# 2013 HPC Challenge Class 2 Submission XcalableMP for Productivity and Performance

Masahiro Nakao[†], Hitoshi Murai [‡]
Takenori Shimosaka[‡], Mitsuhisa Sato [†‡]

† Center for Computational Sciences, University of Tsukuba, Japan
‡ RIKEN Advanced Institute for Computational Science, Japan

# Outline

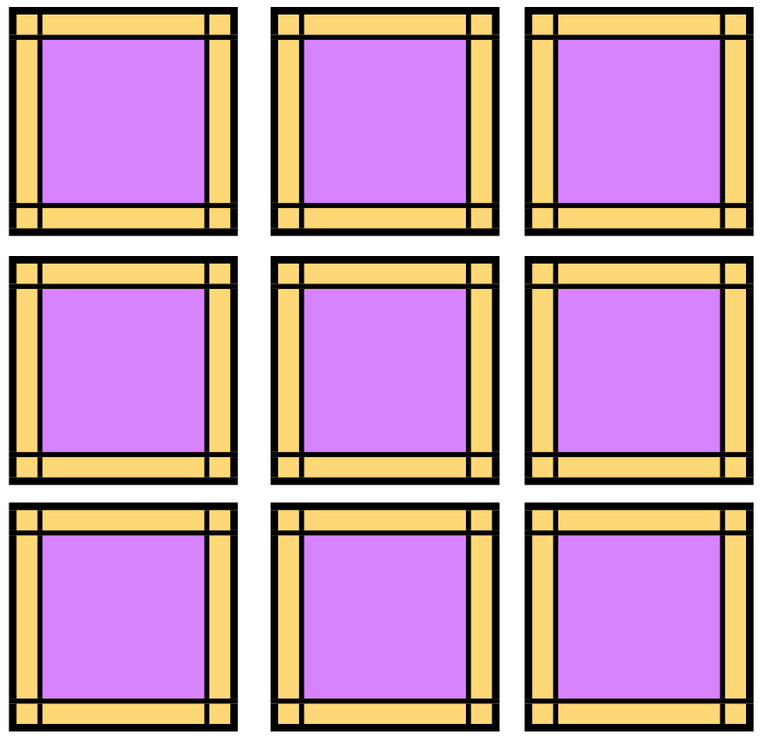
1. What's XcalableMP ?
2. Implementations
    1. HPL
    2. RandomAccess
    3. FFT
    4. STREAM
    5. HIMENO Benchmark (Optional)
3. Performances
4. Conclusion

# What is XcalableMP?

- XcalableMP (XMP)
  - Directive-based language extension of Fortran 2008 and C99
    - **XMP/Fortran** language and **XMP/C** language exist (same directives)
  - **Global-view** and **Local-view** memory models
    - XMP global-view model enables parallelizing the original sequential code using minimal modification with simple directives
    - **Coarray syntax** is available in both languages
      - XMP/Fortran is upward compatible with Fortran 2008
- **XMP specification ver. 1.2** at ***http://www.xcalablemp.org***
  - **XMP specification working group** consists of members from academia, research labs, and industries
    - Univ. of Tsukuba, RIKEN AICS, Fujitsu, NEC, Hitachi, and so on
- **Omni XMP Compiler ver. 0.7** at ***http://www.hpcs.cs.tsukuba.ac.jp/omni-compiler/xcalablemp/***

# Benchmarks

- HPCC benchmarks : **HPL, RandomAccess, FFT, STREAM**
- Optional benchmark : **Himeno benchmark**
  - Typical stencil application (http://accc.riken.jp/2444.htm)
  - Evaluates performance of incompressible fluid analysis code
  - Why do we select the HIMENO benchmark ?
    - To demonstrate parallelization by XMP directives which synchronizes the overlapped regions



# Benchmarks


## Omni XcalableMP Compiler

HOME

Download

Document

Benchmark

Publication

Mailing List

Links

### HPC Challenge Benchmarks

- High Performance Linpack
- RandomAccess
- Fast Fourier Transform
- EP STREAM triad

