XcalableMPによるHPC Challenge Class 2への挑戦

中尾 昌広（理化学研究所 計算科学研究機構）

HPCS2015@東京大学 武田先端知ビル，2015年5月

目的

HPC Challenge Class 2の参加経験をもとに，XcalableMP（XMP）を使った並列アプリケーション開発のノウハウの共有

目次

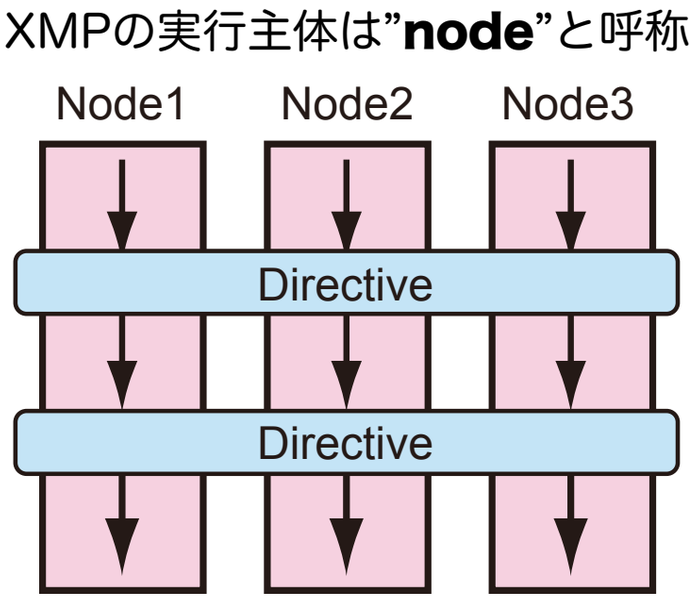
- XMPの概要
  - グローバルビューモデル
  - ローカルビューモデル
  - Omni XMP Compiler
- HPC Challenge Class 2における実装の工夫（2009年 → 2014年）
- HPC Challenge Class 2から得た経験
- まとめと今後の課題

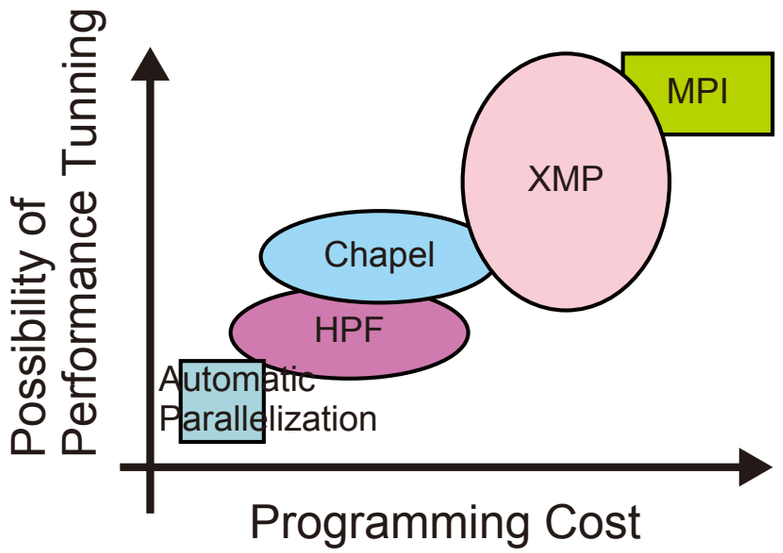
目次

- XMPの概要
  - グローバルビューモデル
  - ローカルビューモデル
  - Omni XMP Compiler
- HPC Challenge Class 2における実装の工夫（2009年 → 2014年）
- HPC Challenge Class 2から得た経験
- まとめと今後の課題

XcalableMPの概要

- 既存言語（C99とFortran95）を指示文と拡張構文で並列化
  - 記法のいくつかはCoarray Fortran，HPF，OpenMPから継承
- 実行モデルはMPIと同じSPMD
  - 指示文がなければ、重複実行
  - 指示文とCoarray記法により通信・同期を実行
- 様々なアプリケーションに対応するための並列化ビュー
  - グローバルビュー：指示文による典型的なデータ並列
  - ローカルビュー：Coarrayによる片側通信
- PCクラスタコンソーシアムの「並列プログラミング言語XcalableMP規格部会」で仕様書の策定
  - http://xcalablemp.org
  - メンバは企業・大学・国研など





