University of Tsukuba | Center for Computational Sciences

XcalableMP: Directive-Based Language eXtension for Scalable and Performance-Aware Parallel Programming

Overview of XcalableMP

- XcalableMP (XMP) is a PGAS language for distributed memory system
- XMP extends C99 and Fortran 95 with directives, Coarray syntax, and user APIs
- XMP supports typical parallelization under global-view programming model
  - XMP global-view model enables parallelizing the original sequential code using minimal modification with simple directives, like OpenMP
  - The directives can describe data mapping, work mapping, and inter-node communication
  - Many ideas on global-view programming are inherited from High Performance Fortran
- XMP includes Coarray Fortran syntax as local-view programming model
- Coarray syntax in XMP describes one-sided communication
- The important design principle of XMP is performance-awareness
  - All actions of communication and synchronization are taken by directives, different from automatic parallelizing compilers
  - The user should be aware of what happens by XMP directives in the execution model on the distributed memory architecture

Laplace Solver

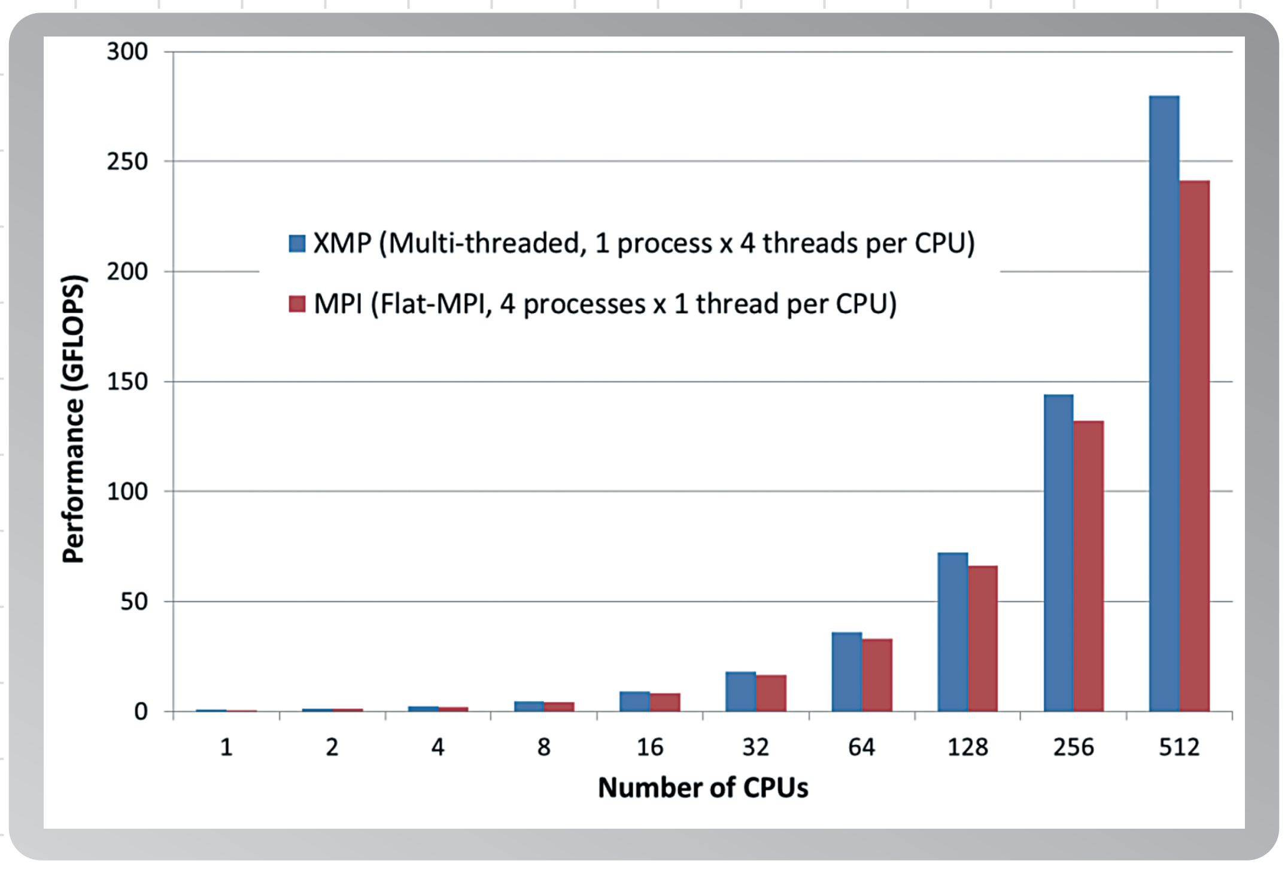
```
#define N (10*1024)
#pragma xmp nodes p(2,2)
#pragma xmp template t(0:N-1, 0:N-1)
#pragma xmp distribute t(BLOCK, BLOCK) onto p
#pragma xmp align [i][j] with t(j, i) :: u, uu
#pragma xmp shadow u[1][1]
...
for(k=0; k<TIMES; k++){
#pragma xmp loop (x, y) on t(x, y) threads
   for(y=1; y<N-1; y++)
      for(x=1; x<N-1; x++)
         u[y][x] = uu[y][x];
#pragma xmp reflect (u)
#pragma xmp loop (x, y) on t(x, y) threads
   for(y=1; y<N-1; y++)
      for(x=1; x<N-1; x++)
         uu[y][x] = (u[y-1][x] + u[y+1][x] + u[y][x-1] + u[y][x+1]) / 4.0;
}
```