### HIMENO Benchmark

- HIMENO Benchmark

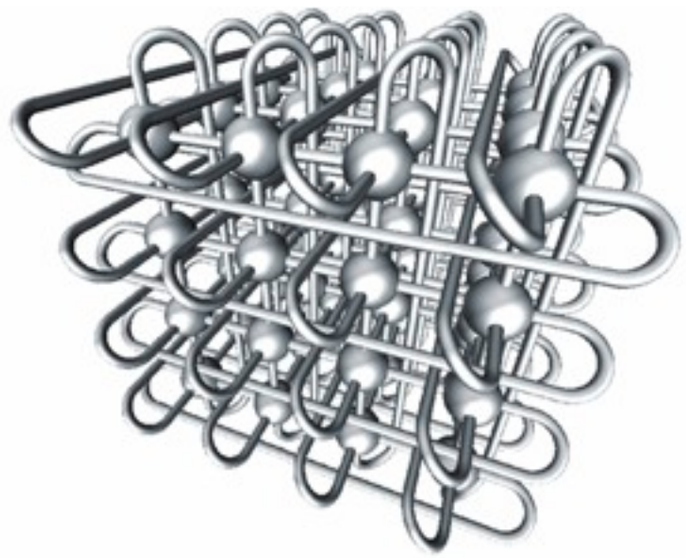
You can download these implementations at

***http://www.hpcs.cs.tsukuba.ac.jp/omni-compiler/xcalablemp/***

# The K computer

XcalableMP

- SPARC64 VIIIfx Chip
  - 128 GFlops (2GHz x 8Cores x 8)
  - 6MB L2 shared cache
  - 58W
- DDR3 SDRAM
  - 16GB
  - 64GB/s
- Tofu Interconnect
  - 6D mesh/torus network
  - 5GB/s x 4links x 2
- 82,944 compute nodes (+ 5,184 IO nodes)

*http://www.aics.riken.jp*

# Outline

1. What's XcalableMP ?
2. Implementations
    1. HPL
    2. RandomAccess
    3. FFT
    4. STREAM
    5. HIMENO Benchmark (Optional)
3. Performances
4. Conclusion

# Summary

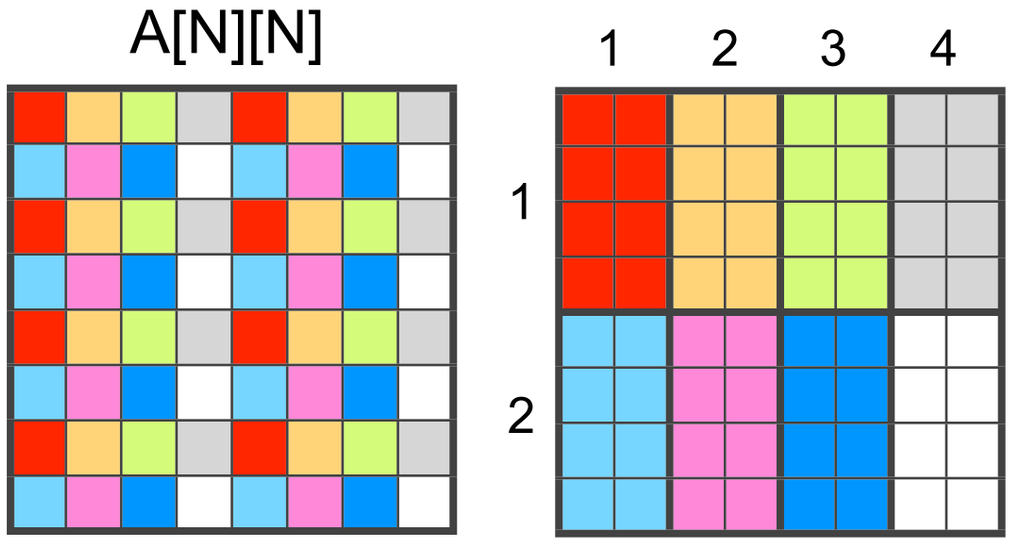
| Benchmark | # Nodes | Performance | SLOC |
|---|---|---|---|
| HPL | 16,384 | 933.8 TFlops (44.5% of peak) | 306 |
| RandomAccess | 16,384 | 162.6 GUPs | 250 |
| FFT | 36,864 | 50.1 TFlops (1.1% of peak) | 239 + 283 + 1892 |
| STREAM | 16,384 | 481.8 TB/s | 66 |
| HIMENO | **82,944** | 1.3 PFlops (12.7% of peak) | 137 |

Full compute nodes

# HPL

- Source lines of Code (SLOC) is **306**, written in XMP/C
- Block-Cyclic Distribution

