XcalableMP Workshop 2019 へようこそ!

PCクラスタコンソーシアム XcalableMP規格部会長
朴　泰祐（筑波大学 計算科学研究センター）

XcalableMPについて

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
- MPIとの inter-operability
- 一般的なPCクラスタの他、京コンピュータ・ポスト京コンピュータでも稼働（「ポスト京」は予定）

PCクラスタコンソーシアムXcalableMP規格部会の活動

- 年3回の部会開催 (6/5, 10/5, 2/25, いずれも東京)
  - 新仕様V2.0に向けた新しい議論
    - 動的なタスクモデルの実現検討
    - MPIとの連携、C等の言語とのインタラクションのための関数群
  - 現仕様V1.4規格のリリース
    http://xcalablemp.org/download/spec/xmp-spec-1.4.pdf
  - アクセラレータに向けたOpenACCとの融合⇨XcalableACCの規格化の検討
- 第6回XMPワークショップ開催(11/1@筑波大)
  - 共催:理化学研究所、筑波大学計算科学研究センター、高性能Fortran推進協議会
  - Invited Talk 5件 (日独仏共同研究SPPEXAにおけるMYXプロジェクト支援による)
- XMPの仕様・研究内容を本にまとめ出版予定
  - Springerより

XcalableMPの状況

- PCクラスタコンソーシアムで規格を議論
  - 2018/10 Version 1.4仕様を公開。
    - Combined directive
    - Tasklet機能 (draft、V2.0に向けて)
  - 2015年より、次期仕様「XMP2.0」の検討を開始。
    - PGAS + Multitasking for Multicore
    - Code transformation for Optimization
    - XcalableACC for Accelerator

