

Large-scale Simulation of the Global SWEs on Hybrid CPU-GPU Platforms

Lin Gan

Computer Science & Technology
Center for Earth System Science

Tsinghua University, China

Email: l-gan11@mails.tsinghua.edu.cn

Tel: +86-15810537053



Sep 13, 2012 @ NCAR, Boulder, US

ISCAS

About us

- ✧ A team work over the last 5 months:
 - Chao Yang (*Institute of Software Chinese Academy of Sciences*)
 - Lin Gan, Wei Xue, Haohuan Fu, Yangtong Xu (*Center for Earth System Science, Tsinghua University*)
- ✧ Achievements:
 - Peta-scalable global SWE simulation, 809TFlops in double precision in 3750 nodes(45,000 CPU cores & 52,500 GPU cores)
 - Adjustable partition between CPUs/GPUs
 - Effective comm.-comp. overlap to hide comm. cost
 - “Pipe-flow” scheme to arrange message-passing
 - Systematic optimizations on CPU and CUDA code, 130 speedup



Contents



Background



Mesh & Equations



Algorithm & Implementations



Large-scale results

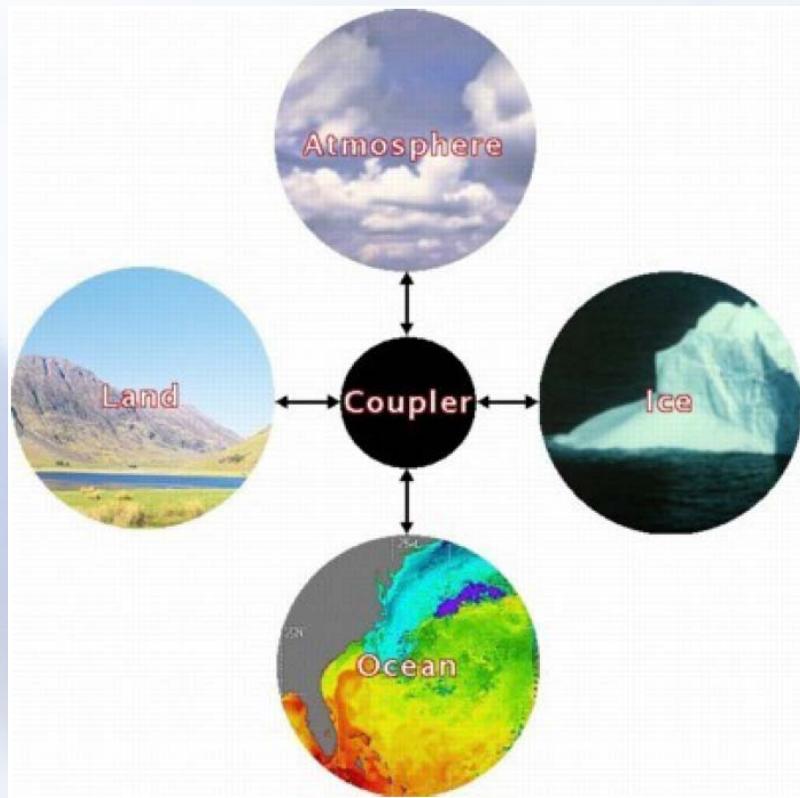


Conclusions



Background

- ❖ Global atmospheric modeling is a key component in climate simulation.



(Courtesy: Mark Taylor et al., 2009)

- Land model: petascale-ready
- Ice model: petascale-ready
- Ocean model: nearly petascale-ready
- Atmosphere model: bottleneck

Note: petascale-ready means scalable to $O(100K)$ processors at target climate resolution (<10km)



Background

❖ Rapid development of heterogeneous supercomputers

- On the recent Top500 list (Jun 2012)
 - #5:Tianhe-1A, 4.7Pflops, 86K CPU-cores + 100K GPU-cores
 - #10:Nebulae, 2.98Pflops, 55K CPU-cores + 65K GPU-cores
- Coming soon
 - Titan, successor to Jaguar, ORNL
 - Blue Waters, NCSA

❖ Successes in hybrid CPU-GPU algorithms

- N-body simulations (Hamada et al. SC'09, GB winner)
- NWP simulations (Takashi Shimokawabe et al. SC'10)
- Biofluidics simulations (Bernaschi et al. SC'11, GB final list)
- Phase-field simulations (Shimokawabe et al. SC'11, GB winner)

Efficient hybrid algorithms in global atmospheric modeling



Contents



Background



Mesh & Equations



Algorithm & Implementations



Large-scale results

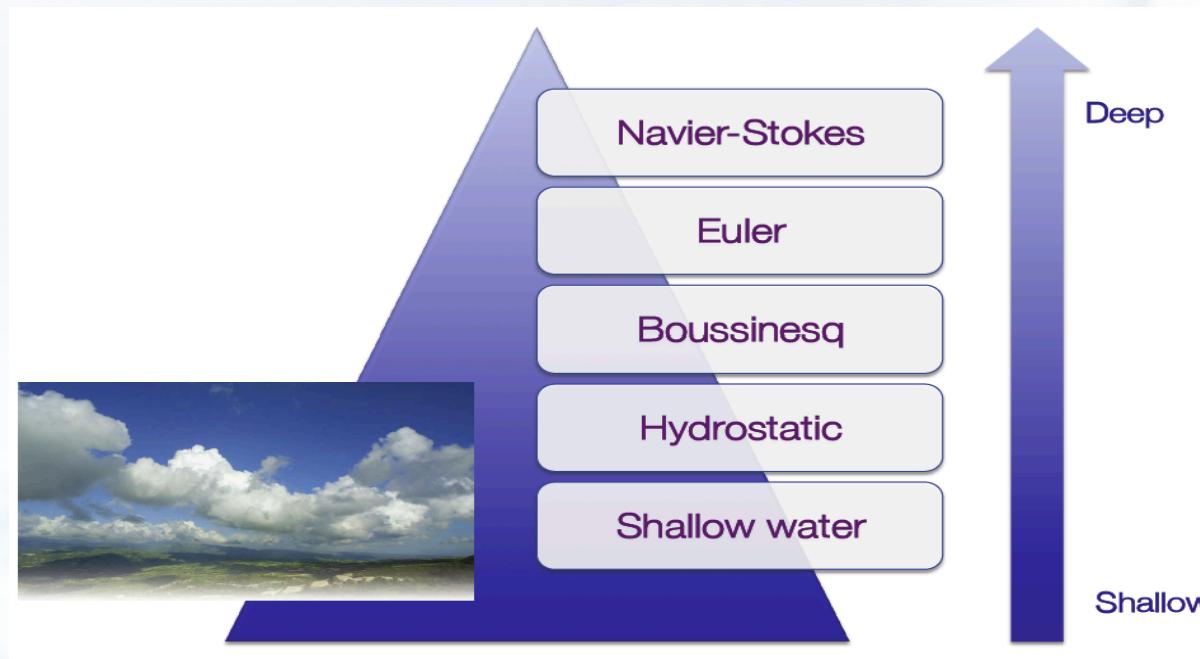


Conclusions



Mesh & Equations

❖ Equations: Shallow Water Equations (SWEs)