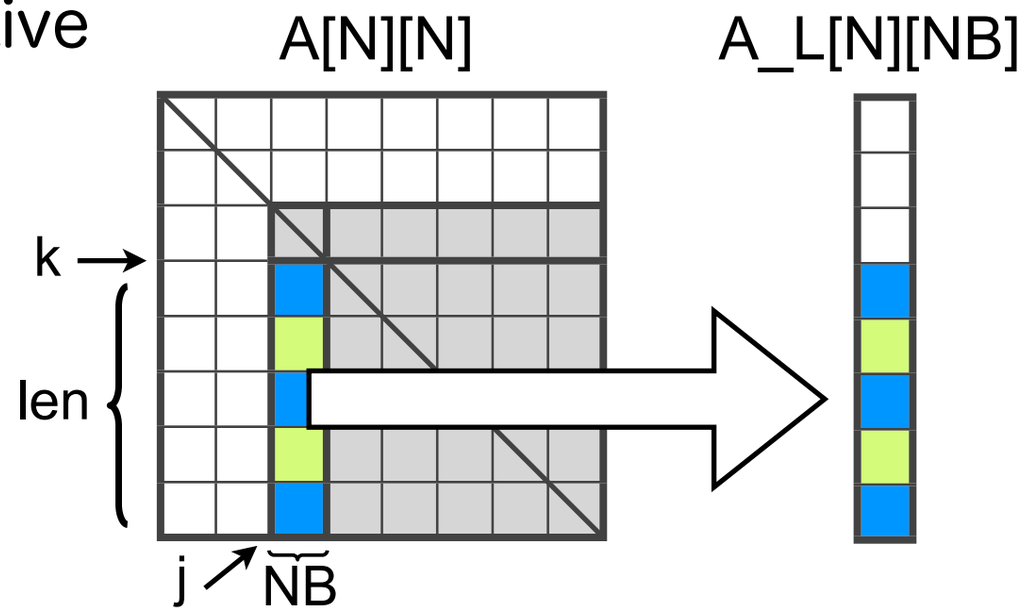
```
double A[N][N];
#pragma xmp nodes p(P,Q)
#pragma xmp template t(0:N-1, 0:N-1)
#pragma xmp distribute t(cyclic(NB), \
                         cyclic(NB)) onto p
#pragma xmp align A[i][j] with t(j,i)
```




**Programmer can use BLAS for distributed array.**

- Panel Broadcast by using **gmove** directive

```
double A_L[N][NB];
#pragma xmp align A_L[i][*] with t(*,i)
     :
#pragma xmp gmove
A_L[k:len][0:NB] = A[k:len][j:NB];
```



# RandomAccess

- SLOC is **253**, written in XMP/C
- The XMP RandomAccess is iterated over sets of CHUNK updates on each node
- Local-view programming with XMP/C coarray syntax

```
u64Int recv[LOGPROCS][RCHUNK+1]:[*];
...
for (j = 0; j < logNumProcs; j++) {
  recv[j][0:num]:[i_partner] = send[i][0:num];

#pragma xmp sync memory
#pragma xmp post(p(i_partner), 0)
    :
#pragma xmp wait(p(j_partner))
}
```

Define coarray

Put operation

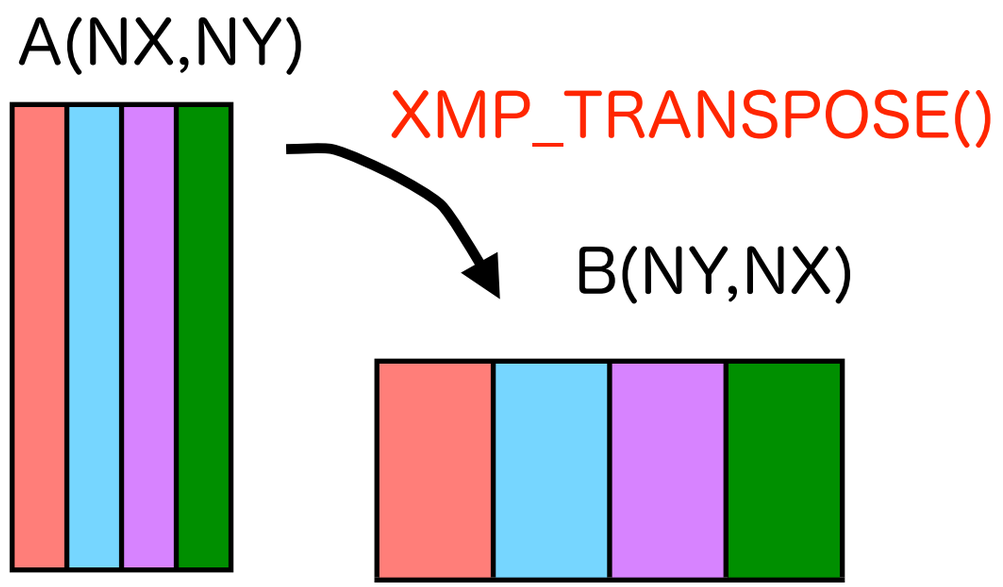
- A point-to-point synchronization is specified with the XMP's post and wait directives to realize asynchronous behavior of this algorithm

