2-GDRの環境変数を設定し，
    通信のチューニングを実施
    （詳細は研究報告の付録を参照）
- 連続通信とブロックストライド通信の性能比較

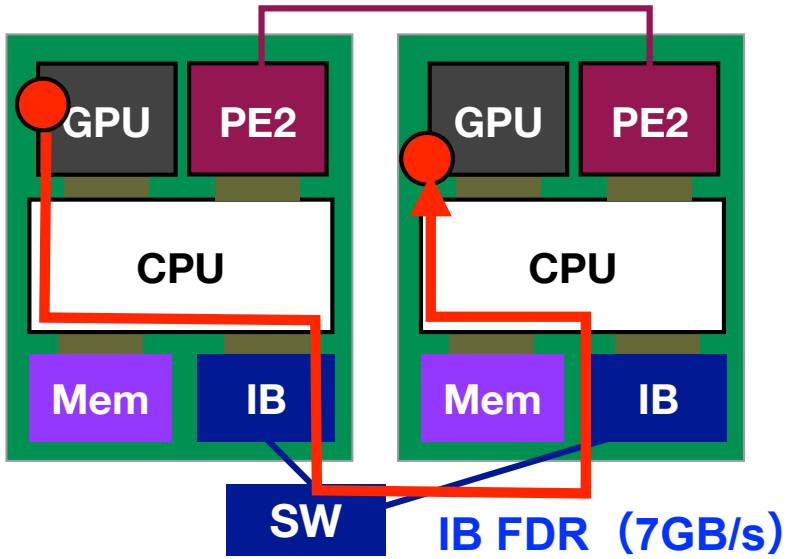
CPU	Intel Xeon-E5 2680v2 2.8 GHz x 2 Sockets
Memory	DDR3 1866 MHz x 4 channel, 128GB
GPU	NVIDIA Tesla K20X x 4 GPUs, GDDR5 6GB
Network	InfiniBand FDR 7GB/s
Software	Intel compiler 16.0.4, CUDA 7.5.18 MVAPICH2-GDR 2.2

TCAとGPUDirect RDMA（HA-PACS/TCA）

- TCA（PEACH2）

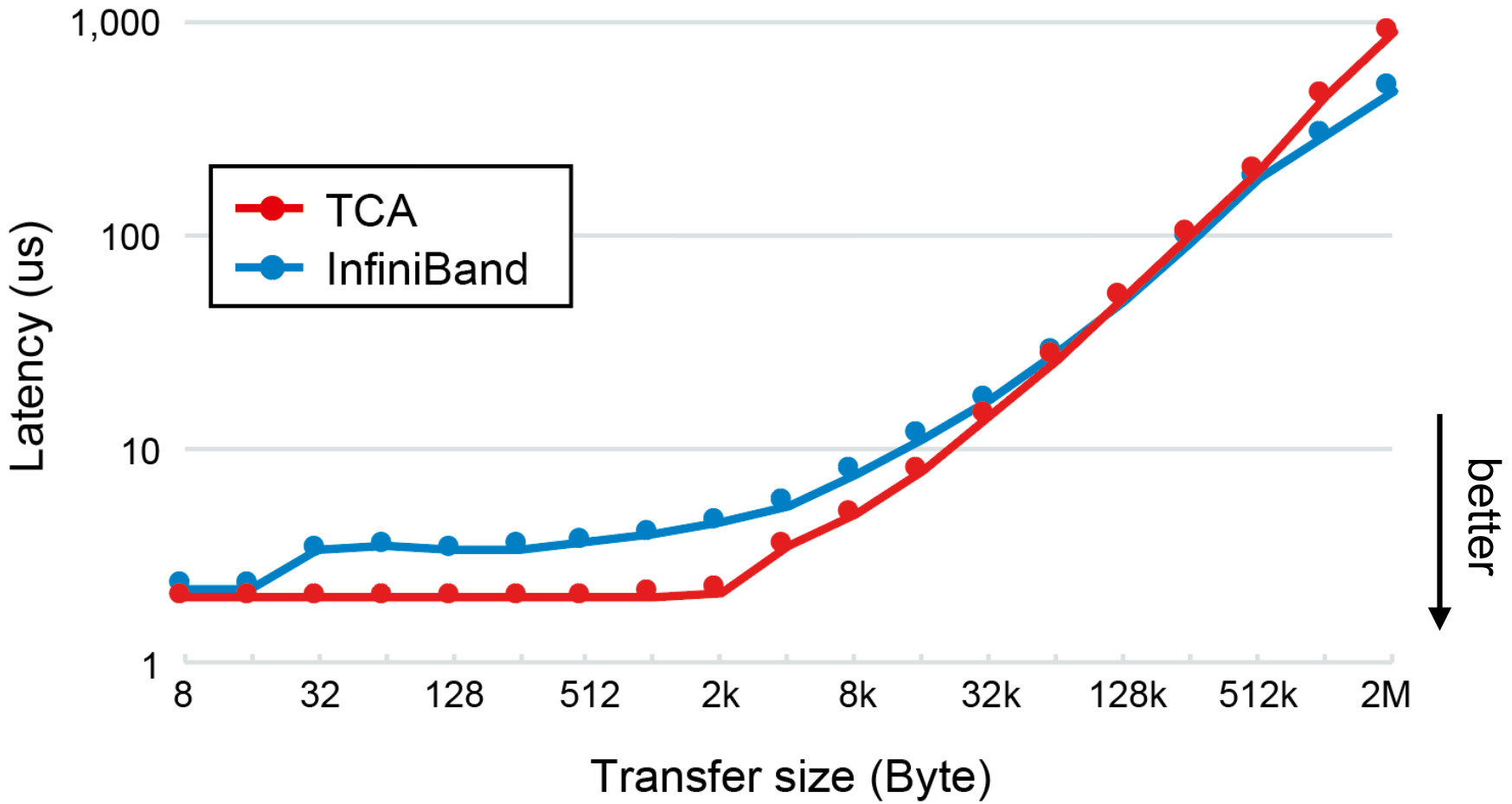


- MVAPICH2-GDR



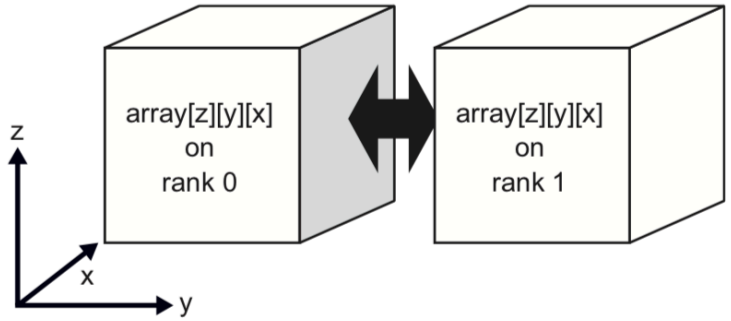
- TCAは通信プロトコルにPCIeを利用
- PEACH2が用意している専用のAPIを用いて操作する（C言語）
- PEACH2は高性能なDMAコントローラを搭載
  - 小データの通信やブロックストライド通信において高性能
  - **ただし，PEACH2はPCIe Gen2 x 8なので、最大バンド幅は4GB/s**