コード例：グローバルビュー

```
int a[MAX];
#pragma xmp nodes p(4)
#pragma xmp template t(0:MAX-1)
#pragma xmp distribute t(block) on p
#pragma xmp align a[i] with t(i)

main(){
  int val = 0;

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

データ分散の定義

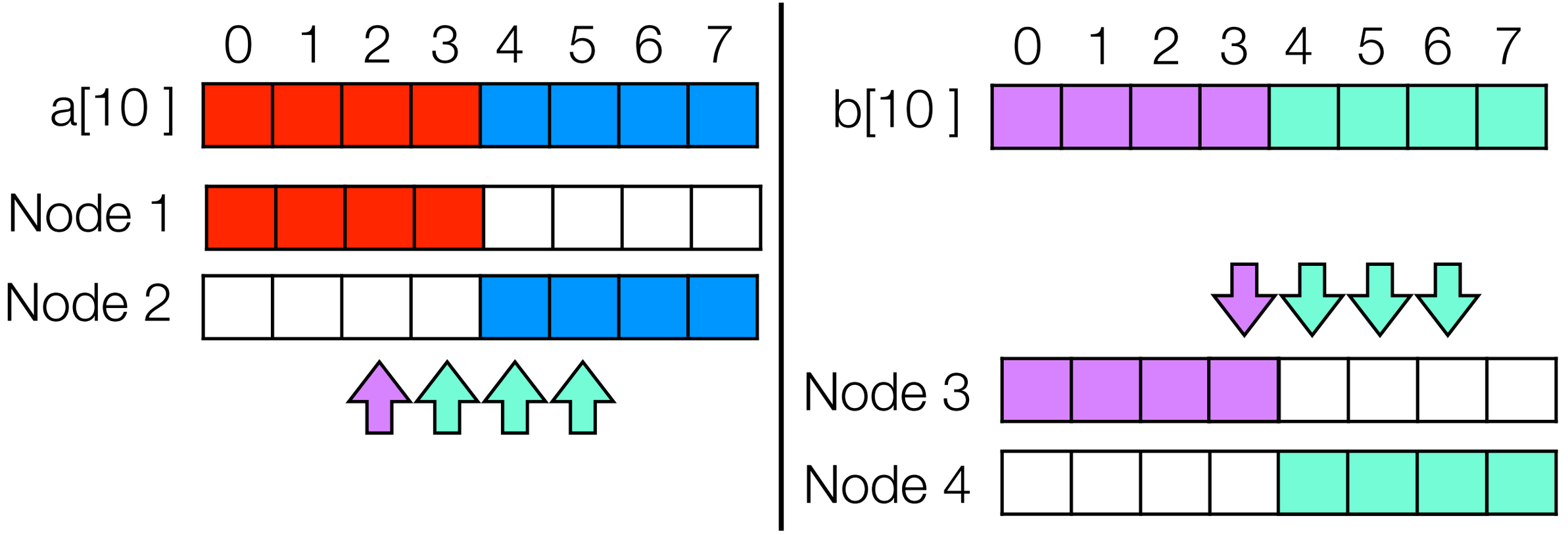
ループ文の並列実行

コード例：グローバルビュー

- gmove 指示文
  - グローバルなイメージを保ちつつ，通信を記述
  - array section notation

[start_index:length]

```
#pragma xmp gmove
a[2:4] = b[3:4];
```



コード例：ローカルビュー

```
double a[100]:[*], b[100]:[*];
int me = xmp_node_num();

if(me == 2)
   a[:]:[1] = b[:];

if(me == 1)
   a[0:50] = b[0:50]:[2];
```

Coarrayの定義

Put通信

Get通信

Coarray synax in XMP/C