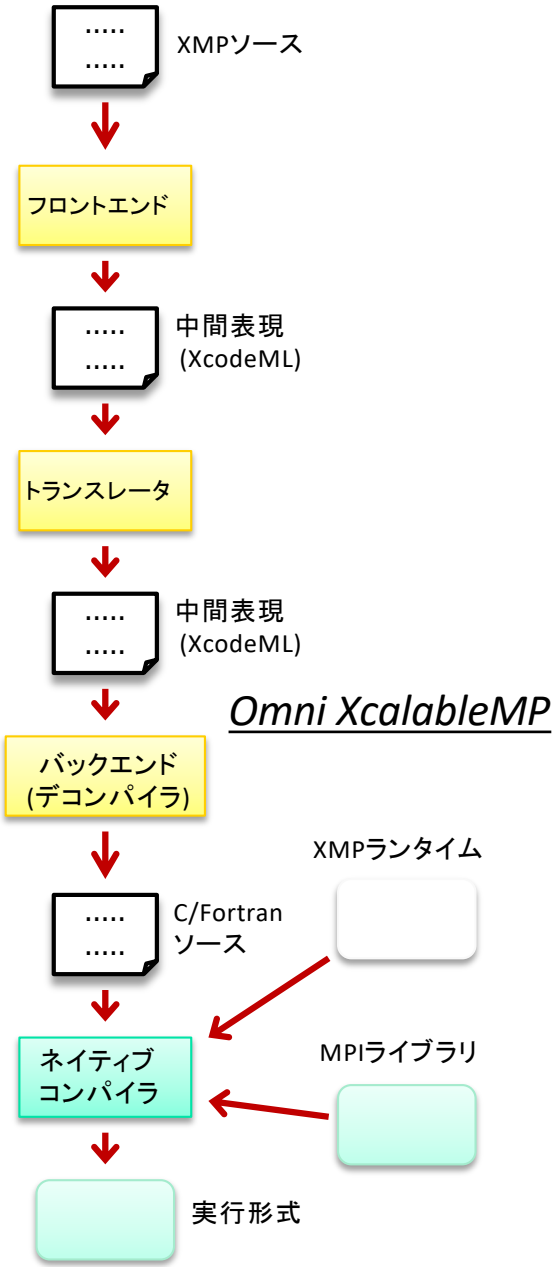
- 理研・筑波大で、リファレンス実装
  - Omni XMP コンパイラ

www.xcalablemp.org

omni-compiler.org

Omni XcalableMP

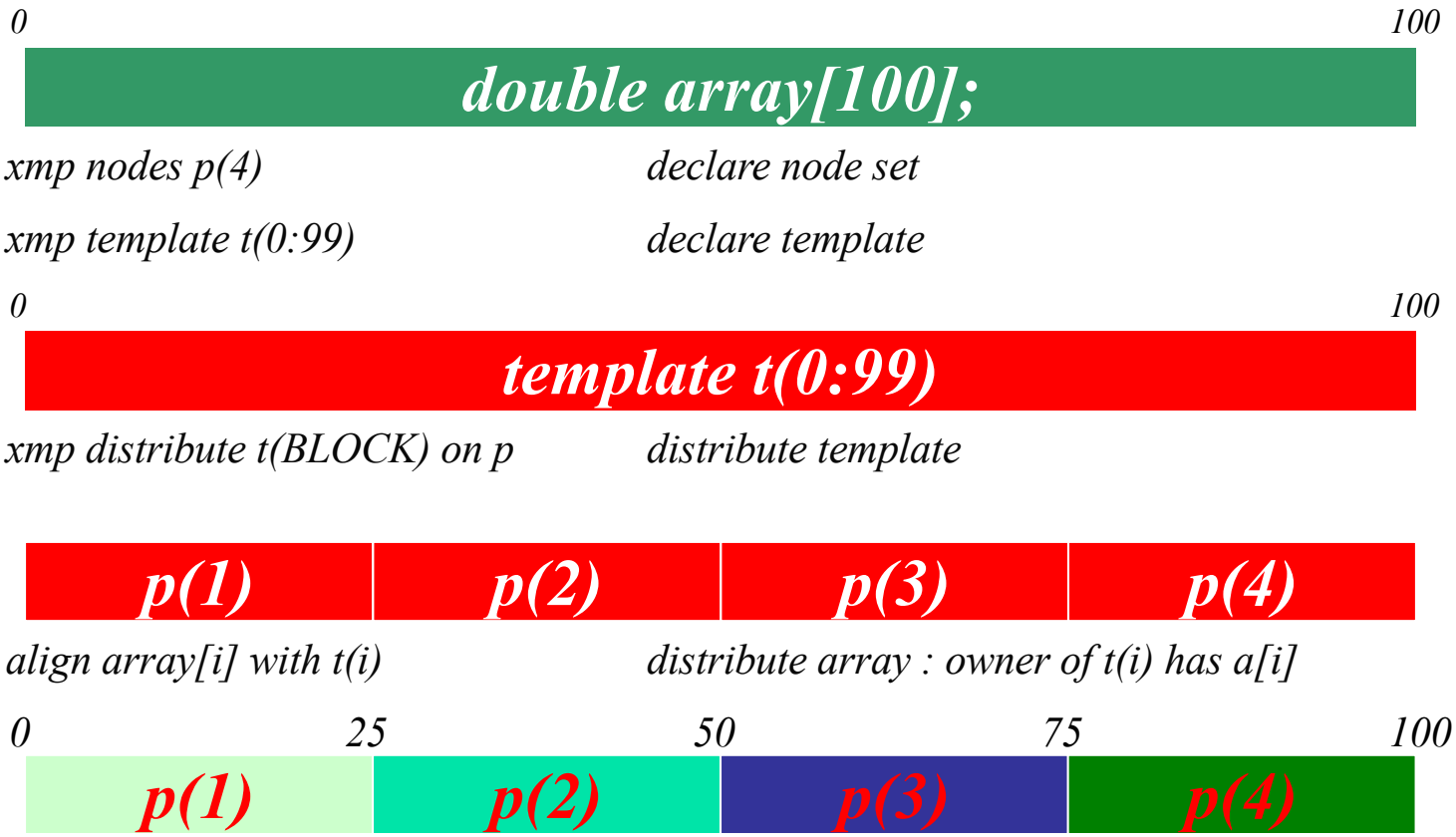
- 理研R-CCSと筑波大で開発中のXMP処理系
  - XMP/C
  - XMP/Fortran
- オープンソース
- トランスレータ + ランタイム(MPIベース)
- OpenACC、XcalableACC対応



Data Distibution Using Template

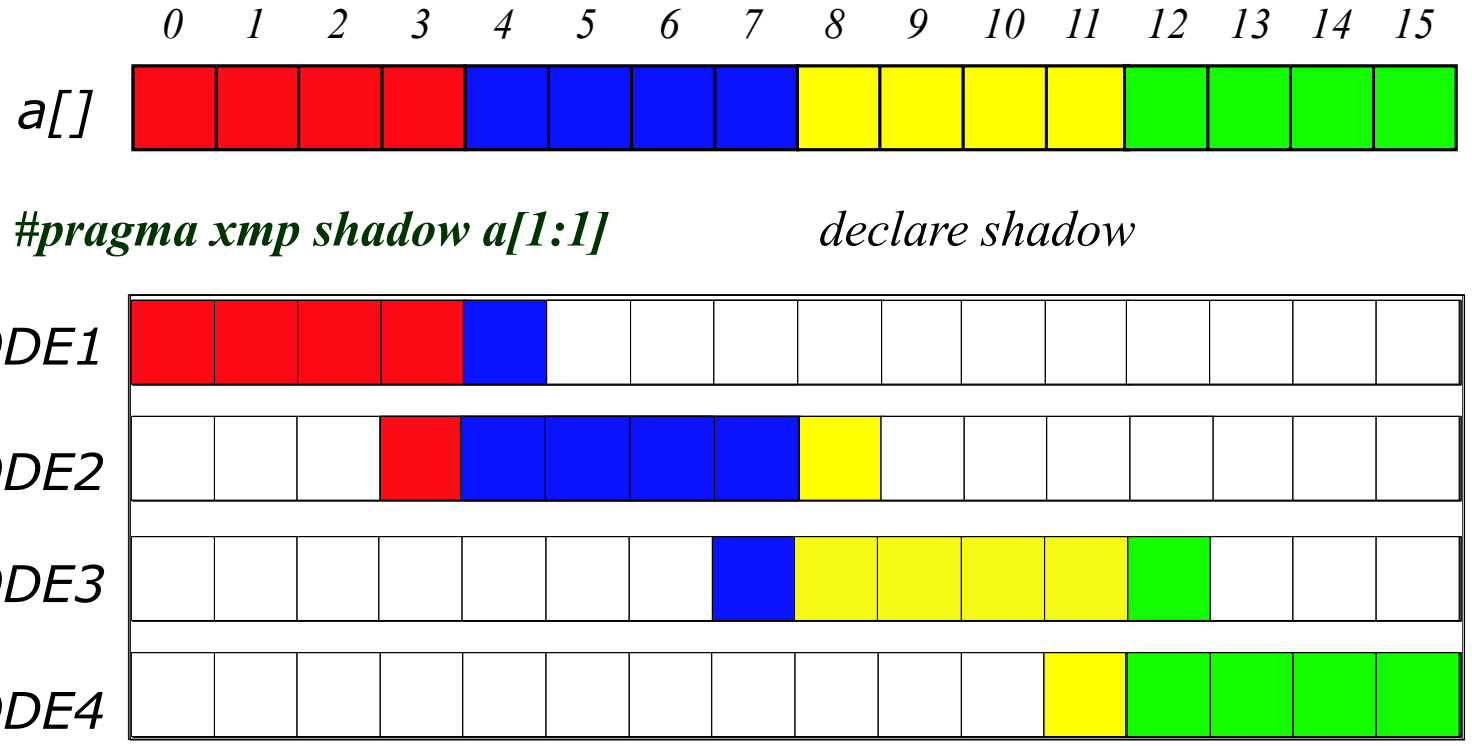
- Template
  - virtual array representing data(index) space
  - array distribution, work-sharing must be done using template

Example)



Data Synchronization of Array(shadow)