Definition of data mapping

Definition of shadow area and its width

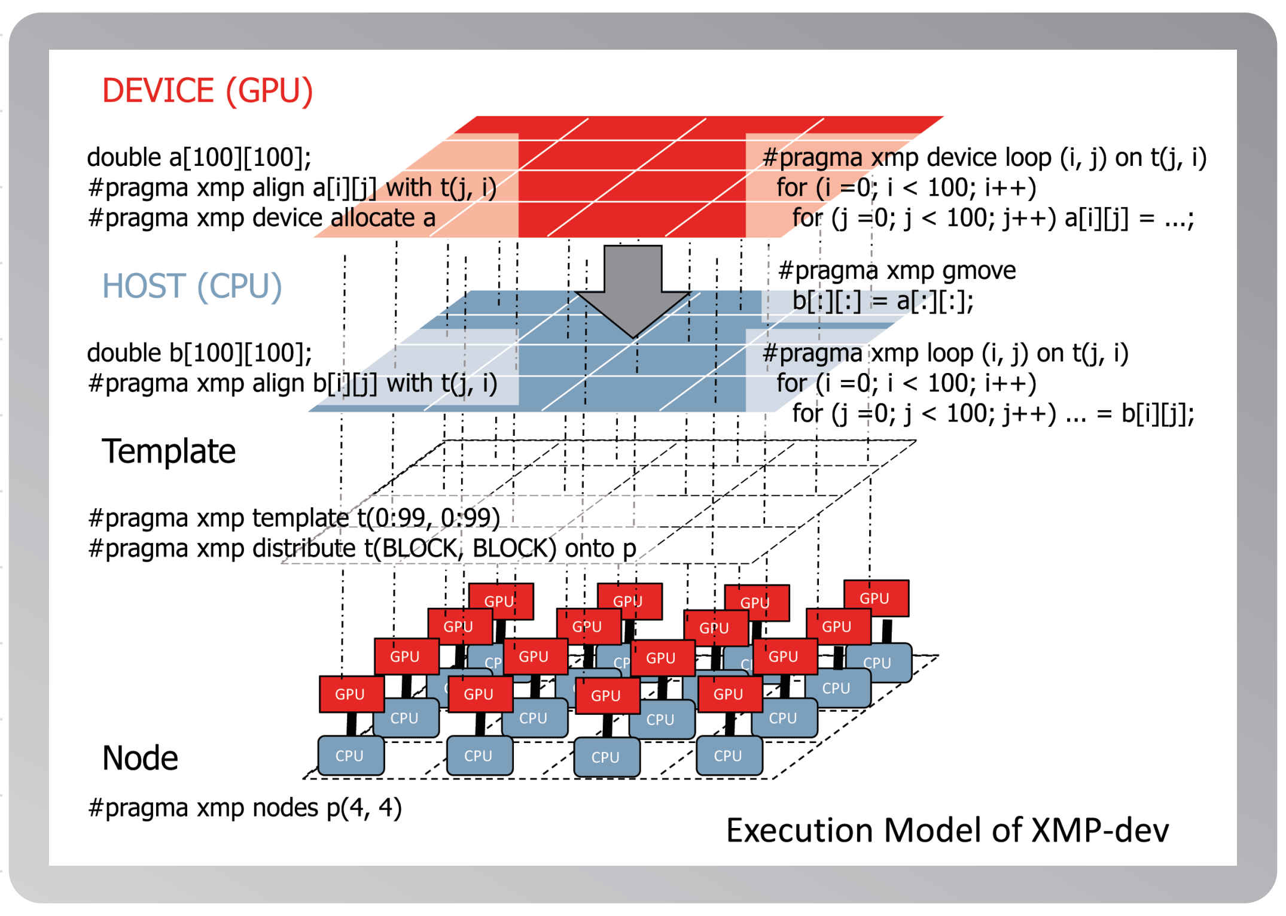
Parallelization for loop statement

Synchronization data only on shadow area



XcalableMP Acceleration Device Extension (XMP-dev)

- XMP-dev is an extension of XMP for acceleration devices such as GPUs
- XMP-dev supports clusters equipped with acceleration devices
- XMP-dev provides directives to describe typical processes of data parallelism for accelerators such as data allocation, transfer and task offloading onto devices
- Data distribution and inter-node communication for cluster computing can be described in XMP-dev



```
double a[100][100];
#pragma xmp align a[i][j] with t(j, i)
#pragma xmp device allocate a
```

```
#pragma xmp device loop (i, j) on t(j, i)
 for (i =0; i < 100; i++)
   for (j =0; j < 100; j++) a[i][j] = ...;
```

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

```
double b[100][100];
#pragma xmp align b[i][j] with t(j, i)
```

```
#pragma xmp loop (i, j) on t(j, i)
 for (i =0; i < 100; i++)
   for (j =0; j < 100; j++) ... = b[i][j];
```

```
#pragma xmp template t(0:99, 0:99)
#pragma xmp distribute t(BLOCK, BLOCK) onto p
```

```
#pragma xmp nodes p(4, 4)
```

Execution Model of XMP-dev

Sample Code

```
#pragma xmp nodes p(*)
#pragma xmp template t(0:N-1)
#pragma xmp distribute t(block) onto p
#pragma xmp align [i] with t(i) :: a, hb, db
#pragma xmp shadow a[*]
#pragma xmp device replicate (a)
#pragma xmp device allocate (db)
...
#pragma xmp loop on t(i)
for(i=0; i<N; i++)    a[i] = i + 1;
#pragma xmp reflect (a)

#pragma xmp device replicate_sync in (a)

#pragma xmp device loop on t(i)
for (i=0; i<N; i++){
   db[i] = 0;
   for(j=0; j<N; j++)    db[i] += a[j];
}

#pragma xmp gmove
   hb[:] = db[:];
```