```
array_name[start_index:length]:[node_number];
```

グローバルビューとローカルビューを混合利用（ハイブリッドビュー）できる仕様
（例：核融合シミュレーションコード：津金（筑波大））

Omni XcalableMP Compiler

http://omni-compiler.org

Omni Compiler Project

Omni compiler is a collection of programs and libraries that allow users to build code transformation compilers. Omni Compiler is to translate C and Fortran programs with **XcalableMP** and/or **OpenACC** directives into parallel code suitable for compiling with a native compiler linked with the Omni Compiler runtime library. The Omni compiler project is proceeded by **Programming Environment Research Team** of RIKEN AICS and **HPCS Lab.** of University of Tsukuba, Japan.

- 理化学研究所 計算科学研究機構と筑波大学が開発
- オープンソース
- 京，富士通FX10，NEC SX，日立SR，Crayシステム，OS X，Linuxクラスタ
  - 京とFX10では，CoarrayはFujitsu RDMAを利用
- XMPで実装したHPC Challengeベンチマークなども公開しています
- Source-to-Source Compilerなので既存プロファイリングツールの利用可

目次

- XMPの概要
  - グローバルビューモデル
  - ローカルビューモデル
  - Omni XMP Compiler
- HPC Challenge Class 2における実装の工夫（2009年 → 2014年）
- HPC Challenge Class 2から得た経験
- まとめと今後の課題

XMPの挑戦の歴史

提出ベンチマークと結果

年	追加ベンチマーク	結果	主なシステム
2009		Honorable Mention	T2K Tsukuba
2010	Laplace, NPB-CG	Honorable Mention	T2K Tsukuba
2012	姫野ベンチマーク		京
2013	姫野ベンチマーク	**Award**	京
2014		**Best Performance**	京

2009年と2014年の比較

ベンチマーク	2009年		2014年	
	測定値（ピーク比）	#nodes	測定値（ピーク比）	#nodes
HPL	16 GFlops (0.3%)	32	423 TFlops (80.7%)	4,096
RA	0.00004 GUPs	32	254.2 GUPs	16,384
FFT	2 GFlops (0.04%)	32	212 TFlops (2.0%)	82,944
STREAM	700 GB/s (51.2%)	32	3,583 TB/s (67.5%)	82,944

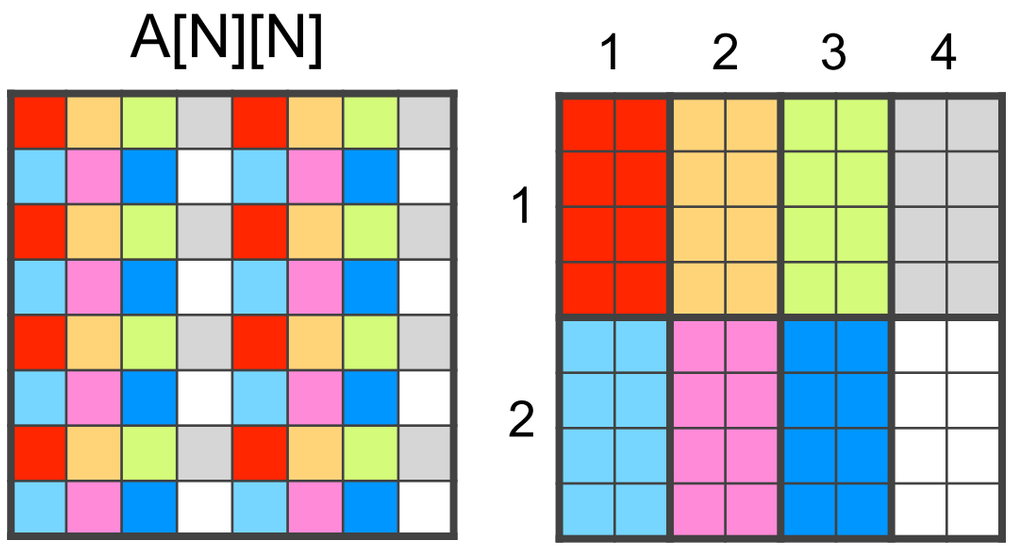
参照：第148回HPC研究会「PGAS言語XcalableMPを用いたHPC Challengeベンチマークの実装と評価」

HPL（2009年→2014年）

- 連立一次方程式をLU分解を用いて解くベンチマーク
- 2009年は，（非常にシンプルな）ブロック化されていない逐次コードのLU分解に指示文を挿入したのみ
- 2014年は，ブロック化した逐次コードのLU分解を作成し，並列化を行った
  - 係数行列をブロックサイクリック分散 + DGEMMの利用
  - 行列積はBLASのDGEMMを利用
- パネル転送は非同期通信（比較のため，同期版と非同期版の2つを作成）

ブロックサイクリック分散のコード