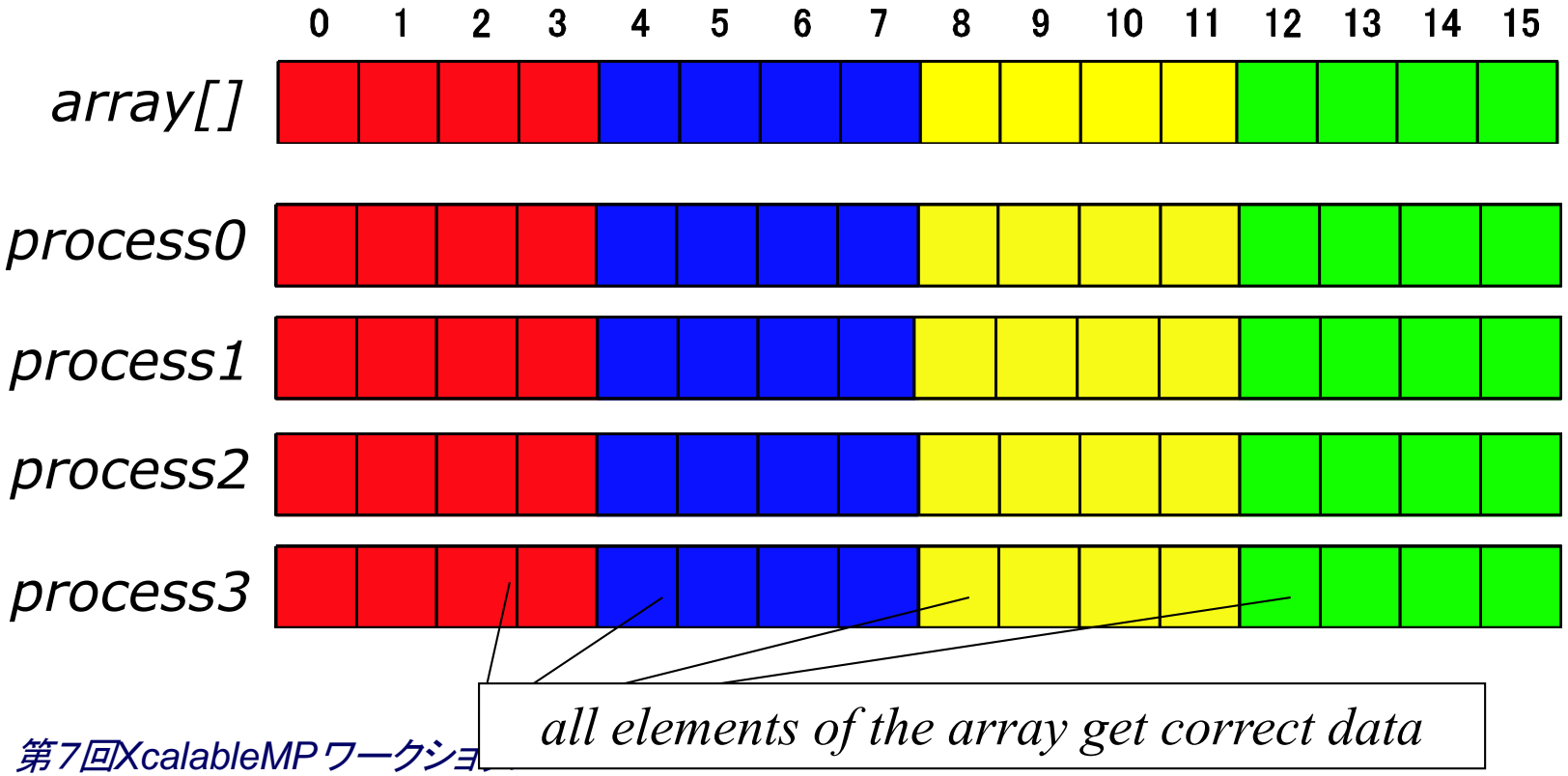
- **Shadow Region**
  - in XMP, memory access is always local
  - duplicated overlapped data distributed onto other nodes
  - data synchronization: **reflect directive**



Data Synchronization of Array(gather)

- *gather array data (collect entire elements)*
- *#pragma xmp gather(var=list)*

#pragma xmp gather(var=array)

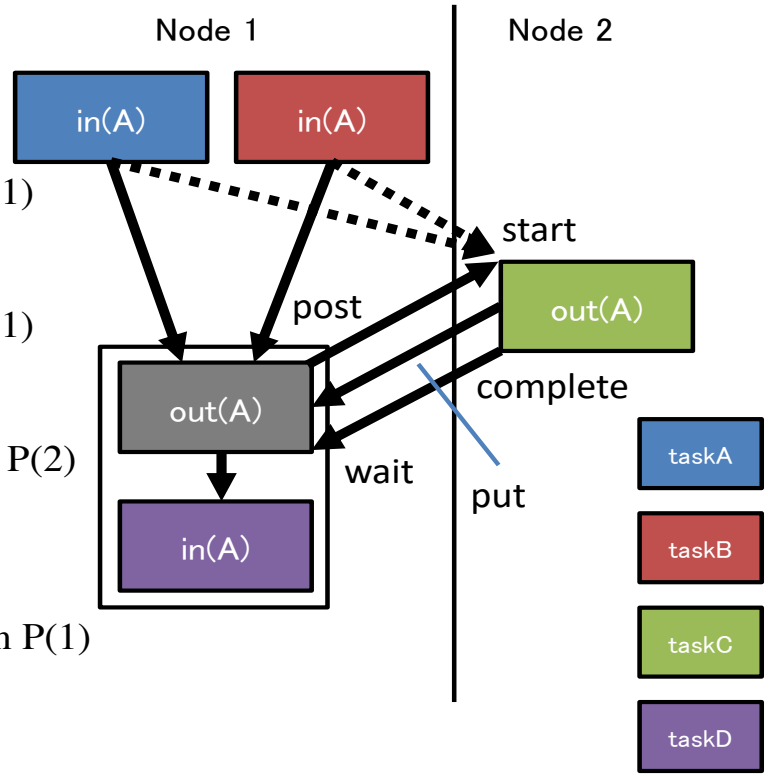


Put operation in tasklet

- put_ready clause: indicates that the specified data may be written by the associated PUT operation
  - This clause has the dependence-type out for the specified data on a node since its values are overwritten by the remote node.
- put clause: indicates that the PUT operation may be performed in the associated structured block.
  - At the beginning of the block, the task waits to receive the post notification with the tag by the put_ready clause to indicates that the data is exposed in the target node for the PUT operations.

