## 本日のチュートリアルの趣旨

- 今回は、PGAS言語の動向とXcalableMPについての概要だけでなく、Dockerを使ったデモや、post-peta CRESTで研究開発されたGPUクラスタのための拡張 XcalableACC, レファレンス実装であるOmni XMPコンパイラの解説、現在、規格部会で議論しているタスク並列への拡張 XMP 2.0まで、以下のスケジュールでご紹介いたします。

RIKEN AICS

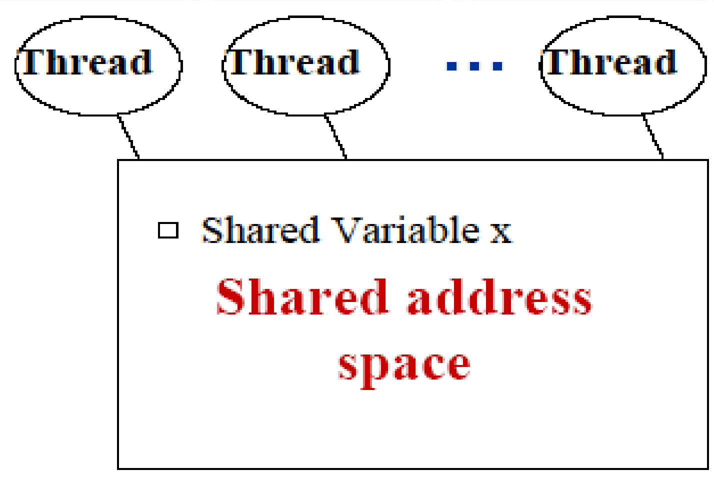
## 本日の予定

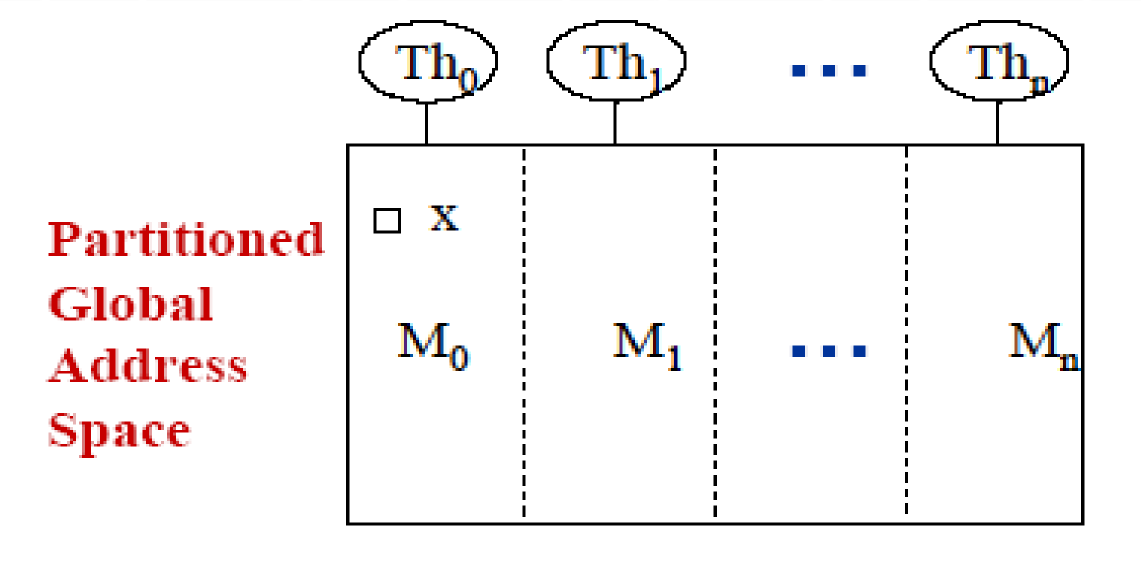
- **13：00～13：20　：PGASの動向とXcalableMP**
  - 佐藤 三久（理化学研究所）
- **13：20～14：00　：  XcalableMPの詳細**
  - 李 珍泌（理化学研究所）
- **14：00～14：20：XcalableMPデモ**
  - 中尾 昌広（理化学研究所）
- **(14：20～14：30　　　　休憩（コーヒーブレイク））**
- **14：30～15：30　：XcalableACCの概要**
  - 中尾 昌広
- **15：30～16：30　：Omniコンパイラの概要とXMP2.0について**
  - 佐藤 三久

