

# HPC Challenge Benchmarkによる 並列プログラミング言語XcalableMPの性能評価

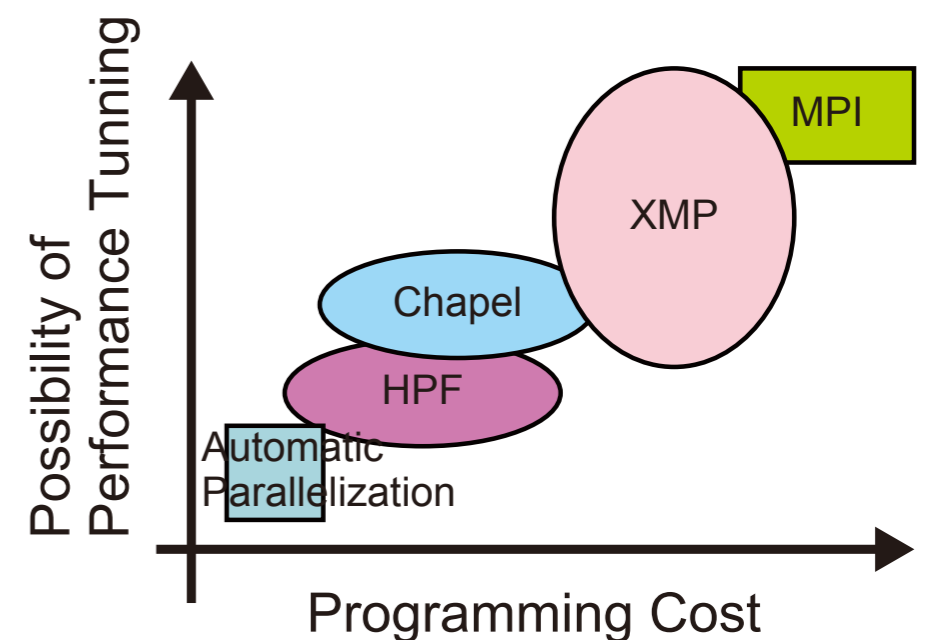
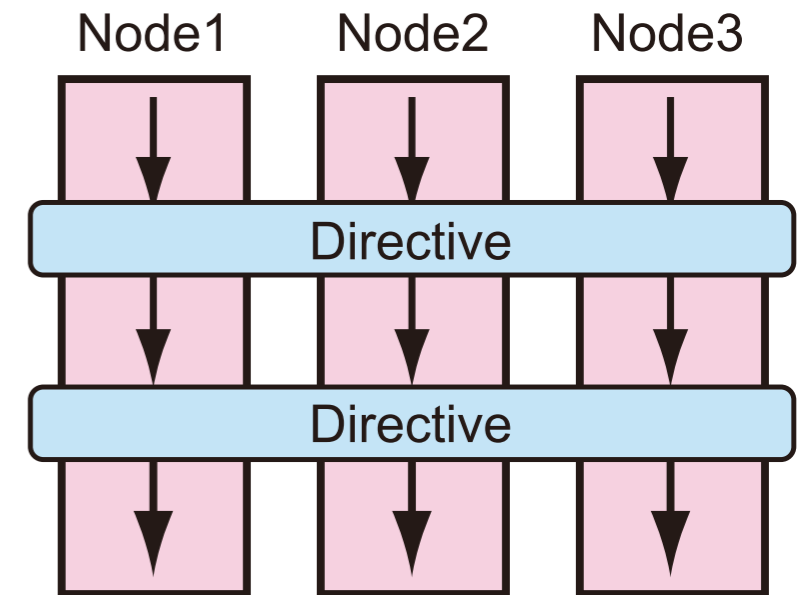
---

- 中尾 昌広 (理化学研究所 計算科学研究機構)  
佐藤 三久 (理化学研究所 計算科学研究機構)

# XcalableMP (XMP)



- 既存言語（C99とFortran95）を指示文と拡張構文で並列化
  - 低い教育コスト
  - 記法のいくつかはCoarray FortranとHPF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
- Performance-aware programming
  - 指示文とCoarray記法により通信・同期を実行
  - チューニングが行い易い
- 2つのメモリモデル
  - グローバルビュー：指示文による典型的なデータ並列のサポート
  - ローカルビュー：Coarrayによる片側通信



# Code example (グローバルレビュー)

```
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)
```

データ分散の定義

```
main(){  
    int i, j, res = 0;
```

```
#pragma xmp loop on t(i) reduction(+:res)
```

```
    for(i = 0; i < MAX; i++){  
        a[i] = func(i);  
        res += array[i];  
    }
```

データ並列と集約演算

# Code example (ローカルビュー)

```
double a[100]:[*], b[100]:[*];
```

Coarrayの定義

```
int me = xmp_node_num();
```

```
if(me == 2)
```

```
    a[:,1] = b[:,];
```

Put通信

```
if(me == 1)
```

```
    a[0:50] = b[0:50]:[2];
```

Get通信

## Coarray syntax in XMP/C

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

# 最近の成果 (1/4)

---

- 2014年11月に仕様書1.2.1をリリース (マイナーアップデート)
  - <http://xcalablemp.org>
- 同日にOmni Compiler Ver. 0.9をリリース
  - <http://omni-compiler.org>
  - 「京」のRDMAを使った片側通信機能の実装など
- total社のアプリReverse Time Migration (RTM) をXMPで実装
- 核融合プラズマシミュレーションコードGTC-PをXMPで実装
- FX10, SR16000, SX-9, XE6, BG/Qなどへの移植
  
- 第2回XMPワークショップ@秋葉原 (2014年10月24日)
  - 参加者は約40名：14件の発表
  - 基調講演は3次元流体コードなどのアプリの話
  - <http://www.pccluster.org/ja/event/2014/09/2xcalablemp.html>
  - 来年度もします



