並列プログラミング言語XcalableMPと大規模シミュレーション向け並列プログラミングモデルの動向

理研AICS プログラミング環境研究チーム

村井 均

はじめに

- 大規模シミュレーションなどの計算を行うためには、クラスタのような分散メモリシステムの利用が一般的
- 分散メモリ向け並列プログラミングの現状
  - 大半はMPI (Message Passing Interface)を利用
  - MPIはプログラミングコストが大きい

➡ 高性能と高生産性を兼ね備えた新しいプログラミングが必要

HPF (High Performance Fortran)

- Fortran90 + 指示文
  - データ分散→プログラマ、通信と並列化→処理系
- MPI代替として期待されたが、2000年ごろまでに失速。
- 敗因:
  - 初期の処理系の品質が悪く、早々に見切りをつけられた。
  - 処理系の解析・最適化に強く依存する仕様

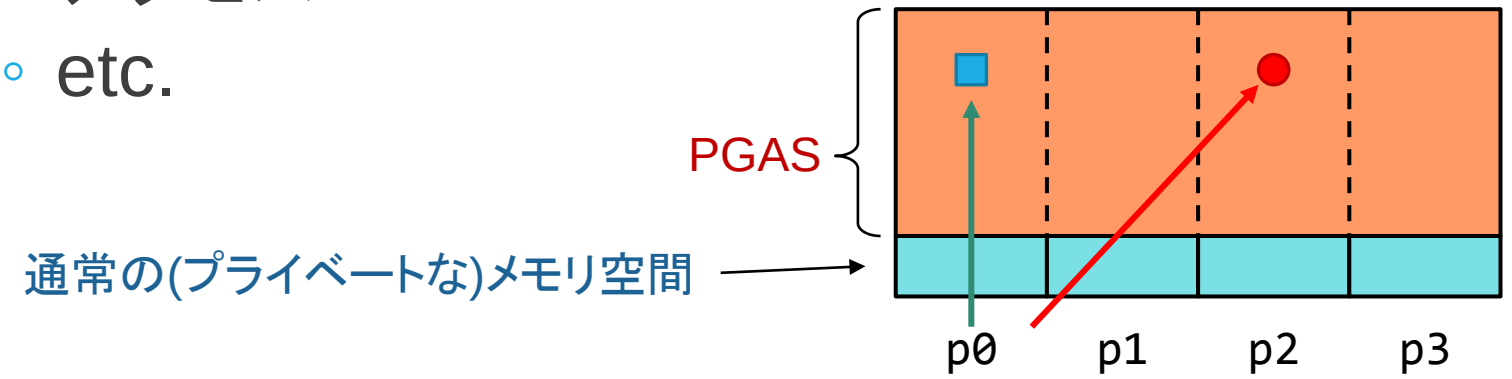
  → チューニング困難、低い性能移植性

Partitioned Global Address Space (PGAS)

- “Global”
  - 全てのプロセスはアドレス空間を共有する (リモートデータを参照できる)。
- “Partitioned”
  - リモートデータとローカルデータは区別され、参照の方法やコストは異なる →「データの局所性」
- PGASに基づく新しい並列プログラミングモデルが多く提案・開発されている。

Partitioned Global Address Space (PGAS) 続き

- 個々の言語やプログラミングモデルにおけるPGASの「実装方法」は様々。
  - OSやハードウェアのサポートの有無
  - 片側通信ライブラリの利用
  - 明示的(特別な記法による)または暗黙的なリモートアクセス
  - etc.



PGASの長所と短所

- 長所
  - よりシンプルな表現で「通信」を記述できる。
  - OSやハードウェアのサポート次第では、性能はメッセージパッシングを上回る。
  - コンパイラによる最適化やエラーチェックを期待できる？
- 短所
  - メモリコンシステンシを意識する必要あり。
  - ポータビリティ

PGAS言語/プログラミングモデル

- coarray (in Fortran 2008)
- Unified Parallel C (UPC)
- OpenSHMEM
- X10
- Chapel
- XcalableMP

用語: グローバルビューとローカルビュー

- グローバルビュー
  - 解くべき問題全体を記述し、それをN個のノードが分担する方法を示す。
    - 「問題1～100を4人で分担して解け」
  - グローバルなインデックス空間
  - 分かりやすい
- ローカルビュー
  - 各ノードが解くべき問題を示す。
    - 「ノードnは問題((n-1)*25+1)～(n*25)を解け」
  - ローカルなインデックス空間
  - 自由度が高いが、やや難しい。

coarray (in Fortran 2008)

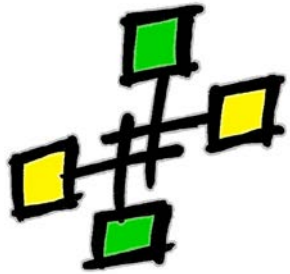
- Fortran 2008標準に含まれるPGAS機能
- 「coarray」として宣言されたデータは、PGAS上に配置され、他イメージ(プロセスに相当)から参照可能。
- SPMD + ローカルビュー → MPIプログラムの通信関数をcoarray代入に置き換えたものに相当
- Intel, Cray, IBM

サンプルコード

```
real, save :: a(0:101)[*] ! aをcoarrayとして宣言

me = this_image()         ! イメージ番号を取得

a(0) = a(100)[me - 1]     ! 隣接イメージ上のaを参照
a(101) = a(1)[me + 1]     !

sync all                  ! 同期

do i = 1, 100
  b(i) = (a(i-1) + a(i) + a(i+1)) / 3
end do
```

[ ]がない場合、通常のデータとしてアクセスされる。

OpenSHMEM



- 各社（SGI, Quadrics, HP, ...）が提供してきた片側通信ライブラリSHMEMのオープンソース実装
- SPMD + ローカルビュー→ MPIプログラムの通信関数をshmem_put等に置き換えたものに相当