```
double A[N][N];
#pragma xmp nodes p(P,Q)
#pragma xmp template t(0:N-1, 0:N-1)
#pragma xmp distribute t(cyclic(NB), \
                         cyclic(NB)) onto p
#pragma xmp align A[i][j] with t(j,i)
```

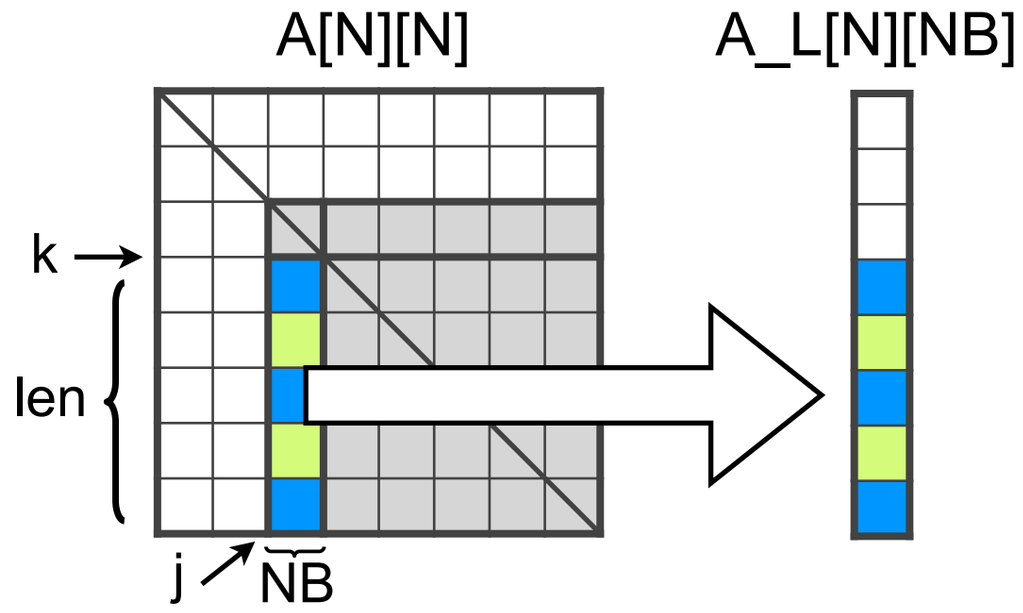


HPL（2014年：非同期版）

- gmove指示文とasync節を用いたLookaheadアルゴリズムの実装

通信と計算のオーバラップ