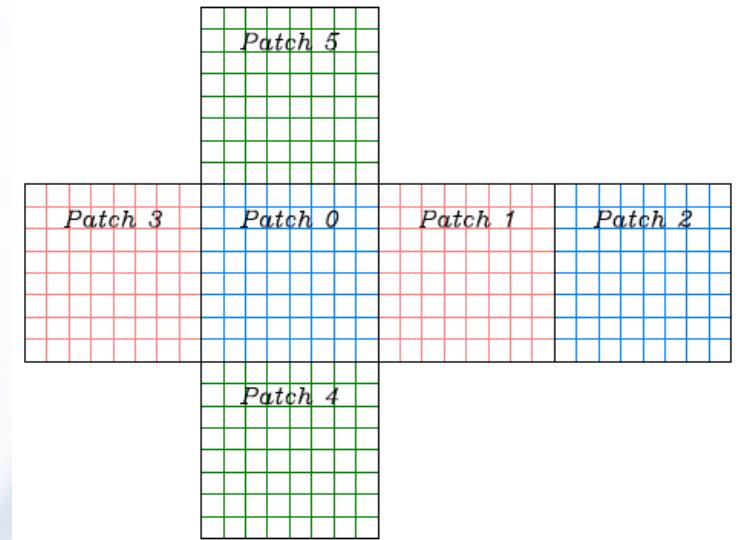
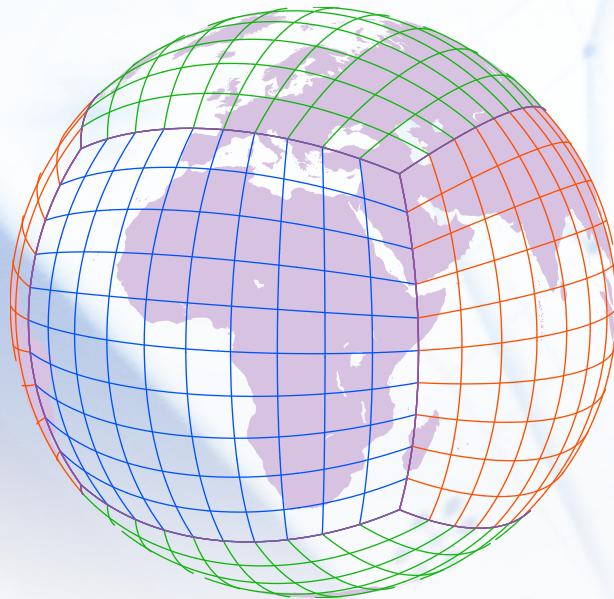
- SWEs exhibit most basic horizontal dynamics of the global atms

$$\begin{cases} \frac{\partial h}{\partial t} + \nabla \cdot (hv) = 0, \\ \frac{\partial hv}{\partial t} + \nabla \cdot (hv \otimes v) + gh\nabla(h + b) + fh\hat{k} \times v = 0 \end{cases}$$



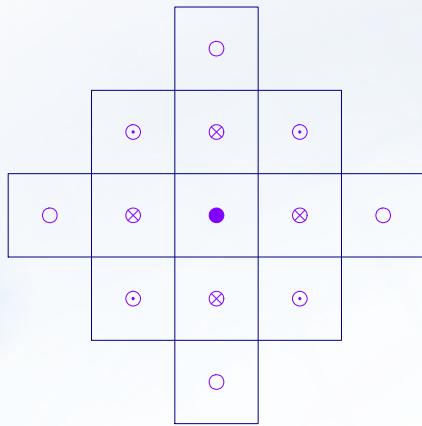
Mesh & Equations

- ✧ Cubed-sphere mesh
 - Mapping inscribed cube to the surface of the earth
- ✧ Computational domain
 - Six patches covered with rectangular meshes



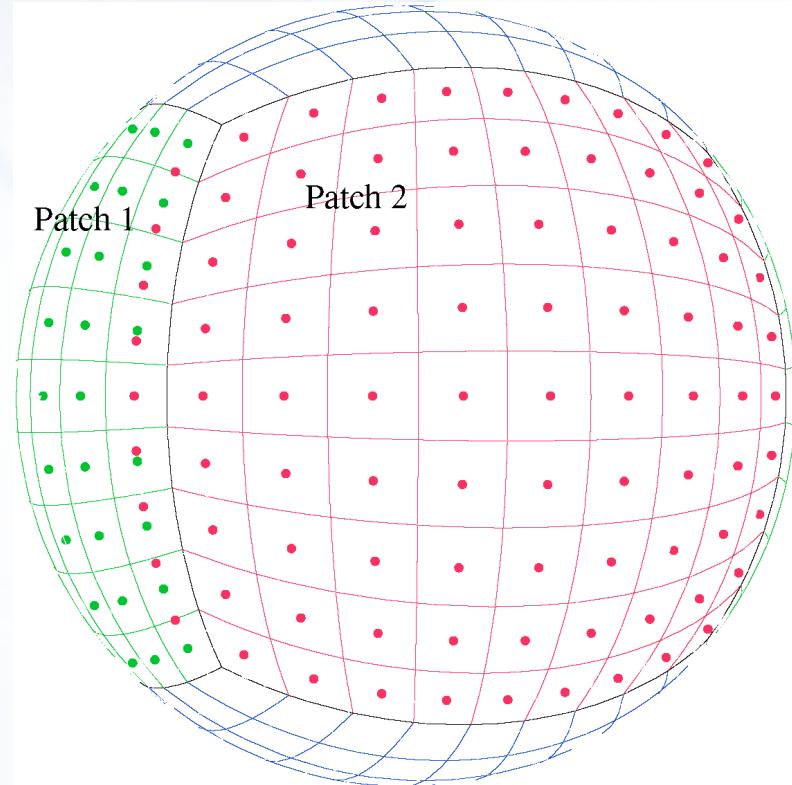
Mesh & Equations

- ❖ 13-point stencil



- ❖ Interp. Across patches

- 1-d linear interpolation



- ❖ Discretize using cell-centered finite volume scheme
- ❖ Intergrade using second order TVD Runge-Kutta method



Contents



Background



Mesh & Equations



Algorithm & Implementations



Large-scale results



Conclusions



Algorithms & Implementations

In this part:

- ✧ Tianhe-1A supercomputers
- ✧ Hybrid algorithm
- ✧ Pipe-flow scheme
- ✧ Systematic optimization



Algorithms & Implementations

✧ The Tianhe-1A

- TH-net: fast proprietary network
- 7168 computing nodes, 4.7PFlops Rpeak (#5 in Top500, 2012)
- In each computing node
 - Two 6-core Intel X5670 CPUs: 12 cores, 140Mflops
 - One NVIDIA M2050 GPU: 14 cores (448 CUDA cores), 515Mflops



NSCC-Tianjin



The Tianhe-1A Supercomputer

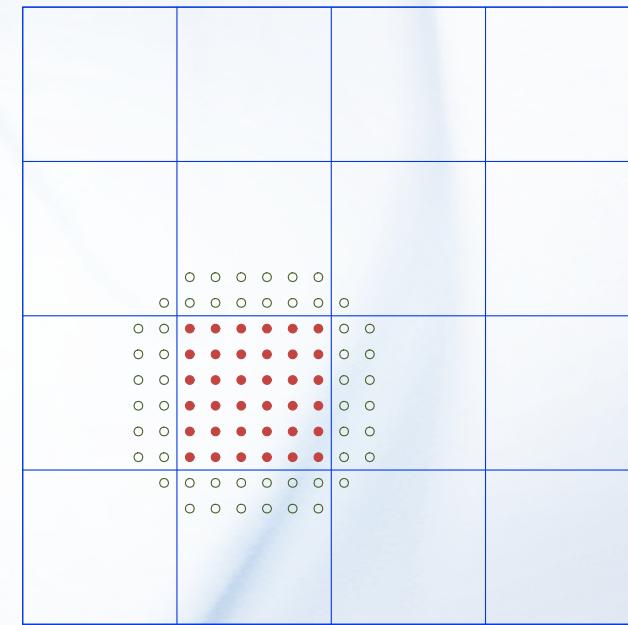


Algorithms & Implementations

✧ Domain decomposition of the cubed-sphere

- 6 MPI groups for 6 patches
- Divide each patch into $N \times N$ sub-blocks (here 4×4)