サンプルコード

```
real, save :: a(0:101)                          ! 対象データ

me = my_pe()                                    ! pe番号を取得

call shmem_get(a, a(100), 4, me-1)     ! 隣接pe上のaを参照
call shmem_get(a(101), a(1), 4, me+1) !

call shmem_barrier                              ! 同期

do i = 1, 100
  b(i) = (a(i-1) + a(i) + a(i+1)) / 3
end do
```

Unified Parallel C (UPC)

- C99の拡張
- GWU, UC Berkley/LBNLが主導
- 「共有データ」
  - 全スレッドからシームレスにアクセス可能
  - 一次元ブロックサイクリック分散
- グローバルビュー
- Berkley UPC, GNU UPC, IBM, HP, Cray

サンプルコード（通常版）

```
shared [*] float a[400];        ! aを共有データとして宣言

upc_forall (i = 1; i < 399; i++; &a[i]){
  b[i] = (a[i-1] + a[i] + a[i+1]) / 3
}
```

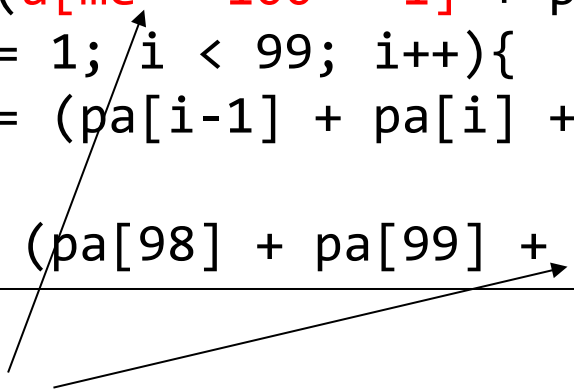
aの分散に合わせて並列化

ブロック幅（「*」は均等ブロックを意味する）

共有データであるaの全ての参照は、高コストなランタイム呼び出しに変換される。

サンプルコード（高速版）

```
shared [*] float a[400];      ! aを共有データとして宣言
float *pa = (float *)a;   ! aに対するローカルなエイリアス

b[0] = (a[me * 100 - 1] + pa[0] + pa[1]) / 3;
for (i = 1; i < 99; i++){
  b[i] = (pa[i-1] + pa[i] + pa[i+1]) / 3
}
b[99] = (pa[98] + pa[99] + a[(me + 1) * 100]) / 3;
```



リモートアクセス

グローバルビューの利点はなくなっている？

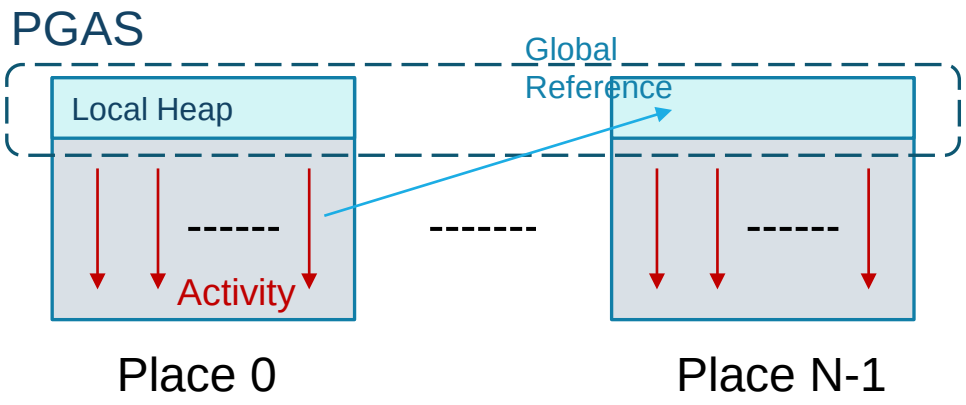
X10

- IBMが提案・開発中の新言語 ← DARPAのHPCSプログラム（2002～2010）
- Javaベース（OO）
- 階層的並列処理（スレッド+Place）
- グローバルビューに基づく分散配列
- 明示的な通信（リモートオブジェクトへのポインタによる参照）

サンプルコード

```
val R = Region.make(1..1000);
val D = Dist.makeBlock(R);
val a = DistArray.make[Float](D);

for (p in D) a(p) = ...;
```

データ並列処理の例




```
def fib(n:Int):Int {
  if (n < 2) return 1;
  var f1:Int;
  var f2:Int;
  finish {
    async f1 = fib(n-1);
    f2 = fib(n-2);
  }
  return f1+f2;
}
```

マルチスレッド処理の例（フィボナッチ）

Chapel

- Crayが提案・開発している新言語 ← DARPAのHPCSプログラム（2002～2010）
- Pascalっぽい文法（OO）
- 階層的並列性（スレッド+Locale）
- グローバルビューに基づく分散配列（HPF/ZPL由来のデータ並列処理）
- 暗黙的な通信

サンプルコード

```
const Space = {1..8, 1..8};
const D: domain(2) dmapped Block(boundingBox=Space) = Space;
var A: [D] int;

forall a in A do
  a = ...;
```

データ並列処理の例

```
proc fib(n:int):int {
  if (n < 2) return 1;
  var f1:int;
  var f2:int;
  sync {
    begin f1 = fib(n-1);
    f2 = fib(n-2);
  }
  return f1+f2;
}
```

マルチスレッドの例（フィボナッチ）

XcalableMP

- 次世代並列プログラミング言語検討委員会 / PCクラスタコンソーシアムXcalableMP規格部会で検討中。
- MPIに代わる並列プログラミングモデル
- 目標:
  - Performance
  - Expressiveness
  - Optimizability
  - Education cost

www.xcalablemp.org

XcalableMPの特徴（1）

- Fortran/Cの拡張（指示文ベース）
  → 逐次プログラムからの移行が容易
- SPMDモデル
  - 各ノード（並列実行の主体）が独立に（重複して）実行を開始する。