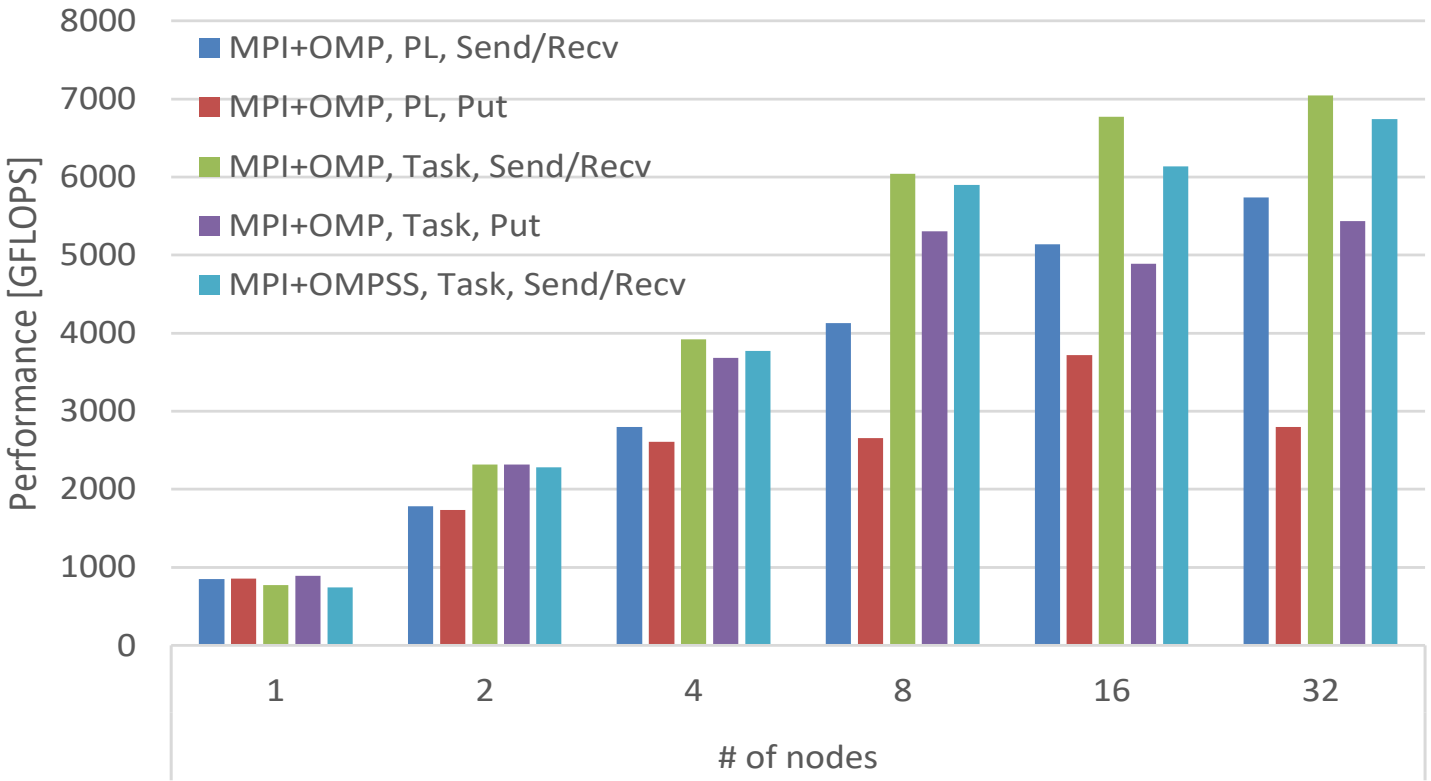
- When output dependencies for the data are satisfied before executing the block, the clause exposes the data for the PUT operation from the specified set of nodes by sending the post notifications to these nodes, starting the PUT operations eventually in remote nodes. Then, it waits until remote operations are done. When the task receives the completion notification of the PUT operation, the block is immediately scheduled.
- When the post notification is received, the task is scheduled to execute the calculation and PUT operation in the block. When the execution of the block is finished, the data written by the PUT operation is flushed and the completion notification is sent to the node matched by the tag.

```
#pragma xmp nodes P(2)
   int A:[*], B, C, D, tag;

#pragma xmp tasklet in(A) out(B) on P(1)
   B = A;        /* taskA */

#pragma xmp tasklet in(A) out(C) on P(1)
   C = A;        /* taskB */

#pragma xmp tasklet out(A) put(tag) on P(2)
   A:[1] = 1;    /* taskC */

#pragma xmp tasklet in(A) out(D) ¥
                 put_ready(A, P(1), tag) on P(1)
   D = A;        /* taskD */
```



Block-Cholesky Benchmark: Results on OFP

- Comparison with “Parallel Loop” (PL) and Task-based
- “Put” and “Send/Recv”
- (OMPSS)