# PGASとはなにか？
# その動向…

# PGAS: Partitioned Global Address Space Programming Model

- ユーザがlocal/globalを宣言する（意識する）
- Partitioned Global Address Space (PGAS) model
- スレッドと分割されたメモリ空間は、対応ずけられている（affinity)
  - 分散メモリモデルに対応



# Partitioned Global Address Spaceプログラミング言語

- 「shared/global」の発想は、いろいろなところから同時に出てきた。
  - Split-C
  - PC++
  - UPC
  - CAF: Co-Array Fortran
  - X10, Chapel
  - Global Array
  - OpenShmem
  - XcalableMP

# U P C

- **Unified Parallel C**
  - Lawrence Berkeley National Lab.を中心に設計開発
- **Private/Shared を宣言**
- **SPMD**
  - MYTHREADが自分のスレッド番号
  - 同期機構
    - **Barriers**
    - **Locks**
    - **Memory consistency control**
- **User's view**
  - 分割されたshared spaceについて、複数のスレッドが動作する。
  - ただし、分割されたshared spaceはスレッドに対してaffinityを持つ。

```
//vect_add.c
#include <upc_relaxed.h>
#define N 100*THREADS
shared int v1[N], v2[N],
 v1plusv2[N];

void main(){
     int i;
  for(i=0; i<N; i++)
    if (MYTHREAD==i%THREADS)
     v1plusv2[i]=v1[i]+v2[i];
}
```

```
//vect_add.c
#include <upc_relaxed.h>
#define N 100*THREADS
shared int v1[N], v2[N],
  v1plusv2[N];

void main()
{
     int i;
    upc_forall(i=0; i<N; i++; i)
        v1plusv2[i]=v1[i]+v2[i];
}
```

Affinityが書けるfor構文

# UPC: Shared宣言

- Sharedというqualifierを導入する
- Shared array の要素とブロックはthreadのメモリ空間に分散配置される。

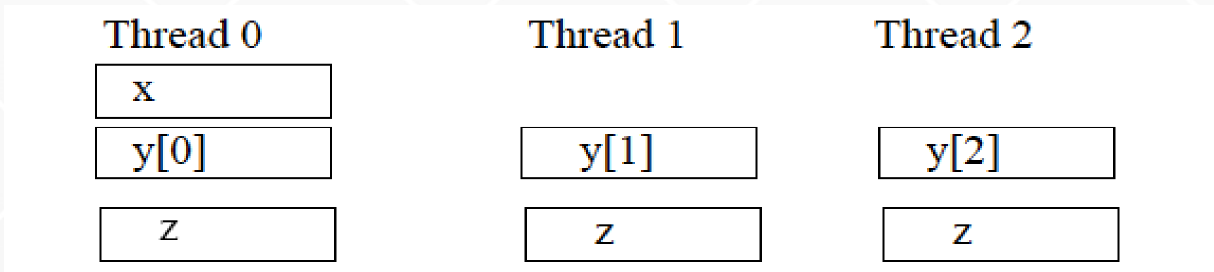
```
shared int x[THREADS] /*One element per thread */
shared int y[10][THREADS] /*10 elements per thread */
```

- sharedなスカラー変数

```
shared int a;
      /*One item on system (affinity to thread 0) */
int b; /* one private b at each thread */
```