TCAとIBとの性能比較：連続通信

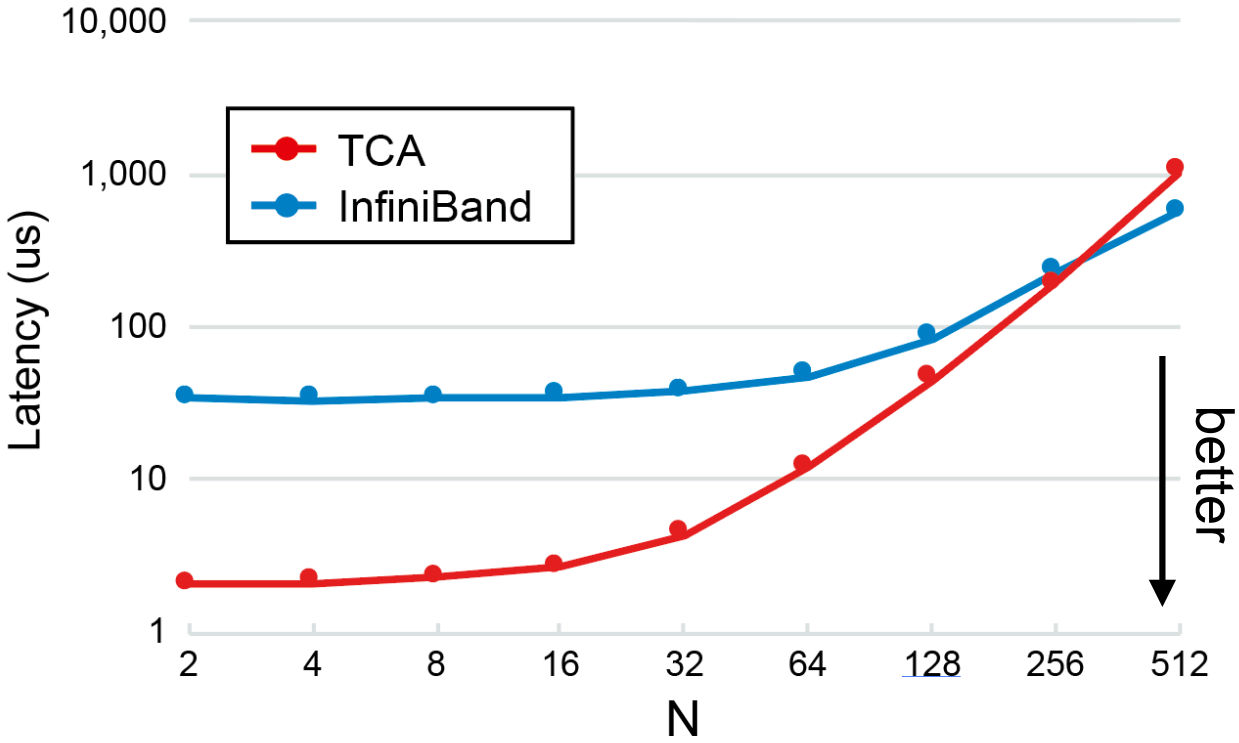


- 転送サイズが128 kBまで，TCAの方が高い性能を発揮する
- 転送サイズが128 kB以上では，最大バンド幅が異なるため，TCAはIBより遅い

TCAとIBとの性能比較：ブロックストライド通信



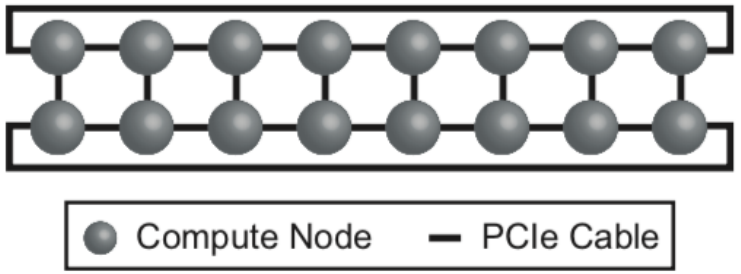
- 3次元配列に対する袖通信を想定した通信パターン
- 配列の各次元の要素数をNとすると，N要素の連続データ転送がN x N周期で現れる



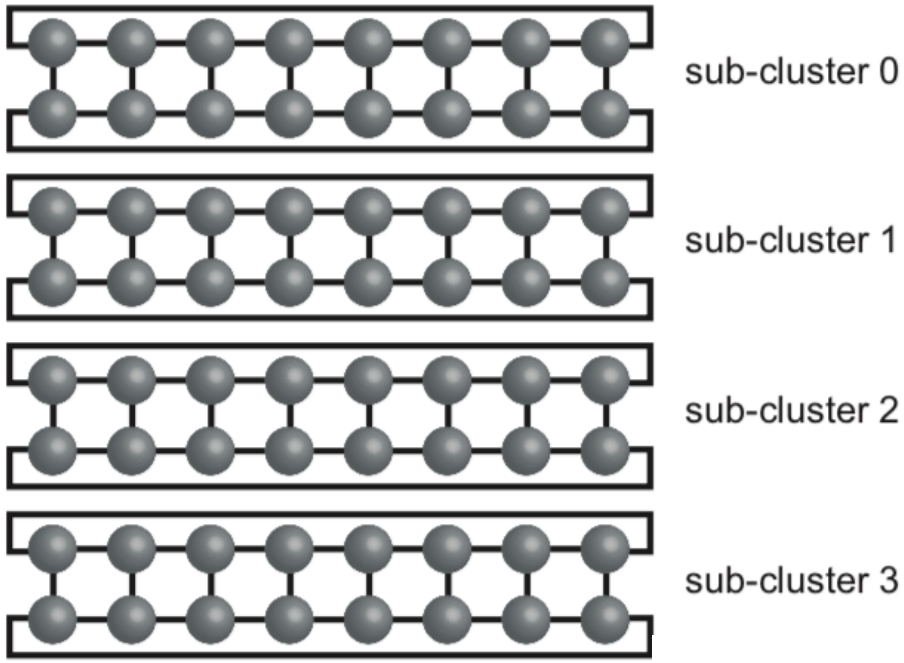
- IBの評価は，CUDAを使ってGPU上のバッファにpackし，それを送信し，相手プロセスでunpackしている（MPI_Datatypeを使うと遅いから）
- TCAはブロックストライド通信が高速に行える（pack/unpackしなくてよい）
- N=256（1要素は8バイト）まで，TCAの方が高い性能を発揮する

TCAの課題

- TCAだけではPCIe外部ケーブル長の限界やホップ数の増加に伴う性能低下により，大規模なクラスタは構成できない
  - 16計算ノードで「サブクラスタ」を構成
  - 3本のPCIe外部ケーブルが他の計算ノードと接続されている



- HA-PACS/TCAのネットワーク構成
  - 64計算ノード
  - 4つのサブクラスタ
  - 全計算ノードはIBでフラットに接続されている（図では省略）
  - サブクラスタ間はIB/MPIによる通信

