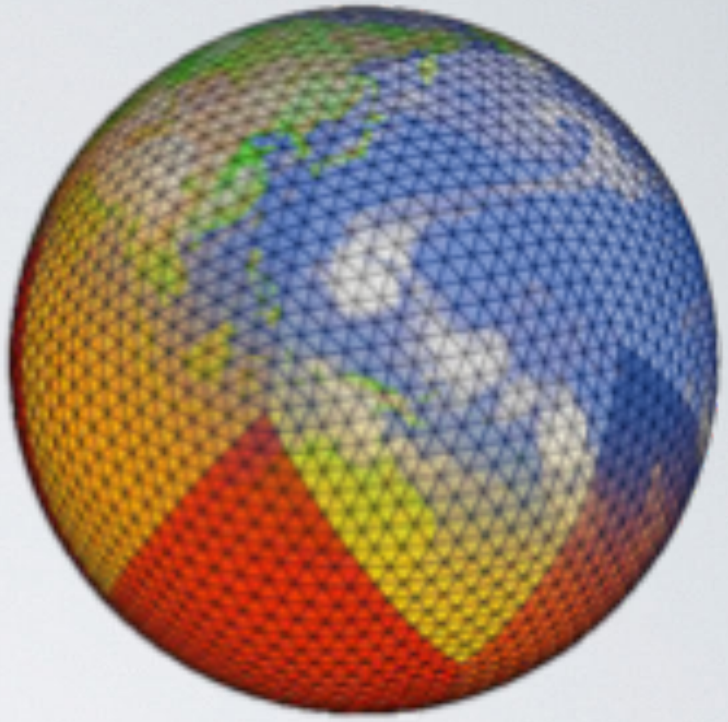
RECENT PERFORMANCE OF NICAM ON THE K-COMPUTER AND ACTIVITIES TOWARDS POST-PETASCALE COMPUTING

H.YASHIRO

RIKEN/Advanced Institute of Computational Science, Kobe, JAPAN

NICAM



- Non-hydrostatic Icosahedral Atmospheric Model (NICAM)
  - Development was started since 2000
    Tomita and Satoh (2005, Fluid Dyn. Res.), Satoh et al. (2008, J. Comp. Phys.)
  - First global dx=3.5km run in 2004 using the Earth Simulator
    Tomita et al. (2005, Geophys. Res. Lett.), Miura et al. (2007, Science)
  - Main developers: JAMSTEC, U. of Tokyo, RIKEN/AICS
  - International collaboration
    Athena project (2009-10): COLA, NICS, ECMWF, JAMSTEC, Univ. of Tokyo

The K computer

- Rmax = 10.51PFLOPs (93% efficiency)
- Won the TOP500 list twice (2011)
- Two Gordon Bell prize
- Shared use started on September, 2012

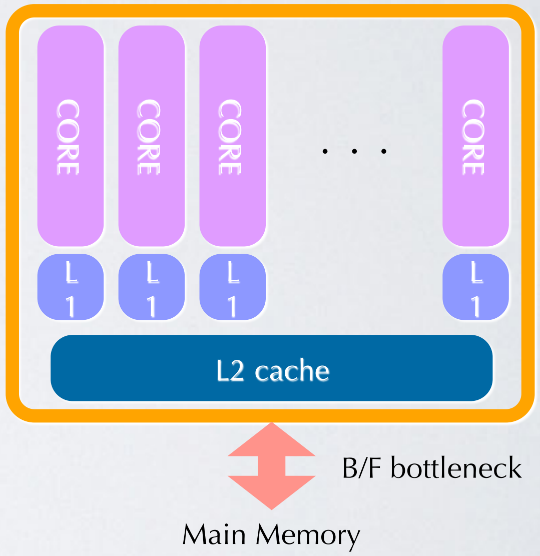
- CPU: Fujitsu SPARC64VIIIfx (128GFlops, 8core)
- 2FMA, 2SIMD
- 1PB memory, 64GB/s bandwidth (B/F=0.5)
- 82944 nodes, 6D mesh/torus network
- 10PB local file system, 30PB storage

京 K computer

Notable features of K computer

- Resiliency
  - LINPACK ran during 29hour
  - 160k core x 9hour NICAM simulation x 24 times : we met job failure only once!
  - Why?
    - Fujitsu has a history of mainframe
    - CPU is kept at a lower temperature, strong error detection & retry system
    - Duplicated 3D torus network (ToFu)
    - File stage-in/out system
- Strategy of thread parallelism
  - Fast hardware barrier of the threads, shared L2 cache
  - Strong hardware prefetching
  - Reasonable B/F ratio(0.5), extended floating-point register(256word)
  - Compiler-aided auto parallelization
    - Small overhead of thread folk/join: It's OK to apply at Inner loop