				<i>Patch 5</i> MPI-group 5
<i>Patch 3</i> MPI-group 3	<i>Patch 0</i> MPI-group 0	<i>Patch 1</i> MPI-group 1	<i>Patch 2</i> MPI-group 2	
				<i>Patch 4</i> MPI-group 4

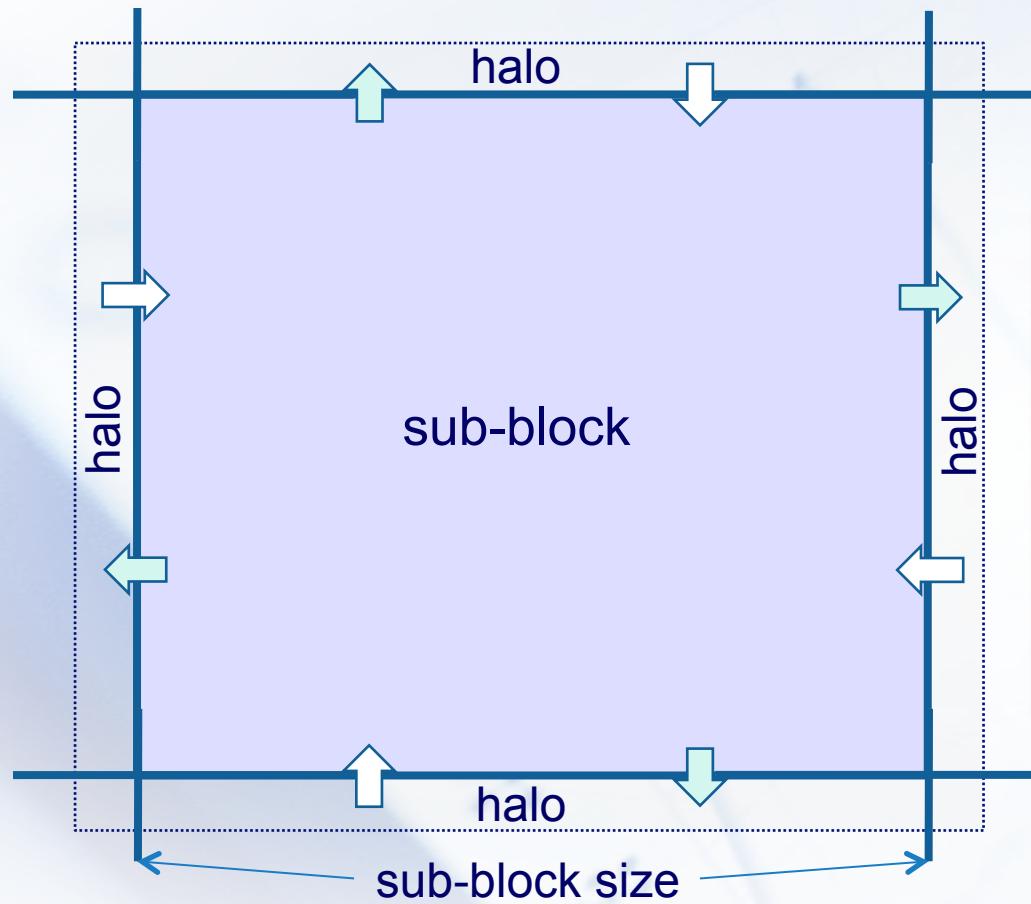


Mesh points (filled) and halos (empty)
for a sub-block



Algorithms & Implementations

✧ CPU-only algorithm

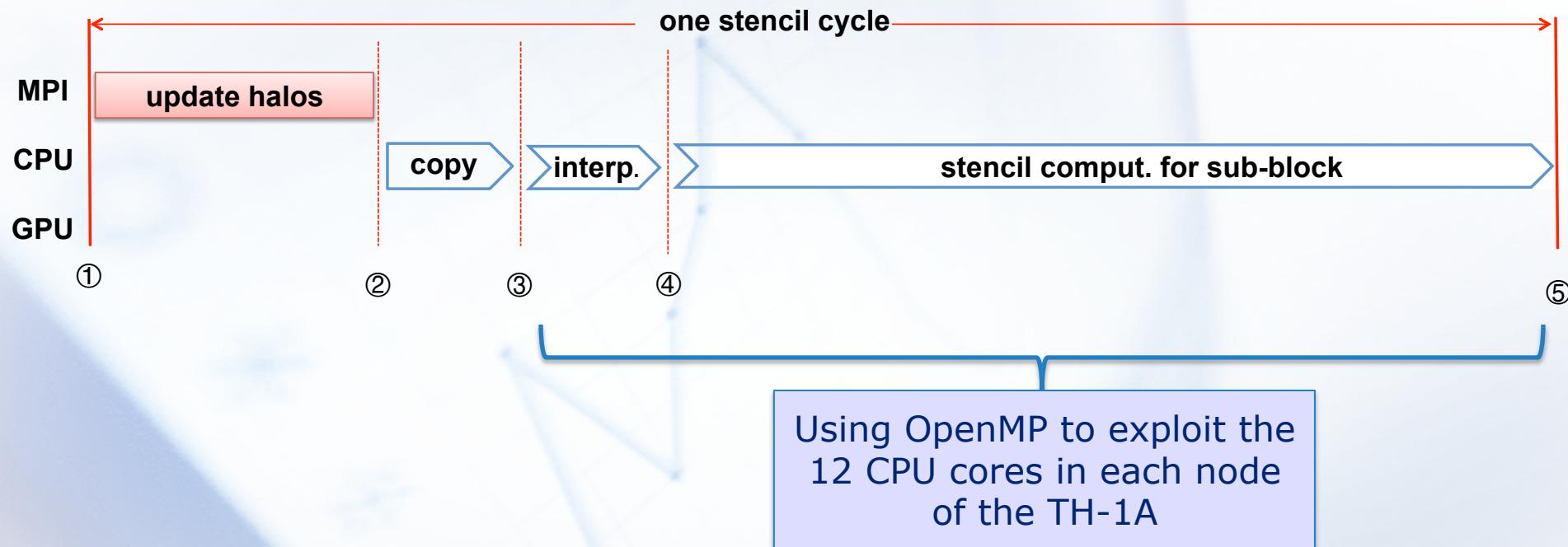


- For each stencil cycle
- ① Update halos
 - ② Prepare buffer copy
 - ③ Interpolate ghost cells if necessary
 - ④ Calculate stencils