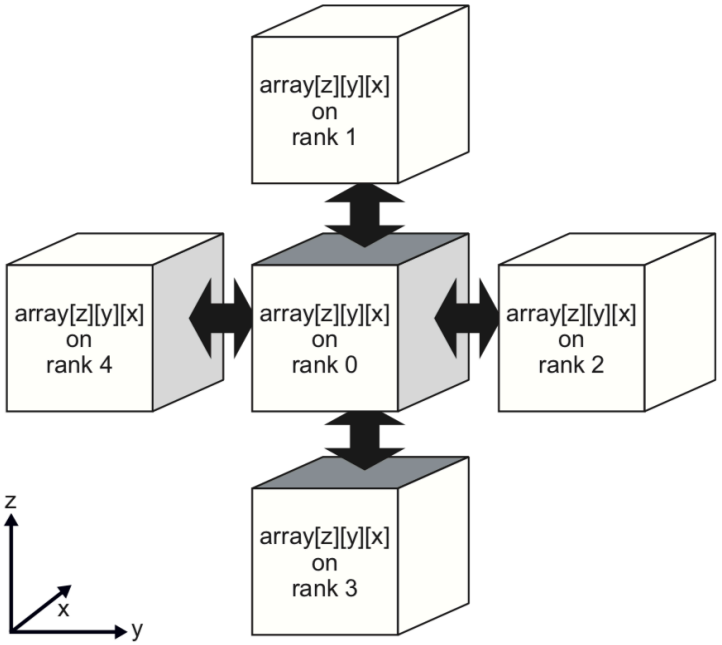


TCA/IBハイブリッド通信

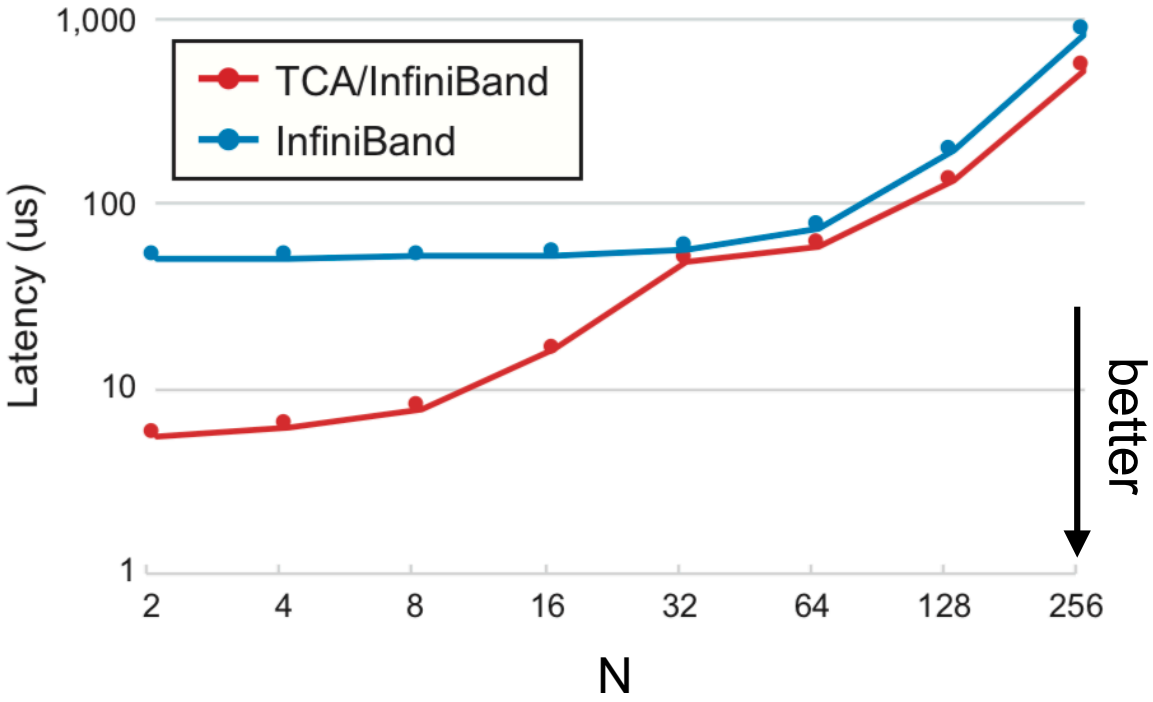
- TCAの課題
  - データサイズの大きな通信はTCAよりもIBの方が有利
  - 16計算ノードで構成されるサブクラスタ内の通信しか行えない

- TCA/IBハイブリッド通信
  - 低レイテンシのTCAと高バンド幅とスケーラビリティのIBとを組合せた通信
  - 条件に応じて，得意な通信路を用いる
    - サブクラスタ間通信は必ずIBを用いる
    - サブクラスタ内の場合は、データパターン（連続 or ストライド）や転送サイズに応じて，IBを使うかTCAを使うかを選択する
    - この2種類の通信路は同時に利用可能

TCA/IBハイブリッド通信とIBとの性能比較



- 3次元配列に対する袖通信を想定
- 横方向：ブロックストライド通信（TCA）
- 縦方向：連続通信（IB）



- 常にTCA/IBハイブリッド通信の方が高い性能を発揮する
- TCA/IBハイブリッド通信は2本の通信路を同時に用いており，さらにIBが不得手とするブロックストライド通信をTCAが行っているから

XACC with TCA/IBハイブリッド通信

- TCA/IBハイブリッド通信を用いると全体的なネットワークのレイテンシ削減が見込めるが，そのようなプログラムは難しい
  - TCAでは，ハードウェア資源の管理を考えながら，専用のAPIを用いたプログラミングを行う必要があり、さらにMPIも書く必要があるため，そんな面倒なプログラムは普通にはできない

- XACCの**既存の指示文**を用いることで，上記の通信をユーザは簡易に利用可能
  - 既存の指示文を使うことで，同じコードが異なるマシンでも動作する
  - TCAを搭載していない計算環境では，MPIで通信する

目次

- 研究背景と研究目的
- TCAとTCA/IBハイブリッド通信の性能評価
- XACCプログラミングモデル
- 評価
  - 性能について
  - 生産性について

XcalableACC（XACC）

- **アクセラレータ用並列言語XcalableACC (XACC)**
  - 並列言語XcalableMP（XMP）のアクセラレータ拡張
  - CとFortranに対応（C++も検討中）
  - XMPプログラム中にOpenACC指示文の記述を許す
  - XMPによって各ノードに分散された配列や演算を対象に，OpenACCによるデータ移動・演算オフローディングを行う
  - XMP指示文とOpenACC指示文は基本的に直交関係だが，単純なXMPとOpenACCの組合せではなく、協調動作できるように工夫
    - XACC = XMP + OpenACC + XACC extensions

XMP	クラスタ用並列言語（MPIの代わりに利用)
OpenACC	アクセラレータ用並列言語（CUDAの代わりに利用）
XACC extensions	アクセラレータ間の通信など（TCA/IBハイブリッド通信）

XACCのプログラミングモデル

- XMP：分散配列の定義やループ文の分割
- OpenACC：アクセラレータの利用

Parallell code in XACC/C

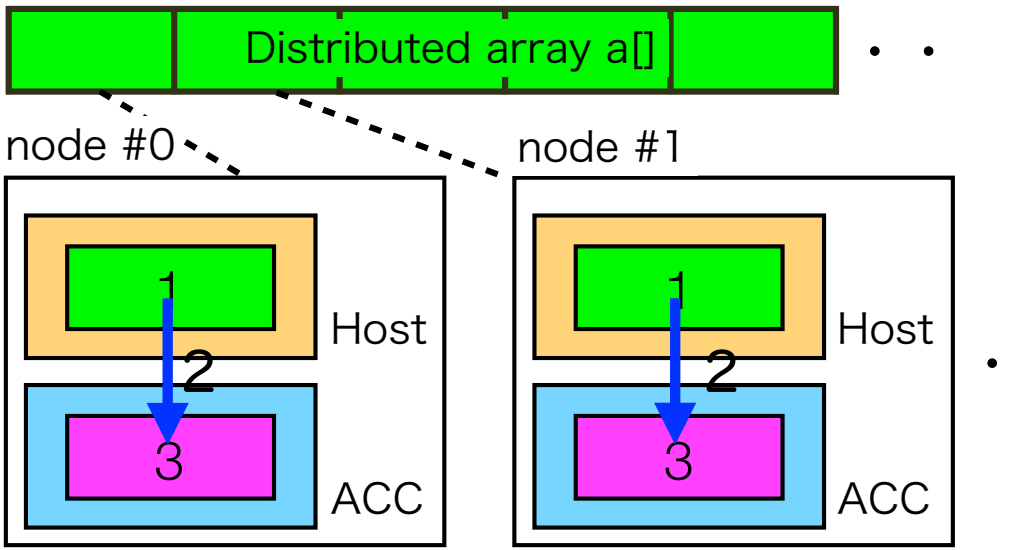
```
int main(){
  double a[MAX];
#pragma xmp nodes p[4]
#pragma xmp template t[MAX]
#pragma xmp distribute t[block] on p
#pragma xmp align a[i] with t[i]
#pragma acc enter data copyin(a)

#pragma xmp loop on t[i]
#pragma acc parallel loop
  for(int i=0; i<MAX; i++){
    a[i] = i;
      :
```