# 最近の成果 (2/4)

- XMP講習会@計算科学振興財団 in 神戸

## 開催情報

日付	場所	内容
2016.01.08 2015.07.03	<b>FOCUS (公益財団法人 計算科学振興財団)</b>	初級編1
2014.12.18 2014.09.18	<b>FOCUS (公益財団法人 計算科学振興財団)</b>	初級編1
2014.05.19	<b>大阪大学 レーザーエネルギー学研究センター</b>	初級編2
2013.12.11 2013.09.13 2013.07.18	<b>FOCUS (公益財団法人 計算科学振興財団)</b>	初級編1

来年度は東京で開催予定  
(ハンズオンができる所を募集中)



AICS eラーニングアーカイブ  
理研AICSのウェブサイト  
or  
YoutubeでXcalableMPと検索

# 最近の成果 (3/4)

- HPC Challenge Awards Competition Class 2への応募（発表は2014年11月）
  - <http://www.hpcchallenge.org>
  - HPCシステムを多角的に評価するためのベンチマークセット
  - In Class 1, only the performance of an HPC system is evaluated
  - In Class 2, the productivity and performance of a programming language are evaluated（コードのエレガントさを50%、性能を50%で評価）

Benchmark	説明
RandomAccess	分散テーブルをランダムにアクセス
STREAM	実効メモリバンド幅を計測
HPL	密行列連立一次方程式を解く
FFT	1次元離散複素フーリエ変換に対する速度

# 最近の成果 (4/4)

- Best Performance賞とMost Elegant Language賞
- 京を使って評価
- 2013年はAwardを受賞 (日本初)
- 2014年は2013年のに対してチューニング
  - XMPはBest Performance賞を受賞
  - Most Elegant Language賞はPCJというJava拡張

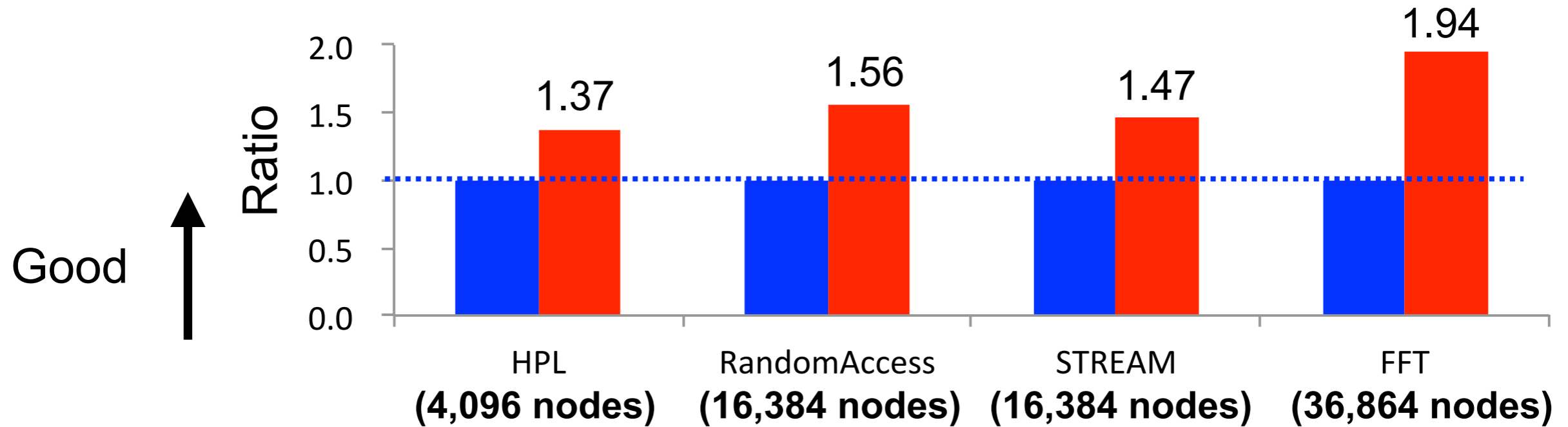
	Best	Most
XMP	○	
PCJ		○
??		
??		