- Shared Pointer

```
shared int *p;
```

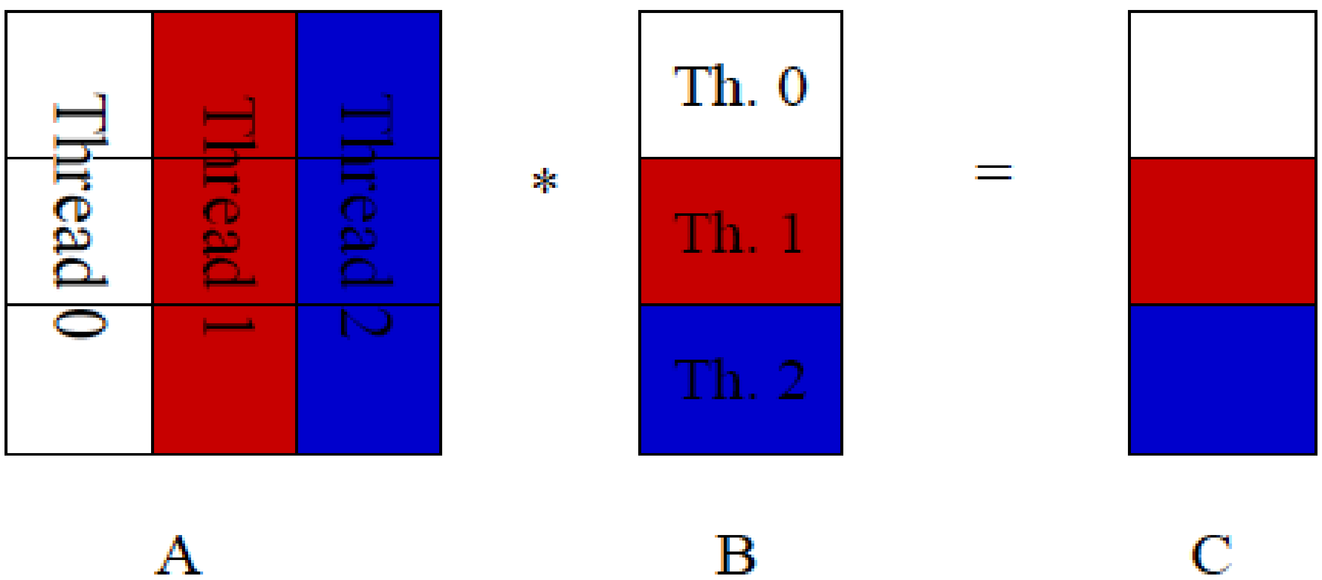



- THREADS = 3の場合

```
shared int x;
     /*x will have affinity to thread 0 */
shared int y[THREADS];
int z;
```

# 行列積の例

```
#include <upc_relaxed.h>
shared int a[THREADS][THREADS];
shared int b[THREADS], c[THREADS];
void main (void) {
     int i, j;
     upc_forall( i = 0 ; i < THREADS ; i++; i) {
         c[i] = 0;
         for ( j= 0 ; j < THREADS ; j++)
              c[i] += a[i][j]*b[j];
     }
}
```



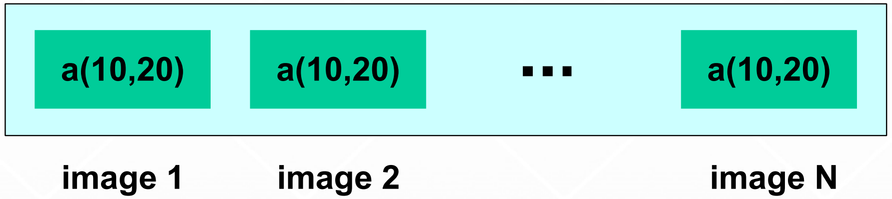
# Co-Array Fortran

- **Fortran 2008 で正式サポート**
- **PGAS programming model**
  - one-sided communication (GET/PUT)
- **SPMD 実行を前提**
- **Co-array extension**
  - 各プロセッサで動くプログラムは、異なる"**image**"を持つ。

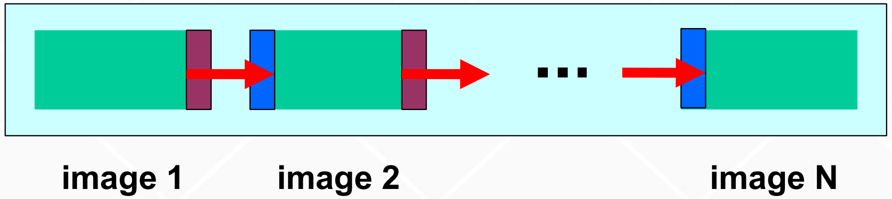
```
real, dimension(n)[*] :: x,y
x(:) = y(:)[q]
```

**qのimageで動くyのデータをローカルなxにコピする(get)**

```
integer a(10,20)[*]
```




```
if (this_image() > 1)
    a(1:10,1:2) = a(1:10,19:20)[this_image()-1]
```