➡ 8 cores work just like 1 CPU



Porting NICAM to K

- **Optimization starts from 2010**
  - Porting from vector machine (Earth Simulator) to scalar machine
  - MPI parallelization was OK
    : NICAM have been designed for massive parallel machine

  - Single node performance efficiency
    - Cache optimization applied to some stencil operators
      : efficiency became up to 10-18% of peak in each kernel
    - But...
      - Total performance efficiency improved only ~1% (of peak)
      - Amdahl's law
      - "Flat profile" : difficult to find hotspots

Porting NICAM to K

- Execution setting on K computer
  - MPI(inter-node) and auto parallelization(intra-node)
    - One process per one node
    - Every process input/output distributed file from/to local file system
- Data structure of NICAM
  - (ij,k,l) = horizontal grid, vertical grid, region
  - Thread parallelization is applied to k-loop

  - We neither use OpenMP nor apply cache blocking
    : Large innermost ij-loop was efficient in the many case

Vector-like thread parallelization of K aided our smooth porting

Stencil kernel optimization of NICAM

```
do l = 1, ADM_lall
do k = ADM_kmin, ADM_kmax
   do n = OPRT_nstart, OPRT_nend
      ij      = n
      ip1j    = n + 1
      ijp1    = n     + ADM_gall_1d
      ip1jp1 = n + 1 + ADM_gall_1d
      im1j    = n - 1
      ijm1    = n     - ADM_gall_1d
      im1jm1 = n - 1 - ADM_gall_1d

      scl(n,k,l) = cdiv(0,n,l,1) * vx(ij    ,k,l) &
                 + cdiv(1,n,l,1) * vx(ip1j  ,k,l) &
                 + cdiv(2,n,l,1) * vx(ip1jp1,k,l) &
                 + cdiv(3,n,l,1) * vx(ijp1  ,k,l) &
                 + cdiv(4,n,l,1) * vx(im1j  ,k,l) &
                 + cdiv(5,n,l,1) * vx(im1jm1,k,l) &
                 + cdiv(6,n,l,1) * vx(ijm1  ,k,l) &
                 + cdiv(0,n,l,2) * vy(ij    ,k,l) &
                 + cdiv(1,n,l,2) * vy(ip1j  ,k,l) &
                 + cdiv(2,n,l,2) * vy(ip1jp1,k,l) &
                 + cdiv(3,n,l,2) * vy(ijp1  ,k,l) &
                 + cdiv(4,n,l,2) * vy(im1j  ,k,l) &
                 + cdiv(5,n,l,2) * vy(im1jm1,k,l) &
                 + cdiv(6,n,l,2) * vy(ijm1  ,k,l) &
                 + cdiv(0,n,l,3) * vz(ij    ,k,l) &
                 + cdiv(1,n,l,3) * vz(ip1j  ,k,l) &
                 + cdiv(2,n,l,3) * vz(ip1jp1,k,l) &
                 + cdiv(3,n,l,3) * vz(ijp1  ,k,l) &
                 + cdiv(4,n,l,3) * vz(im1j  ,k,l) &
                 + cdiv(5,n,l,3) * vz(im1jm1,k,l) &
                 + cdiv(6,n,l,3) * vz(ijm1  ,k,l)
   enddo
enddo
enddo
```

Coefficients:
4th dimension → 1st (=SoA → AoS)
: efficient use of cache line

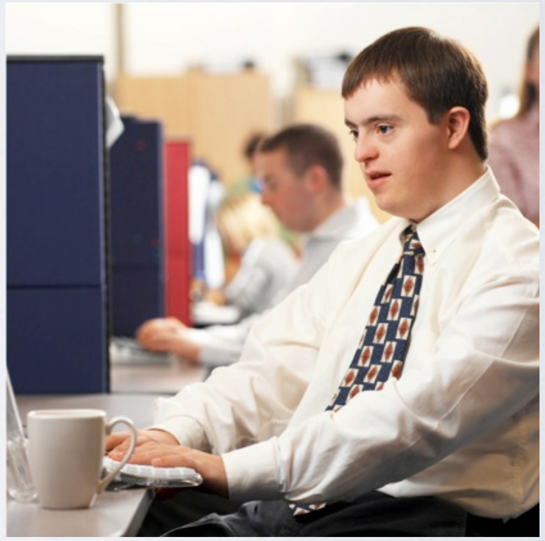
Modification of data structure was very limited. (Only for this case)
K-specific optimization was minor

Where is the problem?

There's no hot-spot...
It's difficult to find the target
in the huge application code !

We don't know
which type of coding
makes slow...