Algorithms & Implementations

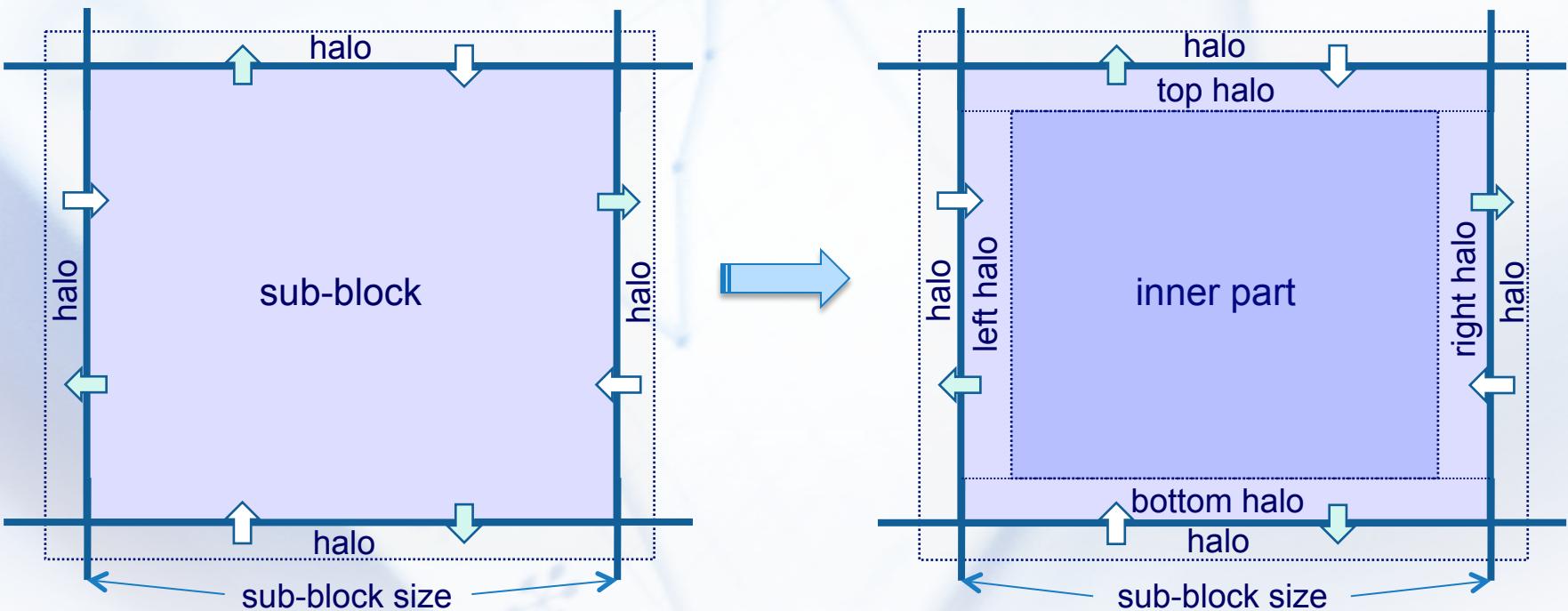
✧ CPU-only algorithm: work flow



Algorithms & Implementations

❖ Hybrid CPU-GPU algorithm

- Divide each sub-block into
 - Outer part: 2 layers of halos for neighbors → CPU computing
 - Inner part: without halo exchanging → GPU computing



(Courtesy: J. Michalakes et al., SC11)



Algorithms & Implementations

❖ Hybrid CPU-GPU algorithm

- Divide each sub-block into
 - Outer part: 2 layers of halos for neighbors → CPU computing
 - Inner part: without halo exchanging → GPU computing

For each stencil cycle

GPU side:

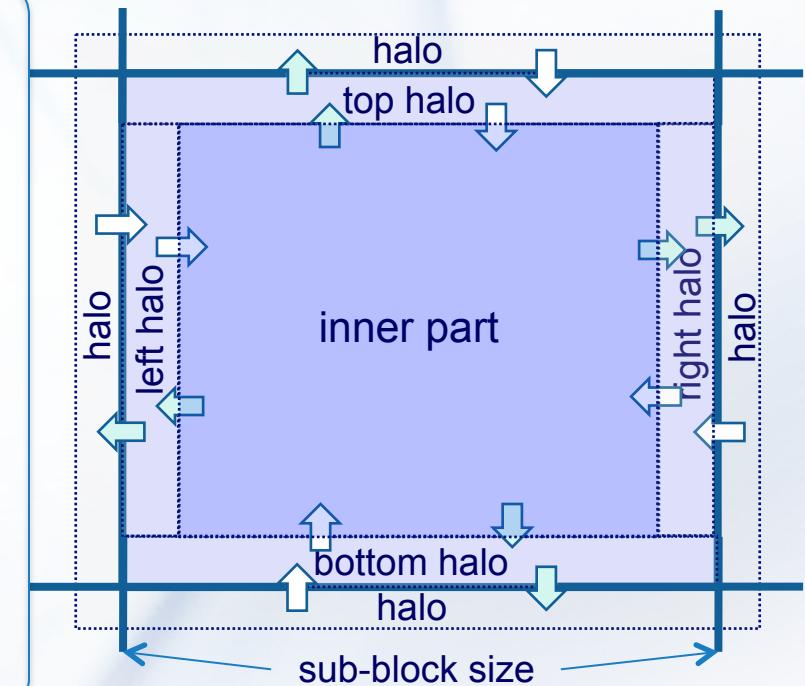
- ① Calculate stencils in inner part

CPU side:

- ① Update halos
- ② Prepare buffer copy
- ③ Interpolate ghost cells if necessary
- ④ Calculate stencils in outer part

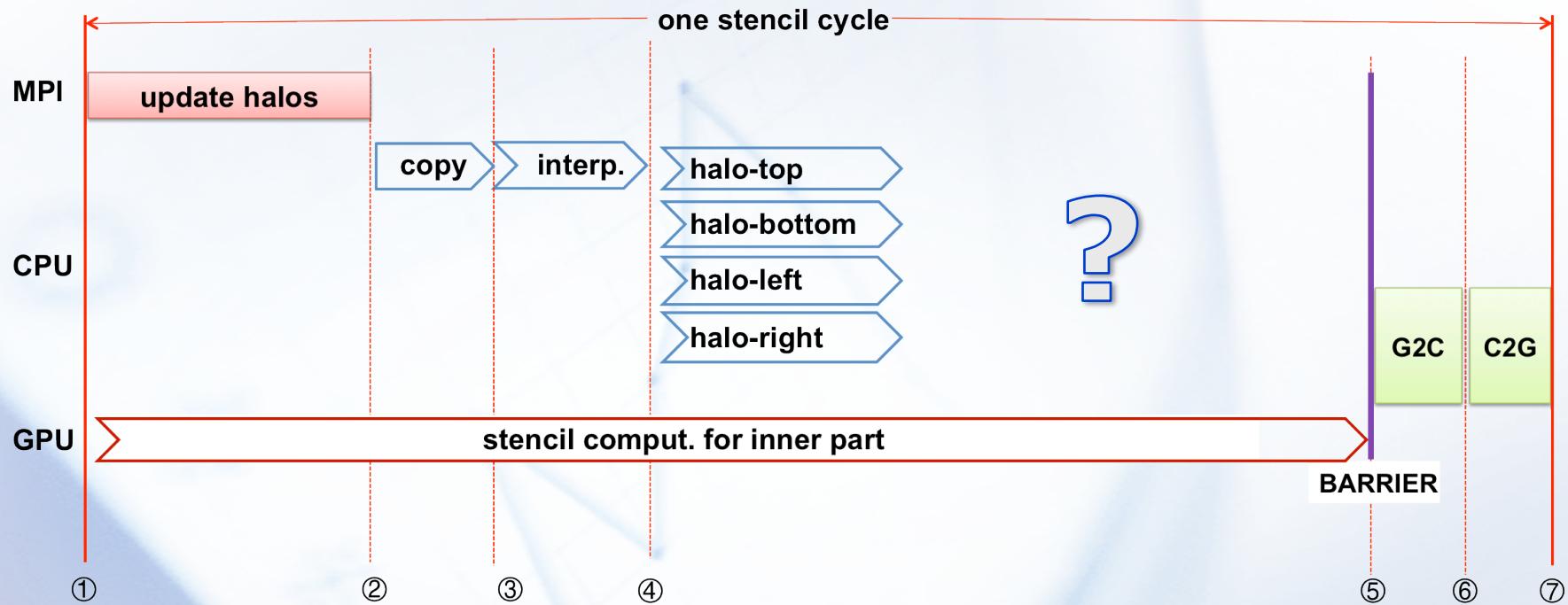
BARRIER:

- ⑤ Exchange data between CPU/GPU



Algorithms & Implementations

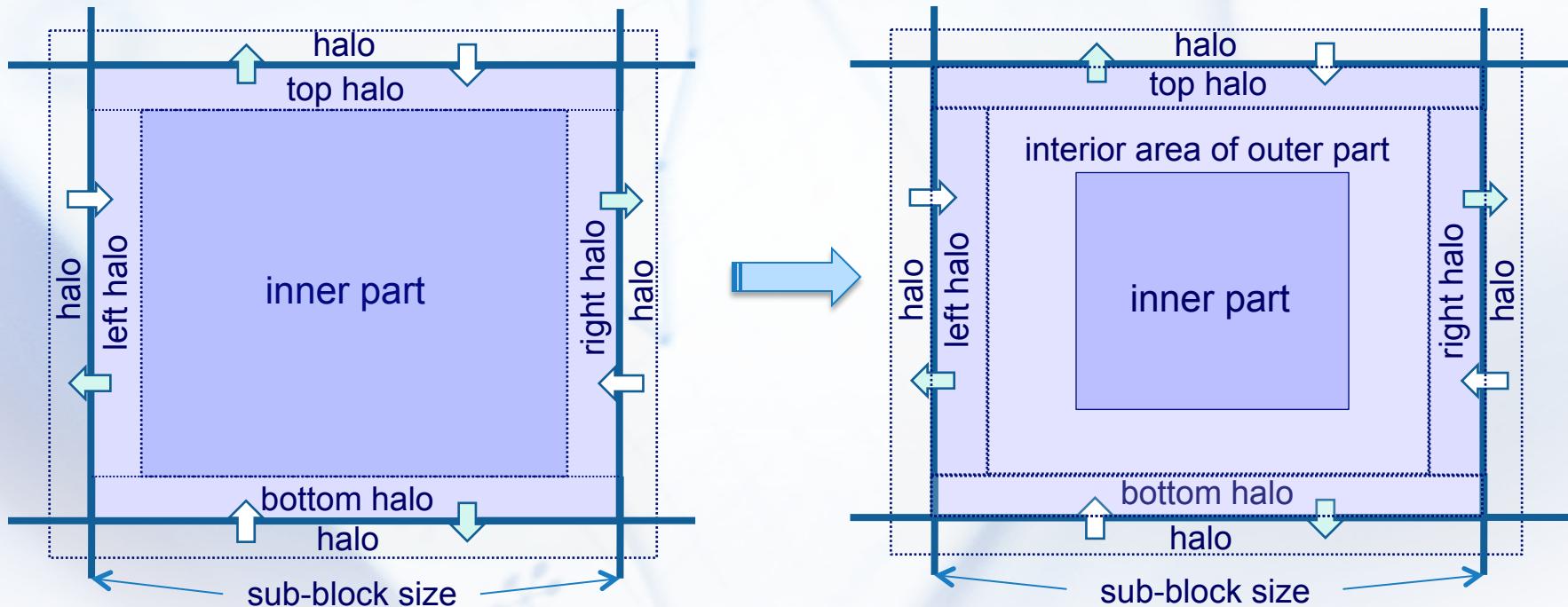
❖ Hybrid CPU-GPU algorithm: work flow



Algorithms & Implementations

❖ Adjustable partition CPU-GPU algorithm

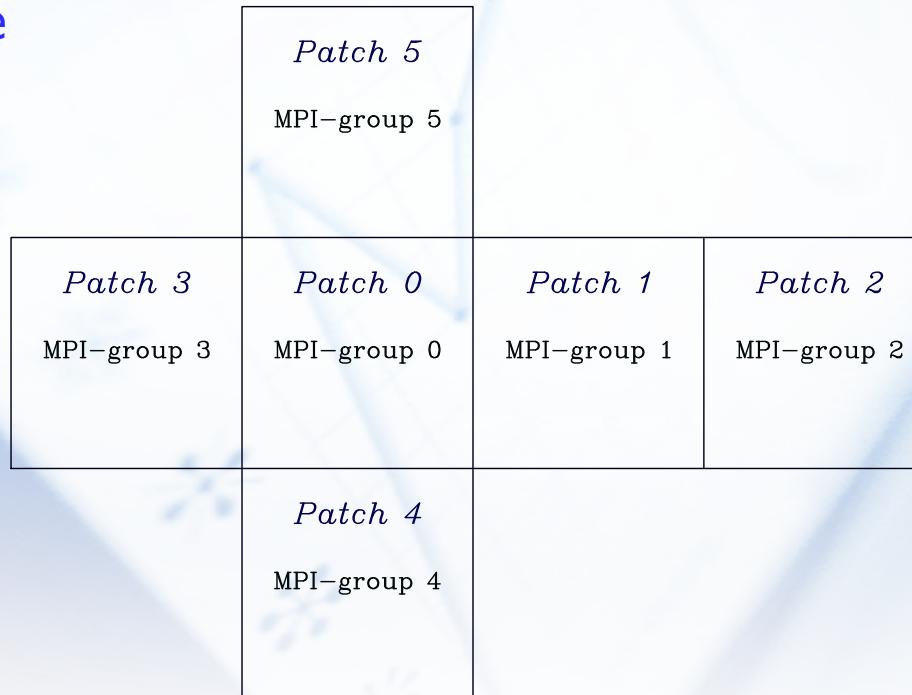
- Divide each sub-block into
 - Outer part: n layers of points → CPU computing
 - Inner part: without halo exchanging → GPU computing



Algorithms & Implementations

❖ Hybrid CPU-GPU algorithm: improvement

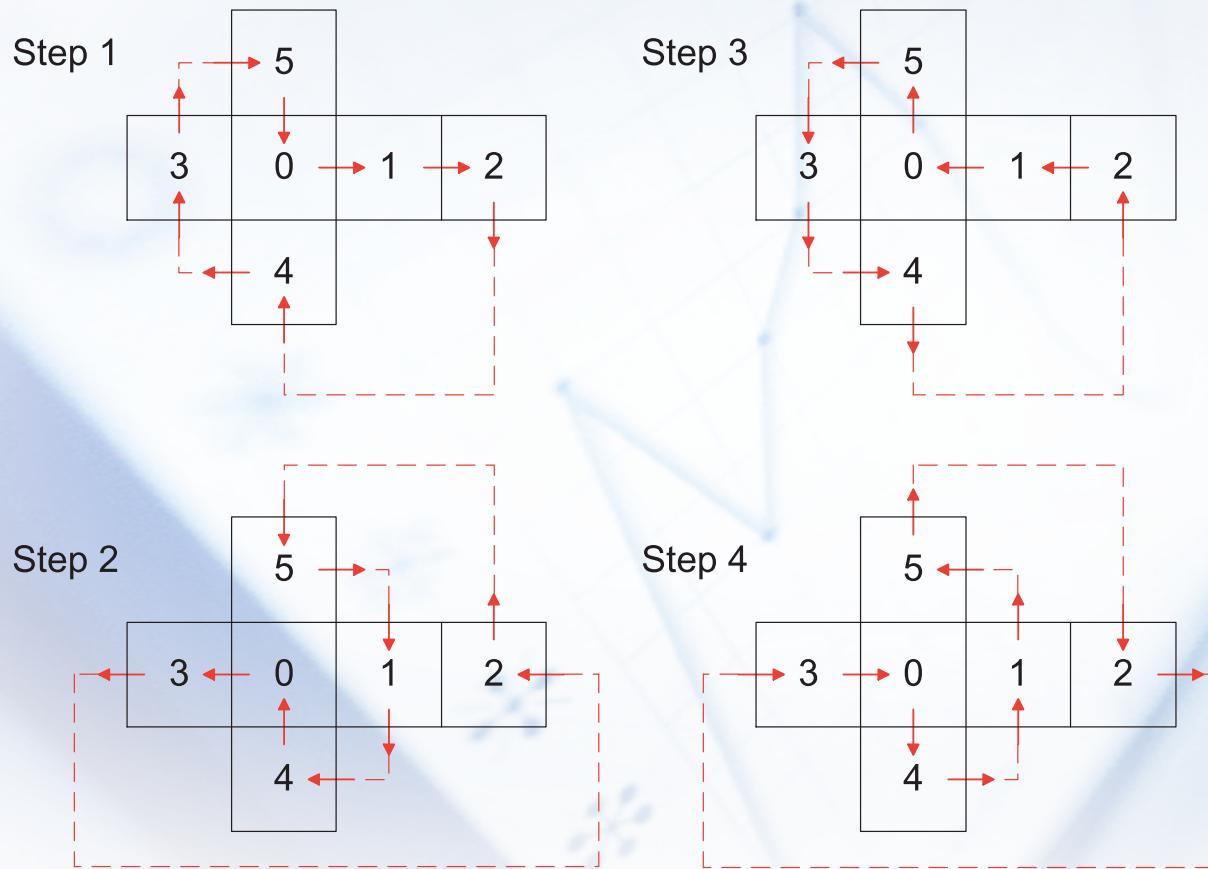
- Communication (halo updating) degrades CPU flops efficiency
 - Strategy for communication-computation overlap
- Imbalanced message-passing on the cubed-sphere
 - e.g. Patch 0,1,2,3 might send top halos to Patch 5 at the same time



