HPC Challenge Awards Class 2

HPC Challenge Benchmarks in XcalableMP

Jinpil Lee, Mitsuhisa Sato

University of Tsukuba, HPCS lab

What is XcalableMP?

- for distributed memory systems (PC clusters)
- directive-based language extension
  - based on C and Fortran 95
  - directives and other language extensions
- not new, but practical
- "performance-aware" parallel programming
  - no automatic parallelization
  - all actions (comm. and work mapping) are taken by directive given by programmers

The specification has been designed by XcalableMP Specification Working Group which consists of members from academia, research labs and industries.

visit **www.xcalablemp.org** for more information

Language Overview

- execution model
  - SPMD(Single Program Multiple Data): MPI-like
  - starting with single thread per process
- explicit parallelism
  - no automatic communication, virtual shared memory
- Two different programming models in one language
  - global view model
    - incremental parallelization with directives (OpenMP-like)
    - many concepts from HPF (High Performance Fortran)
  - local view model
    - supports PGAS features (co-array from Co-Array Fortran)

Global View Programming Model

- incremental parallelization with directives
  - adding directives to serial code
  - programmer describes data distribution, work mapping inter-node comm
    - supports typical techniques for data/task parallelization

```
double array[YMAX][XMAX];
#pragma xmp nodes p(*)                          // declare node group (communicator)
#pragma xmp template t(XMAX, YMAX)              // declare template (range of array index)
#pragma xmp distribute t(*, BLOCK)              // distribute template

main() {
  int i, j, res = 0;
#pragma xmp loop on t(*,i) reduction (+:res)    // work sharing and reduction
  for (i = 0; i < YMAX; i++)
    for (j = 0; j < XMAX; j++) {
      array[i][j] = func(i,j);
      res += array[i][j];
    }
}
```

Local View Programming Model

- PGAS model taken from Co-Array Fortran (extended to C)
- one-sided communication using language extension
- high interoperability with MPI

```
double array[YMAX/PROCS][XMAX];                        // data distribution (manual)
#pragma xmp coarray array                              // declare array as a co-array

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

 for (i = 0; i < YMAX/PROCS; i++)                      // loop parallalization (manual)
   for (j = 0; j < XMAX; j++) {
     array[i][j] = func(i,j);
     res_local += array[i][j];
   }

 for (i = 1; i <= PROCS; i++)
   res += res_local:[i];                               // access res_local on node(i)
}
```

Submission

- XMP/C: a prototype compiler is implemented
  - supports basic functions for data parallelism
- In this submission, we focus on programmability of XcalableMP
  - STREAM, RandomAccess, HPL, FFT are parallelized by XMP

T2K OpenSupercomputer – Tsukuba System (2to32nodes)

CPU	AMD Opteron Quad-core 8000series 2.3Ghz x 4sockets (16 cores)
MEM	32GB
NETWORK	InfiniBand (x4 rails)
MPI lib	MVAPICH2 - 1.2