```
double A_L[N][NB];
#pragma xmp align A_L[i][*] with t(*,i)
     :
#pragma xmp gmove async(1)
A_L[k:len][0:NB] = A[k:len][j:NB];
     :
for(m=j+NB;m<N;m+=NB){
  for(n=j+NB;n<N;n+=NB){
    cblas_dgemm(&A[m][n], ..);
    if(xmp_test_async(1)){
       // receive A[k:len][j:NB];
            :
```

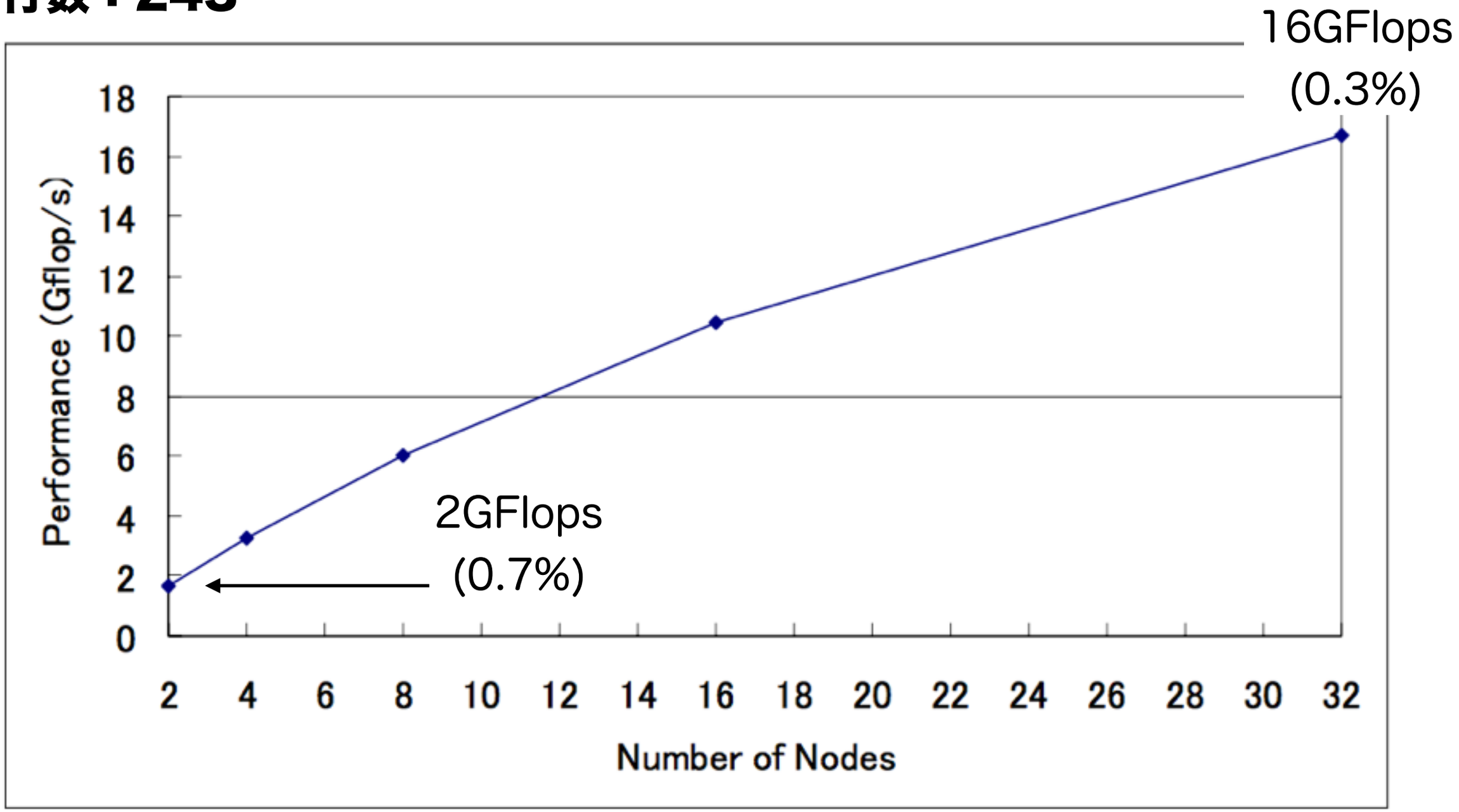