1

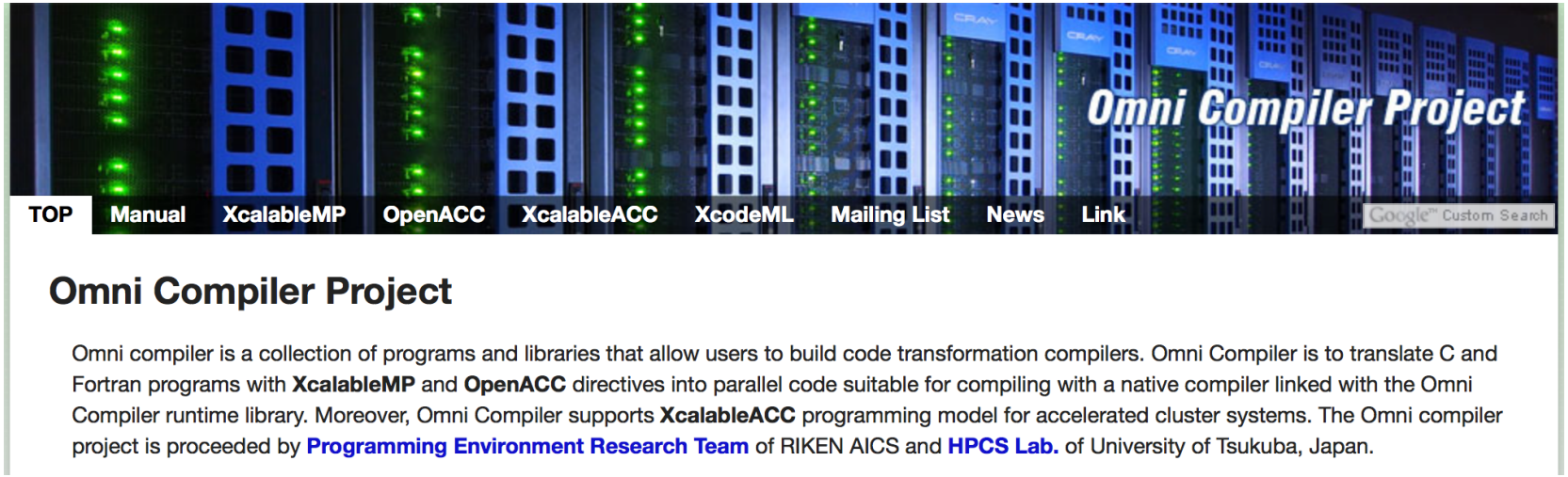
2

3

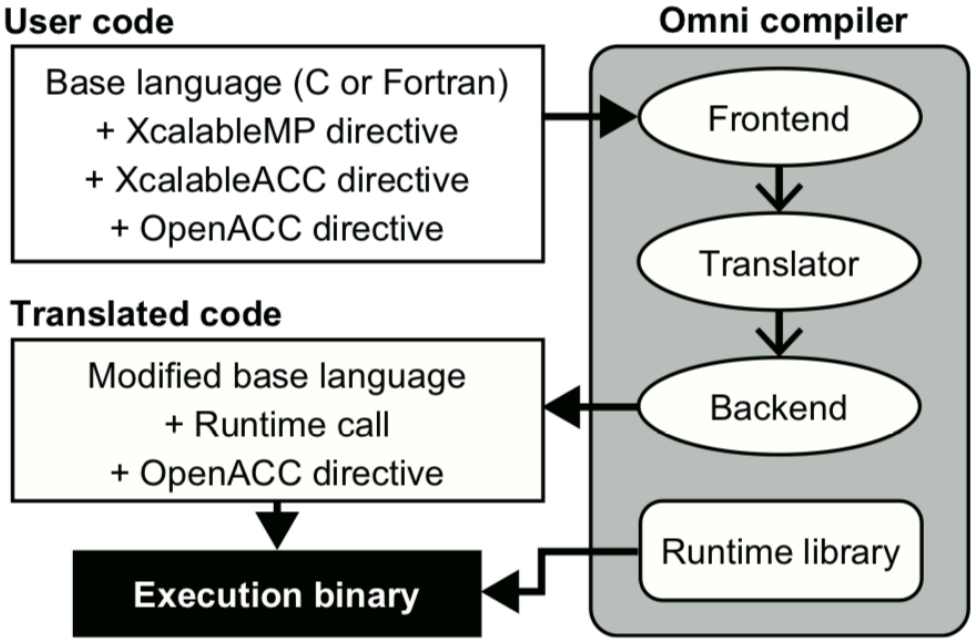


Omni Compiler

https://omni-compiler.org



- Source-to-source compiler
  - OpenACC以外の指示文は,
    ランタイムコールにコード変換
  - 任意のOpenACC compilerを
    用いて変換されたコードを
    コンパイルする
- TCA/IBハイブリッド通信を追加

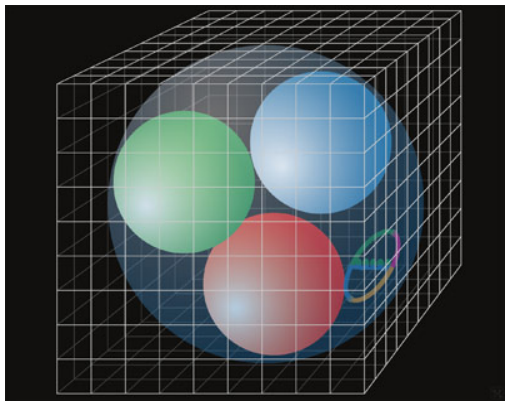


目次

- 研究背景と研究目的
- TCAとTCA/IBハイブリッド通信の性能評価
- XACCプログラミングモデル
- 評価
  - 性能について
  - 生産性について

Lattice QCD（LQCD）ミニアプリケーション

- HPC分野で重要なアプリケーションの1つ
- QCD（Quantum Chromo-Dynamics：量子色力学）は物質の最小単位であるクォークと，クォーク間における相互作用を結ぶグルーオン（糊粒子）を表す基本方程式

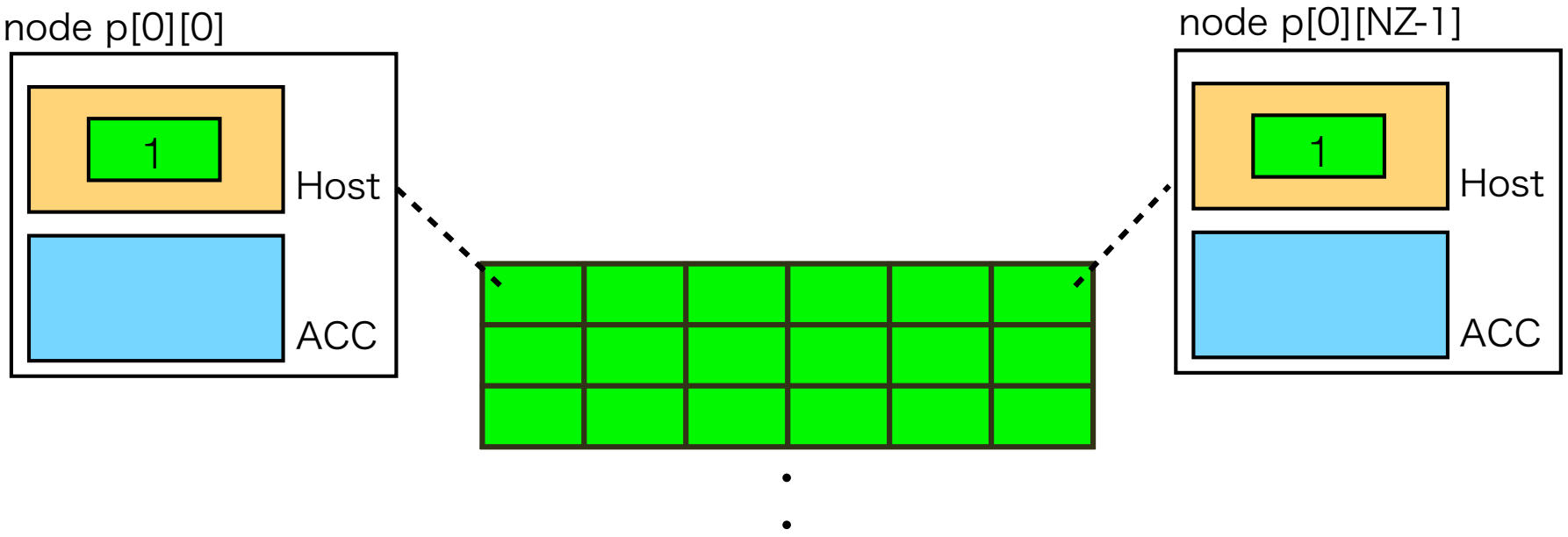


- LQCDは4次元（時間+XYZ軸）の格子上で行うQCDのシミュレーション
- LQCDミニアプリケーション
  - C言語，逐次コード，840行程度
  - 実アプリケーションBridge++のカーネル部分を抽出
  - http://research.kek.jp/people/matufuru/Research/Programs/index.html
- 既存のLQCDミニアプリケーションをベースにXACC，CUDA+MPI，OpenACC+MPIを使ってアクセラレータクラスタ用にそれぞれ実装した

分散配列の定義

```
Quark_t v[NT][NZ][NY][NX];
#pragma xmp template t[NT][NZ]
#pragma xmp nodes p[PT][PZ]
#pragma xmp distribute t[block][block] onto p
#pragma xmp align v[i][j][*][*] with t[i][j]
#pragma xmp shadow v[1][1][0][0]
#pragma acc enter data copyin(v)
```