Weather/Climate
scientists & students



Computer
scientists
Optimization staff

Labor-intensive method

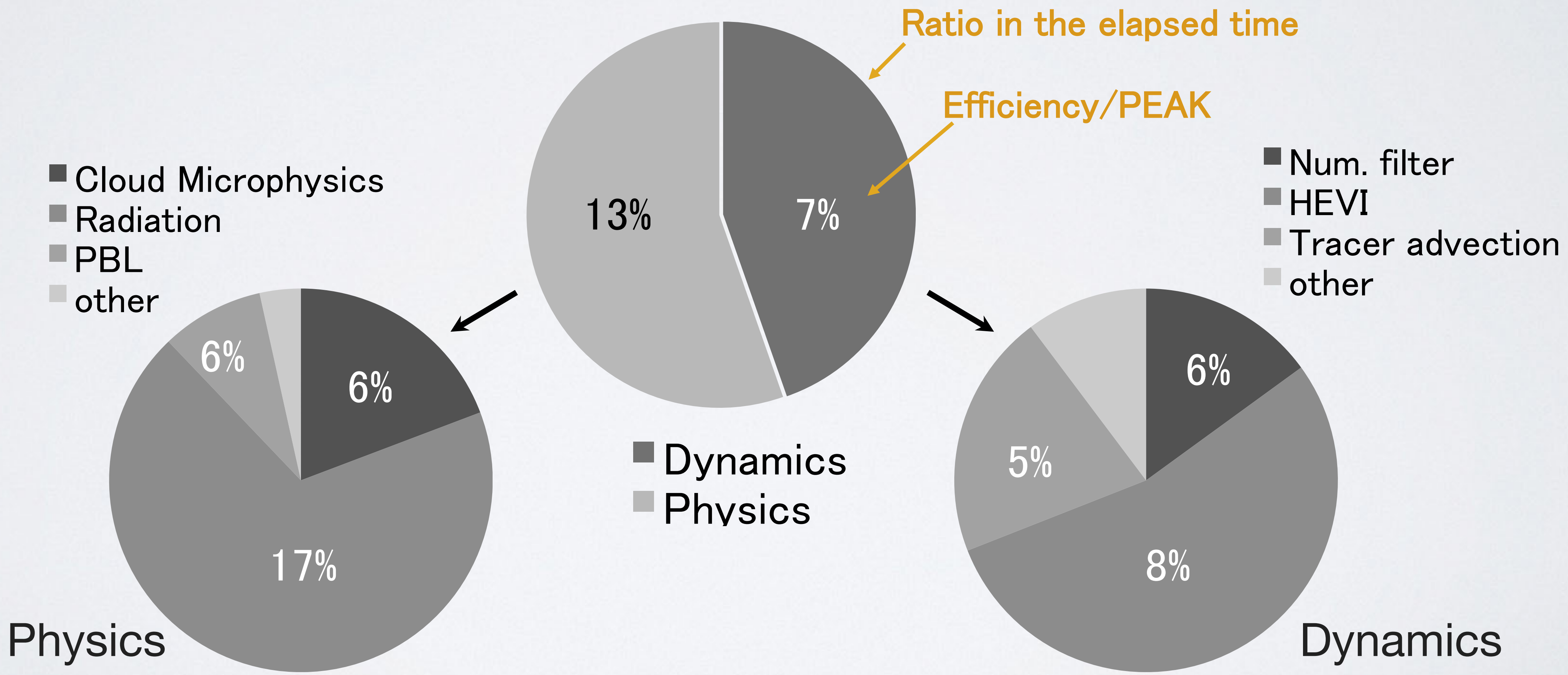
- **Detailed check of performance**
  - Insert FLOP & time counter into the hundreds of sections
  - List up the "time-wasting" sections
    - Less computation, more memory transfer
      : mainly related to array copy, array initialization, tentative array generation in the subroutine call
    - The loops which the compiler gives up
      : “if” branch in innermost loop, complex loop structure

  ➡ Reduce intermediate arrays, avoid unnecessary zero-filling, and turn the conditional branches out of loops

- Our lesson: the compiler is not so intelligent as we have expected.
  - Complier-friendly code and readable code are compatible: simple is **best**