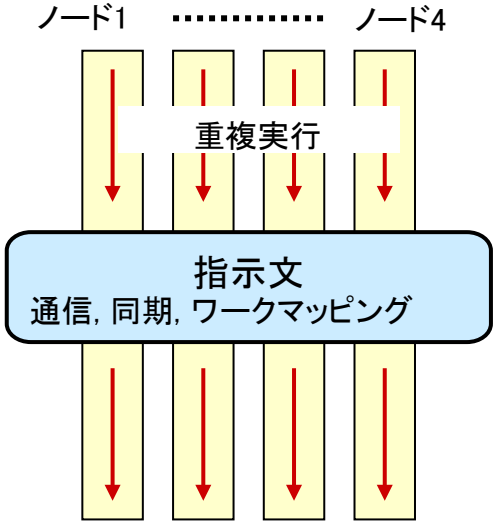
XcalableMPの特徴（2）

- 明示的な並列化と通信
  - ワークマッピング（並列処理）、通信、および同期は「集団的」な指示文によって明示される。

  → チューニングが容易
- 2つのプログラミングモデル
  - グローバルビュー
  - ローカルビュー

XMPの実行モデル(SPMD)

- 各ノードは、同一のコードを独立に(重複して)実行する。
- 指示文の箇所では、全ノードが協調して動作する(集団実行)。
  - 通信・同期
  - ワークマッピング(並列処理)



メモリモデル

- 各ノードは、自身のローカルメモリ上のデータ（ローカルデータ）のみをアクセスできる。
- 他のノード上のデータ（リモートデータ）にアクセスする場合は、特殊な記法による明示的な指定が必要。
  - 通信指示文
  - coarray
- 「分散」されないデータは、全ノードに重複して配置される。

プログラム例（MPIとの比較）

XMP/Cプログラム

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

main(){
#pragma xmp loop on t(i) reduction(+:res)
   for (i = 0; i < MAX; i++){
      array[i] = func(i);
      res += array[i];
   }
}
```

シンプル

MPIプログラム

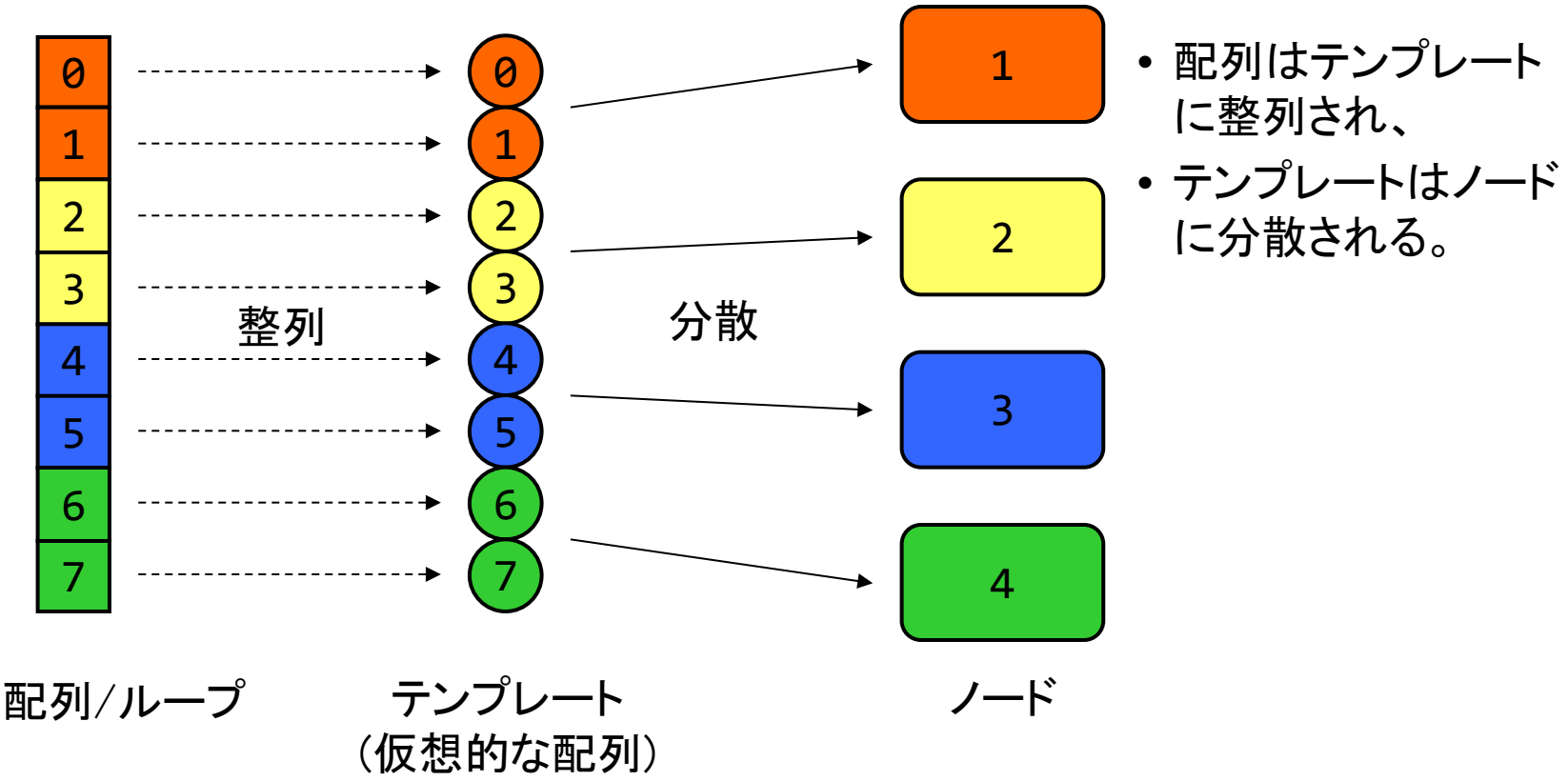
```
int array[MAX];

main(int argc, char **argv){
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    dx = MAX/size;
    llimit = rank * dx;
    if (rank != (size -1)) ulimit = llimit + dx;
    else ulimit = MAX;
    temp_res = 0;

    for (i = llimit; i < ulimit; i++){
       array[i] = func(i);
       temp_res += array[i];
    }

