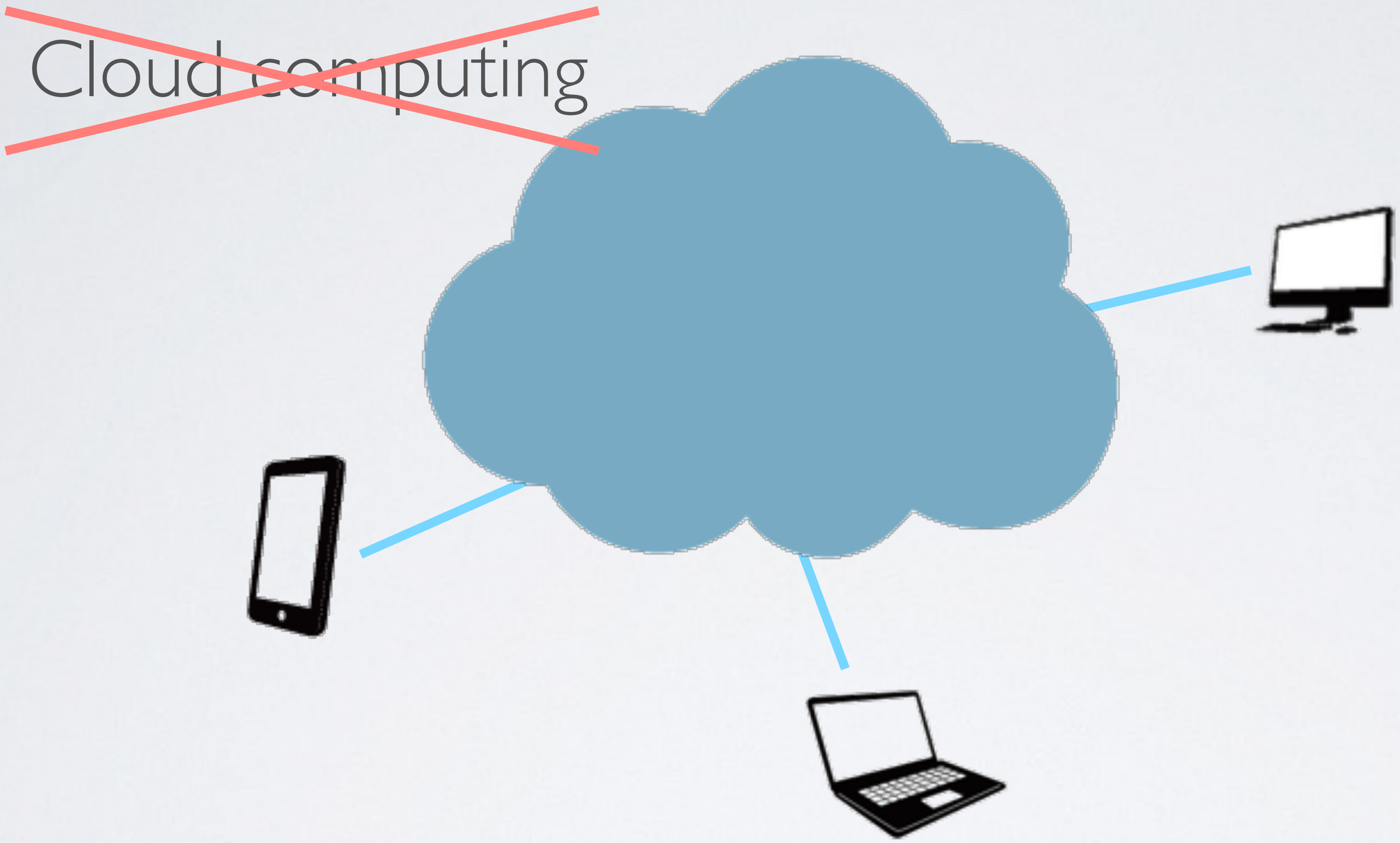
A Simulation of Global Atmosphere Model NICAM on TSUBAME 2.5 Using OpenACC