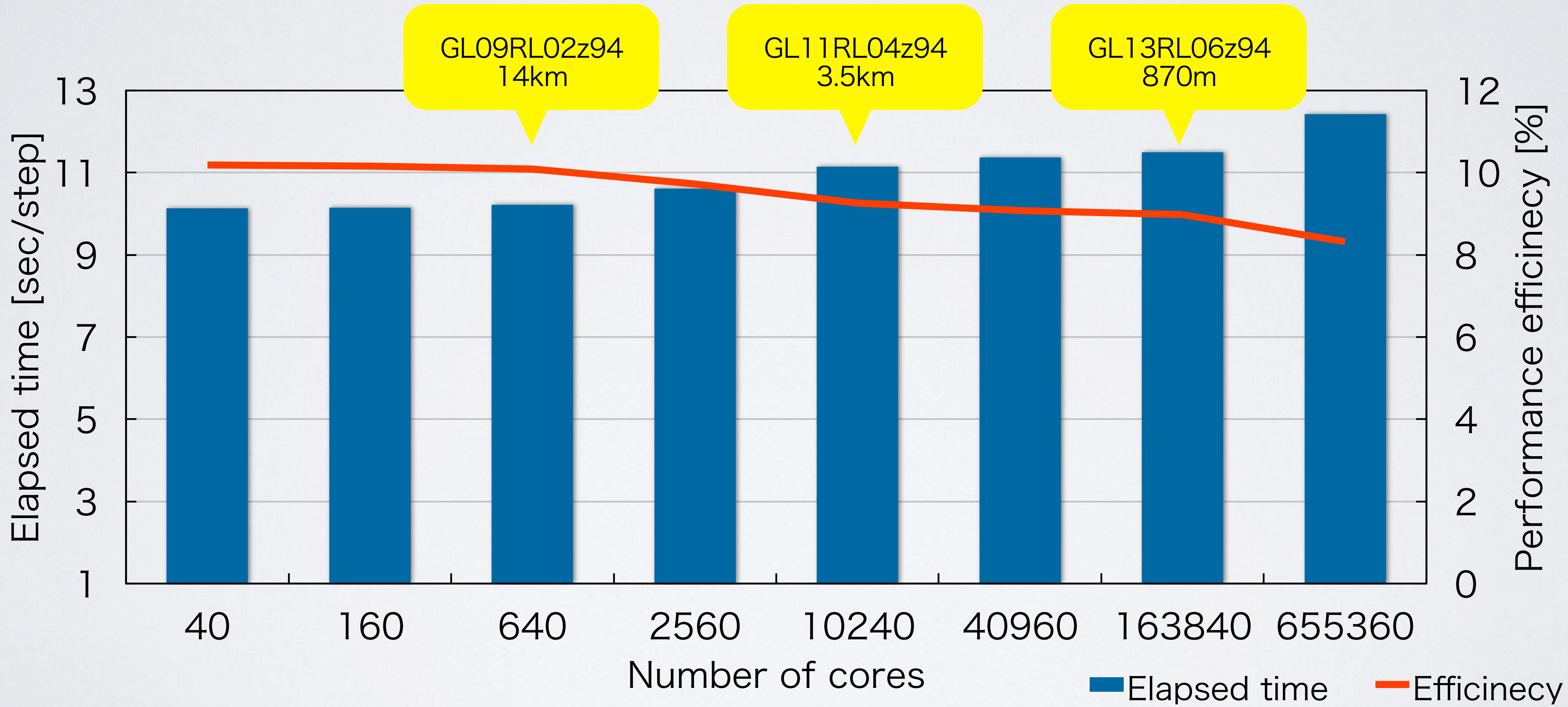
Efficiency of NICAM on K Computer

- Performance efficiency (per peak)
  - Just after porting from Earth Simulator : ~4%
  - Cache optimization to stencil operators : ~5%
  - Cleaning the time-wasting codes : ~7%
  - Modify conditional branches, refactoring : ~10%