    MPI_Allreduce(&temp_res, &res, 1, MPI_INT,
                  MPI_SUM, MPI_COMM_WORLD);

    MPI_Finalize( );
}
```

グローバルビュー・プログラミング

- 基本的に指示文を挿入するだけ。
- 「分担」を指定する方法
  - データマッピング
  - ワークマッピング
  - 通信・同期

データマッピング

- 整列 + 分散による2段階の処理



- 配列はテンプレートに整列され、
- テンプレートはノードに分散される。

データマッピング指示文

- align指示文の例

  配列aの要素iを、テンプレートtの要素i-1に整列させる。

```
#pragma xmp align a[i] with t(i-1)
```

- distribute指示文の例

  ノード集合pに、テンプレートtをブロック形式で分散する。

```
#pragma xmp distribute t(block) onto p
```

他に、サイクリック、ブロックサイクリック、不均等ブロックを指定できる。

ワークマッピング指示文

- task指示文

```
#pragma xmp task on t(k-1)
{
   a[k] = ...;
}
```

t(k)のオーナが、a(k)への代入を実行する。

- loop指示文(並列ループ)

```
#pragma xmp loop on t(i)
for (i = 0; i < n; i++)
{
   a[i] = ...;
}
```

t(i)のオーナが、繰り返しiにおいて、a[i]への代入を実行する。

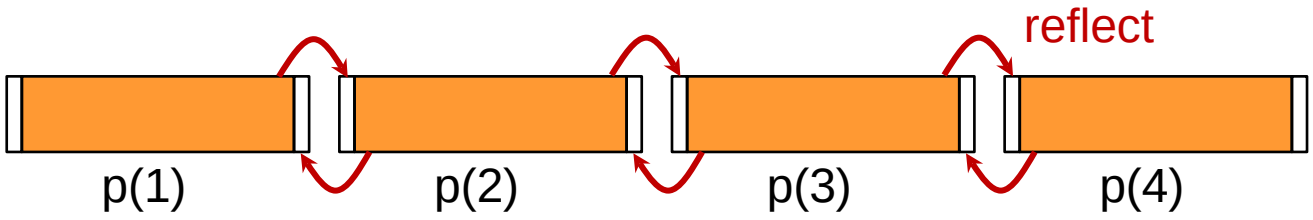
通信指示文(1)

- shadow指示文 & reflect指示文

aの上下端に幅1のシャドウを付加する。

```
#pragma xmp distribute t(block) onto p
#pragma xmp align a[i] with t(i-1)
#pragma xmp shadow a[1:1]
...
#pragma xmp reflect (a)
```

aに対する隣接通信を実行する。

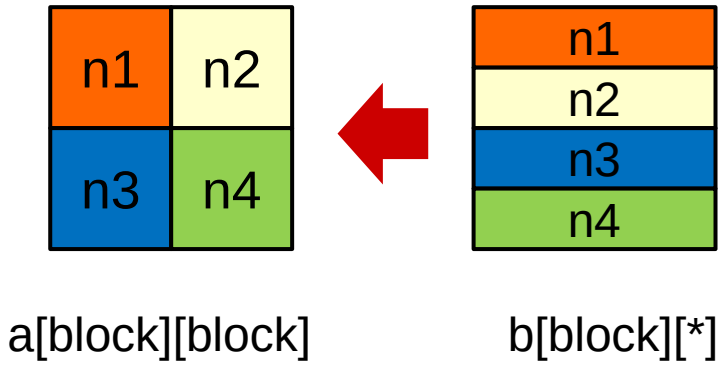


通信指示文(3)

- gmove指示文
  - 通信を伴う任意の代入文を実行する。

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

※ Cで「部分配列」も記述できる。



- その他に、ブロードキャスト(bcast)や集計演算(reduction)を指定できる。

XcalableMPプログラムの例

```
!$xmp nodes p(npx,npy,npz)

!$xmp template (lx,ly,lz) :: t
!$xmp distribute (*,*,block) onto p :: t

!$xmp align (ix,iy,iz) with t(ix,iy,iz) ::
!$xmp&        sr, se, sm, sp, sn, sl, ...

!$xmp shadow (0,0,0:1) ::
!$xmp&        sr, se, sm, sp, sn, sl, ...

      lx = 1024

!$xmp reflect (sr, sm, sp, se, sn, sl)