# FFT

- SLOC is **239**+ 283 +1,892, written in XMP/Fortran + OpenMP
  - We mainly modified "pzfft1d.f" in ffte optimized for the K computer
    - This SLOC is **70** (the SLOC of original "pzfft1d.f" is 101)
  - The total SLOC of the modified files is 239 (**70** + 169)
  - The total SLOC of NOT modified files is 283
  - The total SLOC of C interface and another kernels of the ffte is 1,892

```
CALL XMP_TRANSPOSE(B,A,1)
...
!$XMP loop on tx(I)
!$OMP parallel do
DO I=1,NX
  DO J=1,NY
    B(J,I)=B(J,I)*W(J,I)
  END DO
END DO
```

A() and B() are global arrays



# EP STREAM Triad

- SLOC is **66**, written in XMP/C + OpenMP
- a[], b[], and c[] are local arrays

```
#pragma xmp nodes p(*)
...
for(k=0; k<NTIMES; k++) {
#pragma omp parallel for
  for (j=0; j<size; j++)
    a[j] = b[j] + scalar*c[j];
}
...
*triadGBs = ...  // performance of each node
...
#pragma xmp reduction(+:triadGBs)
```

Parallelize the program

Gather performance results on each node

# HIMENO Benchmark

- SLOC is **137**, written in XMP/Fortran
  - SLOC of the original is **380**, written in MPI/Fortran

To synchronize overlapped regions, we use XMP shadow and reflect directives

```
!$xmp shadow p(0,2:1,2:1)
 ...
!$xmp reflect (p)
 ...
!$xmp loop (J,K) on t(*,J,K)
do K = 2, kmax-1
  do J = 2, jmax-1
    do I = 2, imax-1
      S0 = a(I,J,K,1)*p(I+1,J,K) + ...
       :
    enddo
  enddo
enddo
```