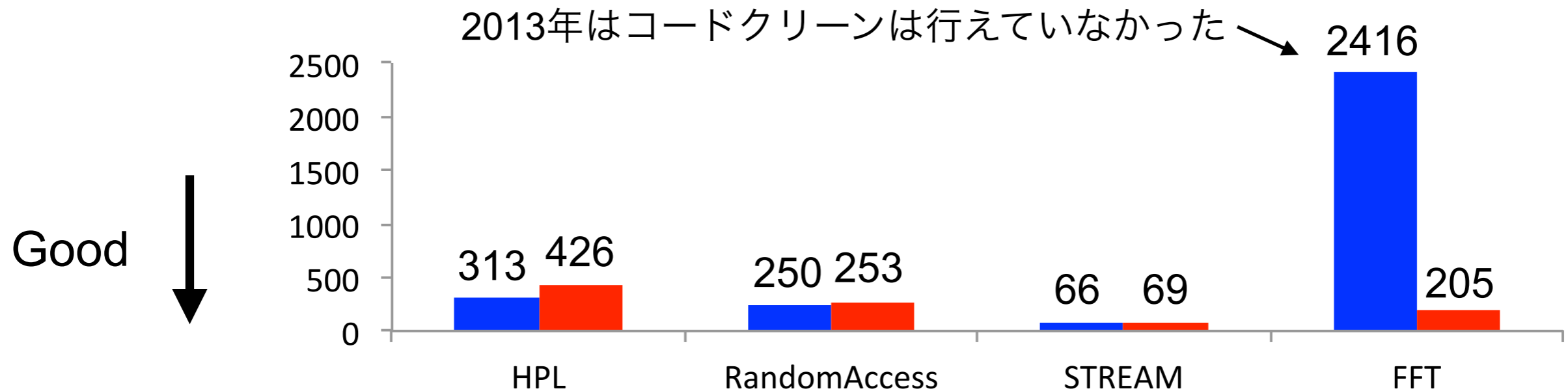


# 2013年と2014年とのXMPの比較

- 同じノード数における性能向上率



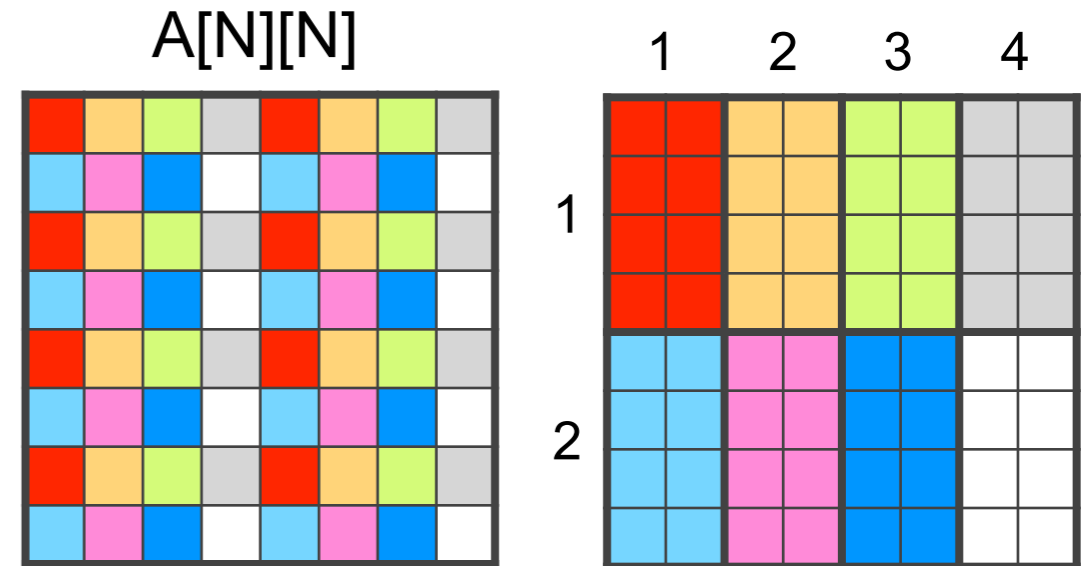
- 行数



# HPL version 1

- Source lines of Code (SLOC) is **313**, written in XMP/C
- ブロックサイクリック分散

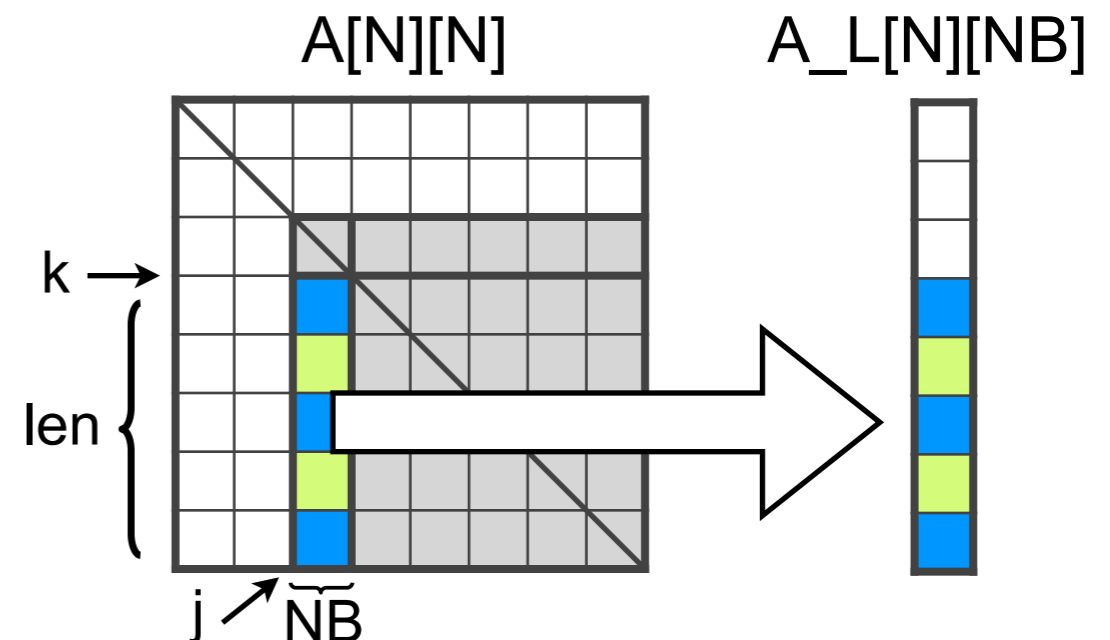
```
double A[N][N];  
#pragma xmp nodes p(P,Q)  
#pragma xmp template t(0:N-1, 0:N-1)  
#pragma xmp distribute t(cyclic(NB), \  
                        cyclic(NB)) onto p  
#pragma xmp align A[i][j] with t(j,i)
```