Hisashi YASHIRO
RIKEN Advanced Institute of Computational Science
Kobe, Japan

GTC2015, San Jose, Mar. 17-20, 2015

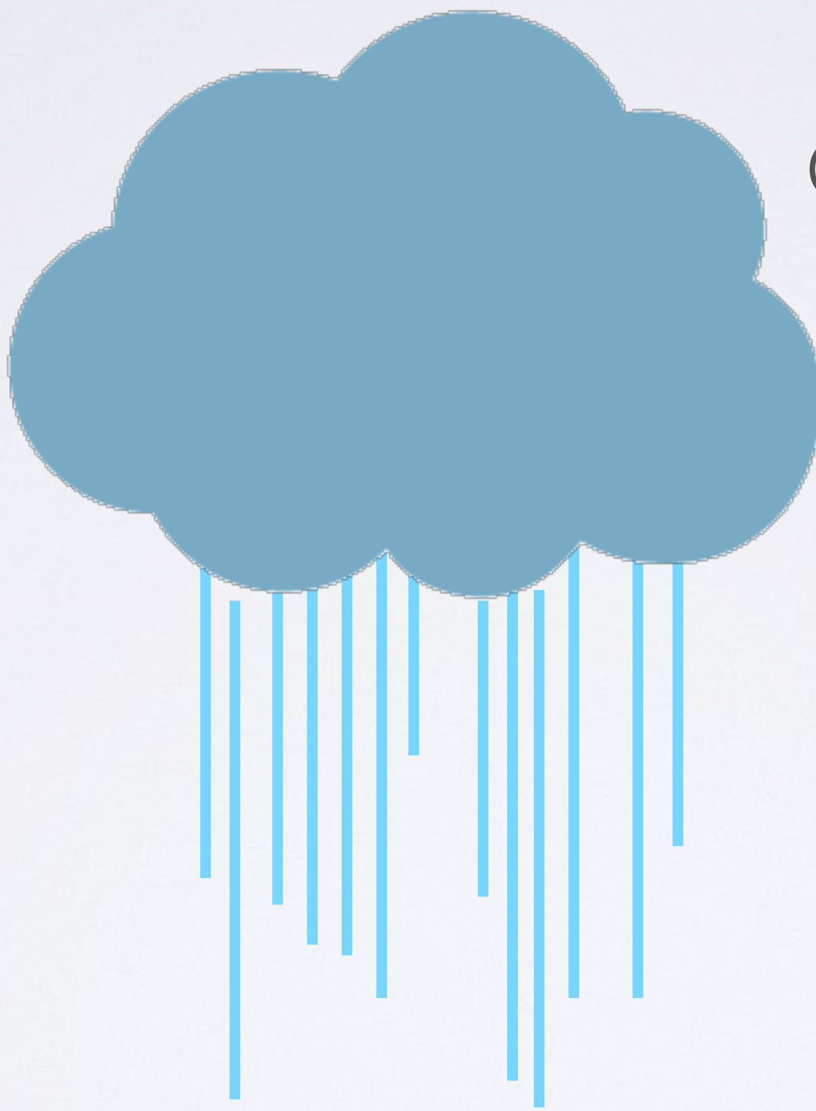
My topic

- The study for…



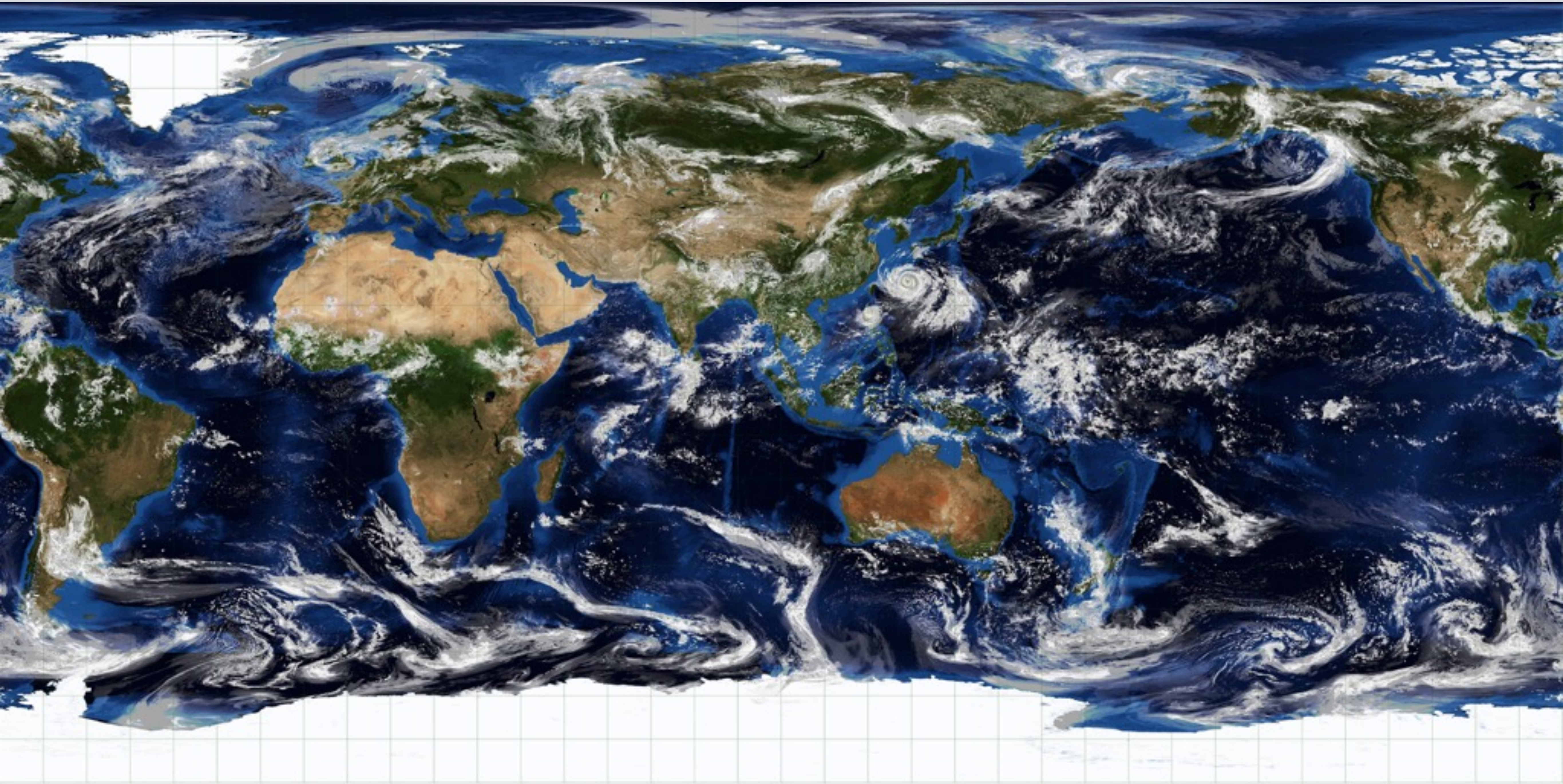