非同期Broadcast通信

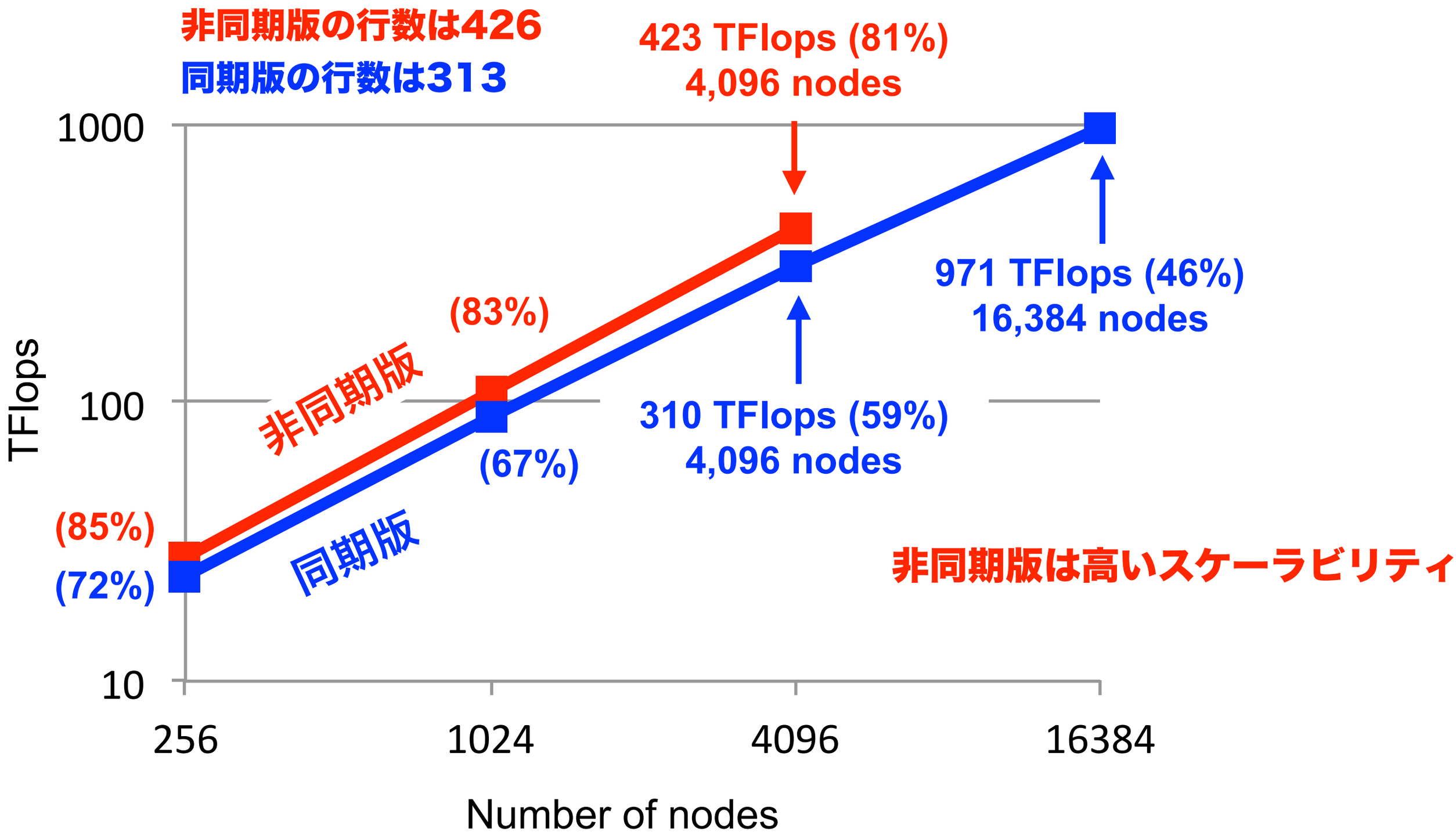
非同期通信が来ているかどうかをチェック

HPLの評価 in 2009年

行数：243



HPLの評価 in 2014年



RandomAccess（2009年→2014年）

- 分散されたテーブルの各要素に対して排他的論理和をとりつつ，各ノードの要素の値を更新する
- Coarrayを用いたローカルビュー
- 2009年は，逐次コードの一部をCoarray化したのみ
  - 一要素ずつデータを送信
- 2014年は，通信時間の削減するため，チャンク毎にデータを送信