Algorithms & Implementations

❖ Improved hybrid CPU-GPU algorithm

- “pipe-flow” scheme for message-passing on cubed-sphere



Four steps to arrange conflict-free message-passing on cubed-sphere.

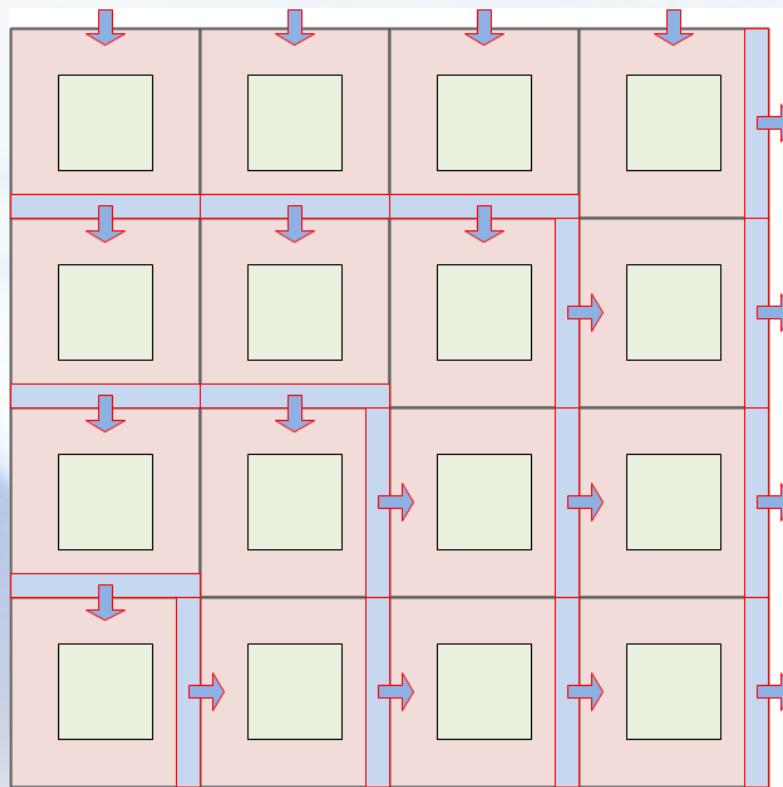
The arrows indicate directions of data entering or exiting patches as a pipe flow.



Algorithms & Implementations

❖ Improved hybrid CPU-GPU algorithm

- “pipe-flow” scheme for message-passing on cubed-sphere



“pipe-flow” pattern for
the 4*4 sub-blocks in a patch

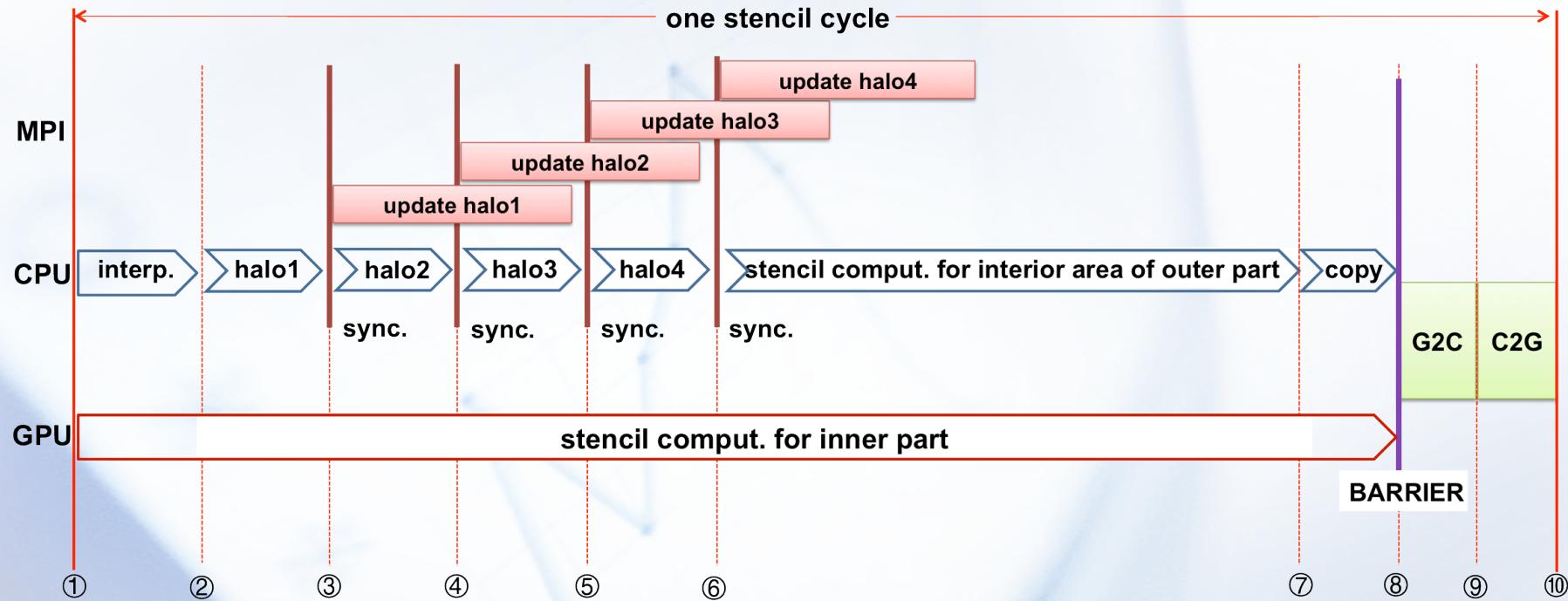
Directions of the “pipe-flow” inside a particular patch of the cubed-sphere.

The square shaded regions represent the inner parts of sub-blocks.

The narrow shade regions represent the halos to be sent.