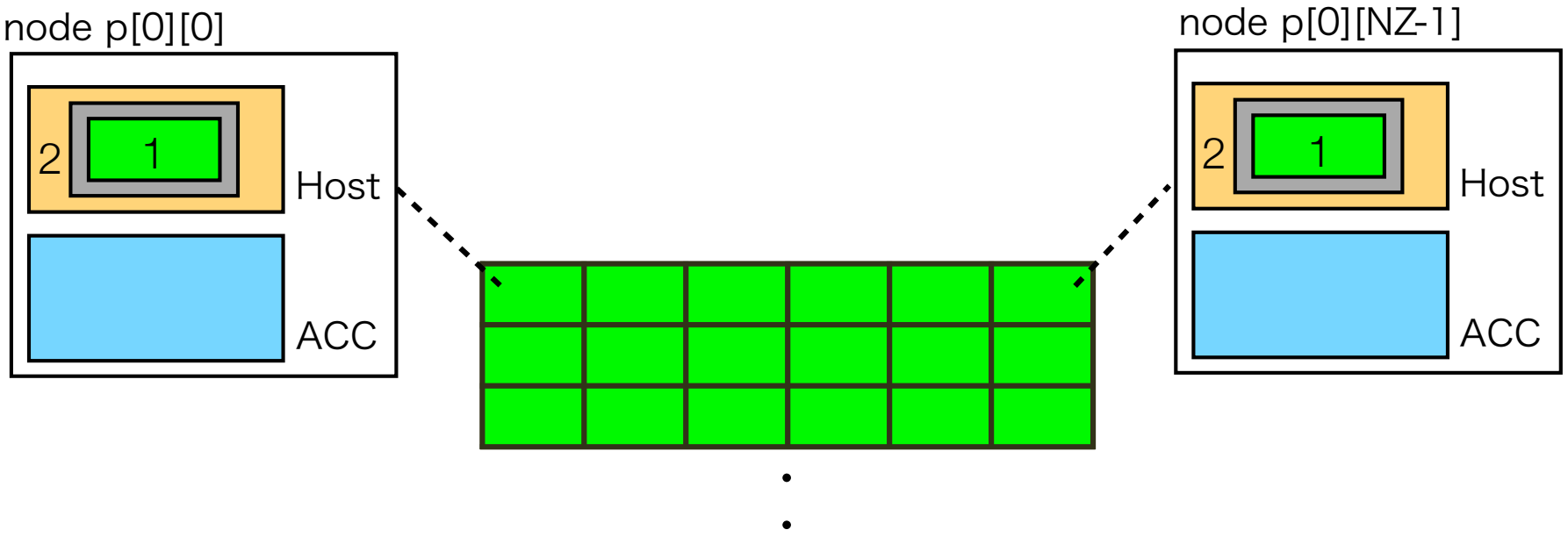
1. 2次元プロセスグリッドを作成し，クォークの4次元配列を各プロセスに分散させる
2. ステンシル計算のために，袖領域を分散配列に確保
3. 分散配列をアクセラレータに転送



分散配列の定義

```
Quark_t v[NT][NZ][NY][NX];
#pragma xmp template t[NT][NZ]
#pragma xmp nodes p[PT][PZ]
#pragma xmp distribute t[block][block] onto p
#pragma xmp align v[i][j][*][*] with t[i][j]
#pragma shadow v[1][1][0][0]
#pragma acc enter data copyin(v)
```

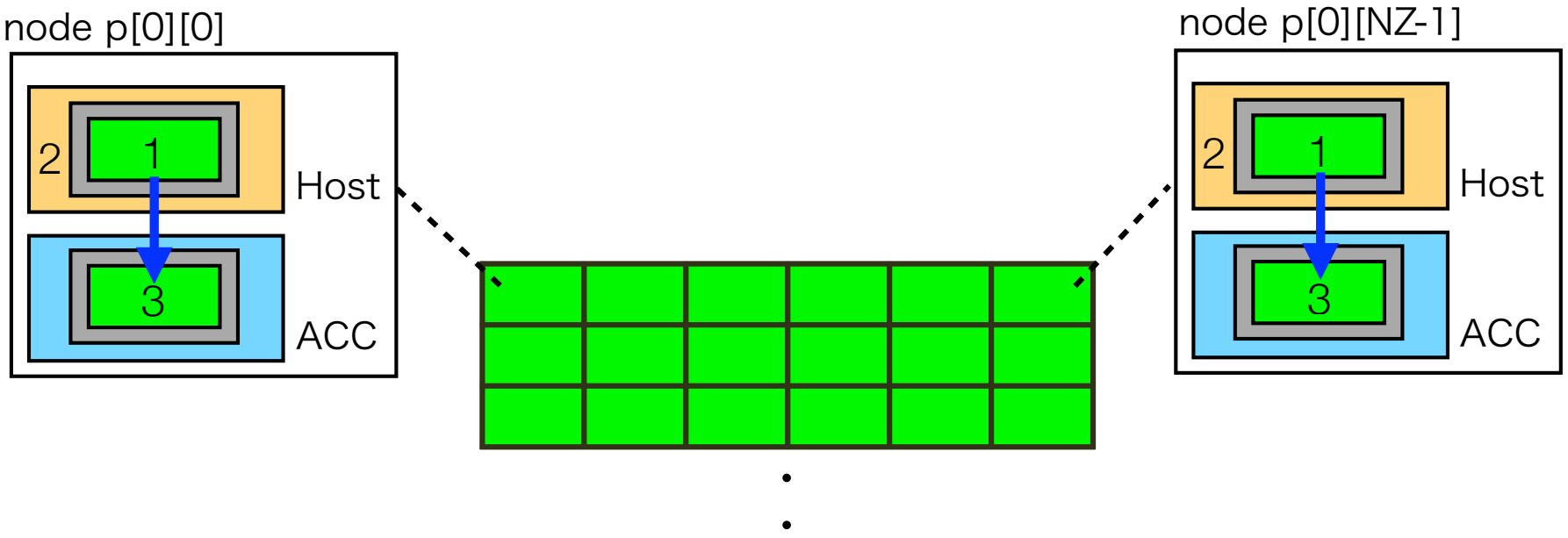
1. 2次元プロセスグリッドを作成し，クォークの4次元配列を各プロセスに分散させる
2. ステンシル計算のために，袖領域を分散配列に確保
3. 分散配列をアクセラレータに転送



分散配列の定義

```
Quark_t v[NT][NZ][NY][NX];
#pragma xmp template t[NT][NZ]
#pragma xmp nodes p[PT][PZ]
#pragma xmp distribute t[block][block] onto p
#pragma xmp align v[i][j][*][*] with t[i][j]
#pragma xmp shadow v[1][1][0][0]
#pragma acc enter data copyin(v)
```

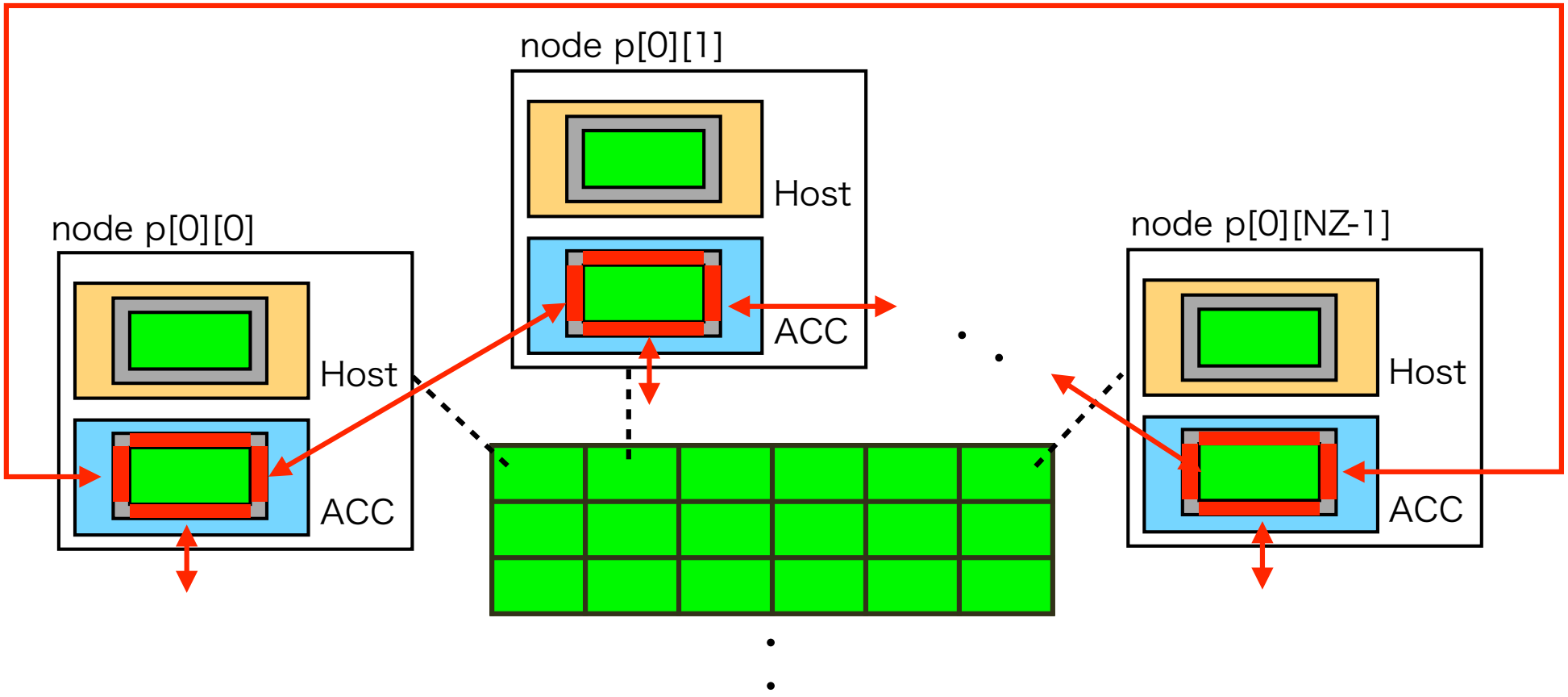
1. 2次元プロセスグリッドを作成し，クォークの4次元配列を各プロセスに分散させる
2. ステンシル計算のために，袖領域を分散配列に確保
3. 分散配列をアクセラレータに転送