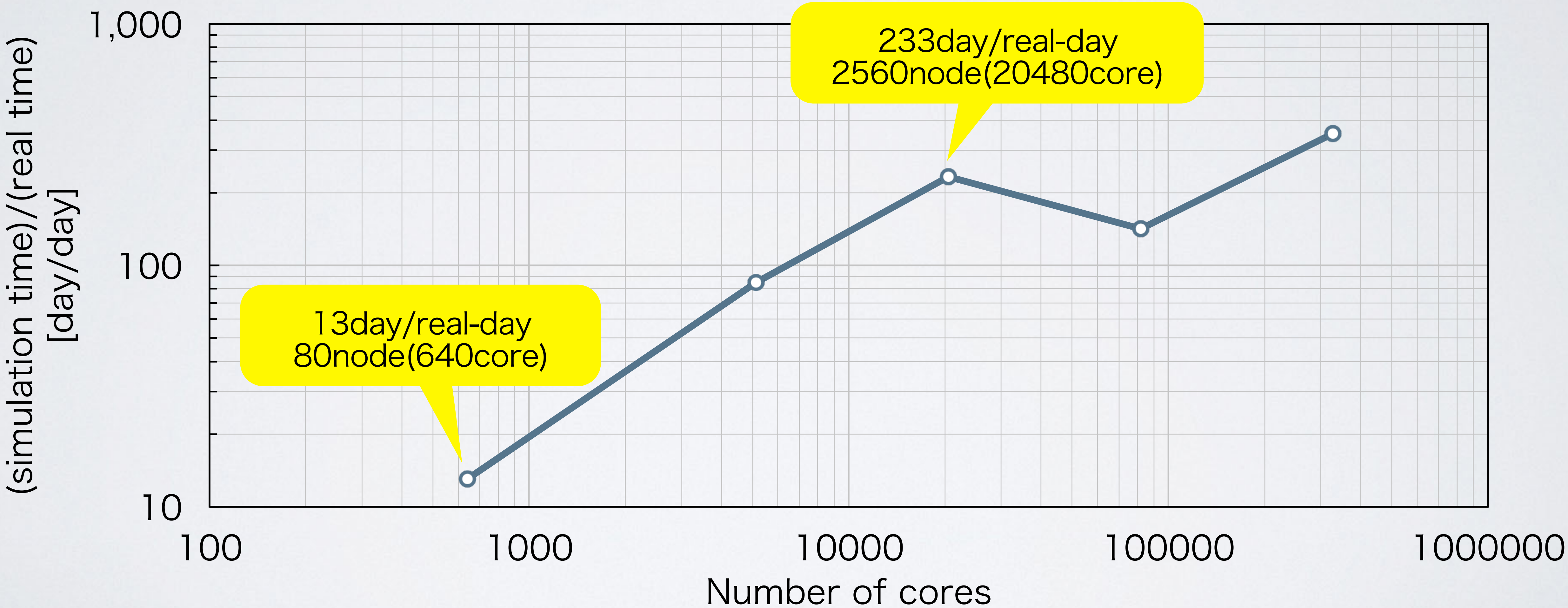
Weak scaling test

- Same problem size per node, same steps
  - Full configuration / full components
  - Realistic boundary / initial data set
- Good scalability up to 81920node x 8threads with **0.9PFLOPS**



Strong scaling test

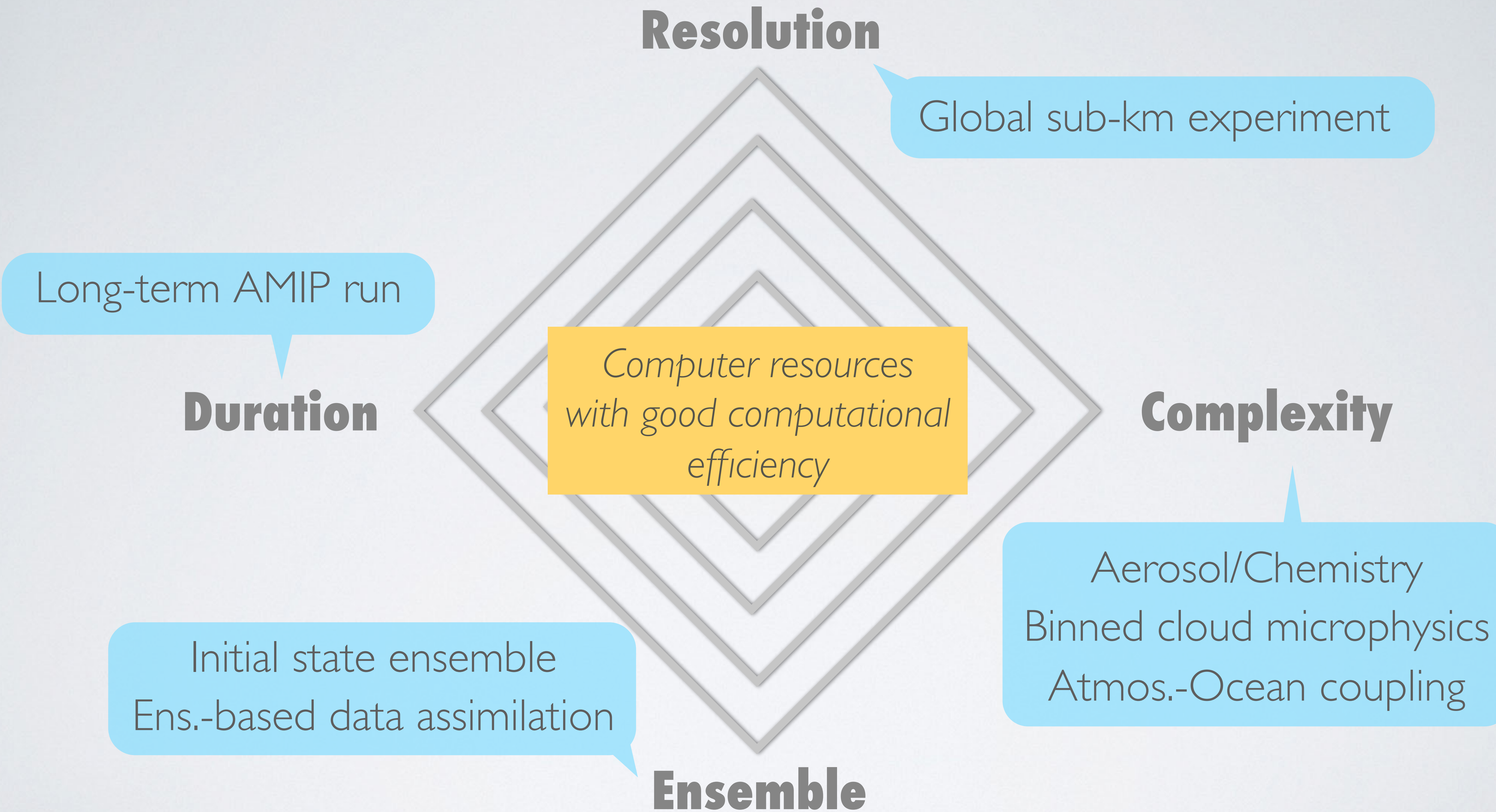
- 14km horizontal, 38layers, total problem size is fixed
- The efficiency decreases rapidly
  : the relative ratio of the communication time increases