Define overlapped region

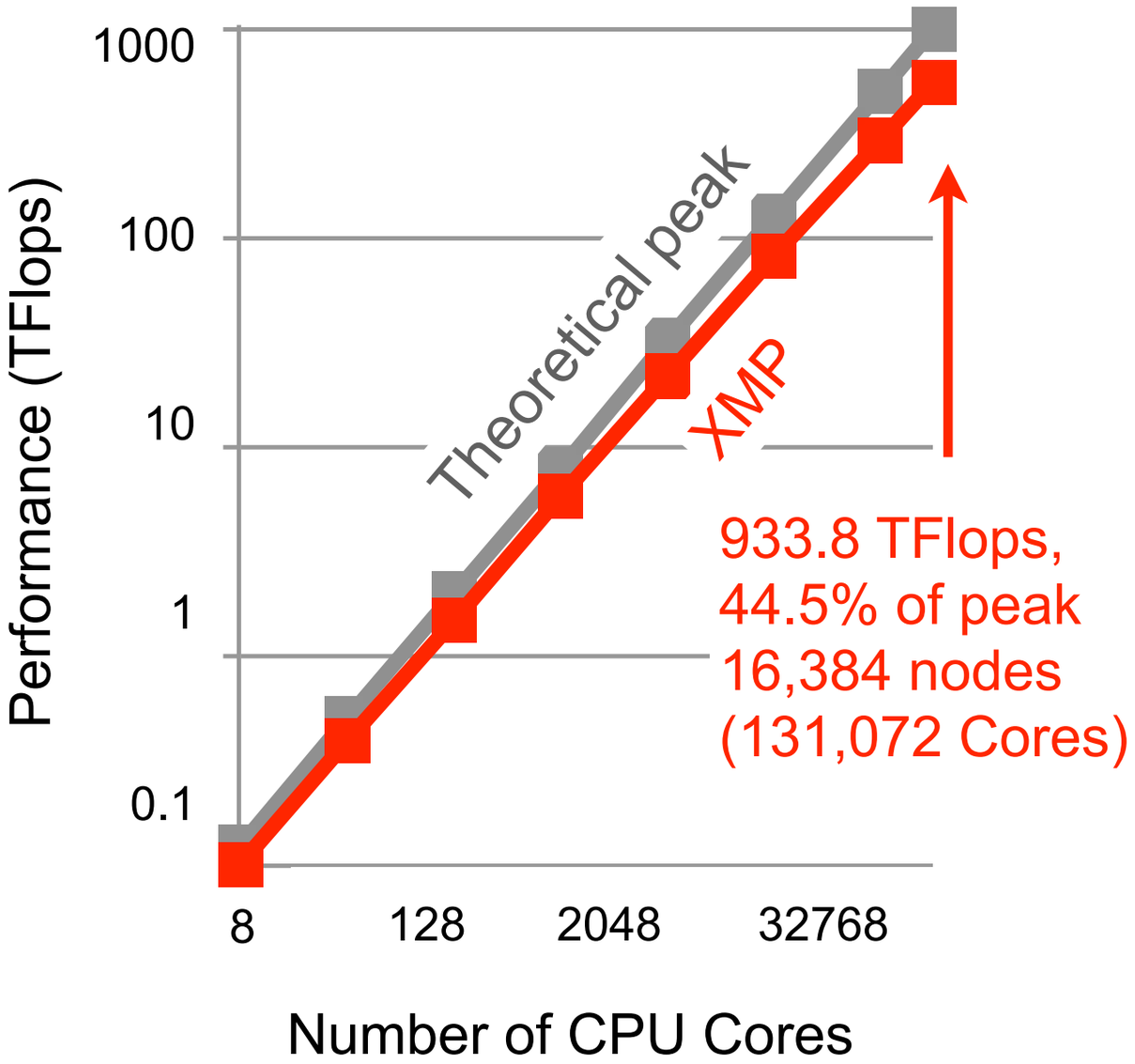
Synchronize overlapped region

Work mapping

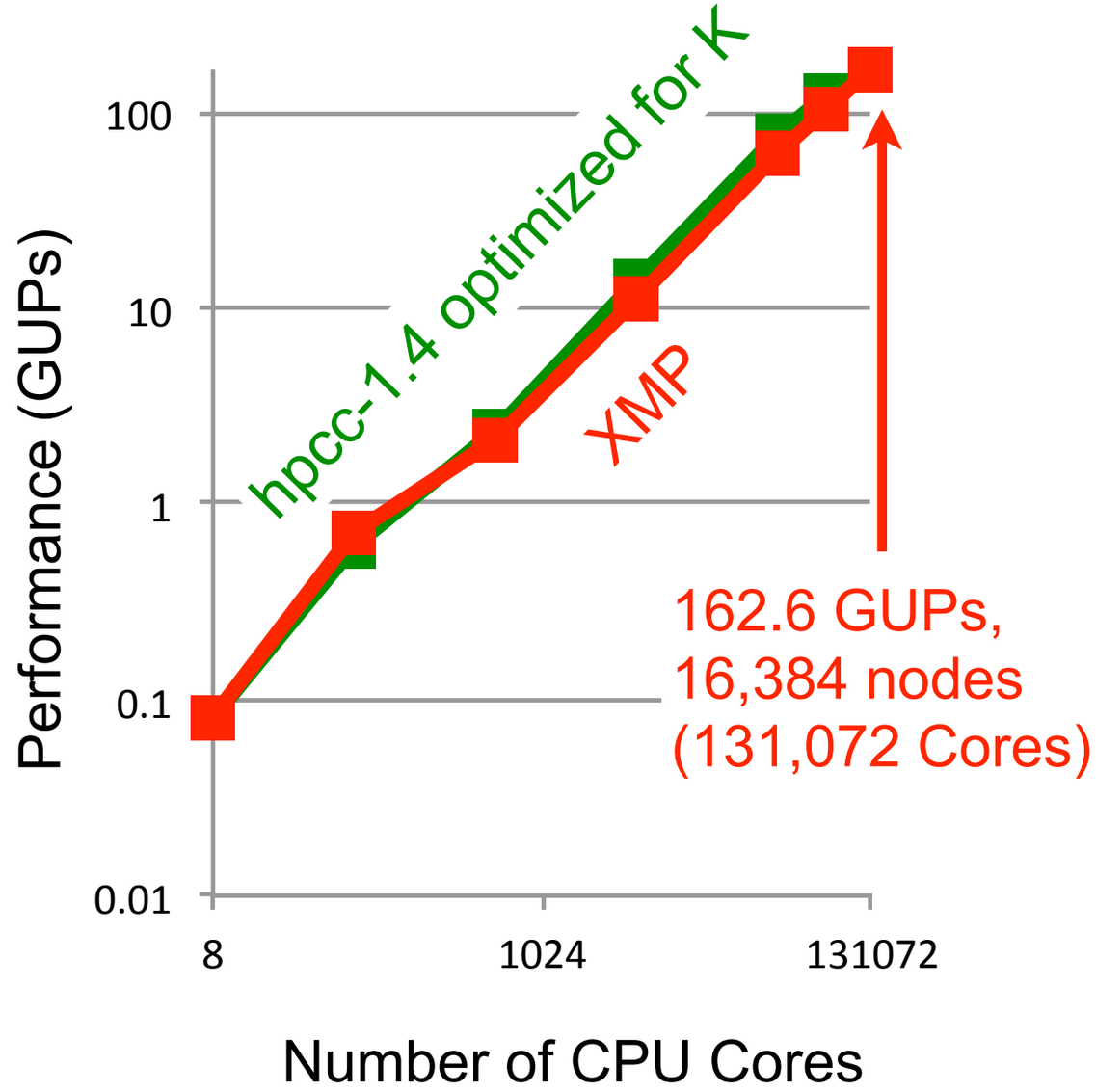
Only add XMP directives into the sequential Himeno benchmark basically.