- **プログラマは、パフォーマンス影響を与える要素に対して制御する。**
  - データの分散配置
  - 計算の分割
  - 通信をする箇所
- **データの転送と同期の言語プリミティブをもっている。**
  - amenable to compiler-based communication optimization

# CAF

- **SPMD process images**
  - fixed number of images during execution
  - images operate asynchronously
- **Both private and shared data**
  - **real x(20, 20)** a private 20x20 array in each image
  - **real y(20, 20)[*]** a shared 20x20 array in each image
- **Simple one-sided shared-memory communication**
  - **x(:,j:j+2) = y(:,p:p+2)[r]** copy columns from image **r** into local columns
- **Synchronization intrinsic functions**
  - **sync_all** – a barrier and a memory fence
  - **sync_mem** – a memory fence
  - **sync_team(**[team members to notify], [team members to wait for]**)**
- **Pointers and (perhaps asymmetric) dynamic allocation**

RIKEN AICS

# 行列積の例

- 2次元のco-Array notationも可能

  ２次元のCo-dimension

```
real,dimension(n,n)[p,*] :: a,b,c
do k=1,n
    do q=1,p
      c(i,j)[myP,myQ] = c(i,j)[myP,myQ]
          + a(i,k)[myP, q]*b(k,j)[q,myQ]
    enddo
enddo
```

# PGAS プログラミングモデルの動向

- **ライブラリ的なアプローチが多くなりつつある。**
  - Global Array
  - OpenShmem
  - MPI3のput/get (モデルではないが)
  - …

- **UPCも、UPC++でtemplate programming でサポート**

- **CAFは、正式にFortran 2008でサポート**

# XcalableMP

- 分散メモリ環境を対象とした指示文ベースの並列言語
- 次世代並列プログラミング言語検討委員会 → 当部会において仕様を検討、提案。
- 2つの並列プログラミングモデルをサポート
  - グローバルビューモデルによる定型的な並列化
  - ローカルビューモデルによる自由度の高い並列化

# XcalableMPの現況

- **PCクラスタコンソーシアムで規格を議論**
  - 2017/April Version 1.3仕様を公開。
    - Version 1の系統はほぼ「収束」
  - 2015年より、次期仕様「XMP2.0」の検討を開始。
    - PGAS + Multitasking for Multicore
    - Code transformation for Optimization
    - (Accelerator )
- **理研・筑波大で、レファレンス実装**
  - Omni XMP コンパイラ

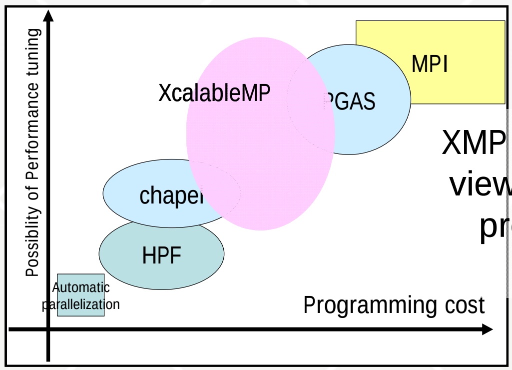
www.xcalablemp.org

omni-compiler.org

RIKEN AICS

# XcalableMP(XMP) http://www.xcalablemp.org

- **What's XcalableMP (XMP for short)?**
  - A PGAS programming model and language for distributed memory , proposed by **XMP Spec WG**
  - XMP Spec WG is a special interest group to design and draft the specification of XcalableMP language. It is now organized under **PC Cluster Consortium**, Japan. Mainly active in Japan, but open for everybody.
- **Project status (as of June 2016)**
  - XMP Spec **Version 1.2.1** is available at XMP site. new features: mixed OpenMP and OpenACC , libraries for collective communications.
  - Reference implementation by U. Tsukuba and Riken AICS: **Version 1.0 (C and Fortran90)** is available for PC clusters, Cray XT and K computer. Source-to-Source compiler to code with the runtime on top of MPI and GasNet.
- **HPCC class 2 Winner 2013. 2014**




XMP provides a global view for data parallel program in PGAS model

- Language Features
  - Directive-based language extensions for Fortran and C for PGAS model
  - Global view programming with global-view distributed data structures for data parallelism
    - SPMD execution model as MPI
    - pragmas for data distribution of global array.
    - Work mapping constructs to map works and iteration with affinity to data explicitly.
    - Rich communication and sync directives such as "gmove" and "shadow".
    - Many concepts are inherited from HPF
  - Co-array feature of CAF is adopted as a part of the language spec for local view programming (also defined in C).