Additional topics

- **Other features of NICAM which contributed to performance**
  - **Simple, less memory**
    - Hexagonal A-grid structure for horizontal
      : Less working array and less operation for averaging values
    - Structured: continuous memory access
  - **Distrubuted file I/O**
    - Maximize file I/O throughput
    - But…
      - Number of files are increasing and increasing!
      - Post-process work takes long time
    - MPI-IO w/ compression is better choice in the future?

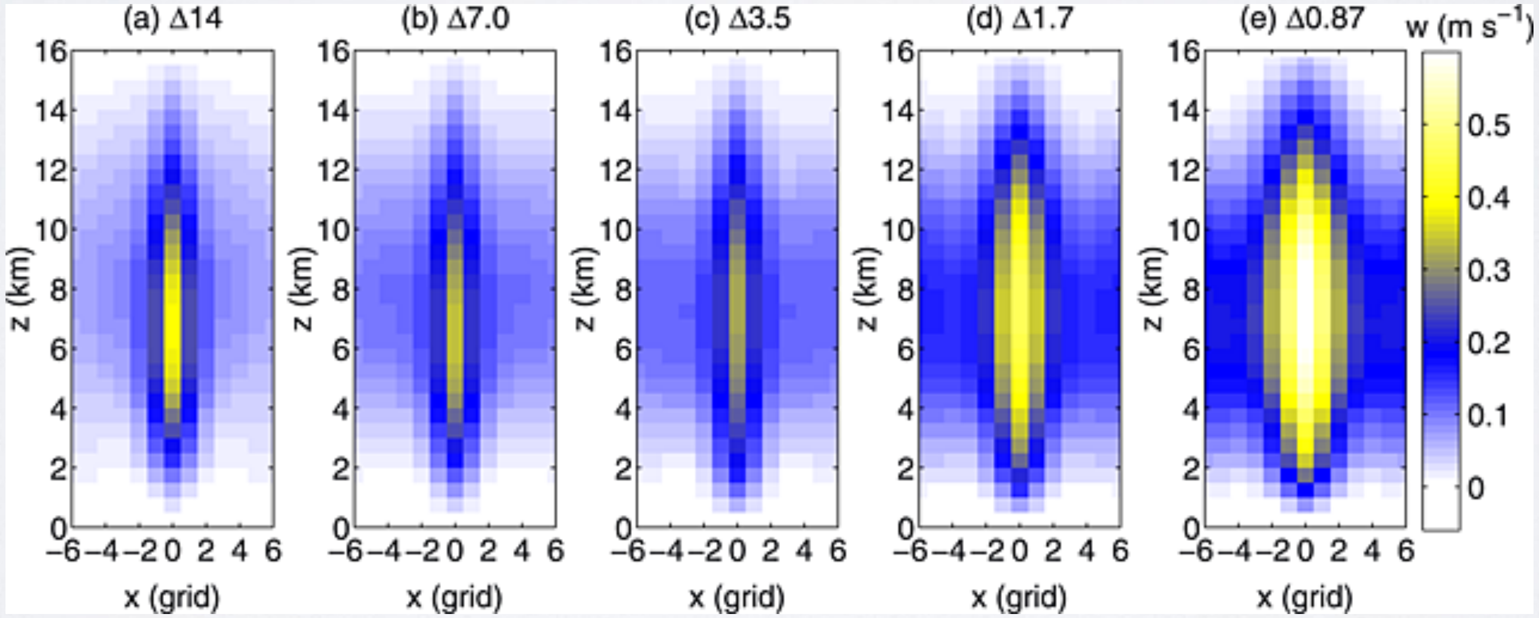
Science target & Model development



Global Sub-km experiment

Y.Miyamoto(RIKEN/AICS)
Y.Kajikawa(RIKEN/AICS)
R.yoshida(RIKEN/AICS)
T.yamaura(RIKEN/AICS)