# Results(1/3)

- HPL (1 process/node + Threaded BLAS)

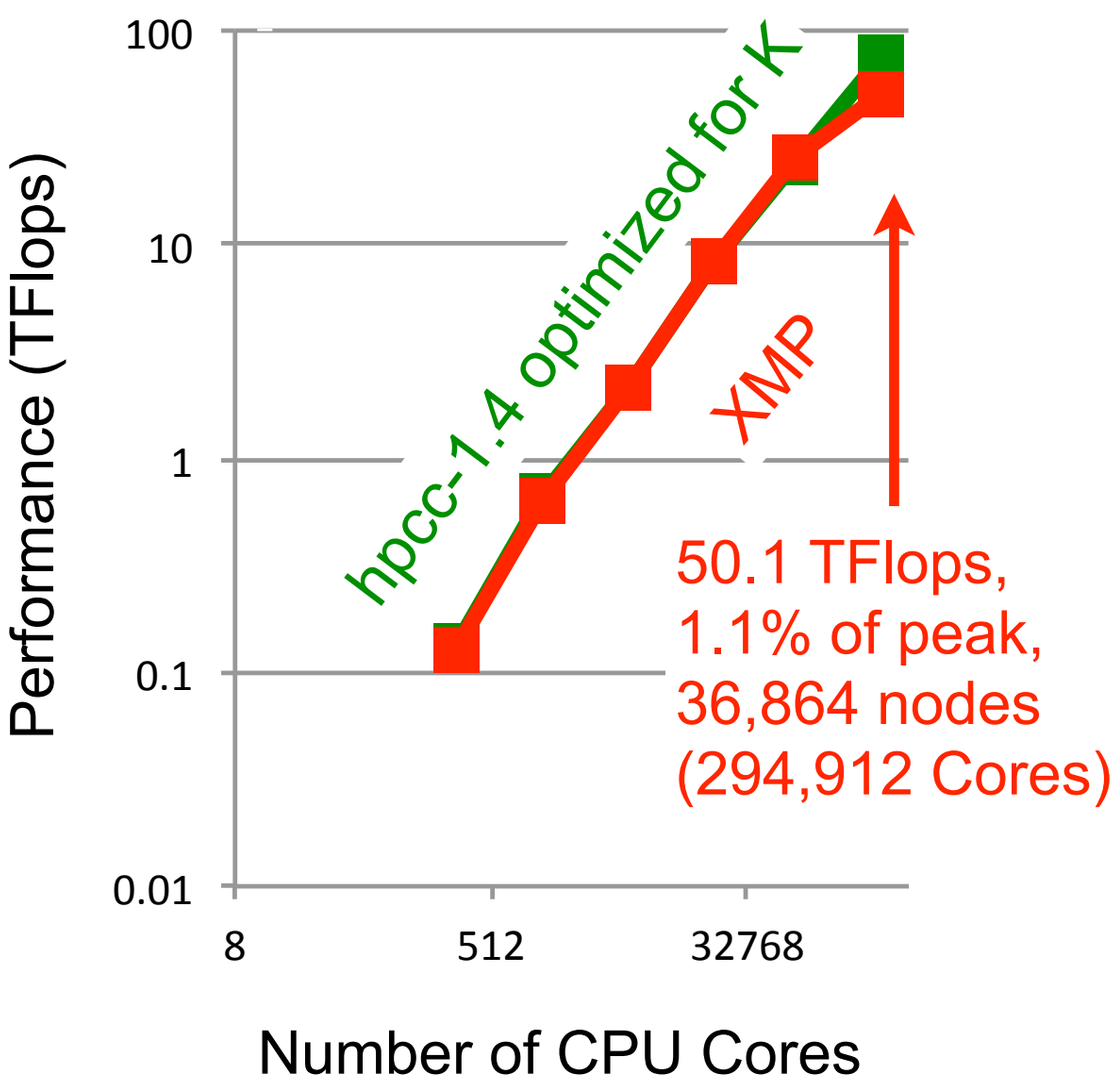



- RandomAccess (8 processes/node)