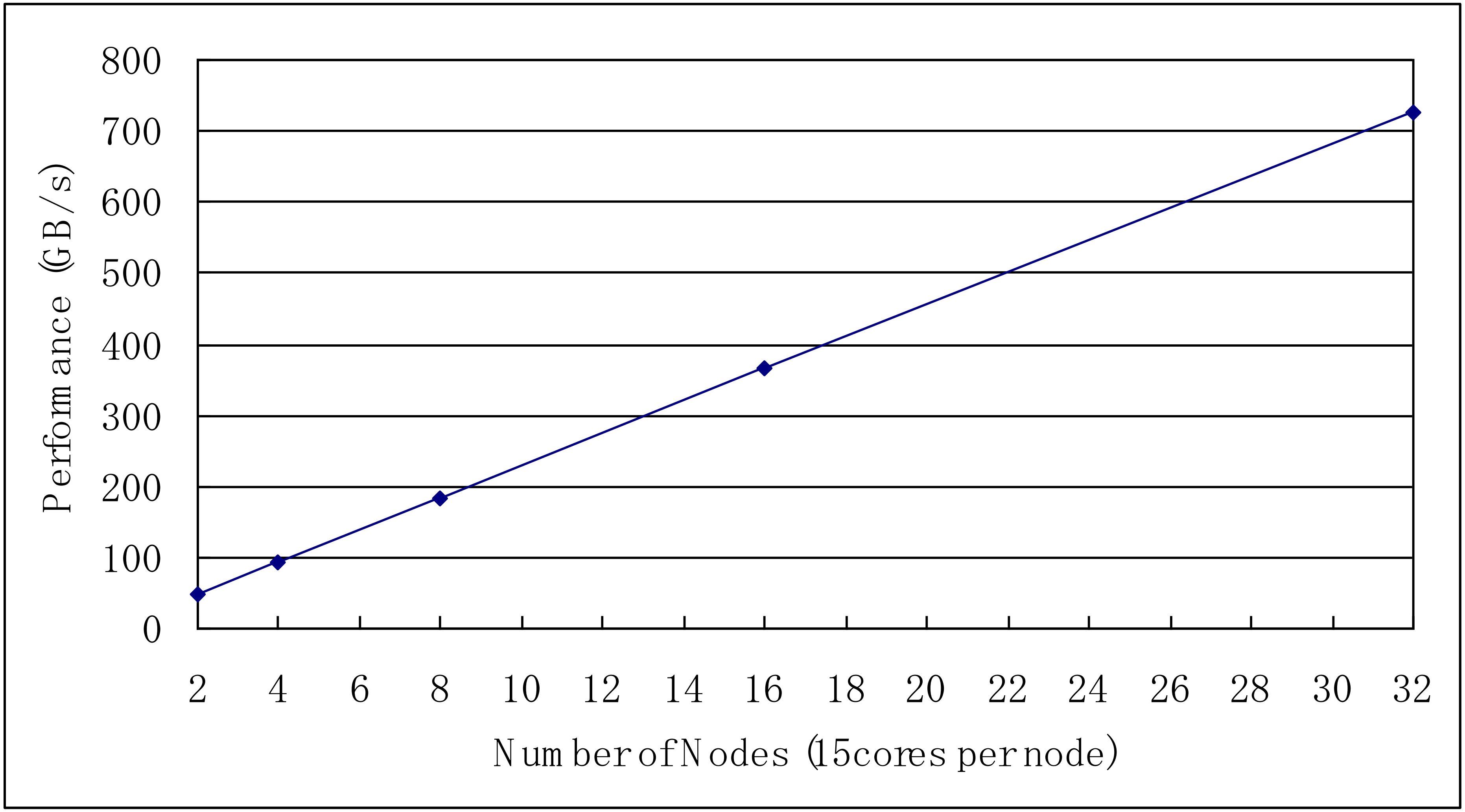
Benchmark1: STREAM

- global view programming with directives
  - very straightforward to parallelize by loop directive

```
double a[SIZE] , b[SIZE] , c[SIZE];
#pragma xmp nodes p(*)
#pragma xmp template t(0:SIZE−1)
#pragma xmp distribute t(block) onto p
#pragma xmp align [j] with t(j) :: a, b, c
. . .
# pragma xmp loop on t(j)
for (j = 0; j < SIZE; j++) a[j] = b[j] + scalar*c[j];
. . .
#pragma xmp reduction(+:triadGBs)
```