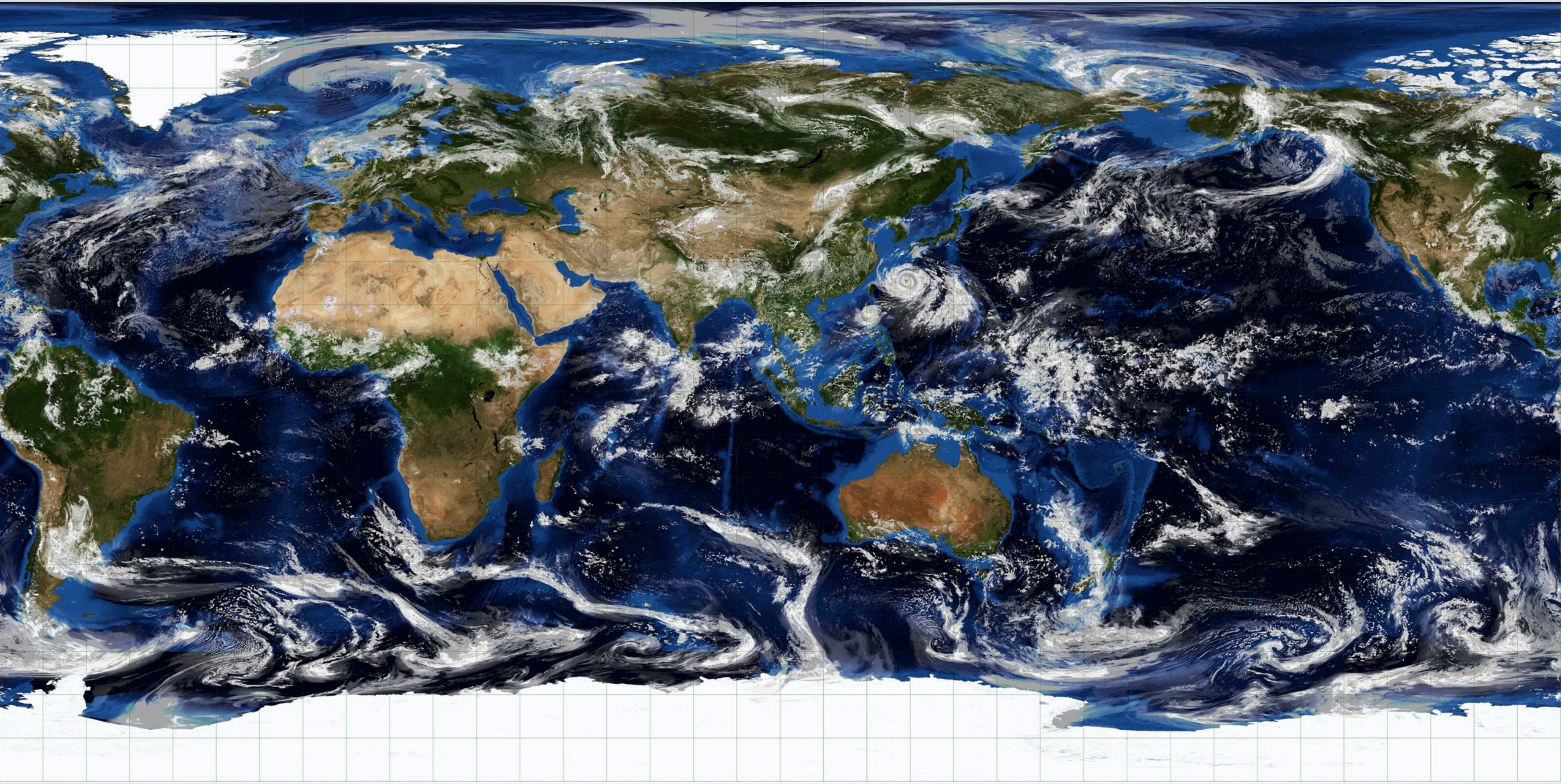
- dx=870m, 97layers, 20480PE
  - 63billion grids, dt=2sec
  - 24hour simulation = 700EFLOP
  - 4.5hour for 1hour simulation
  - 8TB for restart file, total output was 160TB for 24hour simulation
- Composite of convection (vertical velocity)
  - Δx≦1.7km : Convection is represented at multiple grids



(Miyamoto et al. 2013, GRL)

Global Sub-km experiment

Movie by R.Yoshida(RIKEN/AICS)



Japanese post-K (exa-scale) project

- RIKEN is selected to develop an exa-scale supercomputer by 2020.
- **Feasibility study (2012-2013)**
  - 3 architectures
    - Vector machine
    - "K computer"-like machine
    - Accelerators
  - Application side
    - Scientific roadmap to the exa-scale
    - Provide benchmarks and mini-apps