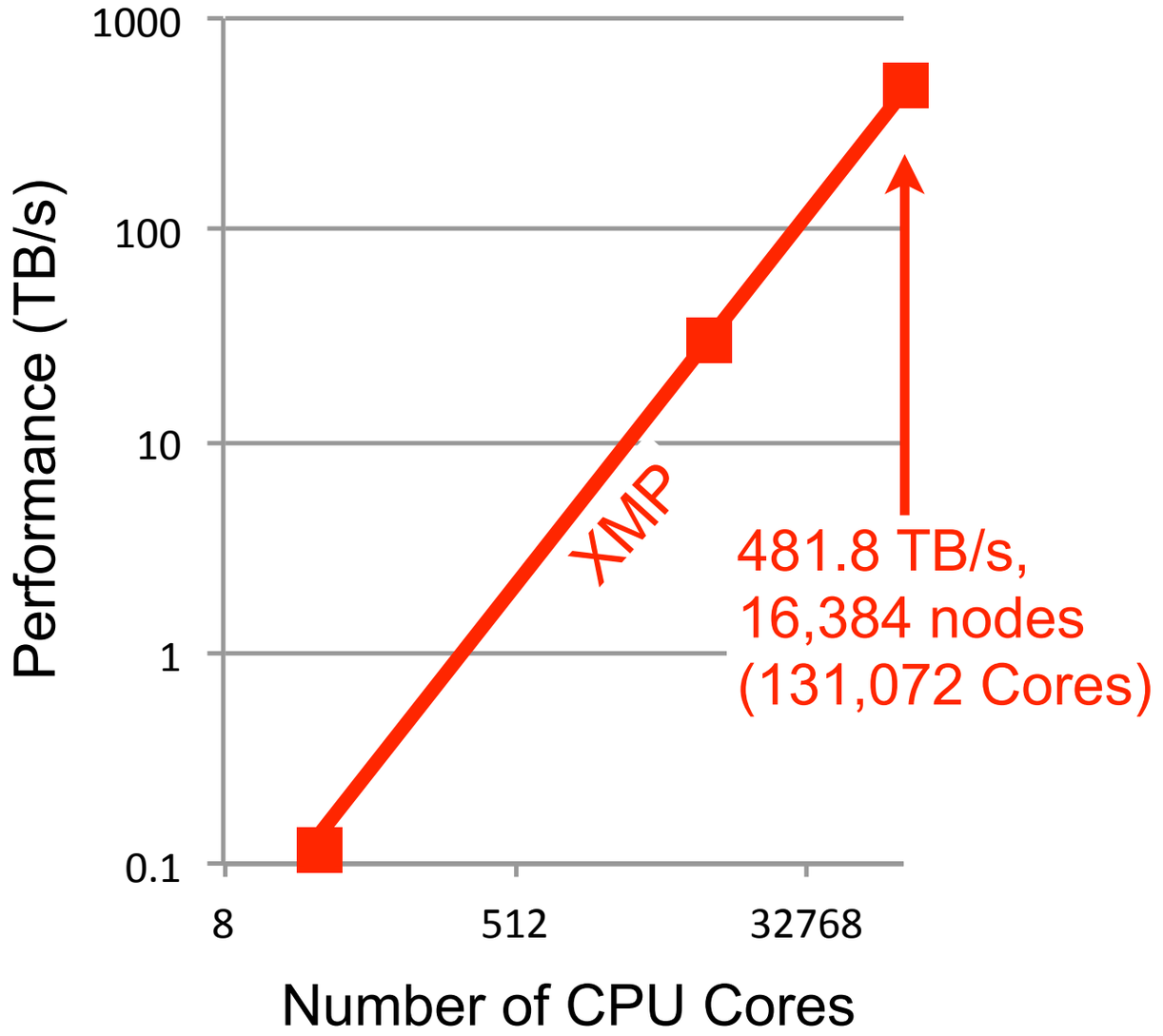


# Results(2/3)

- FFT (1 process/node with 8 threads)