Performance of STREAM

- Lines Of Code: 98



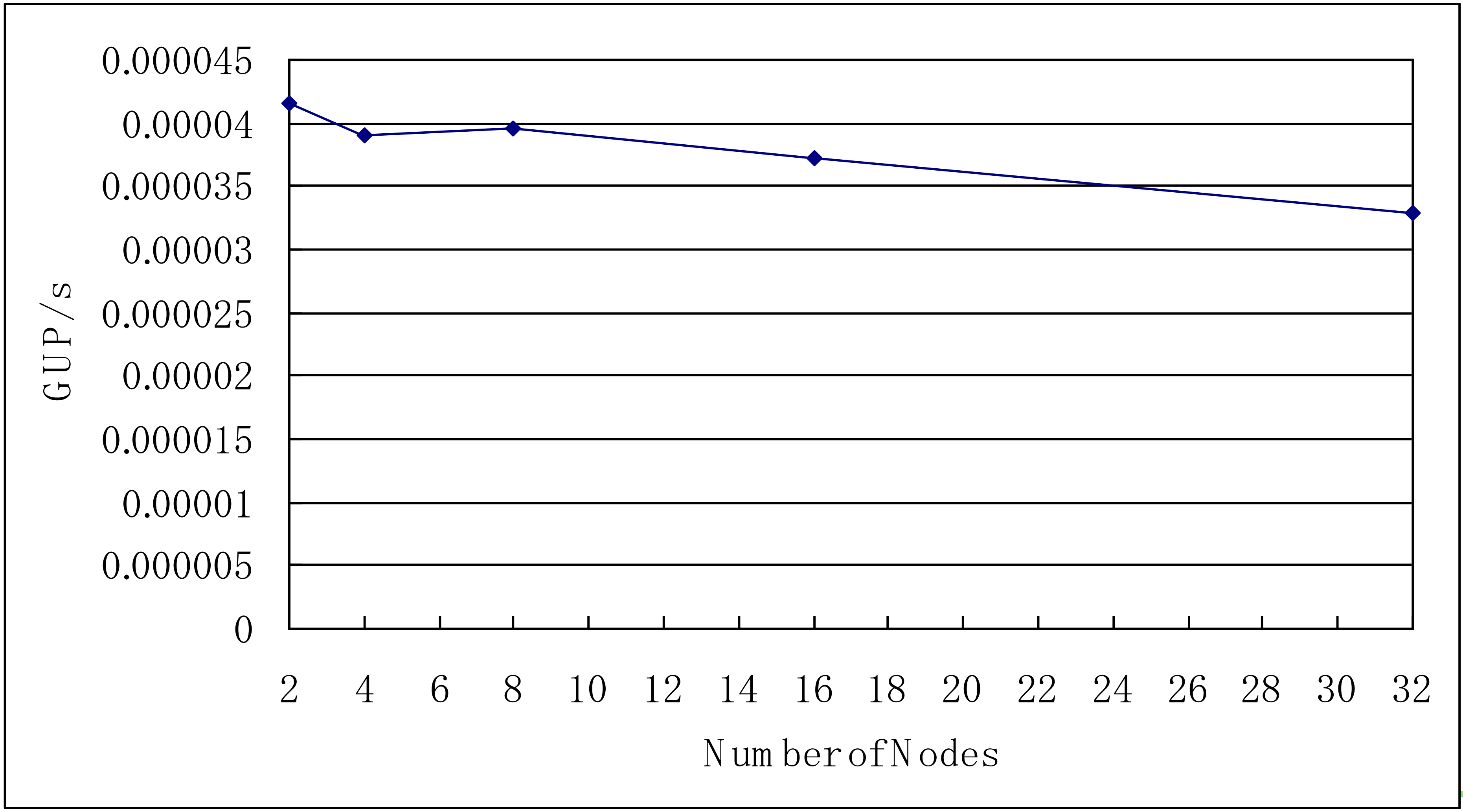
Benchmark2: RandomAccess

- local view programming with co-array

```
#define SIZE TABLE_SIZE/PROCS
u64Int Table[SIZE] ;
#pragma xmp nodes p(PROCS)
#pragma xmp coarray Table [PROCS]
. . .
for (i = 0; i < SIZE; i++) Table[i] = b + i ;
. . .
for (i = 0; i < NUPDATE; i++) {
  temp = (temp << 1) ^ ((s64Int)temp < 0 ? POLY : 0);
  Table[temp%SIZE]:[(temp%TABLE_SIZE)/SIZE] ^= temp;
}
#pragma xmp barrier
```