⇨ taskletの詳細については村井さんのトークで

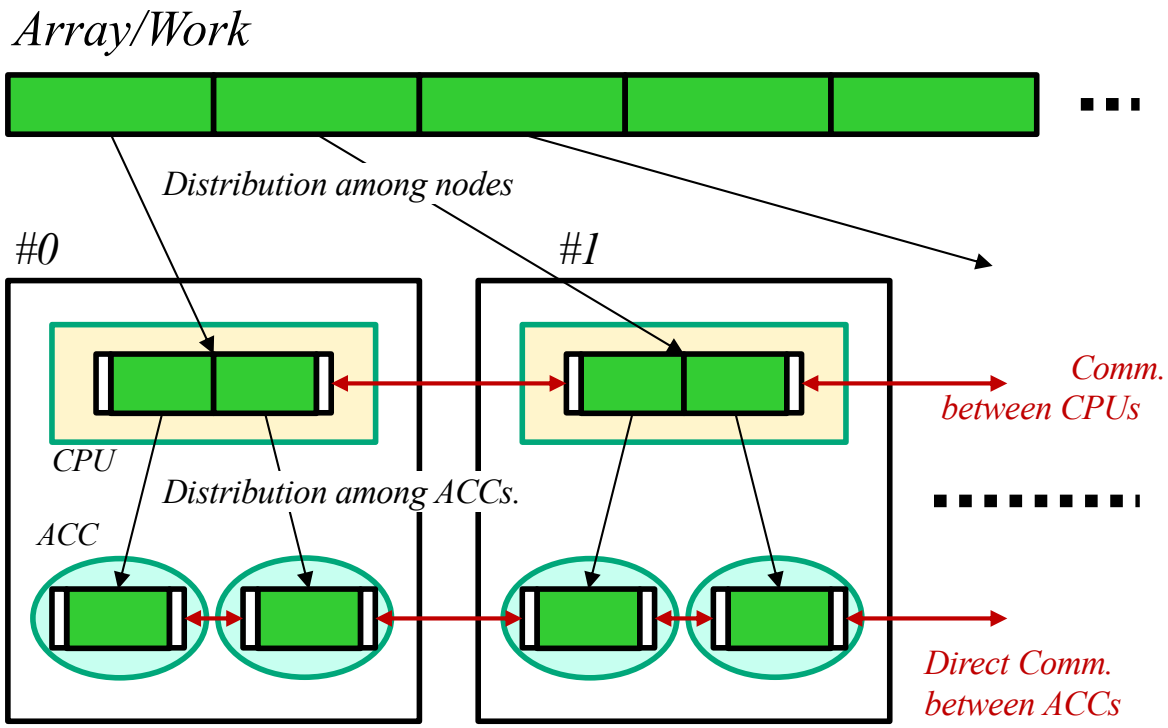
XcalableMPのアクセラレータ対応 (GPU & FPGA plan)

XcalableACC (XACC)の基本的な考え方

XcalableACC = XcalableMP + OpenACC

- XMPプログラム中にOpenACCのdirective記述を許す
- XMPが提供する分散メモリ並列処理 (PGAS)に従い、各ノードに分散された配列イメージを対象に、OpenACCのデータ移動・演算オフローディングを行う。
- OpenACC + MPI で記述する必要があった並列演算加速（GPUコンピューティング）をすっきりした形で記述でき、大規模並列化にも適している。
- XMP directiveと OpenACC directive は基本的に直交関係。

Processing model of XACC




```
#pragma acc device d = nvidia(0:3)

#pragma xmp reflect_init (a) device

#pragma xmp loop (i) on t(i)
for (int i = 0; i < 100; i++){
  #pragma acc kernels loop on_device(d)
  for (int j = 0; j < 100; j++){
    a[i][j] = ...
  }
}

#pragma xmp reflect_do (a)
```

Example of XcalableACC program

```
double u[XSIZE][YSIZE], uu[XSIZE][YSIZE];




…

{
  for(k=0; k<MAX_ITER; k++){


   for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        uu[x][y] = u[x][y];




    for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        u[x][y] = (uu[x-1][y]+uu[x+1][y]+
                   uu[x][y-1]+uu[x][y+1])/4.0;
  } // end k
```

2-D Laplace Eq.

Example of XcalableACC program

```
double u[XSIZE][YSIZE], uu[XSIZE][YSIZE];
#pragma xmp nodes p(x, y)
#pragma xmp template t(0:YSIZE-1, 0:XSIZE-1)
#pragma xmp distribute t(block, block) onto p
#pragma xmp align [j][i] with t(i,j) :: u, uu
#pragma xmp shadow uu[1:1][1:1]
…


  for(k=0; k<MAX_ITER; k++){
#pragma xmp loop (y,x) on t(y,x)

   for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        uu[x][y] = u[x][y];

#pragma xmp reflect (uu)

#pragma xmp loop (y,x) on t(y,x)

    for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        u[x][y] = (uu[x-1][y]+uu[x+1][y]+
                   uu[x][y-1]+uu[x][y+1])/4.0;
  } // end k
```

2-D Laplace Eq.

array distribution and "sleeve" declaration

exchange sleeves on array "uu"

Example of XcalableACC program

```
double u[XSIZE][YSIZE], uu[XSIZE][YSIZE];
#pragma xmp nodes p(x, y)
#pragma xmp template t(0:YSIZE-1, 0:XSIZE-1)
#pragma xmp distribute t(block, block) onto p
#pragma xmp align [j][i] with t(i,j) :: u, uu
#pragma xmp shadow uu[1:1][1:1]
…
#pragma acc data copy(u) copyin(uu)
{
  for(k=0; k<MAX_ITER; k++){
#pragma xmp loop (y,x) on t(y,x)
#pragma acc parallel loop collapse(2)
    for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        uu[x][y] = u[x][y];

#pragma xmp reflect (uu) acc

#pragma xmp loop (y,x) on t(y,x)
#pragma acc parallel loop collapse(2)
    for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        u[x][y] = (uu[x-1][y]+uu[x+1][y]+
                   uu[x][y-1]+uu[x][y+1])/4.0;
  } // end k
} // end data
```

2-D Laplace Eq.

array distribution and "sleeve" declaration

copy partial (distributed) array to device memory

distributed array by XMP is processed according to OpenACC directive

exchange sleeves on array "uu"

"acc" clause indicates to target the array on device memory

Example of XcalableACC program

```
double u[XSIZE][YSIZE], uu[XSIZE][YSIZE];

…
#pragma acc data copy(u) copyin(uu)
{
  for(k=0; k<MAX_ITER; k++){

#pragma acc parallel loop collapse(2)
   for(x=1; x<XSIZE-1; x++)
     for(y=1; y<YSIZE-1; y++)
       uu[x][y] = u[x][y];



#pragma acc parallel loop collapse(2)
    for(x=1; x<XSIZE-1; x++)
      for(y=1; y<YSIZE-1; y++)
        u[x][y] = (uu[x-1][y]+uu[x+1][y]+
                   uu[x][y-1]+uu[x][y+1])/4.0;
  } // end k
} // end data
```

2-D Laplace Eq.