My topic

- The study for…



Computing of the cloud

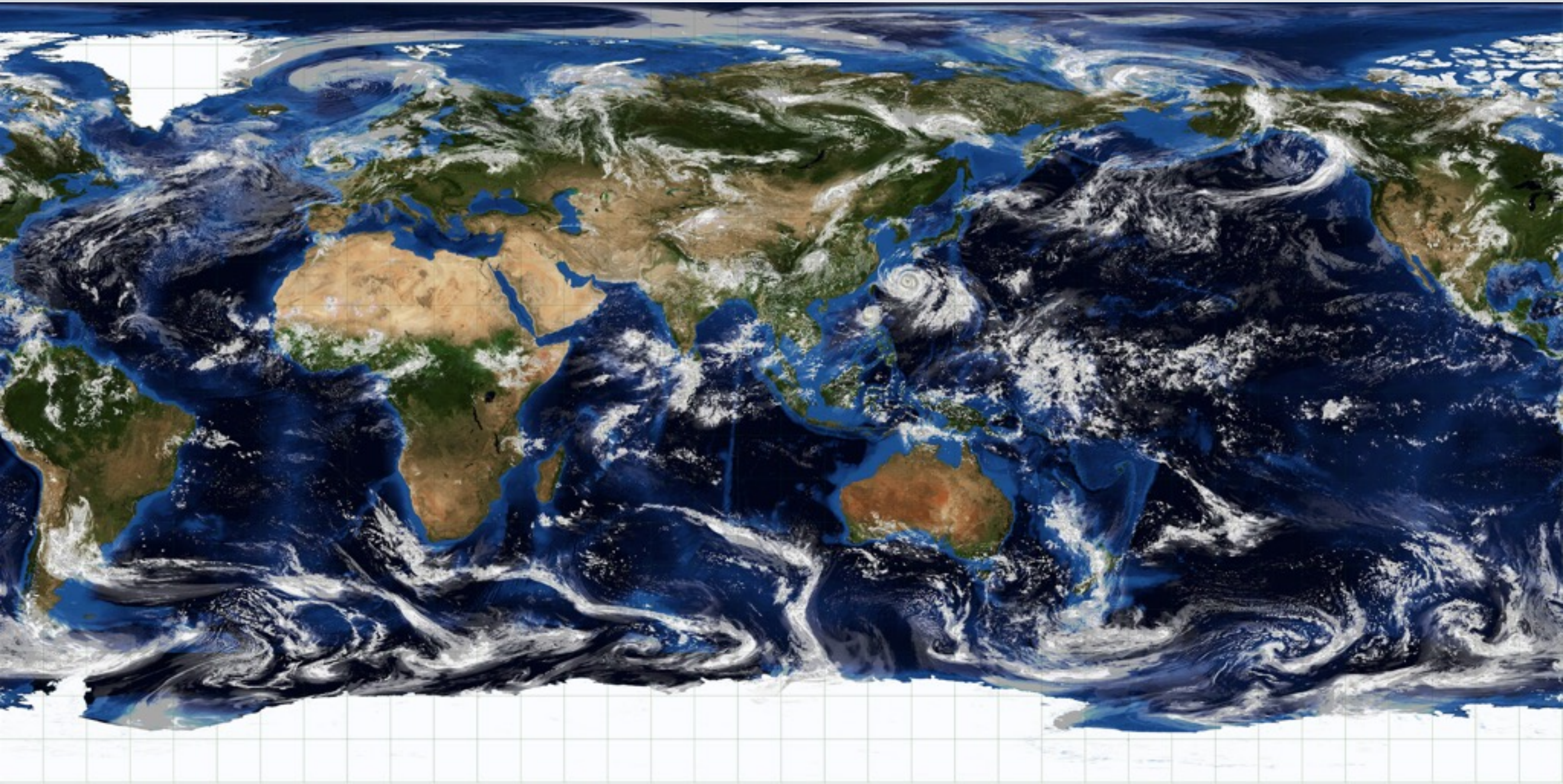
Clouds over the globe



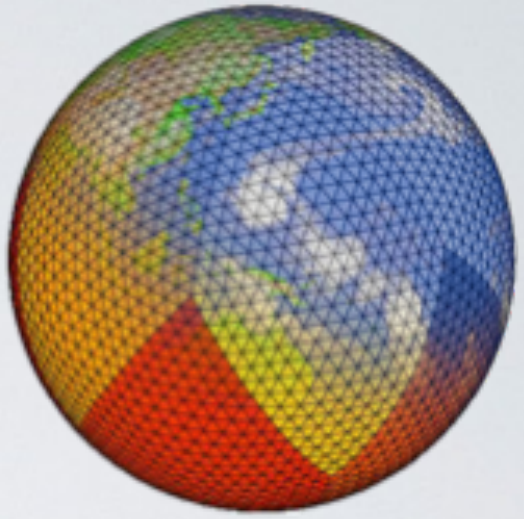
The first global sub-km weather simulation