NICAM-DC

R.yoshida(RIKEN/AICS)

- Dynamical core package of NICAM
  - BSD 2-clause licence
  - From website (http://scale.aics.riken.jp/nicamdc/) or GitHub
  - Basic test cases are prepared
- Application
  - G8 ICOMEX project : scientfic/computational performance evaluation
  - Feasibility studies for Japanese post-K supercomputer

Acceleration by GPU

A.Naruse(NVIDIA)
N.Maruyama(RIKEN AICS)

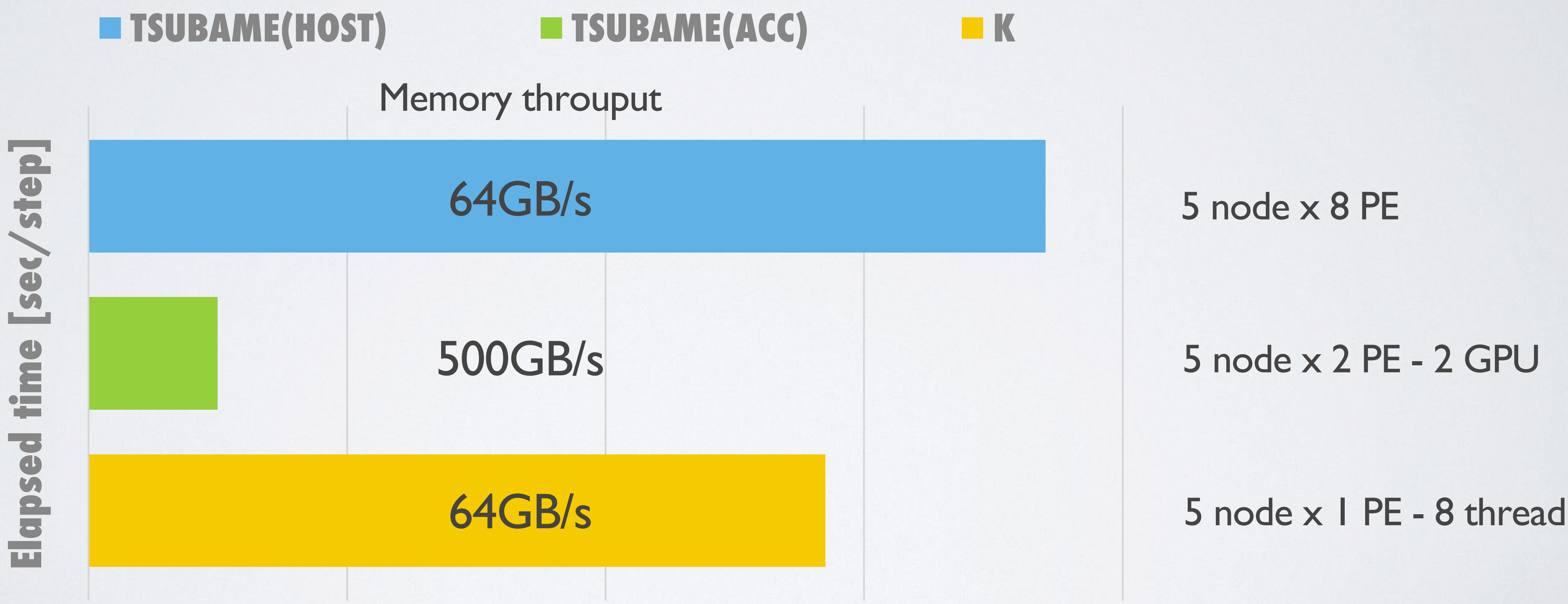
- GPU programming using OpenACC
  - NICAM has 600K lines of source code
  - Active development by researchers and students
    : Most of them are not familiar to the GPU programming
  - We do not want to split the source code (if possible)

  - Optimization by OpenACC experts
    - NICAM-DC was used for testbed : whole dynamical core is ready
    - AoS is changed to SoA again
    - Reduce GPU-to-Host transfer for MPI communication as possible

Acceleration by GPU

- Preliminary results
  - 56km horizontal, 160layers
  - TSUBAME2.5(Tokyo Tec.) and K computer
  - Weak scaling test (56km-3.5km) results are also good

A.Naruse(NVIDIA)
N.Maruyama(RIKEN AICS)



Summary

- NICAM starts shifting to the peta-scale era
  - Good efficiency and scalability enlarge research field of GCSRM
  - Global sub-km simulation study
- Towards to the next generation computer…
  - Keep SIMPLE !
- Ongoing effort
  - Massive incorporation of components : NICAM-ESM
  - OpenACC for physics component