copy partial (distributed) array to device memory

distributed array by XMP is processed according to OpenACC directive

XACCにおける通信の選択

- 現時点ではGPU（NVIDIA GPU）を演算加速器として扱う。
- 一般的な並列通信機構による
  - 通信のためにMPIコードが生成される。
  - XMPとほぼ同じ配列分散を行い、デバイスメモリ上のデータに対する通信の最適化はGDR (GPU Direct RMA)等に対応したMPIによって行う。
- 特殊な並列通信機構向け実装
  - CRESTで開発したTCA (Tightly Coupled Accelerator)コンセプトに基づくGPU間直接通信支援機構であるPEACH2を対象とした実装
  - 通信最適化はPEACH2に依存

XACC on FPGA

- FPGA for HPC 研究が活性化しつつある
- FPGAは直球勝負でGPUと競ってもFLOPS的な勝ち目はない
  - GPUとFPGAを補完的に使えるのでは？
  - AiS (Accelerator in Swtich): FPGAを accelerator と interconnection の両方に使う
    ⇒ FPGAの自律的動作によりGPU memoryのDMAとノード間通信までを行うことで従来の方法より低レイテンシ化
- ORNL (by J. Vetter‘s group)でOpenACCからOpenCL for FPGAのトランスレータ開発が進んでいる
  - GPUとFPGAの両方のカーネルをOpenACCで記述し、適材適所的にデバイス割り当てをして連携させる
  - XACCまで持っていけばFPGAの高速通信チャネル（最大100Gbps x 3ぐらい）を通信レイヤに使うことが可能に
  - 共同研究実施に向け調整中

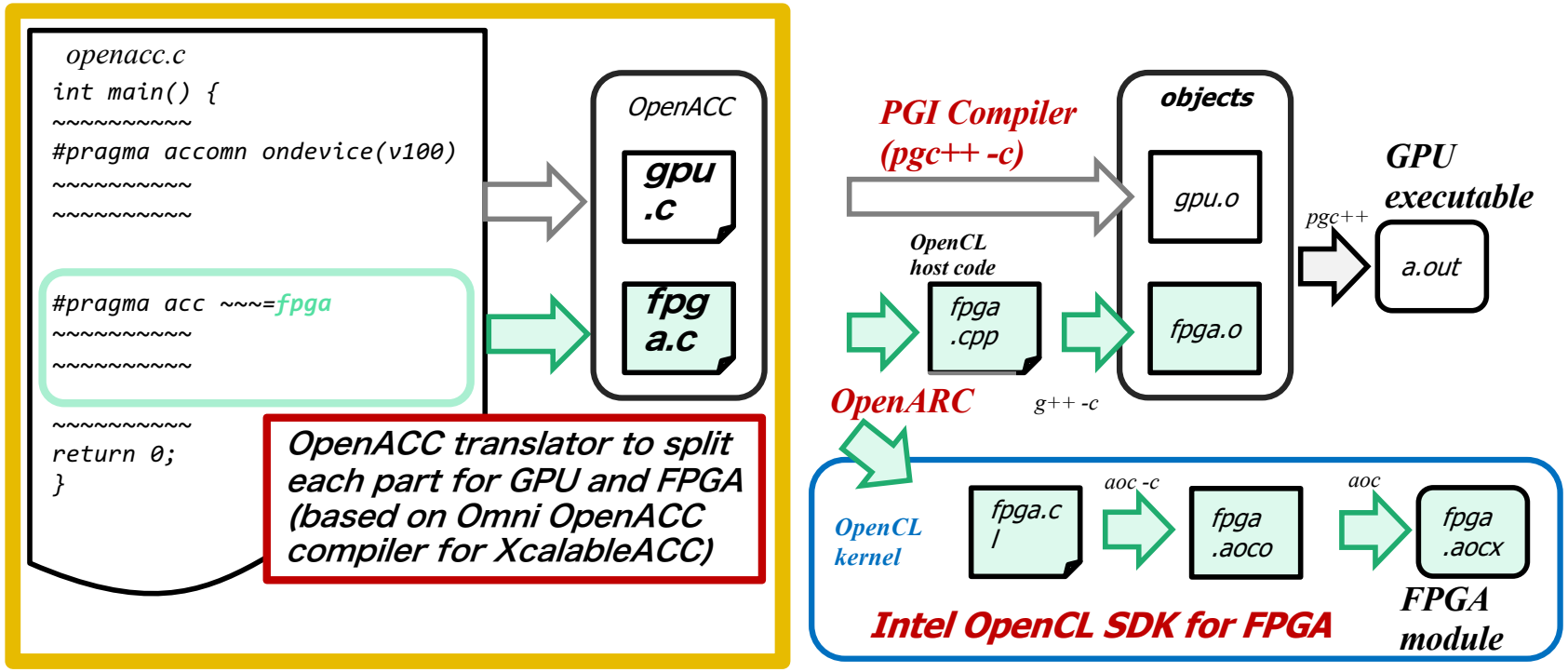
Omni-OpenACCの拡張（筑波大＋理研R-CCS）

- Omni-OpenACC: XcalableACC実装において開発された
- *multi-device support* のための新機能を実装

  “#pragma accomn ondevice(...)”

- Device option: V100, Stratix10, ... (GPU and FPGA currently)
- 2種類のバックエンドコンパイラを使用
  - PGI OpenACC compiler for GPU
  - ORNL’s OpenARC research compiler for FPGA (under collaboration with ORNL)
- 文科省「次世代領域研究開発」プログラムにおける「次世代演算通信融合型スーパーコンピュータの開発」プロジェクト（2017-2021, 代表：朴泰祐）

Multi-device OpenACC compilation



MM & MV on GPU and SpMV on FPGA

```
#pragma acc data copyout(c[:N*N]) copyin(a[:N*N], b[:N*N])
#pragma acc kernels
 {
#pragma acc loop gang(N/32) vector(32) independent
   for (i = 0; i < N; ++i)
   {
#pragma acc loop gang(N/32) vector(32) independent
     for (j = 0; j < N; ++j)
     {
       float sum = 0.0;
#pragma acc loop reduction(+:sum)
       for (k = 0; k < N; ++k)
         sum += a[i * N + k] * b[k * N + j];
       c[i * N + j] = sum;
}}}
```