20480nodes(163840cores) on the K computer

Movie by R.Yoshida(RIKEN AICS)



NICAM



Non-hydrostatic Icosahedral Atmospheric Model (NICAM)

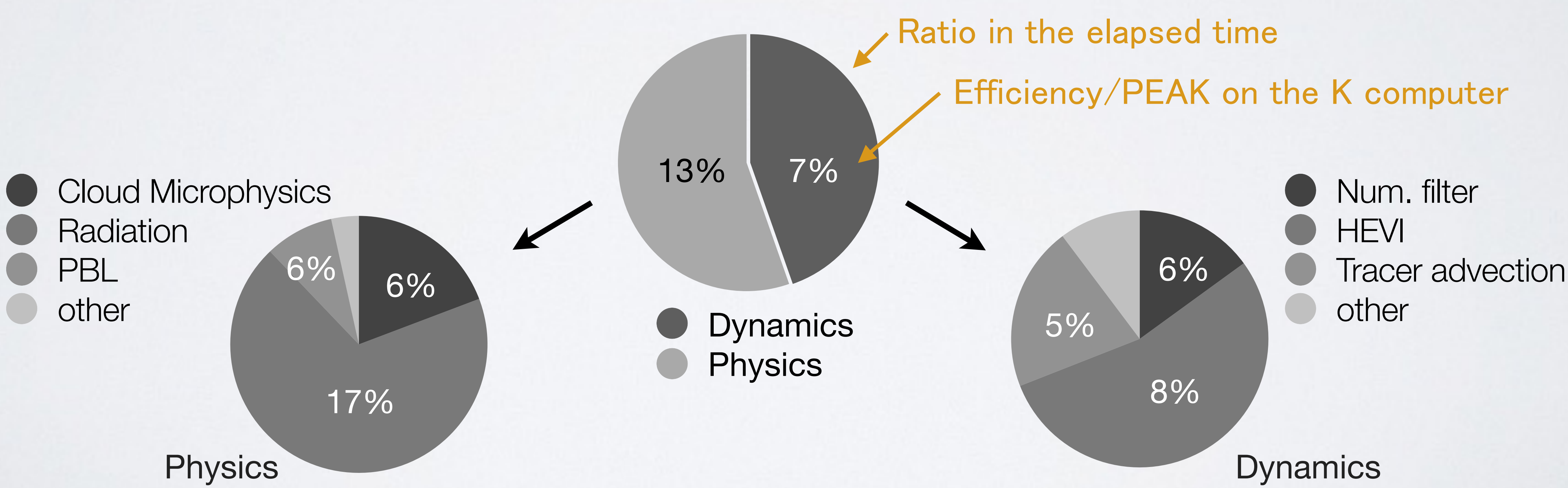
- Development was started since 2000
  Tomita and Satoh (2005), Satoh et al. (2008, 2014)
- First global dx=3.5km run in 2004 using the Earth Simulator
  Tomita et al. (2005), Miura et al. (2007, Science)
- First global dx=0.87km run in 2012 using the K computer
  Miyamoto et al. (2014)

- FVM with icosahedral grid system
- Written by Fortran90

- Selected as a target application in post-K computer development
  : System-Application co-design

“Dynamics” and “Physics” in Weather/Climate Model

- **“Dynamics” : fluid dynamics solver of the atmosphere**
  - Grid method (FDM, FVM, FEM) with horizontal explicit-vertical implicit scheme, or Spectral method
- **“Physics” : external forcing and sub-grid scale**
  - Cloud microphysics, atmospheric radiation, turbulence in boundary layer, chemistry, cumulus, etc..
  - Parameterized, no communication, big loop body with “if” branches



Issues of Weather/Climate Model & Application

The Bandwidth Eater

- **Low computational intensity**
  : Using a lot of variables, low-order scheme

- **Huge code**
  : 10K~100K lines (without comments!)

- **Active development and integration**
  : Fully-tuned codes may replace by the student's new scheme

The Bandwidth Eater

- **It shows "Flat profile"**
  : No large hot-spots of computation

- **Frequent file I/O**
  : Requires the throughput from accelerator to storage disk

➡ We have to optimize everywhere in the application!

Challenge to GPU computation

- **We want to…**
  - Utilize memory throughput of GPU
  - Offload all component of the application
  - Keep portability of the application : one code for ES, K computer and GPU
- **We don’t want to…**
  - Rewrite all component of the application by special language

➡ OpenACC is suitable for our application

NICAM-DC with OpenACC

- **NICAM-DC: Dynamical core package of NICAM**
  - BSD 2-clause licence
  - From website (http://scale.aics.riken.jp/nicamdc/) or GitHub
  - Basic test cases are prepared

- **OpenACC implementation**
  - With the support of the specialist of NVIDIA (Mr. Naruse)

- **Performance evaluation on TSUBAME 2.5 (Tokyo Tech.)**
  - Largest GPU supercomputer in Japan : 1300+ nodes, 3GPUs per node
  - We used 2560GPUs (1280nodes x 2GPUs) for grand challenge run

NICAM-DC with OpenACC

- **Strategy**
  - Transfer common variables to GPU using "data pcopyin" clause
    : After the setup (memory allocation), arrays which use in the dynamical step (e.g. stencil operator coefficient) are transferred all at once
  - Data layout
    : Several loop kernels are reverted from Array of Structure (AoS) to Structure of Array (SoA), which is suitable for GPU computing
  - Asynchronous execution of loop kernels
    : "async" clause is used as much as possible