Algorithms & Implementations

❖ Improved hybrid CPU-GPU algorithm: work flow



Note: halo1/2/3/4 — the 4 steps of the “pipe-flow” communication scheme
adjustable partition between CPU and GPU

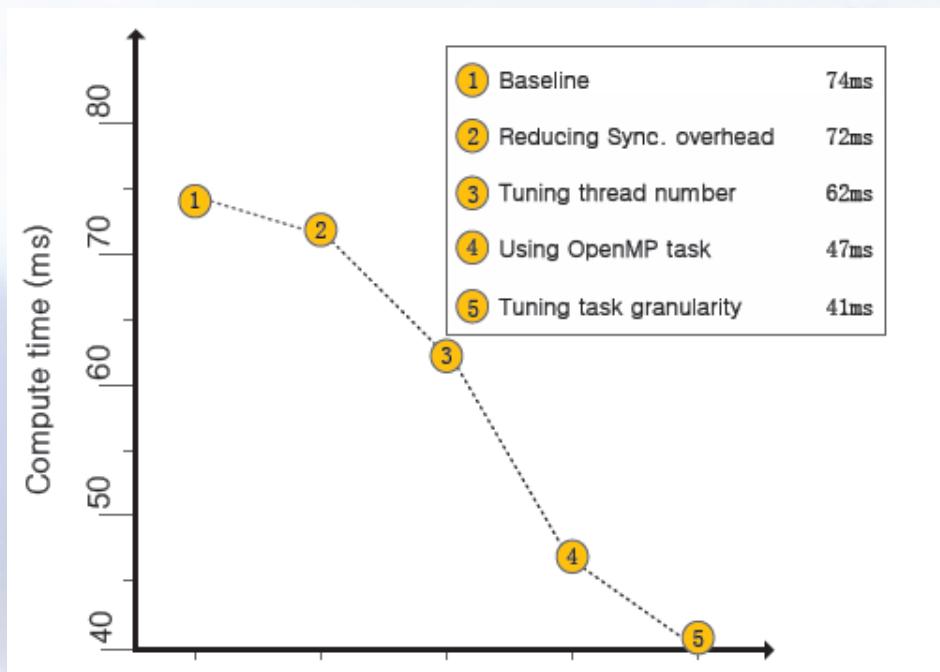


Algorithms & Implementations

❖ CPU optimizations

Mesh size: 1156*1156

- ① Baseline
- ② OpenMP *nowait* clause
- ③ Tuning number of threads to twice the CPU cores
- ④ OpenMP task construct
- ⑤ Tuning the best task granularity

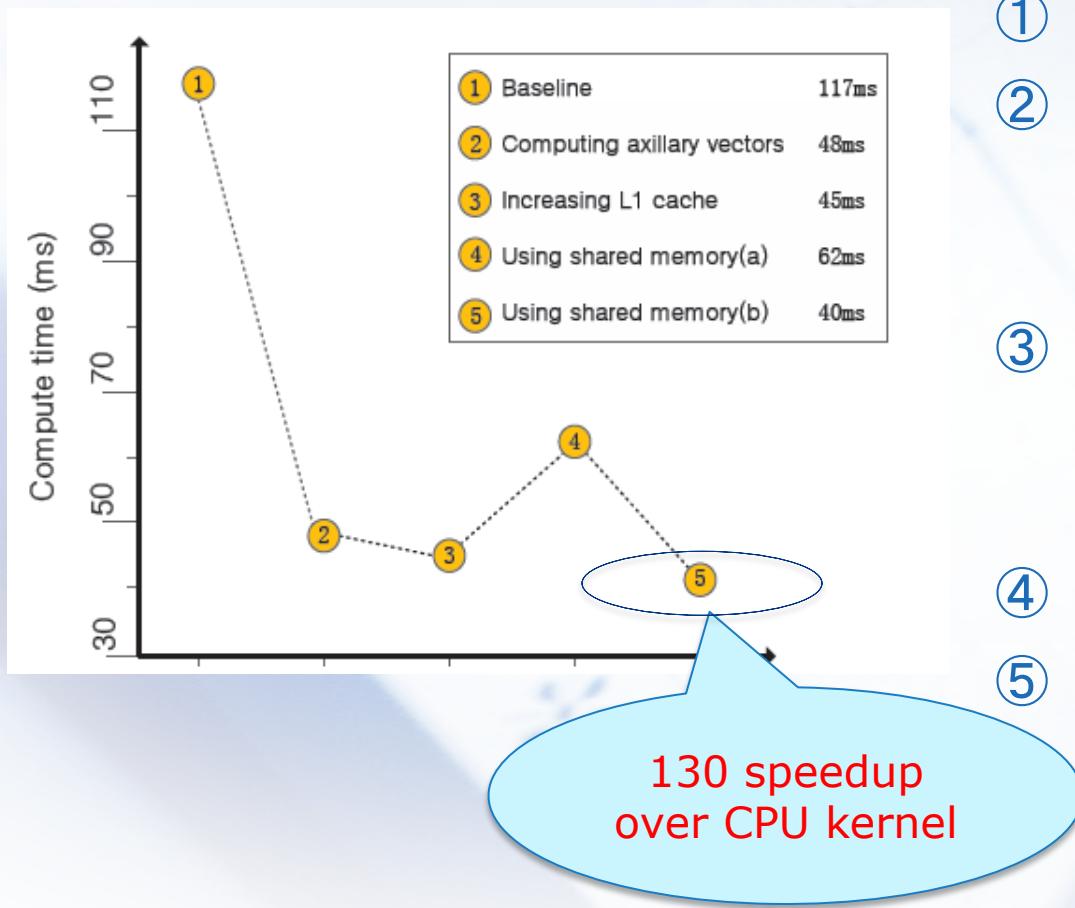


Algorithms & Implementations

✧ GPU optimizations

Mesh size: 1024*1024

- ① Baseline
- ② Computing instead of reading pre-computed axillary vectors (e.g., tensors)
- ③ Increasing L1 cache from 16KB to 48KB, without using the shared memory (SM)
- ④ 48KB L1 cache + 16KB SM
- ⑤ 16KB L1 cache + 48KB SM