行列積には「京」が提供しているBLASを利用

- **gmove**指示文を用いたパネル転送

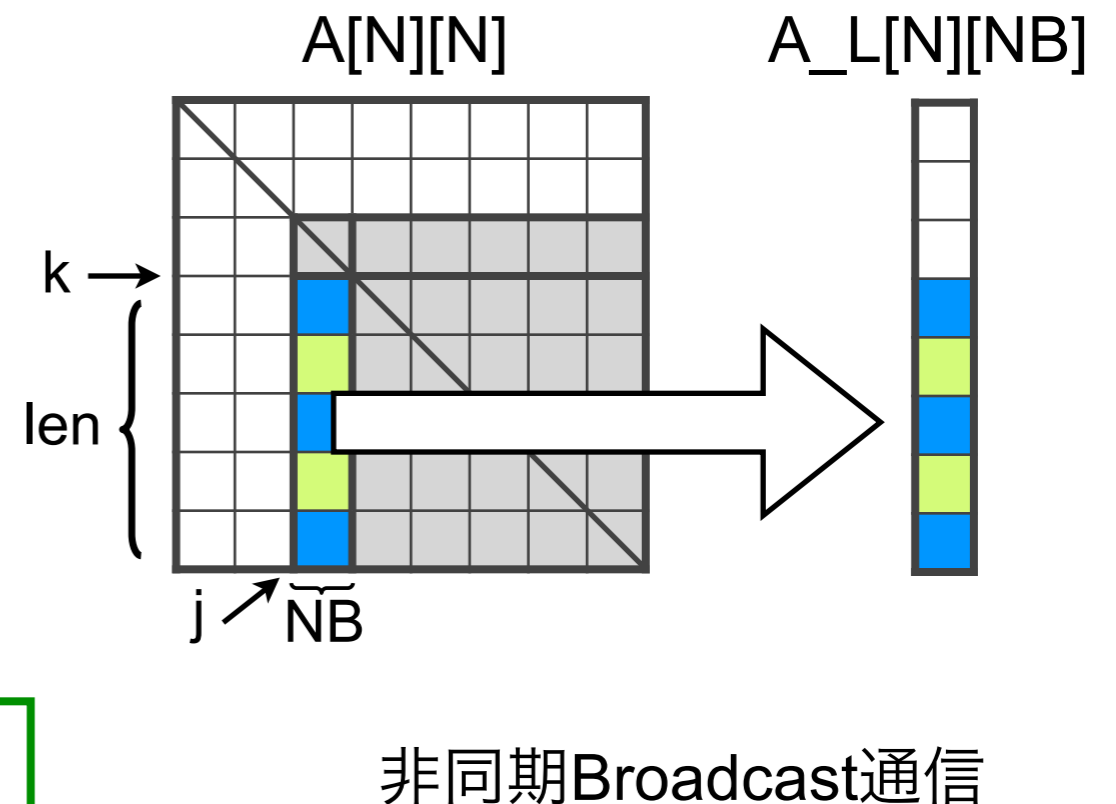
```
double A_L[N][NB];  
#pragma xmp align A_L[i][*] with t(*,i)  
:  
#pragma xmp gmove  
A_L[k:len][0:NB] = A[k:len][j:NB];
```



# HPL version 2

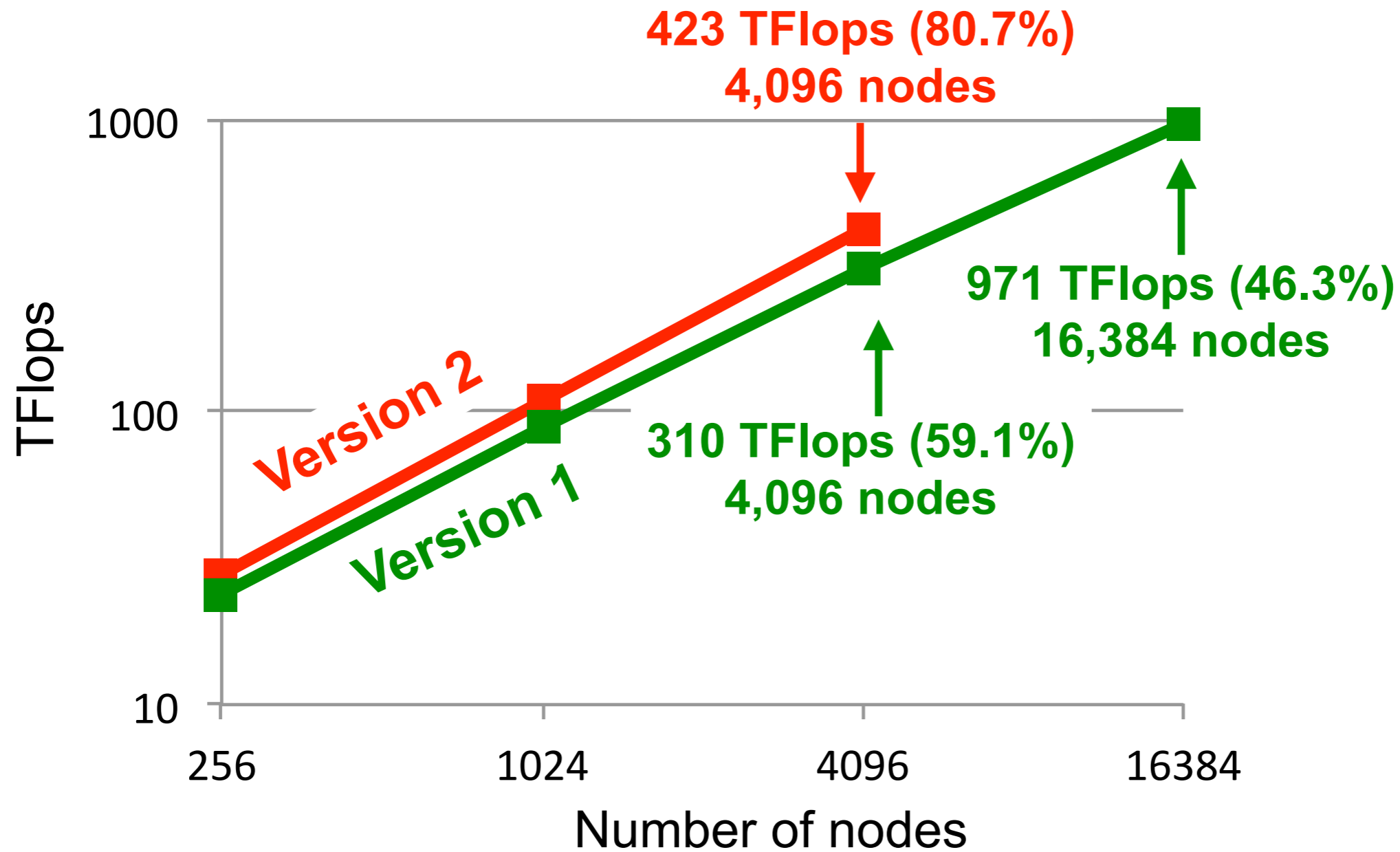
- SLOC is **426**, written in XMP/C
- **gmove**指示文と**async**節を用いたLookaheadアルゴリズムの実装  
通信と計算のオーバラップ

```
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];  
      :  
    }  
  }  
}
```