!$xmp loop on t(ix,iy,iz)
      do iz = 1, lz-1
      do iy = 1, ly
      do ix = 1, lx
         wu0 = sm(ix,iy,iz  ) / sr(ix,iy,iz  )
         wu1 = sm(ix,iy,iz+1) / sr(ix,iy,iz+1)
         wv0 = sn(ix,iy,iz  ) / sr(ix,iy,iz  )
         ...
```

ノード集合の宣言

テンプレートの宣言と分散の指定

整列の指定

シャドウの指定

重複実行される

隣接通信の指定

ループの並列化の指定

ローカルビュー・プログラミング

- 自由度が高いが、やや難しい。
- ローカルビューのための機能として、Fortran 2008から導入したcoarrayをサポート。
  - XMP/Cでもサポート
- グローバルビューとローカルビューを併用可能
  - 全体をグローバルビューで、ホットスポットのみローカルビューで。
  - 場をグローバルビューで、粒子をローカルビューで。

Omni XcalableMP

- 理研AICSと筑波大で開発中のXMP処理系
  - XMP/C
  - XMP/Fortran
- オープンソース
- トランスレータ + ランタイム(MPIベース)
- 対応プラットフォーム
  - Linuxクラスタ、Crayマシン、京コンピュータ、NEC SX、地球シミュレータ
  - その他、MPIが動作している任意のシステム

現況