袖通信

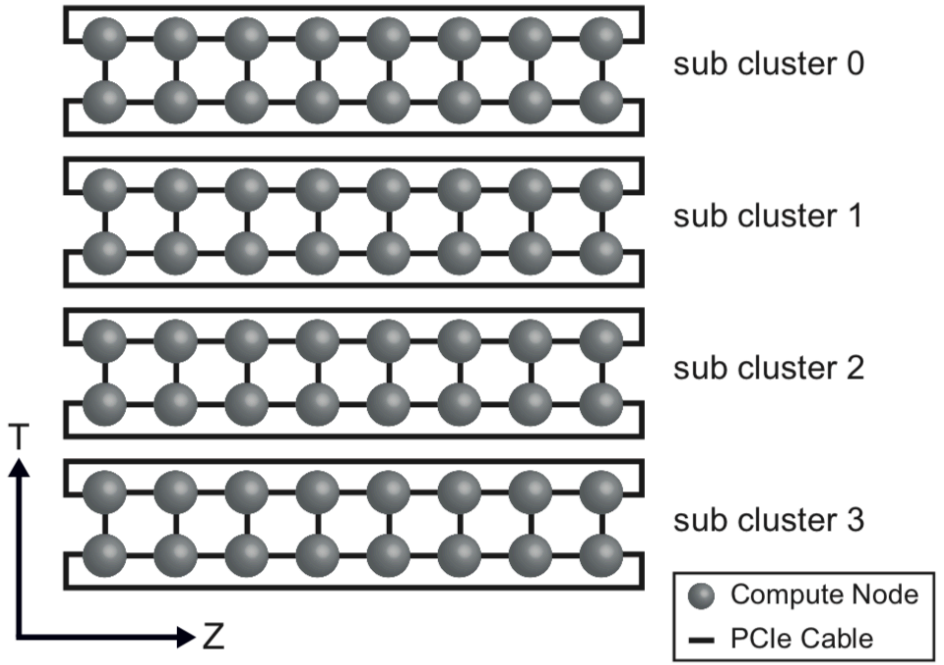
reflect_init/reflect_do指示文により，アクセラレータ上の袖領域の更新を行う

```
#pragma xmp reflect_init(v) width(/periodic/1:1,/periodic/1:1,0,0) orthogonal
while(1){
#pragma xmp reflect_do(v) acc
  WD(..., v); // Stencil calculation
```



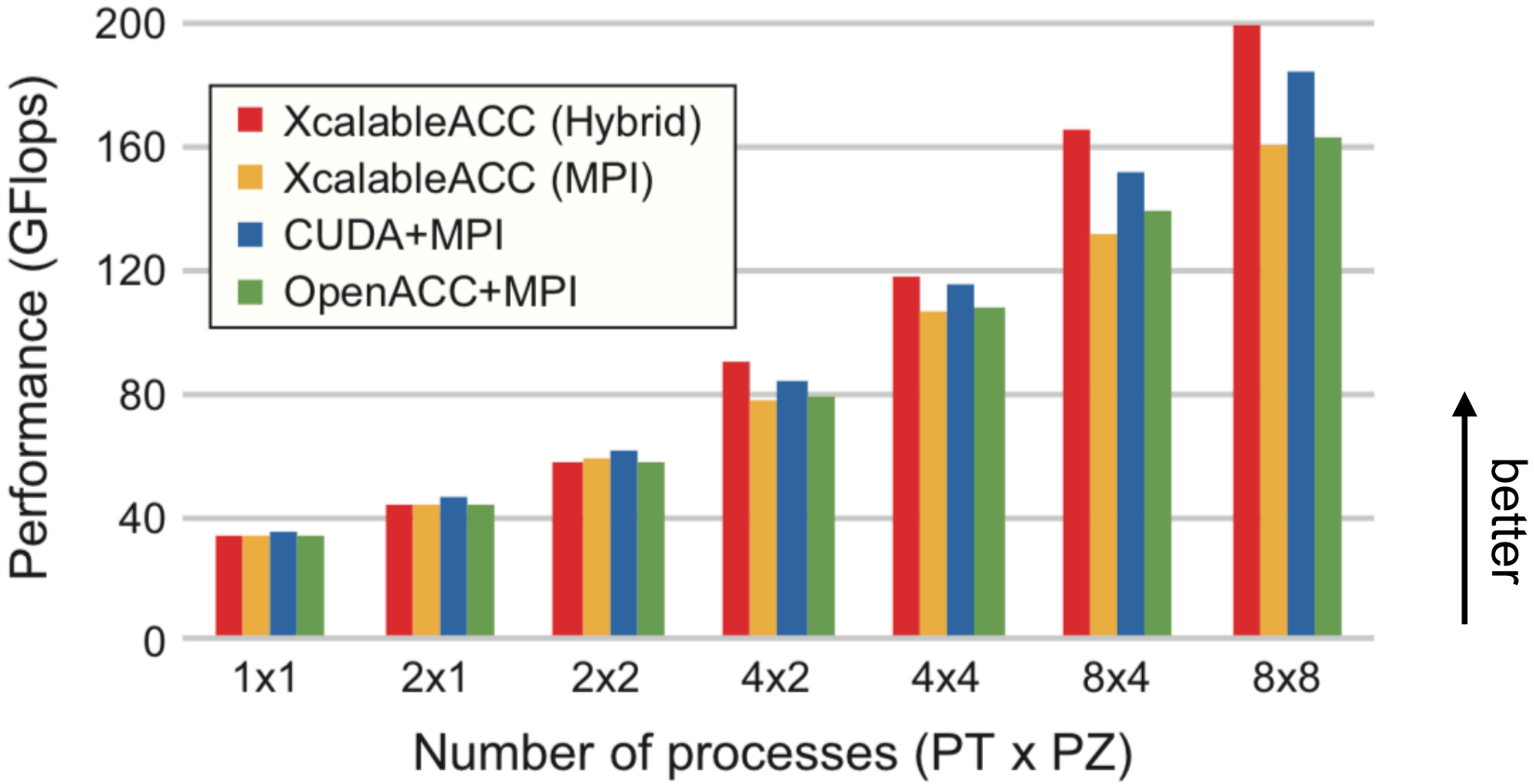
評価

- XACC with TCA/IBハイブリッド通信とCUDA+MPIとOpenACC+MPIとを比較する
- 汎用的な実装であるXACC with MPIとも比較する
- CUDA+MPIとOpenACC+MPIの実装では，CUDA-Aware MPIの永続通信を用いてLQCDを実装した
- TCA/IBハイブリッド通信において，Z方向（ブロックストライド）はTCA，T方向（連続）はIBを用いた通信が行われるように，プロセスを配置した