非同期通信が来ているかどうかをチェック

# Performance of HPL



XMP-HPL Version 2の方がスケーラビリティが高い

# RandomAccess

- SLOC is **253**, written in XMP/C
- XMP/C coarrayを用いたローカルビュー
- Post指示文とWait指示文を用いたP2Pバリア

```
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))  
}
```

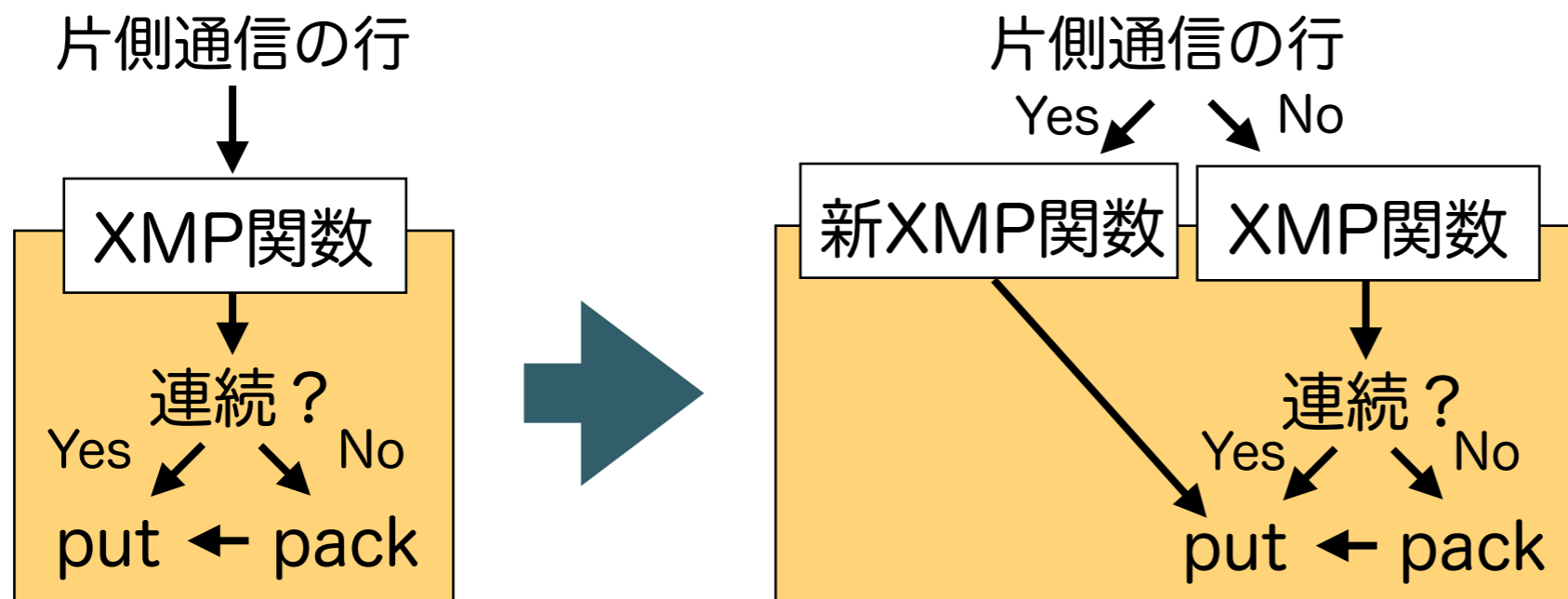
← Coarrayの定義

← Put operation

← post/wait指示文を用いたP2Pバリア

# 片側通信機能

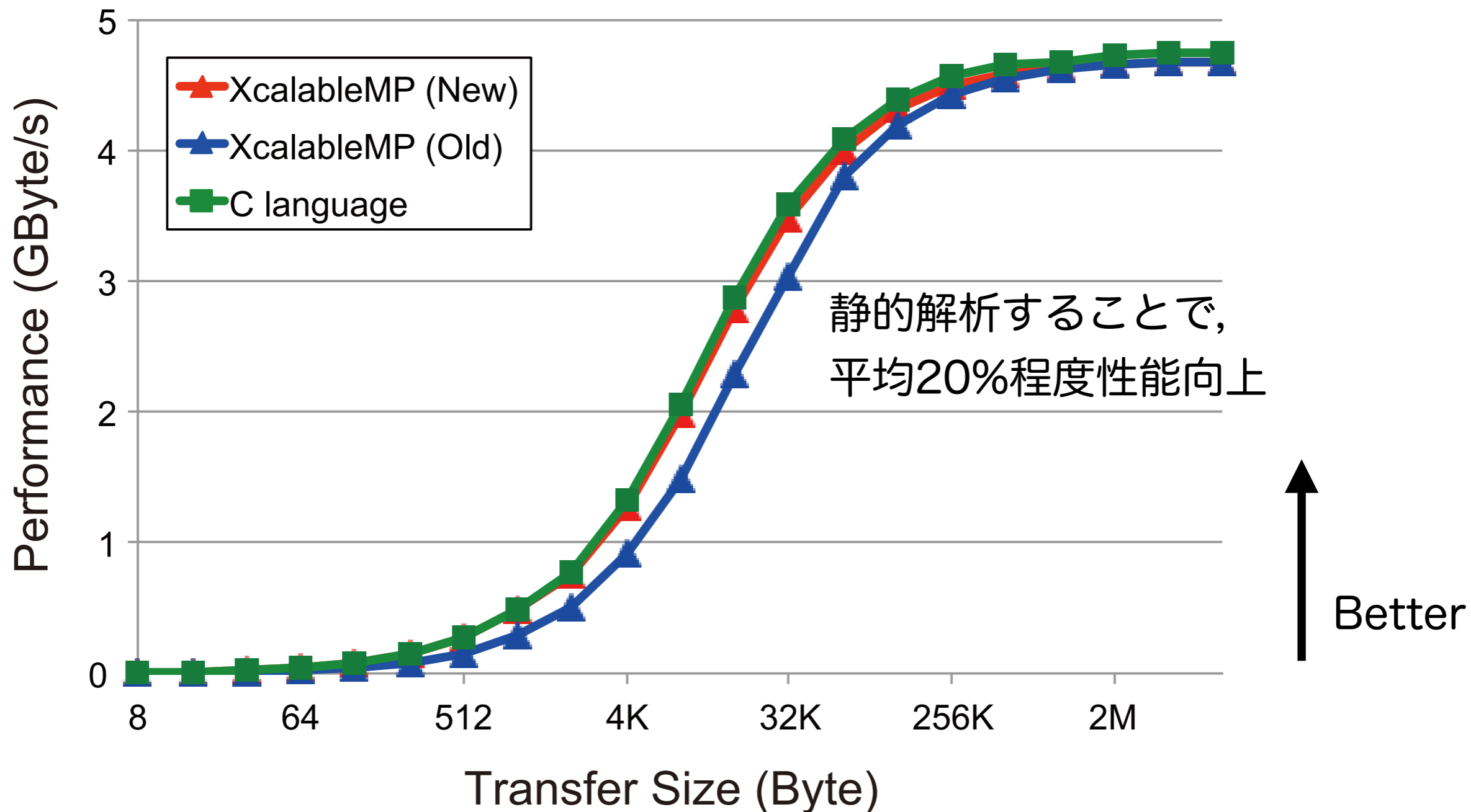
- 片側通信に関する機能 (put/get/post-wait/lock-unlock/sync.)
- 基本的には, 片側通信ライブラリGASNetを利用 (<http://gasnet.lbl.gov>)
- 「京」では, 富士通RDMAを利用
- コードを静的に解析し, 連続領域の転送の場合は, ランタイム内のいくつかの処理をショートカット