- プロトタイプ(ver. 0.7.0)を公開中
  - XMPの主要な機能を実装済み
  - 一部制限事項あり
- 拡張機能
  - アクセラレータ向け拡張 (XMP-dev)
  - プロファイラ・インタフェース
- 今後の予定
  - ver.0.8.0 (4月), ver.1.0 (11月)

ver. 0.8.0の機能（予定）

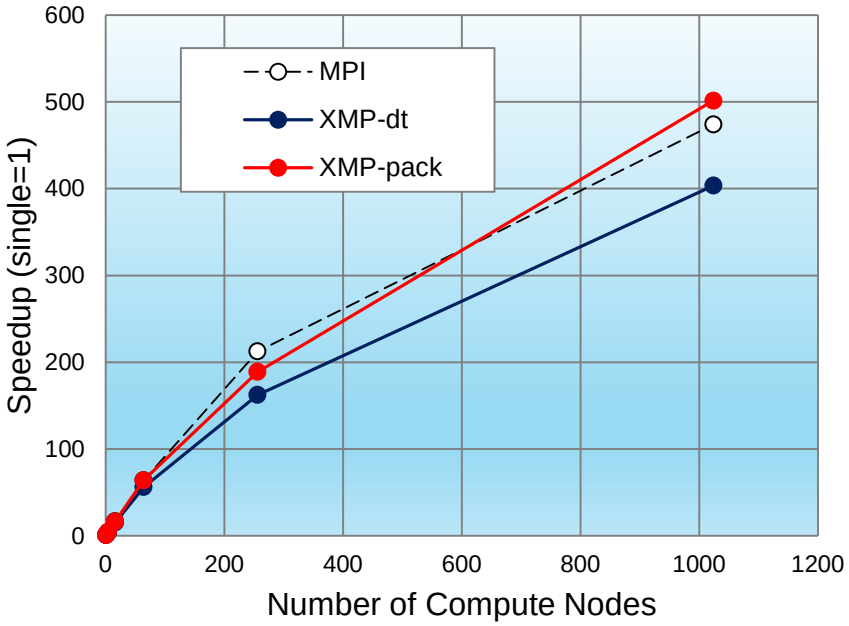
	XMP/C	XMP/F
nodes	○	○
distribute	○	○
align	○	○
shadow	○	○
loop	○	○
task	○	○
reflect	○	○
gmove	△	△
coarray	○	×
組込み手続き	△	△

○ 実装済み。△ 制限あり。× 未実装。赤字:新規

Omni XMPの利用

- ウェブページ
  www.hpcs.cs.tsukuba.ac.jp/omni-compiler/xcalablemp/
  - ソースtarball
  - Debian/Ubuntu/CentOS向けパッケージ
  - チュートリアル
  - サンプルコード
  - サポートML
- 京コンピュータで利用可能
  - /opt/aics/omni にインストール済

性能(1): 気象コード



`reflect` (ステンシル通信)の実装方法
- XMP-packは、並列パック/アンパックを利用
- XMP-dtは、MPIの派生データ型を利用

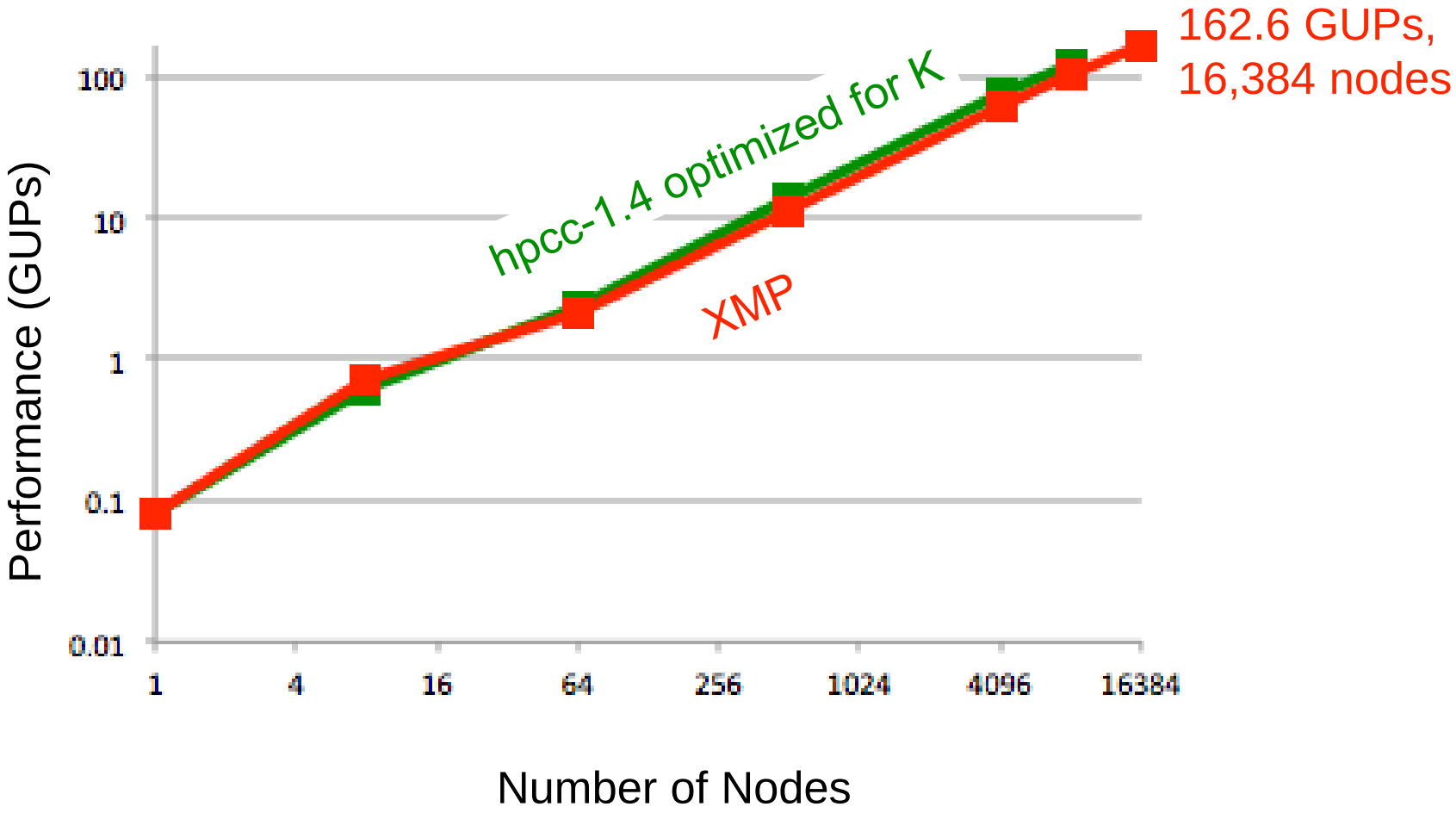