MM on GPU

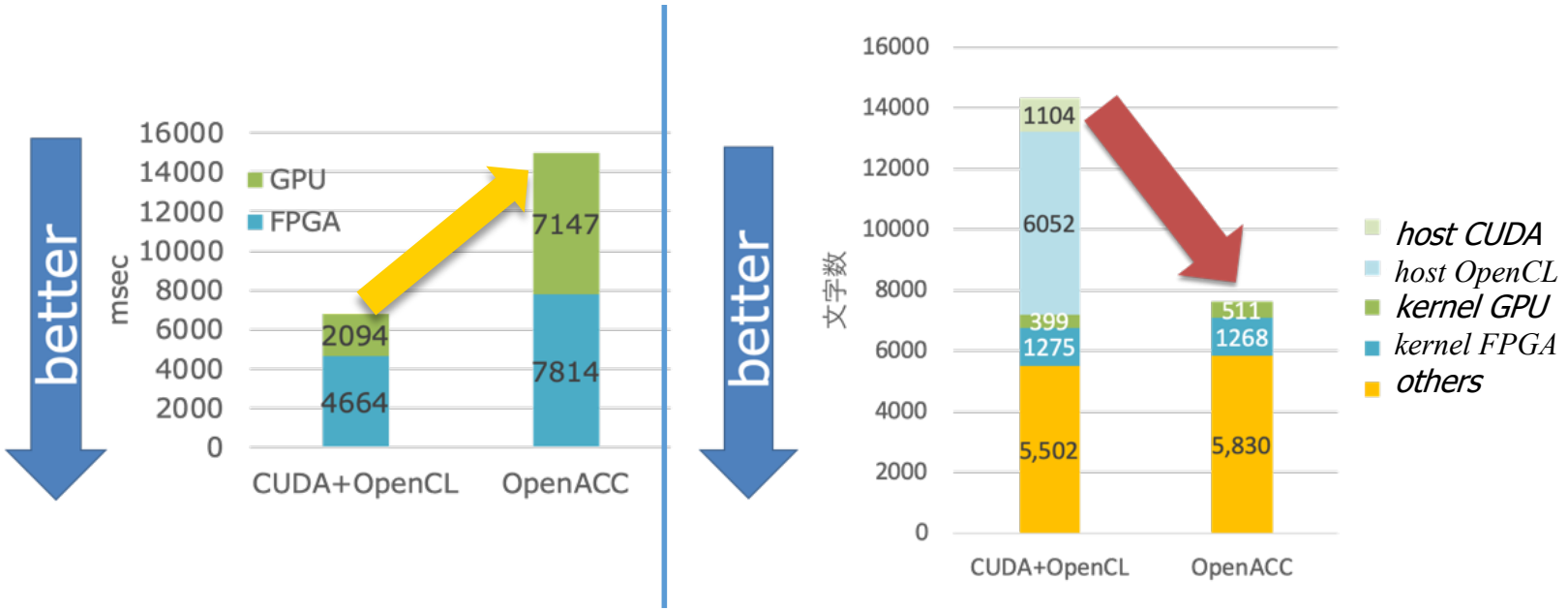
```
#pragma acc data copyin(VAL[0:VAL_SIZE], COL_IND[0:VAL_SIZE],
ROW_PTR[0:N+1], B[0:N], N, K, VAL_SIZE) copyout(X_result[0:N])
#pragma acc parallel num_gangs(1) num_workers(1) vector_length(1) {
~                                   // 初期化計算
#pragma acc loop
   for(int i = 0; i < K; ++i){
      temp_pap = 0.0f;
      int j = 0, l = ROW_PTR_local[0];
#pragma acc loop reduction(+:temp_pap, temp_sum)
      for(int m = 0; m < VAL_SIZE-ROW_PTR_local[0]; ++m){
         temp_sum += p[COL_IND_local[l]] * VAL_local[l];
         ++l;
         if(l == ROW_PTR_local[j + 1]) {
            y[j] = temp_sum;
            temp_pap += p[j] * temp_sum;
            temp_sum = 0.0f;
            ++j;
            l = ROW_PTR_local[j];  } }
...
```

SpMV on FPGA

Making matrix by MM on GPU, MV on GPU then transferred to FPGA for SpMV on FPGA
(Sp matrix data is kept on FPGA, only vector is transferred)

(by R. Tsunashima, U. Tsukuba)

CUDA/OpenCL vs Multi-Device OpenACC



Execution time

#chars in code

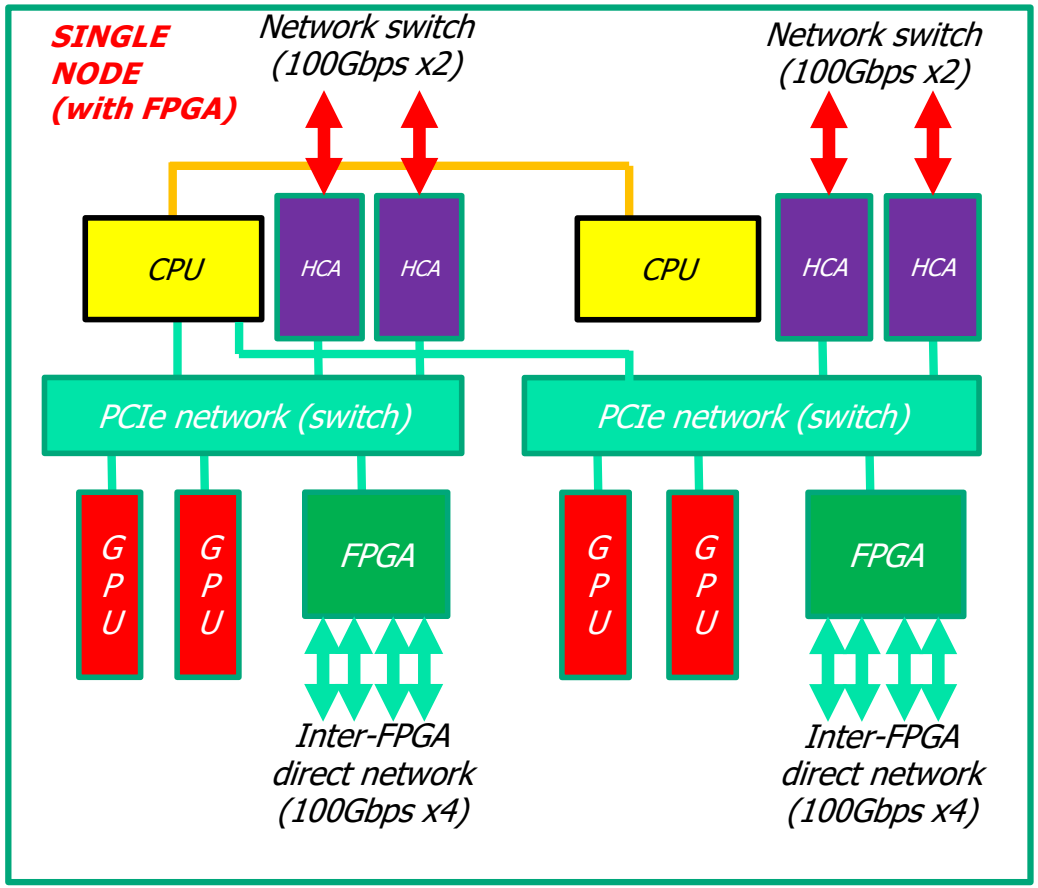
(by R. Tsunashima, U. Tsukuba)