NICAM-DC with OpenACC

- **Strategy (continue)**
  - Ignore irregular, small computation part
    : Pole points are calculated on the host CPU of master rank
    - We don't have to separate kernel for this: It's advantage of OpenACC
  - MPI communication
    : Data packing/unpacking of halo grids are processed on GPU to reduce the size of data transfer between host and device
  - File I/O
    :Variables for output are updated in each time step on GPU
    - At the time to file write, the data is transferred from devise

Node-to-node comparison

TUBAME2.5 GPU
2MPI/node
1GPU/MPI

2620GFLOPS
500GB/s
B/F=0.2
Fat-tree IB

TUBAME2.5 CPU
8MPI/node

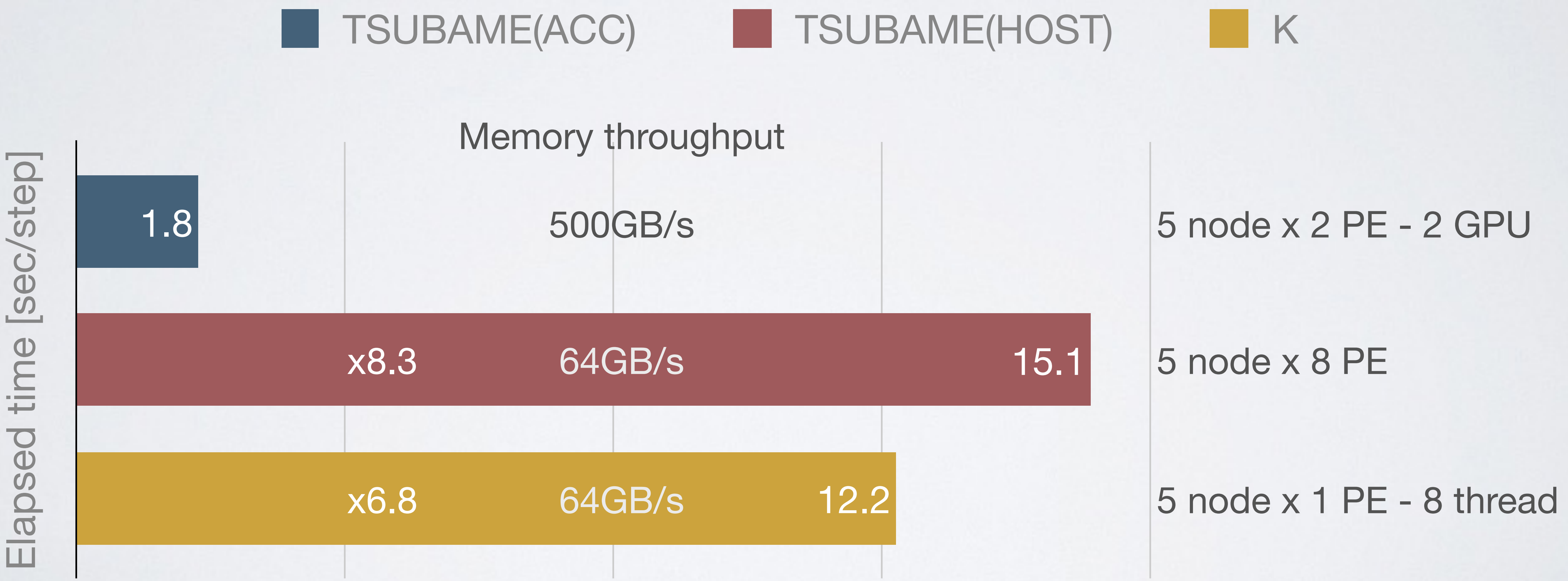
102GFLOPS
64GB/s
B/F=0.6
Fat-tree IB

K computer
1MPI/node
8thread/MPI

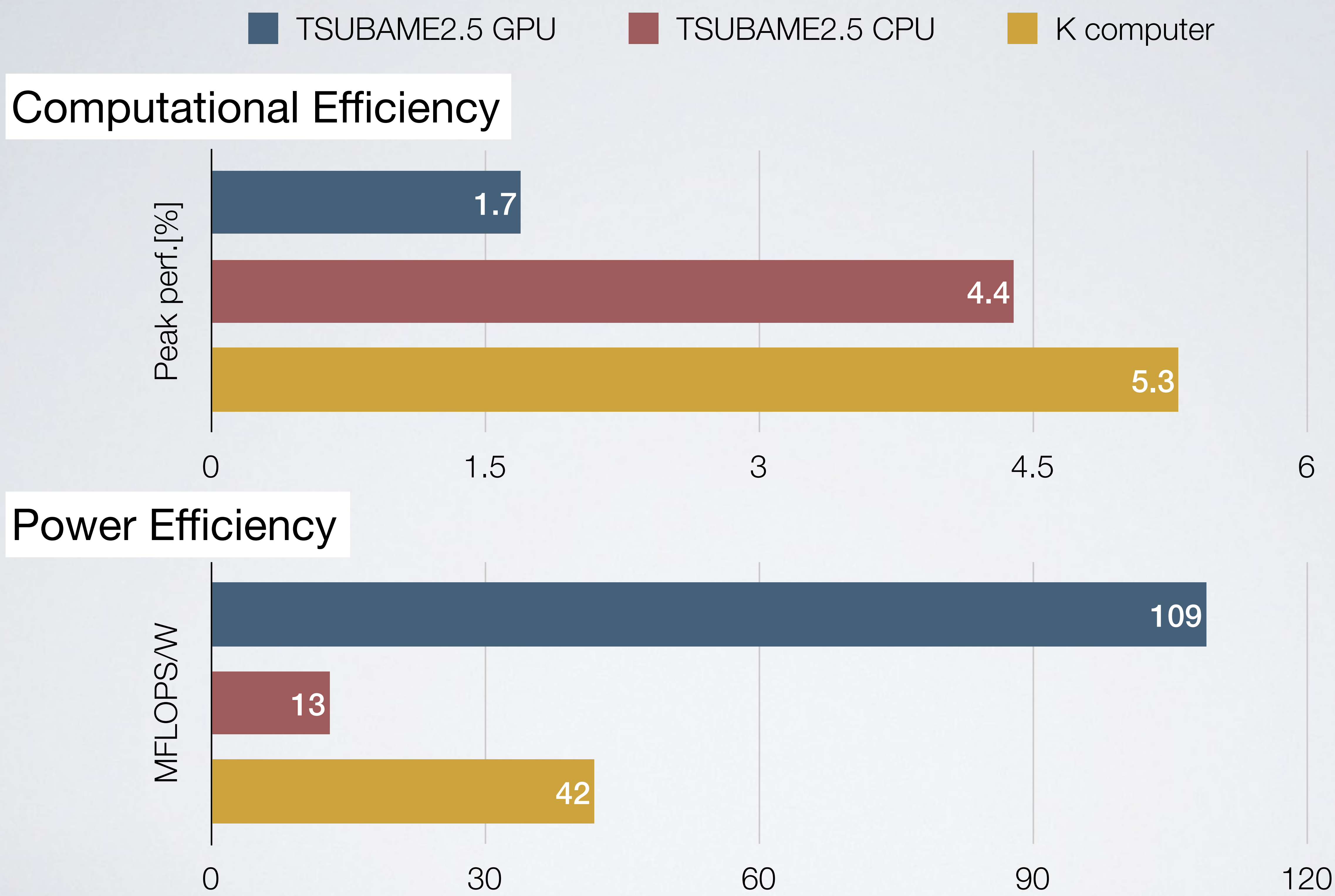
128GFLOPS
64GB/s
B/F=0.5
Tofu

Node-to-node comparison

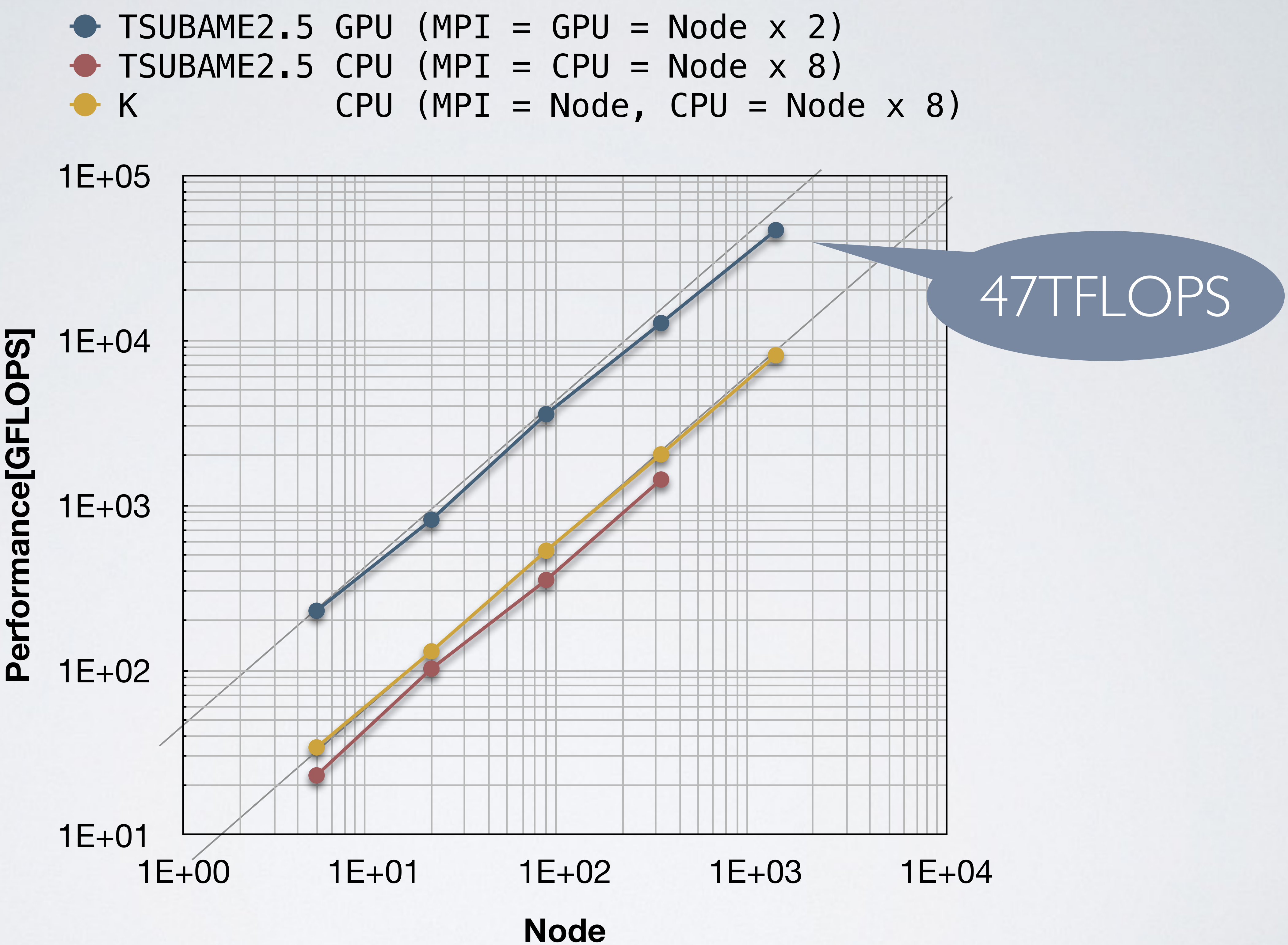
- GPU run is 7-8x faster than CPU run
  :Appropriate to the memory performance
  - We achieved a good performance without writing any CUDA kernels
- Modified/Added lines of the code were only 5% (~2000lines)