- SCALE-LESの力学コアプロトタイプ
  - 512x512x128
  - 水平方向2Dをブロック分散
  - 500タイムステップ
- 京コンピュータ
  - 言語環境K-1.2.0-13
  - Omni XMP 0.6.1

性能(2): HPCCベンチマーク

- 4～5個のベンチマークにより、プログラミング言語の高性能と高生産性を評価する。
  - Global HPL
  - Global RandomAccess
  - EP STREAM (Triad) per system
  - Global FFT
- 2013年HPCC Award (class 2)はXcalableMPが受賞。

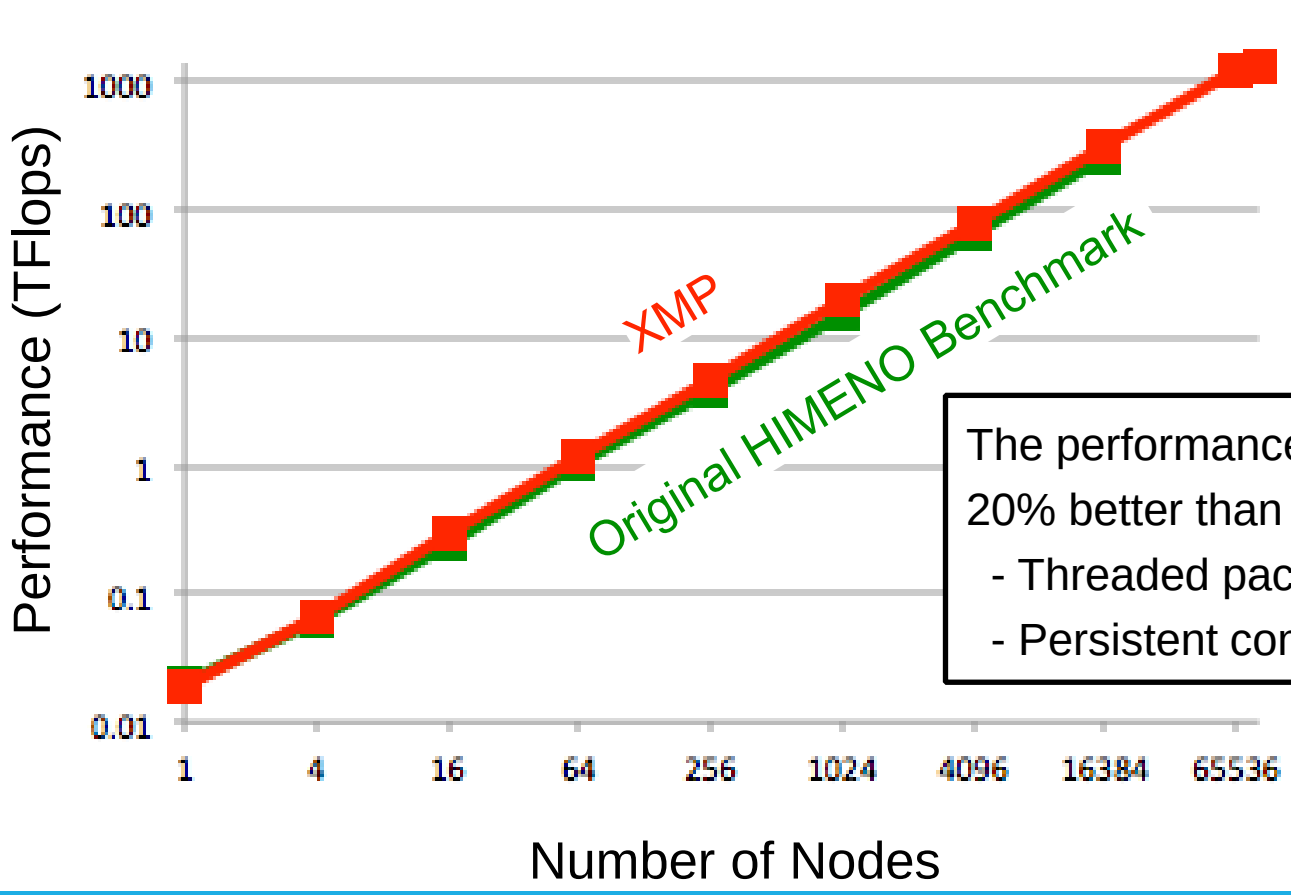
Results of RandomAccess

- RandomAccess（8 processes/node）



Result of HIMENO Benchmark

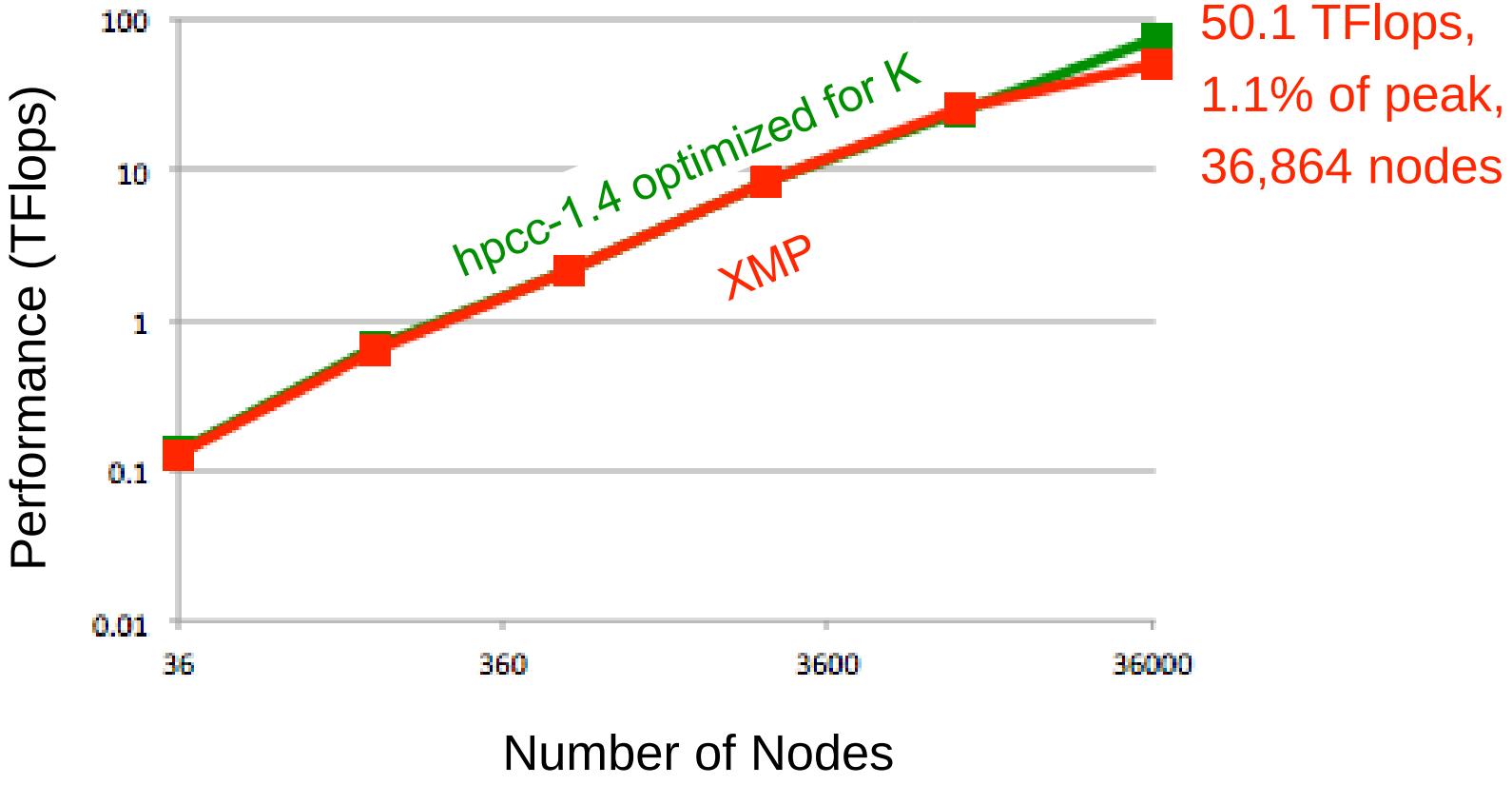
- HIMENO Benchmark (1 process/node with 8 threads)



The performance of the XMP HIMENO is 20% better than that of the original one.
- Threaded pack/unpack operations
- Persistent communication

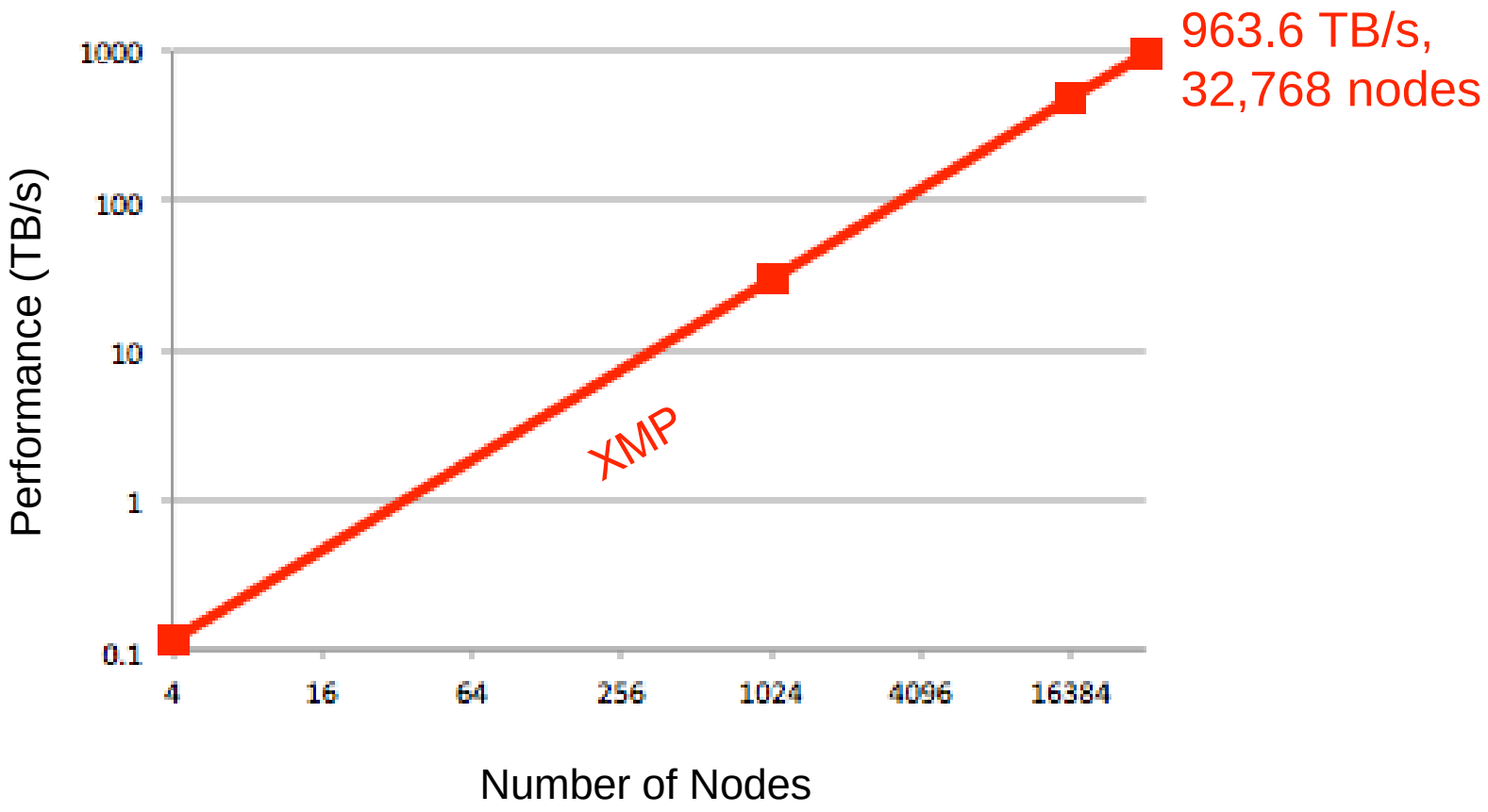