```
u64Int recv[LOGPROCS][RCHUNK+1]:[*];
...
for (j = 0; j < logNumProcs; j++) {
  recv[j][0:num]:[i_partner] = send[i][0:num];

#pragma xmp sync_memory
#pragma xmp post(p(i_partner), 0)
    :
#pragma xmp wait(p(j_partner))
}
```

Coarrayの定義

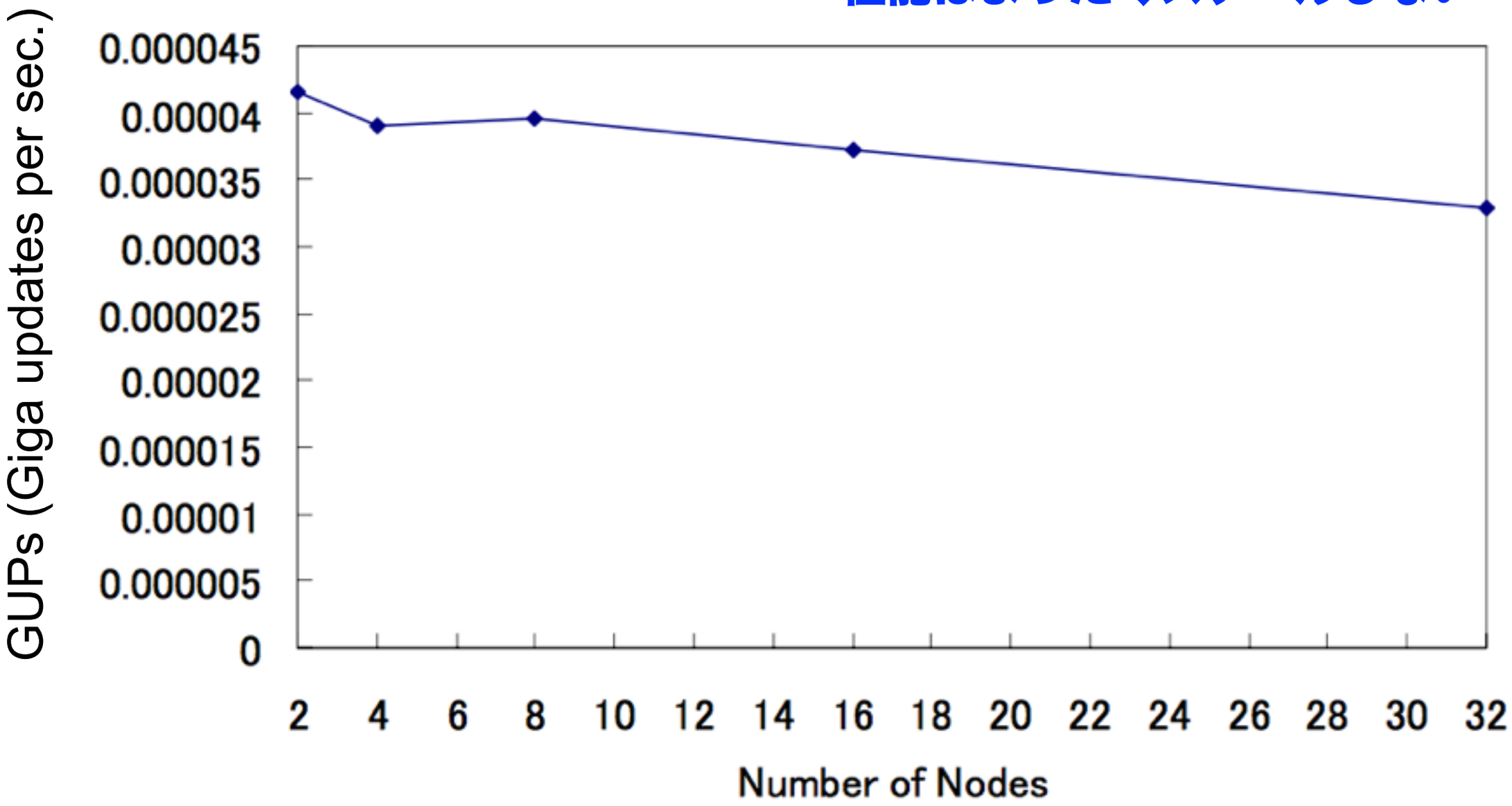
Put operation

post/wait指示文を用いたP2Pバリア

RandomAccessの評価 in 2009年

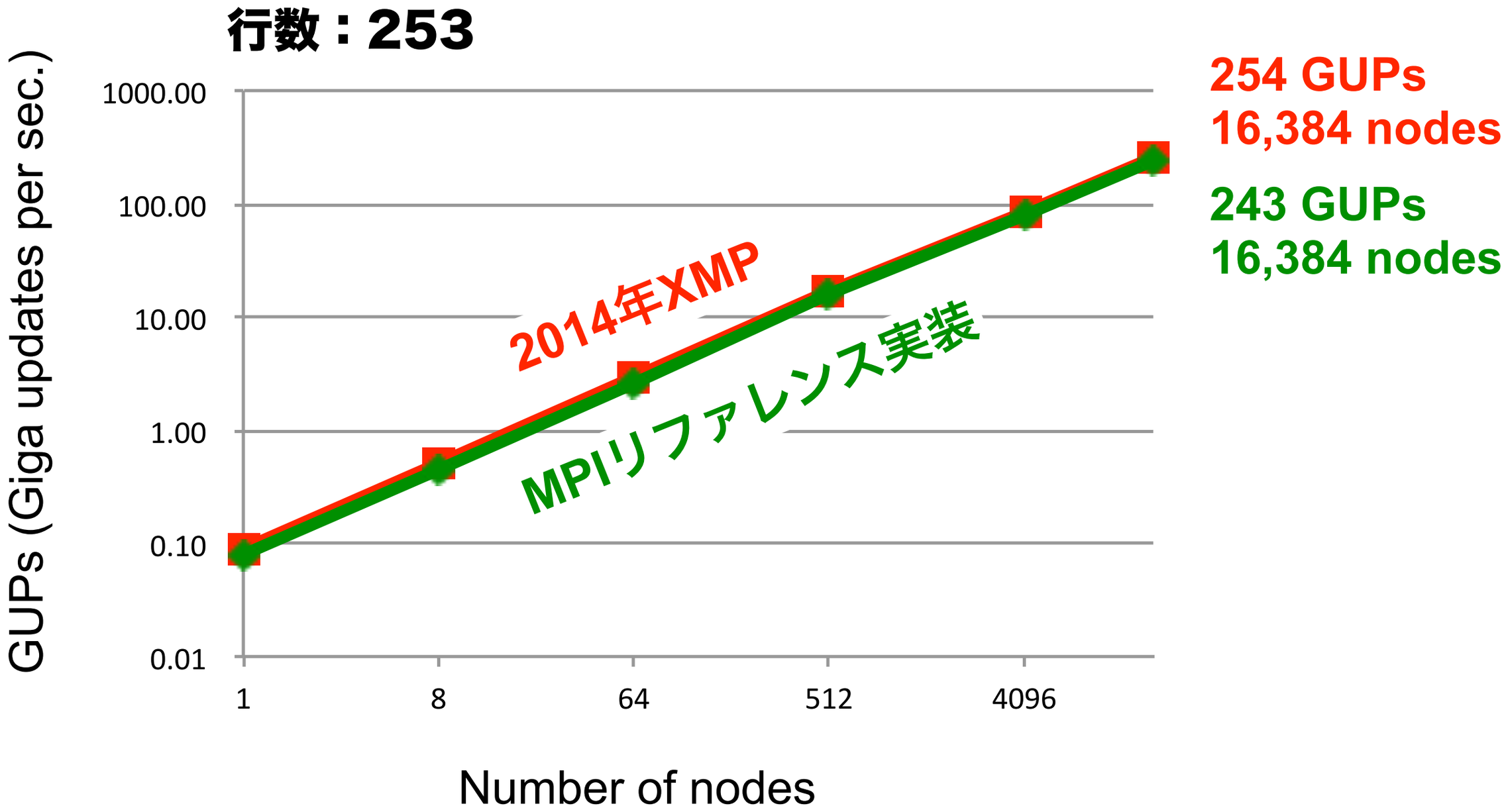
行数：77

性能はまったくスケールしない・・・



RandomAccessの評価 in 2014年

高いスケーラビリティ



目次

- XMPの概要
  - グローバルビューモデル
  - ローカルビューモデル
  - Omni XMP Compiler
- HPC Challenge Class 2における実装の工夫（2009年 → 2014年）
- HPC Challenge Class 2から得た経験
- まとめと今後の課題

HPC Challenge Class 2の経験

- 逐次コードに指示文を挿入するだけでは性能が出ない（こともある）
  - 2009年のHPLは，シンプルなLU分解の逐次コードに指示文を挿入した（243行）
  - 2009年のRAは，エレメントワイズな要素の更新（77行）
- **まず逐次コードで性能を出し，次に並列アルゴリズムを実装する**
  - 2014年のHPLは，ブロックLU分解の効率的な並列アルゴリズムの実装（426行）