Node-to-node comparison



Weak scaling test

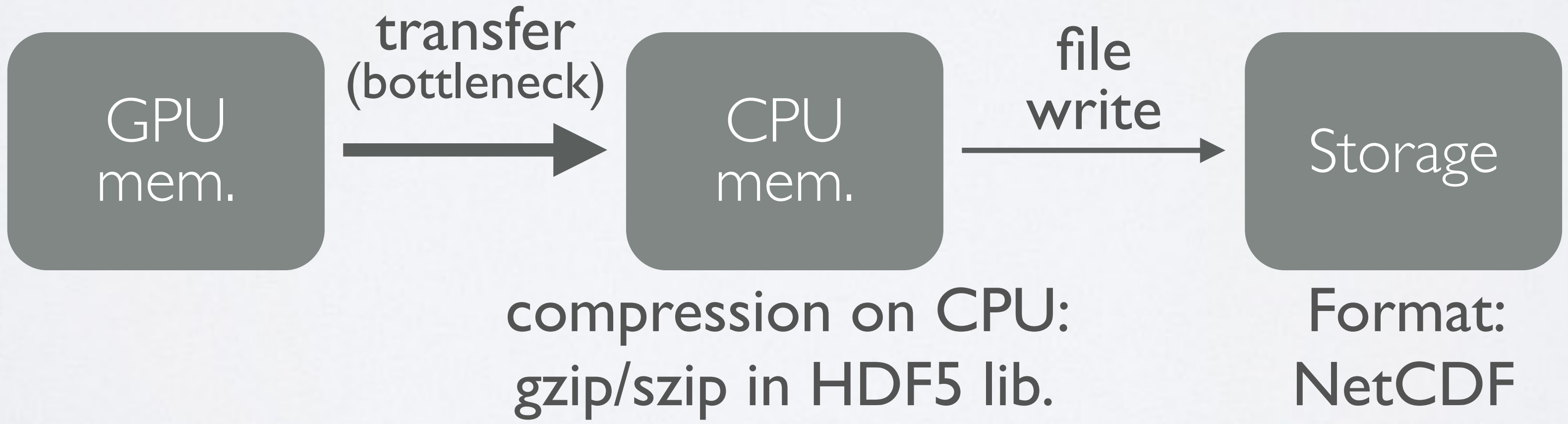


Weak scaling test

- **47TFLOPS in largest problem size**
  - In this case, diagnostic variables were written in every 15 min. of simulation time
  - By selecting the typical output interval (every 3 hours = 720 steps), we achieved **60TFLOPS**

- **File I/O is critical in production run**
  - We can compress output data on GPU

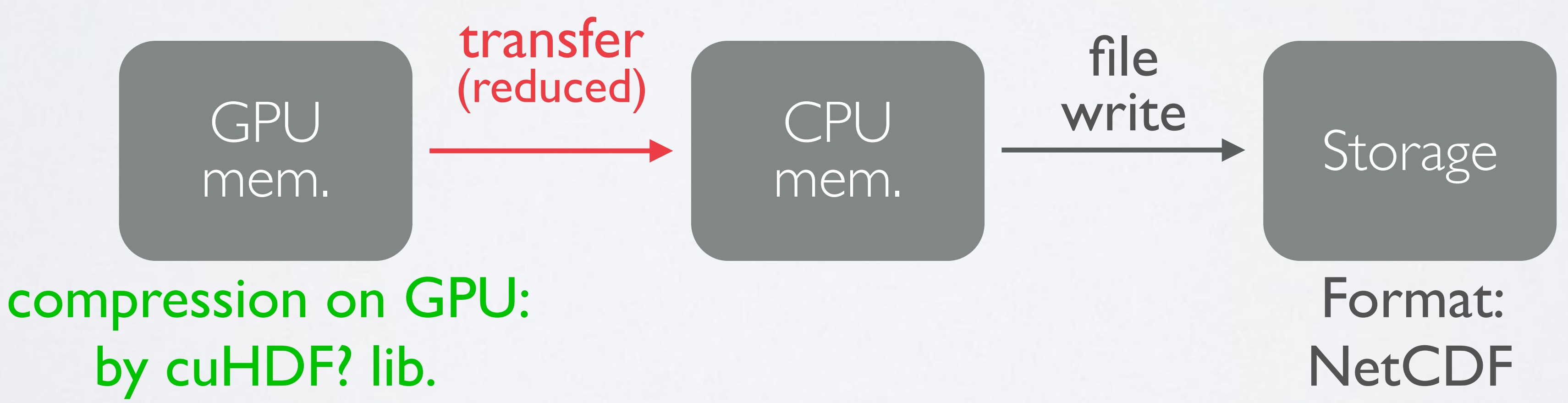
➡ We really need GPU-optimized, popular compression library: **cuHDF?**