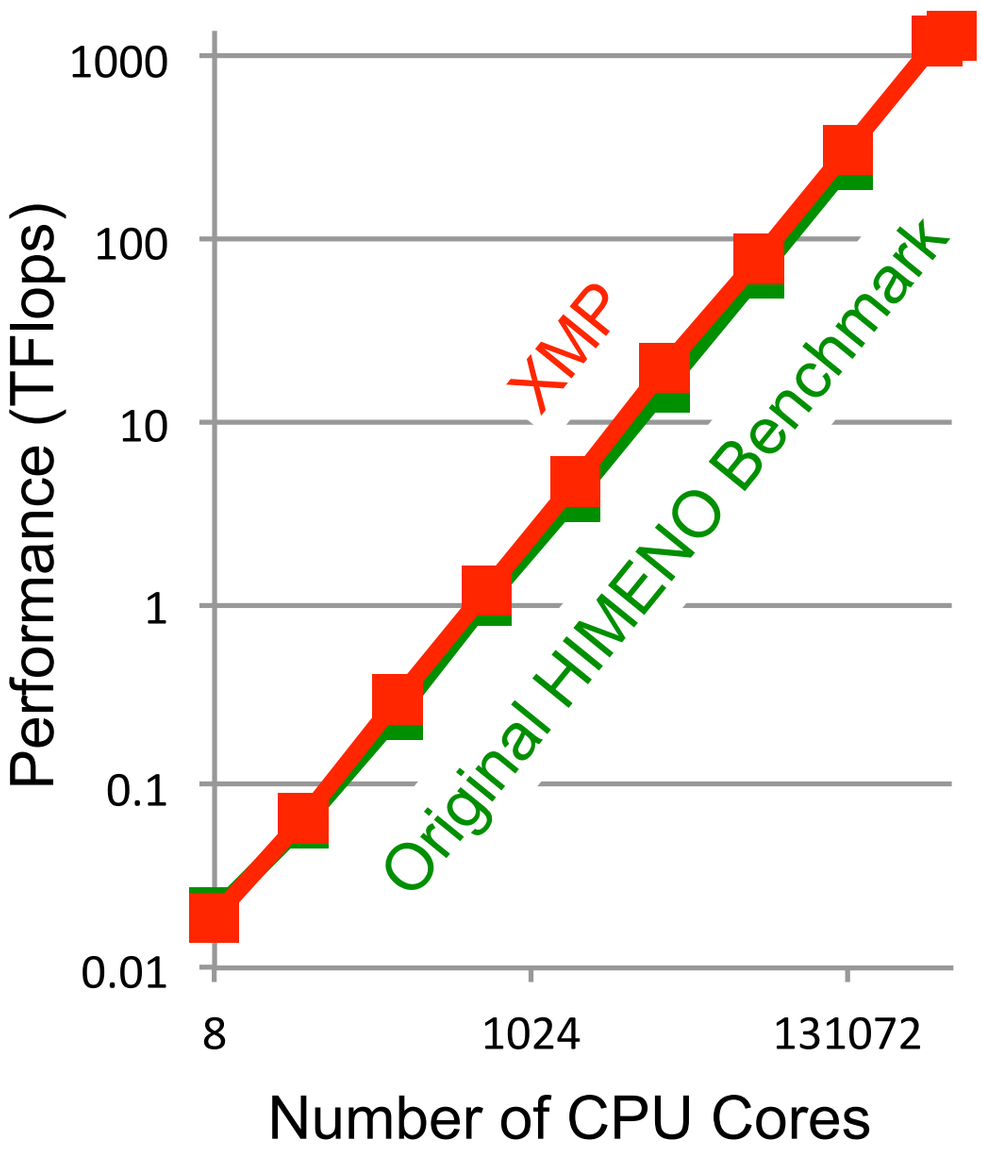



- STREAM (1 process/node with 8 threads)



# Results(3/3)

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




1.3 PFlops, 12.7% of peak
82,944 nodes(663,552 Cores),

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

# Conclusion

| Benchmark | # Nodes | Performance | SLOC |
|---|---|---|---|
| HPL | 16,384 | 933.8 TFlops (44.5% of peak) | 306 |
| RandomAccess | 16,384 | 162.6 GUPs | 250 |
| FFT | 36,864 | 50.1 TFlops (1.1% of peak) | 239 + 283 + 1892 |
| STREAM | 16,384 | 481.8 TB/s | 66 |
| HIMENO | **82,944** | 1.3 PFlops (12.7% of peak) | 137 |

**Masahiro Nakao, et al. "Productivity and Performance of the HPC Challenge Benchmarks with the XcalableMP PGAS language", 7th International Conference on PGAS Programming Models, October, 2013**

For more information, please visit

**#1437 RIKEN AICS (Advanced Institute for Computational Science)**

**#2519 Center for Computational Sciences, University of Tsukuba**