Performance of RandomAccess

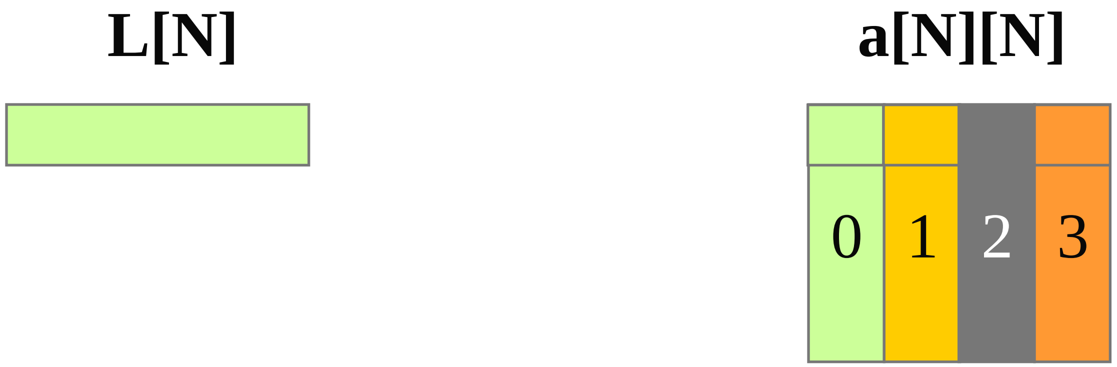
- Lines Of Code: 77
- complied into MPI2 one-sided functions



Gmove Directive

- Data transfer (copy) in global view
  - Very powerful directive to describe communication
  - This operation is "collective", that is, executed by all node, and translated into two-sided comm.
- Array section can be used in C

```
#pragma xmp nodes p(4)
#pragma xmp template t(N)
#pragma xmp distribute t(block) onto p
#pragma xmp align a[*][i] with t(i)
. . .
#pragma xmp gmove
   L[0:N-1] = a[0][0:N-1];   // broadcast
```