静的解析して,  
連続領域の転送で  
あれば新XMP関数を  
呼ぶように置換する

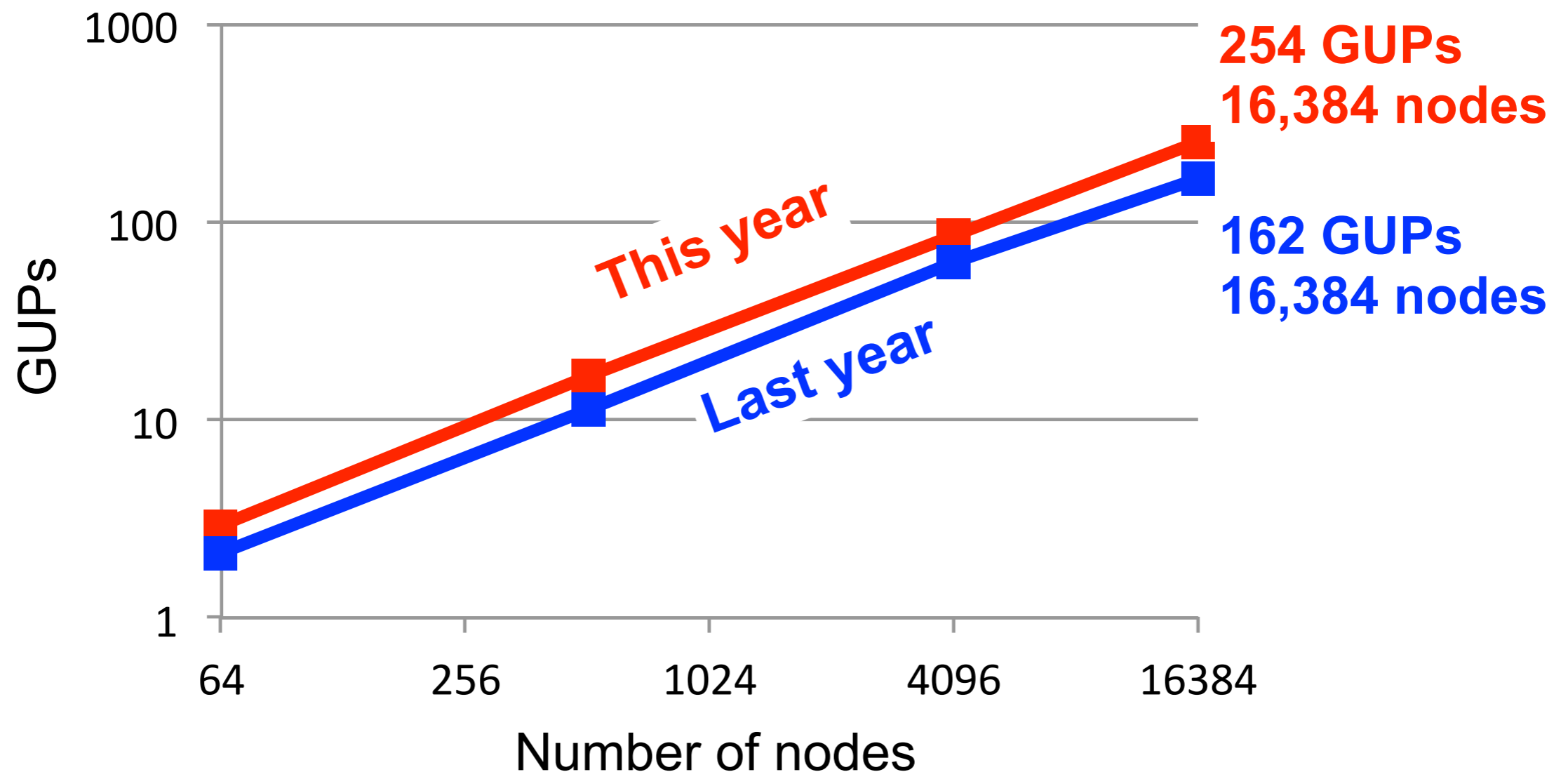
# 片側通信機能

Putの性能比較：C言語から直接呼び出した場合とほぼ同じ性能



# Performance of RandomAccess

MPI\_Send/Recvを用いて実装されていたpost/wait指示文を「京」が提供しているRDMAを用いて再実装.





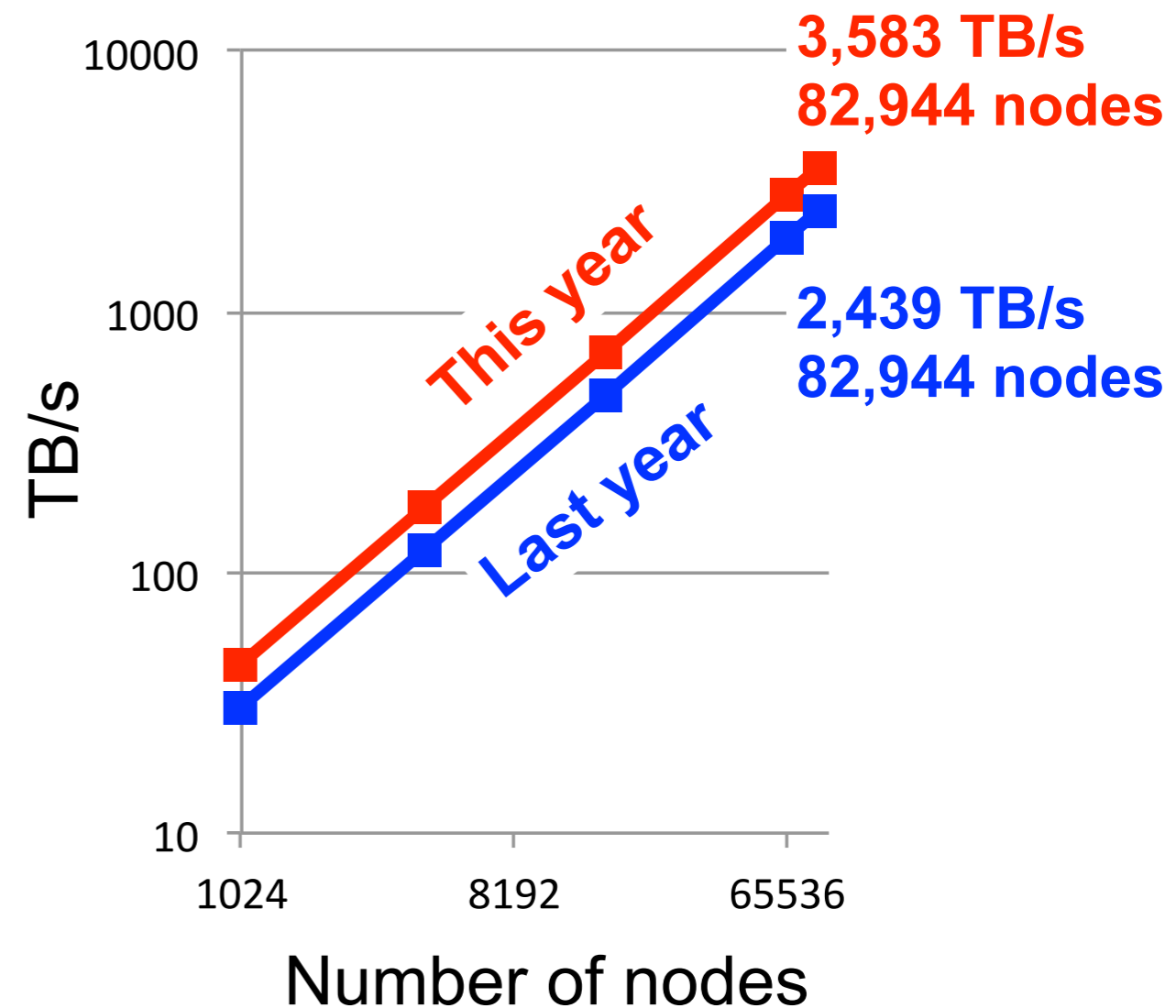
# STREAM

Fujitsuコンパイラの  
最適化指示文を追加

```
#pragma xmp nodes p(*)  
  
#pragma loop xfill  
#pragma loop noalias  
#pragma omp parallel for  
  for (i=0; i<N; i++)  
    a[i] = b[i] + scalar*c[i];  
  
#pragma xmp reduction(+:triadGBs)
```