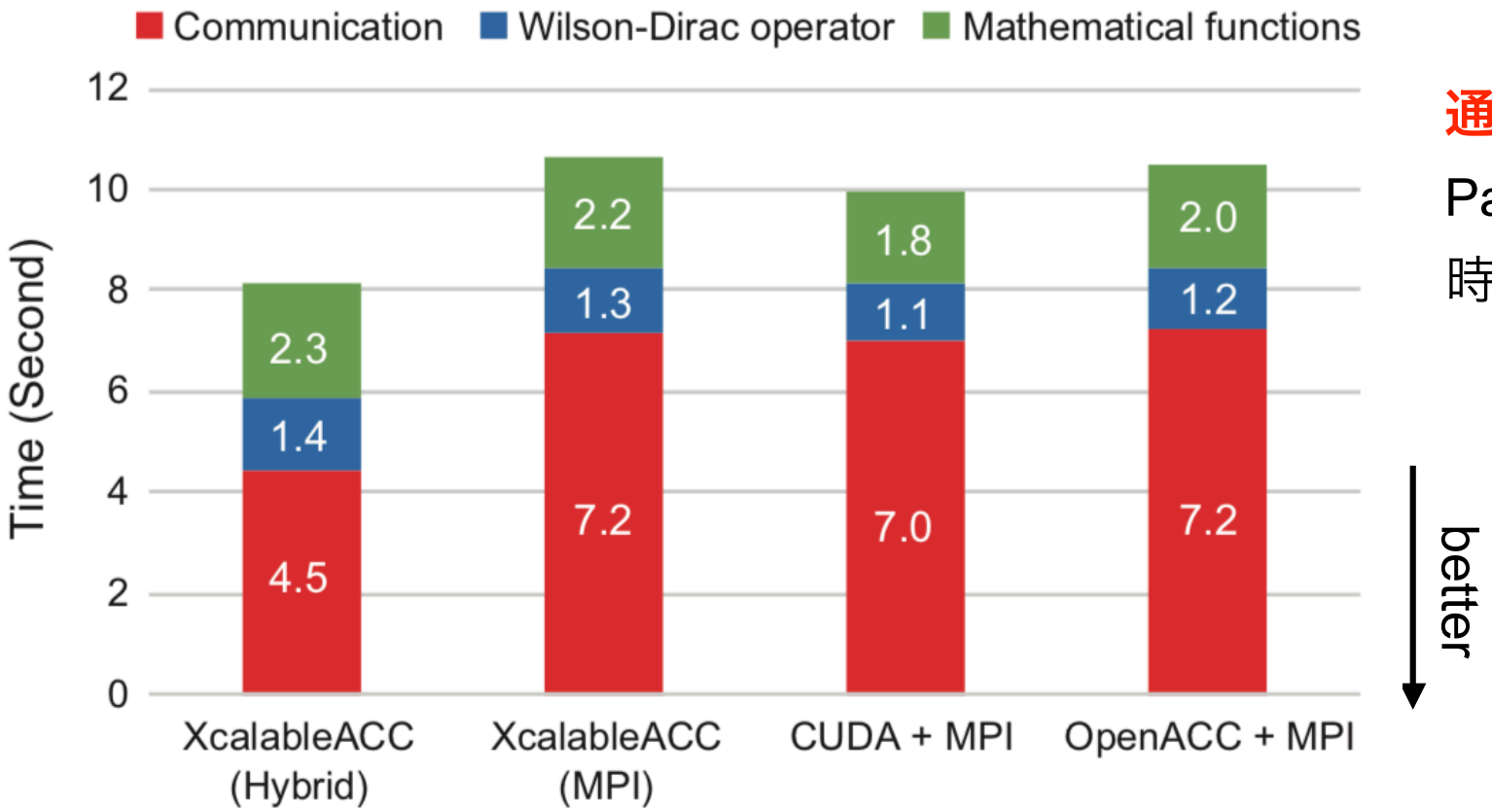
性能評価

データサイズは16x16x16x16 (T x Z x Y x X axes)．強スケーリング．1process/node



- XACC(Hybrid)はCUDA+MPIよりも9%，OpenACC+MPIよりも18%良い
- XACC(MPI)とOpenACC+MPIはほぼ同じ

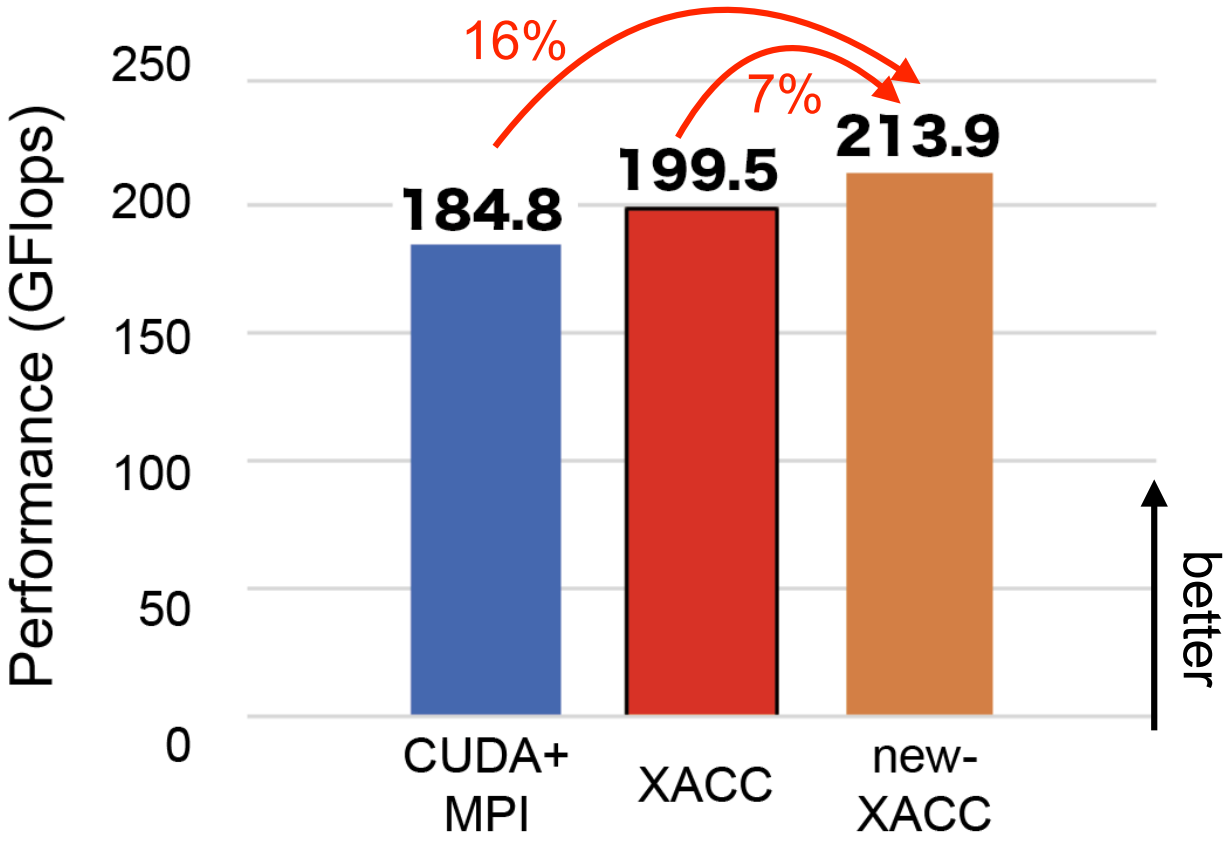
64プロセスにおける計算時間の内訳



通信時間には
Pack/Unpackの
時間も含まれている

- XACC（Hybrid）の**通信時間**は，他のと比較して性能がかなり良い
- CUDA+MPIの**Wilson-Dirac operator**と**数学的関数**は，他のと比較して性能が少し良い（CUDAを用いた実装では，各配列を1次元化しているため）