Omni-OpenACCへのFPGA対応組み込みを行い、XcalableACCに展開
⇨ FPGA間通信にFPGAが持つ光インターコネクトを直接利用

Cygnus – multi-hybrid accelerated supercomputer at U. Tsukuba

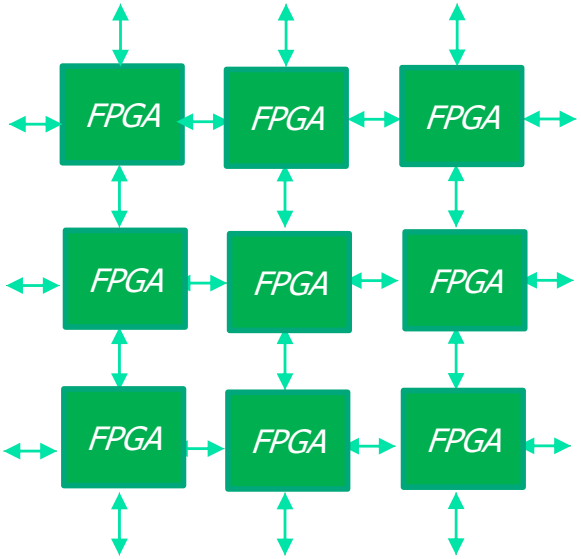
Single node configuration (Albireo)

- *Each node is equipped with both IB EDR and FPGA-direct network*
- *Some nodes are equipped with both FPGAs and GPUs, and other nodes are with GPUs only*

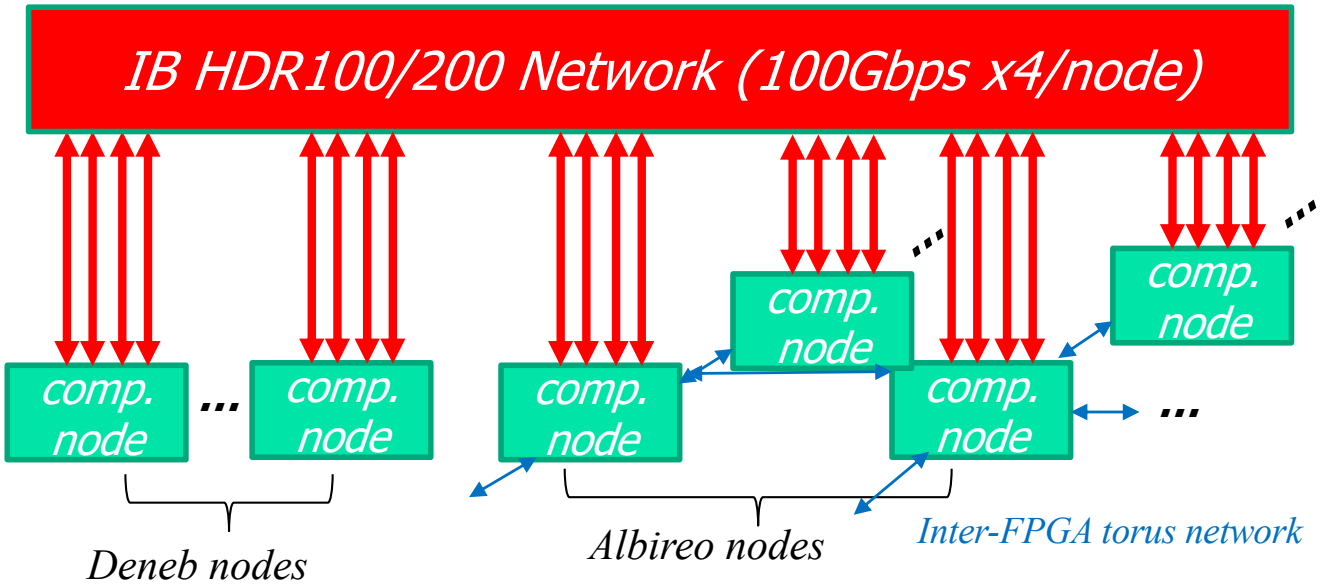


Two types of interconnection network

Inter-FPGA direct network (only for Albireo nodes)



InfiniBand HDR100/200 network for parallel processing communication and shared file system access from all nodes



64 of FPGAs on Albireo nodes (2 FPGAS/node) are connected by 8x8 2D torus network without switch

For all computation nodes (Albireo and Deneb) are connected by full-bisection Fat Tree network with 4 channels of InfiniBand HDR100 (combined to HDR200 switch) for parallel processing communication such as MPI, and also used to access to Lustre shared file system.

おわりに

- XcalableMPの規格は2.0へ、特にtask処理のノード内・ノード間並列実行への展開に重点
- XcalableACCをXcalableMPの正式なブランチへ
  ⇒ GPU, FPGAへの対応
- 従来の OpenACC + MPI のコードをより簡単に記述可能に
- FPGAの特性を活かし、GPUとの協働による高性能処理へ