Benchmark3: HPL

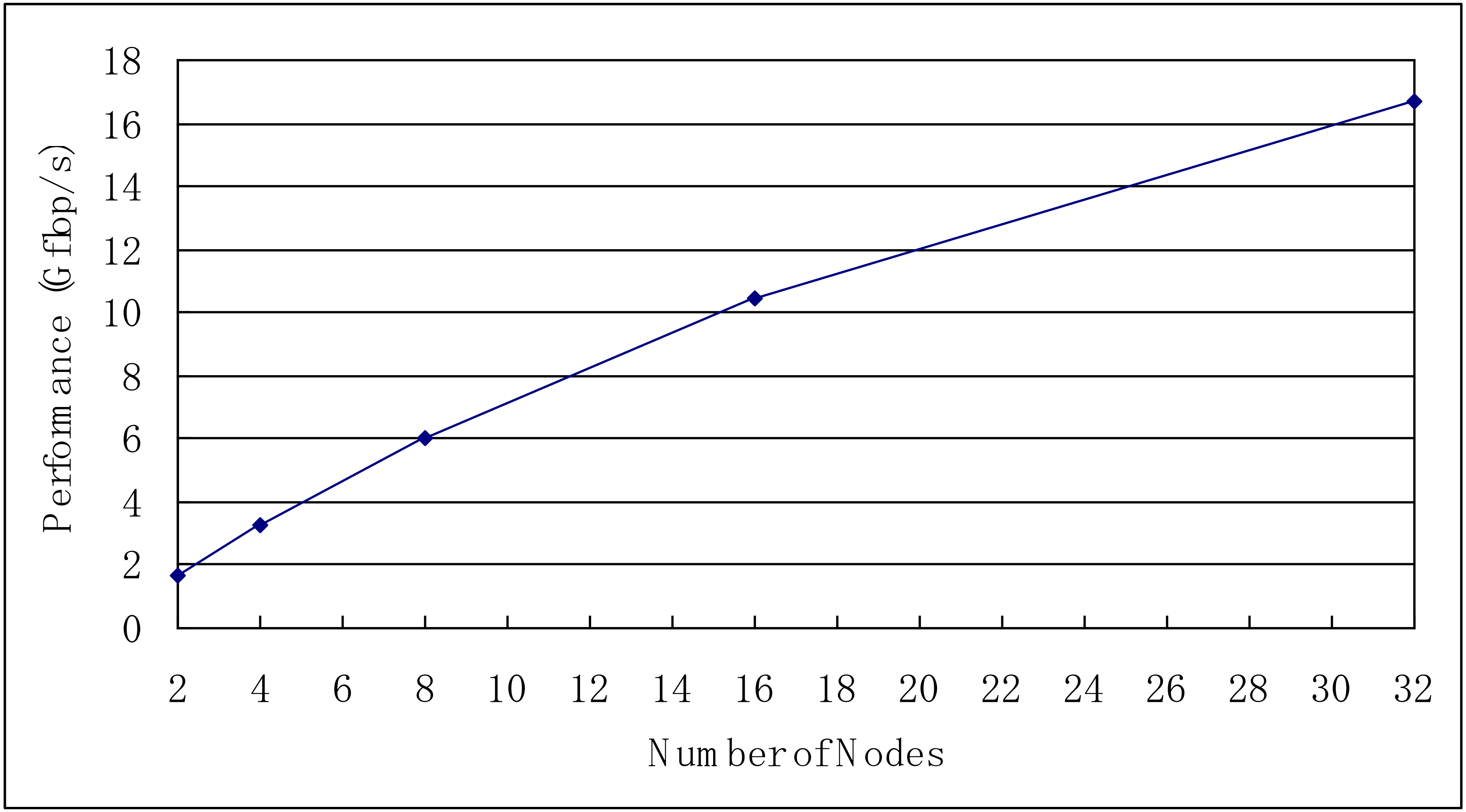
- parallelized in global view
- matrix/vectors are distributed in cyclic manner in one dimension.
- using **gmove** to exchange columns for pivot exchange

```
dgefa function:
#pragma xmp gmove
    pvt_v[k:n-1] = a[k:n-1][l];

    if (l != k) {
#pragma xmp gmove
      a[k:n-1][l] = a[k:n-1][k];
#pragma xmp gmove
      a[k:n-1][k] = pvt_v[k:n-1];
    }
```