Code example

```
int array[YMAX][XMAX];

#pragma xmp nodes p(4)
#pragma xmp template t(YMAX)
#pragma xmp distribute t(block) on p
#pragma xmp align array[i][*] to t(i)

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

#pragma xmp loop on t(i)  reduction(+:res)
  for(i = 0; i < 10; i++)
    for(j = 0; j < 10; j++){
      array[i][j] = func(i, j);
      res += array[i][j];
    }
}
```

data distribution

add to the serial code : incremental parallelization

work sharing and data synchronization

# Example of a Global-view XMP Program

- **Collaboration in Scale project (with Tomita's Climate Science Team)**
- **Typical Stencil Code**

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

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

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

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

      ...

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

!$xmp loop (ix,iy,iz) 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  )
         ...
```

declare a node array

declare and distribute a template

align arrays

add shadow area

stencil communication

parallelize loops

# Local-view XMP program: Coarray

- **XMP includes the coarray feature imported from Fortran 2008 for the local-view programming.**
  - Basic idea: data declared as *coarray* can be accessed by remote nodes.
  - Coarray in XMP/Fortran is fully compatible with Fortran 2008.

```
real b(8)[*]

if (xmp_node_num() == 1) then
  a(:) = b(:)[2]
```

b is declared as a coarray.

Node 1 *gets* b from node 2.

- **Coarrays can be used in XMP/C.**
  - The subarray notation is also available as an extension.

- Declaration: `float b[8]:[*];`
- Put: `a[0:3]:[1] = b[3:3];` ← *puts* b to node 1.
- Get: `a[0:3] = b[3:3]:[2];` ← *gets* b from node 2.
- Synchronization: `void xmp_sync_all(int *status)`

# XcalableMP as evolutional approach

- **We focus on migration from existing codes.**
  - Directive-based approach to enable parallelization by adding directives/pragma.
  - Also, should be from MPI code. Coarray may replce MPI.
- **Learn from the past**
  - Global View for data-parallel apps. Japanese community had experience of HPF for Global-view model.
- **Specification designed by community**
  - Spec WG is organized under the PC Cluster Consortium, Japan
- **Design based on PGAS model and Coarray (From CAF)**
  - PGAS is an emerging programming model for exascale!
- **Used as a research vehicle for programming lang/model research.**
  - XMP 2.0 for multitasking.
  - Extension to accelerator (XACC)

# XcalableACC(ACC) = XcalableMP+OpenACC+α

- **Extension of XcalableMP for GPU**
  - A part of JST-CREST project "Research and Development on Unified Environment of Accelerated Computing and Interconnection for Post-Petascale" of U. Tsukuba leaded by Prof. Taisuke Boku
  - An "orthogonal" integration of XcalableMP and OpenACC
    - Data distribution for both host and GPU by XcalableMP
    - Offloading computations in a set of nodes by OpenACC
  - Proposed as unified parallel programming model for many-core architecture & accelerator
    - GPU, Intel Xeon Phi
    - OpenACC supports many architectures

Source Code Example: NPB CG

```
#pragma xmp nodes p(NUM_COLS, NUM_ROWS)
#pragma xmp template t(0:NA-1,0:NA-1)
#pragma xmp distribute t(block, block) onto p
#pragma xmp align w[i] with t(*,i)
#pragma xmp align q[i] with t(i,*)
double a[NZ];
int rowstr[NA+1], colidx[NZ];
…
#pragma acc data copy(p,q,r,w,rowstr[0:NA+1]¥
                      , a[0:NZ], colidx[0:NZ])
{
   …
#pragma xmp loop on t(*,j)
#pragma acc parallel loop gang
   for(j=0; j < NA; j++){
      double sum = 0.0;
#pragma acc loop vector reduction(+:sum)
      for (k = rowstr[j]; k < rowstr[j+1]; k++)
         sum = sum + a[k]*p[colidx[k]];
      w[j] = sum;
   }
#pragma xmp reduction(+:w) on p(:,*) acc
#pragma xmp gmove acc
   q[:] = w[:];
   …
} //end acc data
```