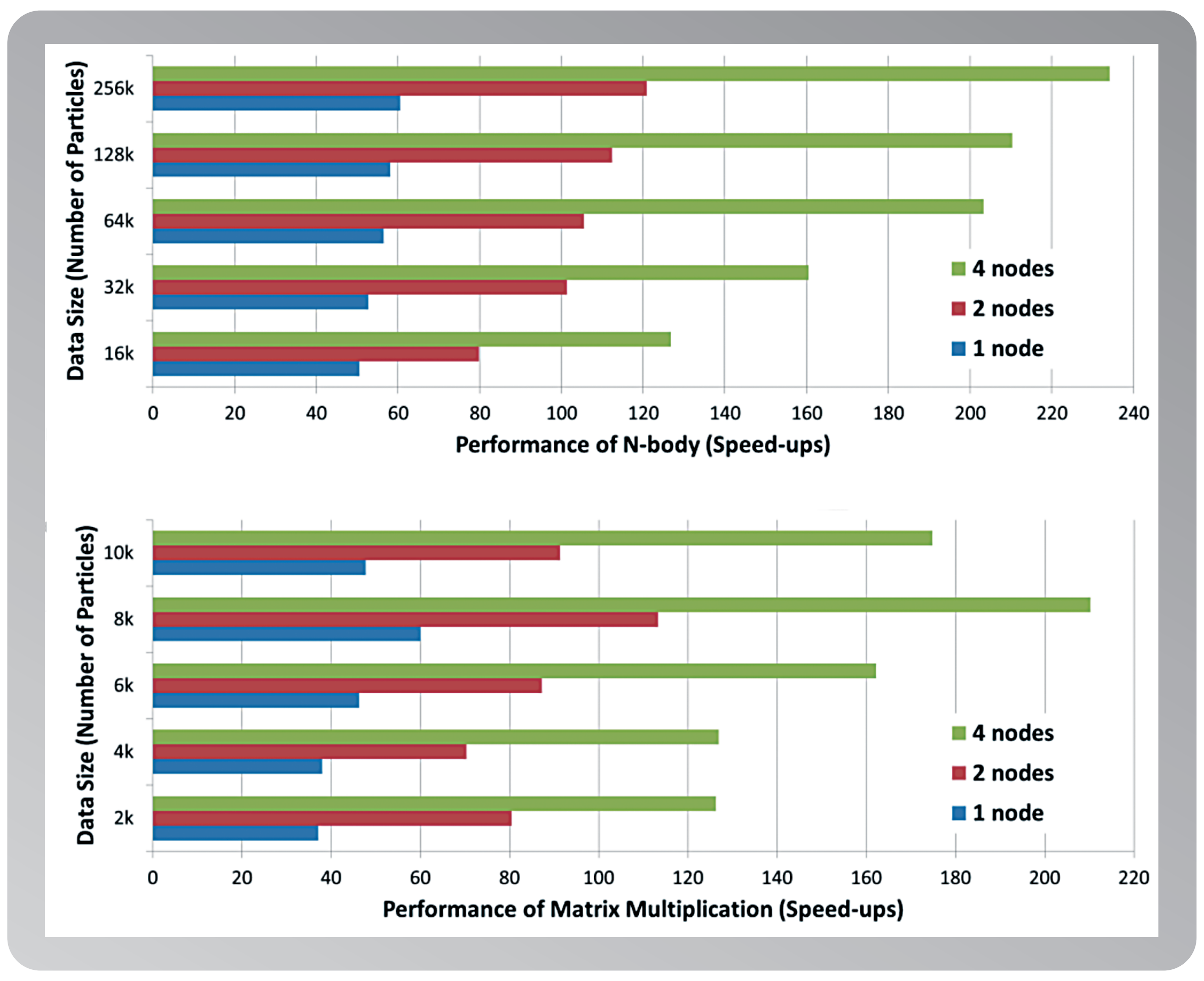
Defintion of data mapping

Execution on HOST

Data copy Host to DEVICE

Execution on DEVICE

Data copy DEVICE to HOST


Data Size (Number of Particles)
256k
128k
64k
32k
16k
4 nodes
2 nodes
1 node
Performance of N-body (Speed-ups)
Data Size (Number of Particles)
10k
8k
6k
4k
2k
4 nodes
2 nodes
1 node
Performance of Matrix Multiplication (Speed-ups)