  - 2014年のRAは，チャンク毎の転送（253行）
  - 通信時間を小さくし，ロードバランシングが良い並列アルゴリズムを実装する
- XMPでは，必要に応じたレベルの最適化が可能
  - 性能を出したい箇所はCoarrayやMPIを利用するなど
    - 例：HPLのパネル転送のスケジュール化
    - Coarrayを多用するとMPIに近くなる（→エレガントでない）
    - コードのエレガントさと性能のバランス調節が可能

XMPのメモリレイアウト問合せ関数（Leading Dim.）

DGEMMやFFTEの利用

OpenMPによるHybrid並列

目次

- XMPの概要
  - グローバルビューモデル
  - ローカルビューモデル
  - Omni XMP Compiler
- HPC Challenge Class 2における実装の工夫（2009年 → 2014年）
- HPC Challenge Class 2から得た経験
- まとめと今後の課題

まとめと今後の課題

- まとめ
  - 発表目的「HPC Challenge Class 2の参加経験をもとに，XMPを使った並列アプリケーション開発のノウハウの共有」
  - 2009年から2014年のHPC Challenge Class2の経験
  - まず逐次コードで性能を出し，効果的な並列アルゴリズムを実装
    - XMPを用いると，必要に応じたレベルの最適化が可能

- 今後の課題
  - http://xcalablemp.orgにおいて，マニュアルとチュートリアルの整備
  - XMP講習会の全国展開
    - 過去動画はAICS eラーニングアーカイブ or YoutubeでXcalableMPと検索
  - アプリケーションの事例を増やす（核融合シミュレーションコード）