Result of FFT

- FFT (1 process/node with 8 threads)



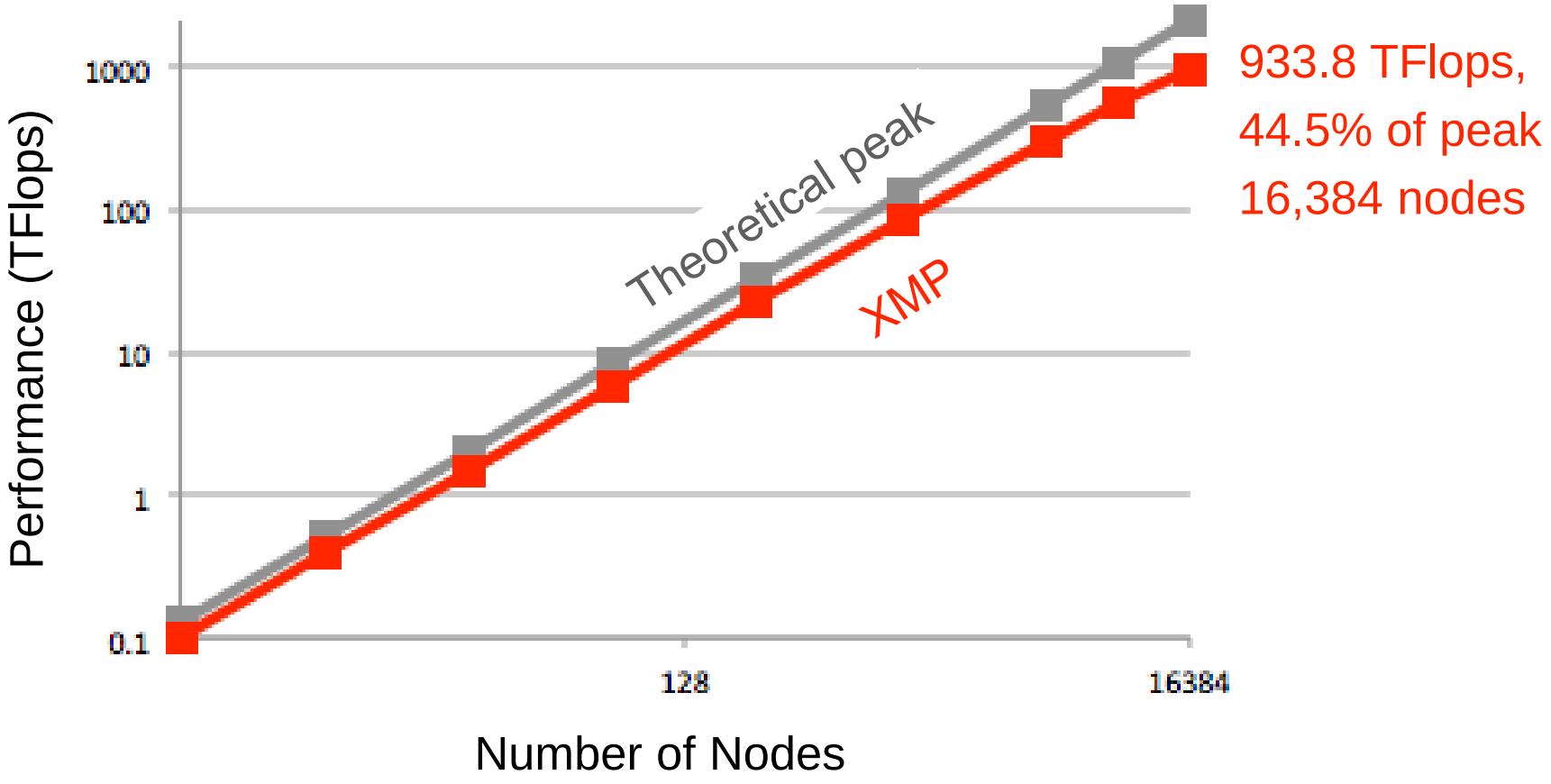
Results of STREAM

- STREAM (1 process/node with 8 threads)



Results of HPL

- HPL (1 process/node + Threaded BLAS)



まとめ

- MPIに代わるプログラミングモデルとして、多くのPGAS言語が提案・開発されている。
- 並列プログラミング言語XcalableMP
  - FortranおよびCに対する拡張（指示文ベース）
  - グローバルビュー＆ローカルビュー
- Omni XcalableMP
  - 理研と筑波大が開発中のXMP処理系
  - 無償でダウンロード・利用可能

個人的な予測

- 地球流体の分野では、coarrayが本命？
  ∵ Fortran
  ∵「標準」に採用されたので、サポートするサイトが増えることが見込まれる。
- XcalableMP/Fortranも有望
  ∵ グローバルビューとローカルビュー（coarray）の両方を利用可能。

XMP講習会

- 2014年度に講習会を予定。
  - 7/16(水), 9/18(木), 12/18(木)
  - 座学（本発表と同内容）および実習
- 計算科学振興財団（FOCUS）のウェブページ（www.j-focus.or.jp）より申込み。
  ※ 現時点では、まだ募集は始まっていない模様。