XACCにおけるさらなる性能向上



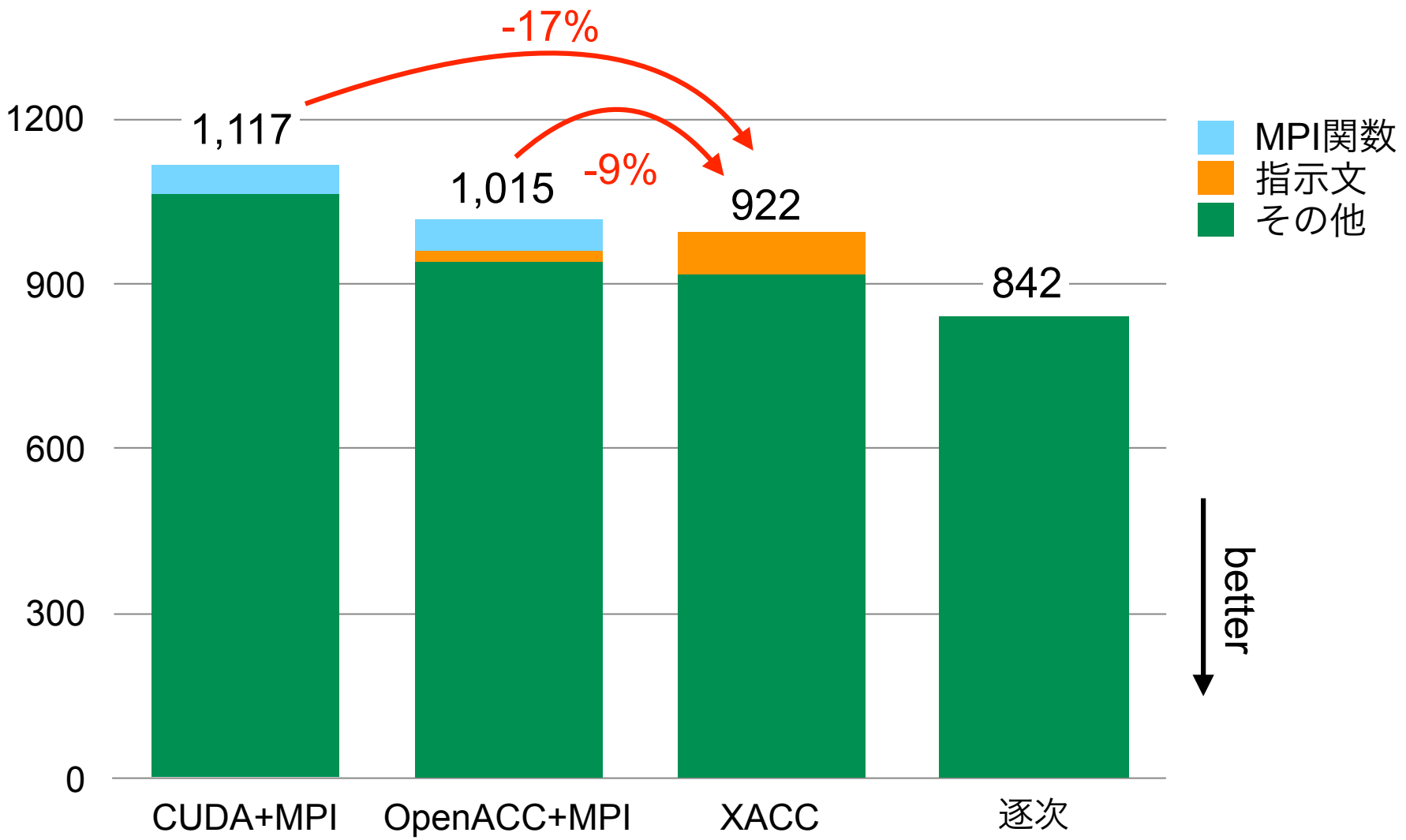

```
#pragma xmp reflect_do(v) acc
WD(..., v); // Stencil calculation
```

↓

```
#pragma xmp reflect_do(v) acc nowait
WD(..., v); // Stencil calculation
```

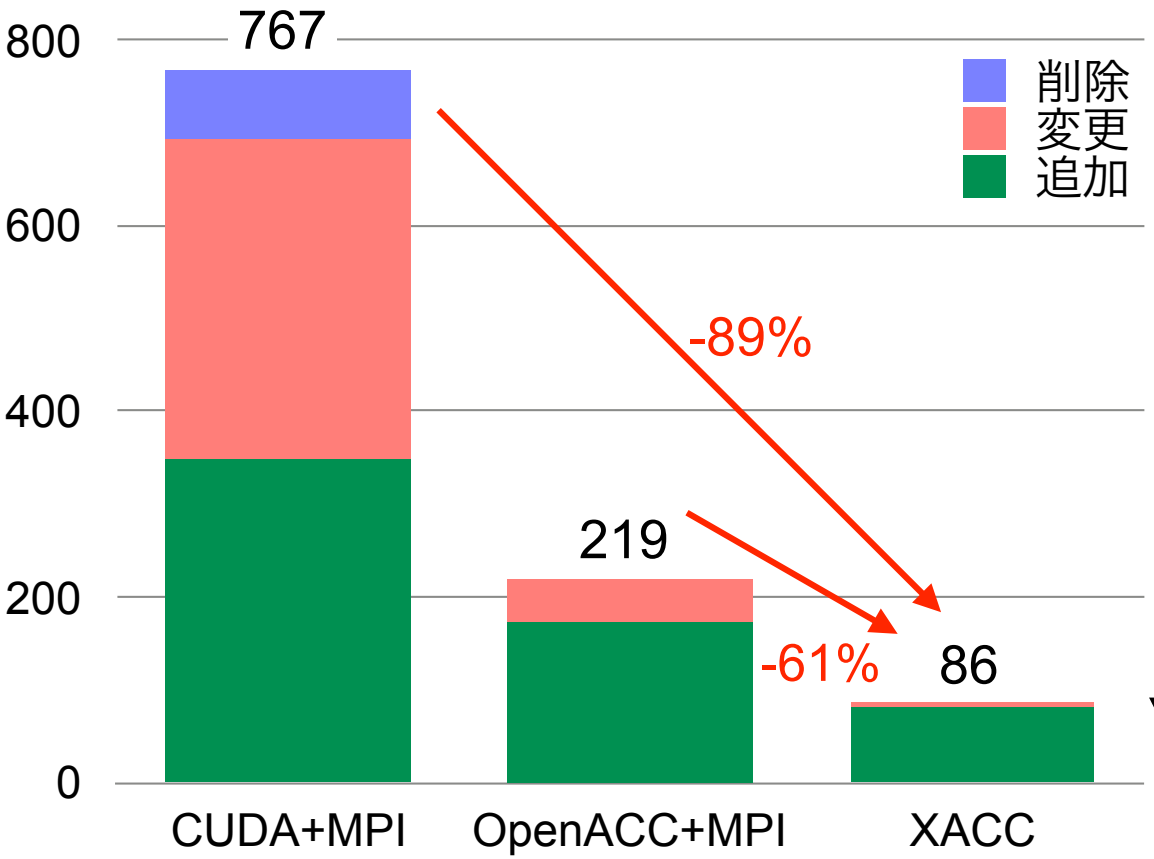
- 他のXACC指示文との兼ね合いのため，袖通信reflect_do指示文の最後にバリア同期が内部的に実行されている．しかし，今回の場合は必要ない
- そこで，そのバリア同期を抑制する**nowait節**を新たに開発
- 最大7%の性能向上を達成した（CUDA+MPIと比較すると16%の性能向上）

生産性評価（SLOC:Source Lines of Code）



- 並列コードの中で，XACCのSLOCは最も少ない

生産性評価（DSLOC:Delta-SLOC）



Delta-SLOC: 逐次コードからの変更した行をカウント

Delta-SLOCが小さいほど、バグが入る可能性やプログラミングコストは低いと言える

better

定量的評価

定性的評価

- 袖領域の更新
- インデックスの変換
- カーネル関数の作成

- 袖領域の更新
- インデックスの変換
- 指示文の追加

- 指示文の追加

まとめと今後の課題

- まとめ
  - TCA/IBハイブリッド通信を適用したXACCの性能と生産性について評価を行った
  - 既存のプログラミングモデルであるCUDA+MPIとOpenACC+MPIと比較し，XACCは高い性能と生産性を示した
  - XACCの利点として，通信の実装を指示文が隠すため，複雑なシステムの効率利用が可能な点およびポータビリティが高い点が挙げられる
    - 本稿のLQCDのコードは，他のアクセラレータクラスタでも動作する

- 今後の課題
  - XACCの性能ポータビリティについて考察する
    - 自動チューニングの検討