C言語やFortranの最適化手法が  
そのまま適用できる

## STREAM (XMP/C)



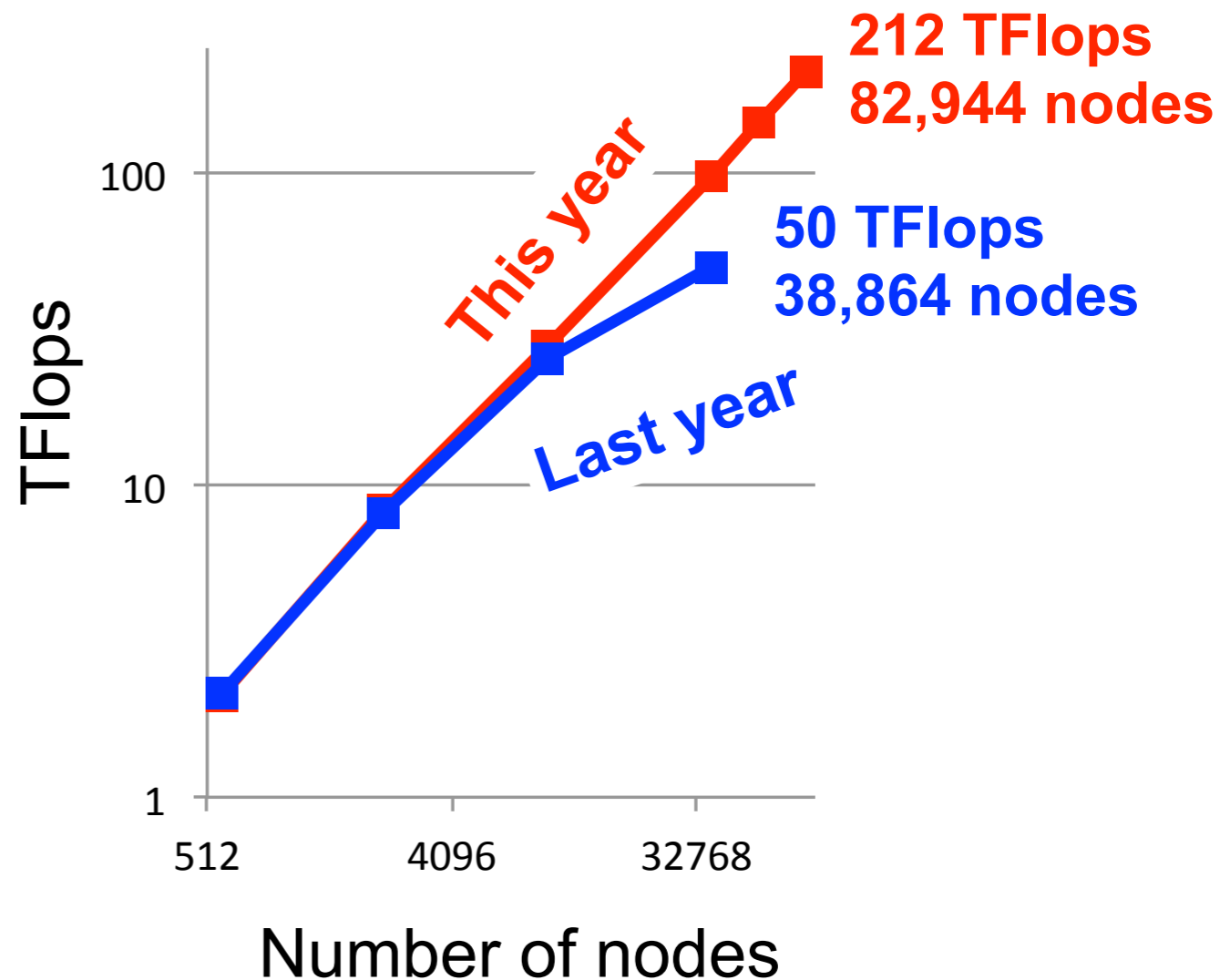
# FFT

1プロセスが、スレッド化したFFTEライブラリを呼ぶように変更。  
コードクリーンナップ。

```
!$xmp loop on tx(i)
do i=1,nx
  call zfft1d(b(1,i),ny,-1,cy)
end do

!$xmp loop on tx(i)
!$omp parallel do
do i=1,nx
  do j=1,ny
    b(j,i)=b(j,i)*w(j,i)
  end do
end do

call xmp_transpose(a,b,1)
```



# Results and Machine

## Summary

Benchmark		# Nodes	Performance (/peak)	SLOC
HPL	Ver. 1	16,384	971 TFlops (46.3%)	313
	Ver. 2	4,096	423 TFlops ( <b>80.7%</b> )	426
RandomAccess		16,384	254 GUPs	253
STREAM		<b>82,994</b>	3,583 TB/s (67.5%)	69
FFT		<b>82,944</b>	212 TFlops (2.0%)	205

## The K computer: 82,944 nodes



<http://www.aics.riken.jp/jp/outreach/photogallery.html>

- SPARC64 VIIIfx Chip, 128 GFlops
- DDR3 SDRAM 16GB, 64GB/s
- Tofu Interconnect
  - 6D mesh/torus network
  - 5GB/s x 4links x 2

# 他のPGAS言語とMPIとの行数の比較

Lang.	HPL	RandomAccess	FFT	STREAM
XMP	313 (ver. 1) 426 (ver. 2)	253	205	69
PCJ	<N/A>	146	498	180
Coarray Fortran	786	409	450	63
Chapel	658	112	<N/A>	72
X10	708	143	236	60
MPI (hpcc-1.4)	8,800	787	1904	329

# まとめと今後の課題

---

- XMPの概要と最近の活動について紹介
- 並列言語のコンテストHPC Challenge Awards Competition Class 2への応募
  - HPL, FFT, STREAM, FFTの実装
  - **Best Performance Awardを受賞**
- 今後の予定
  - 2015年4月にOmni Compiler 0.95のリリース
    - XMP/FortranにおけるCoarray機能
  - ハンドブックの作成
  - C++, Fortran2003 (今はFortran95)
  - アクセラレータ対応 (OpenACCとの組合せ)