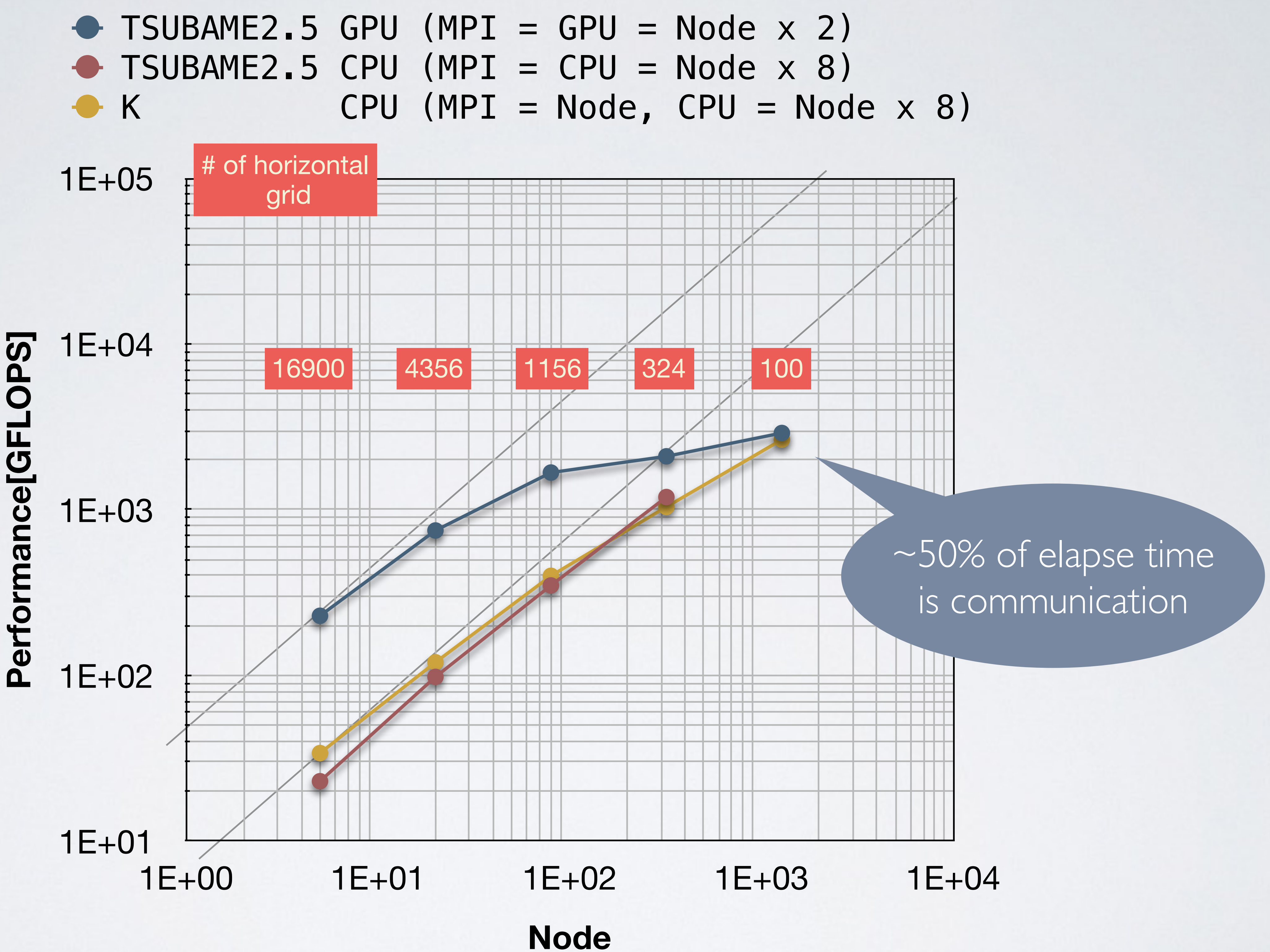
Weak scaling test

- **47TFLOPS in largest problem size**
  - In this case, diagnostic variables were written in every 15 min. of simulation time
  - By selecting the typical output interval (every 3 hours = 720 steps), we achieved **60TFLOPS**

- **File I/O is critical in production run**
  - We can compress output data on GPU
  - ➡ We really need GPU-optimized, popular compression library: **cuHDF?**



Strong scaling test



Summary

- **OpenACC enables easy porting of weather/climate model to GPU**
  - We achieved good performance and scalability with small modification
  - Performance of data transfer limits application performance
    - “Pinned memory” is effective for H-D transfer
    - In near future, NVLink and HBM is expected
    - File I/O issue is critical
    - More effort of application side is necessary
      - ➡ "Precision-aware" coding, from both scientific and computational viewpoint.

- **Ongoing effort**
  - OpenACC for all physics component

Thank you for the attention!