Performance of HPL

- Lines Of Code: 243



Benchmark4: FFT

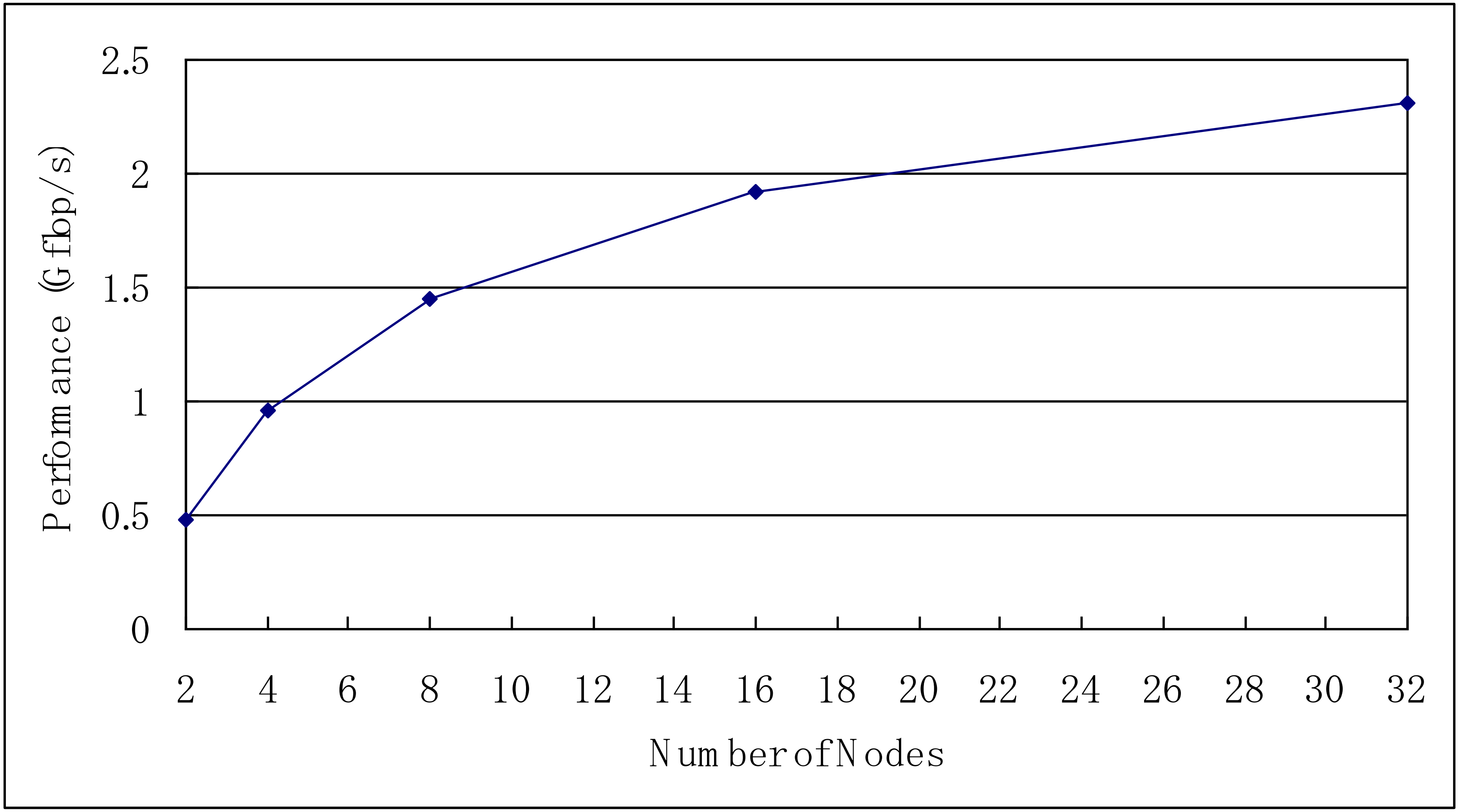
- parallelized in global view
- Using six-step FFT algorithm
  - Matrix transpose is a key operation.
- matrix transpose using **gmove**

```
#pragma xmp align a_work[*][i] with t1(i)
#pragma xmp align a[i][*] with t2(i)
#pragma xmp align b[i][*] with t1(i)
. . .
#pragma xmp gmove
  a_work[:][:] = a[:][:];                    // all to all

#pragma xmp loop on t1(i)
  for(i = 0; i < N1; i++)
    for(j = 0; j < N2; j++)
      c_assgn(b[i][j], a_work[j][i]);       // transpose
```

Performance of FFT

- Lines Of Code: 217



Future Works

- Full implementation of gmove directive, task parallelism ...
- More improvement of the performance, and algorithm.
  - One-sided communication
  - 2-D blocking in HPL
- hybrid programming for SMP clusters:

  XMP with OpenMP

for more information. . .

visit #639 Center for Computational Sciences, University of Tsukuba

#2897 T2K Open Supercomputer Alliance