Contents



Background



Mesh & Equations



Algorithm & Implementations



Large-scale tests



Conclusions

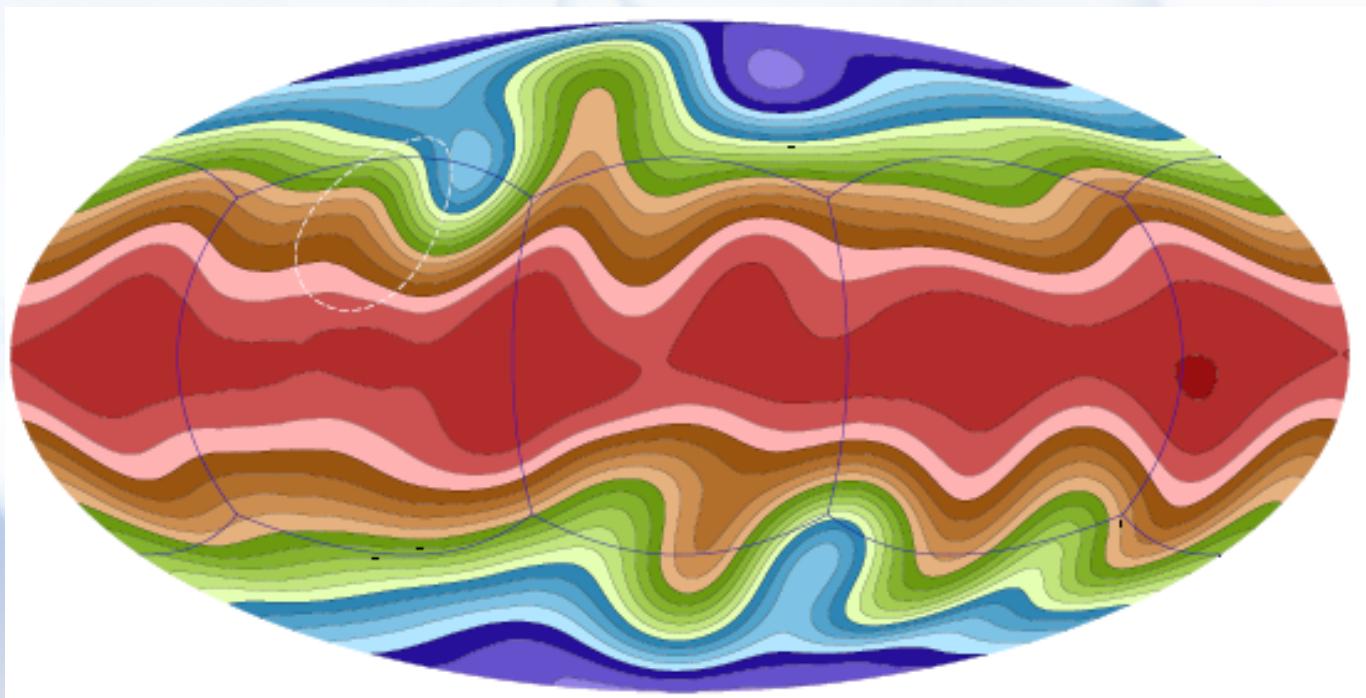


Large-scale tests

❖ Validation of the model

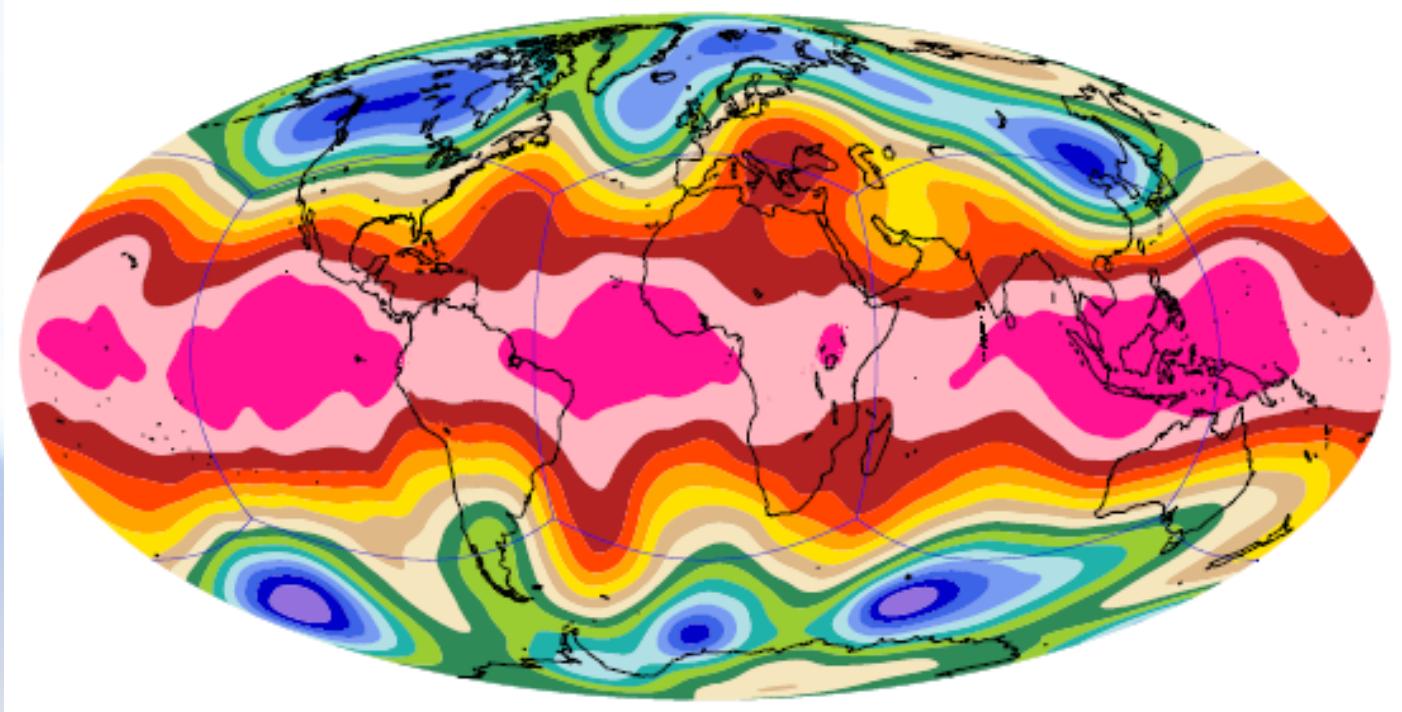
- Isolated mountain (Williamson test set, JCP 1992)

Day 15, 10,240*10,240*6 mesh (1km res), 1536 nodes of TH1A



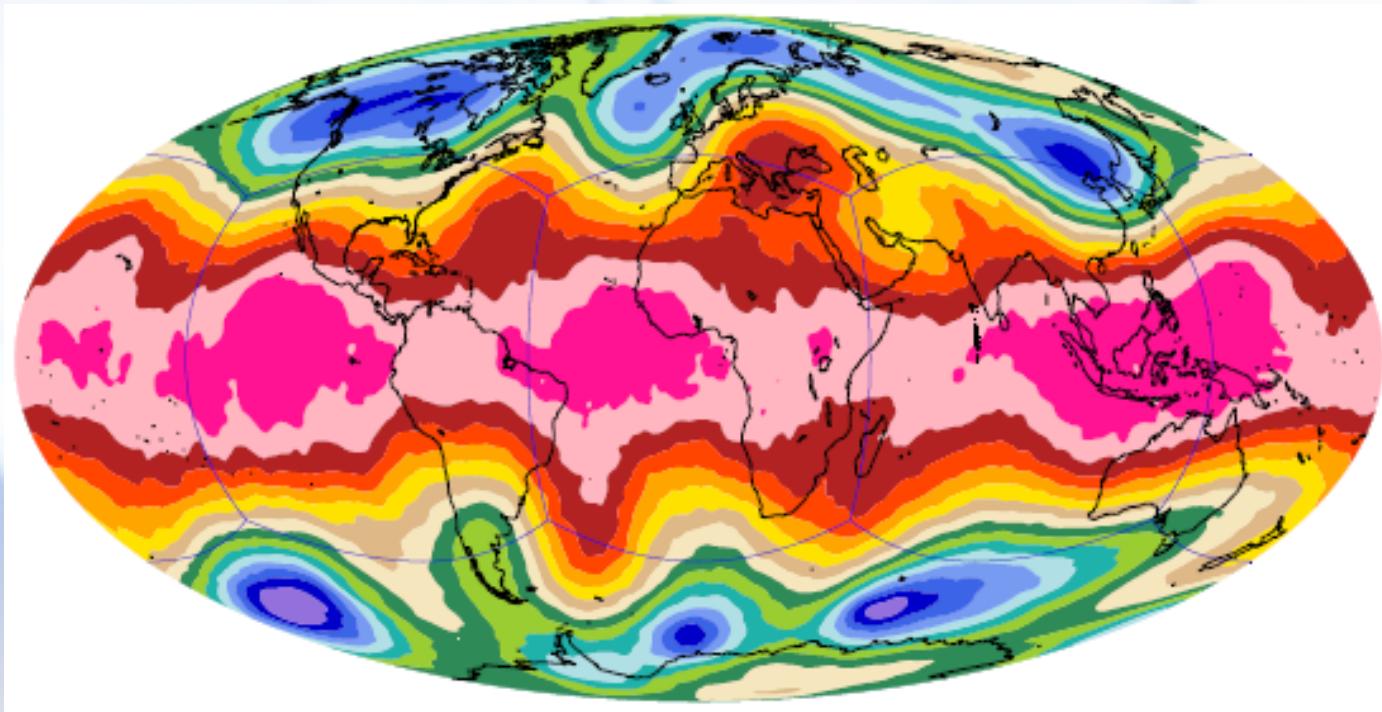
Large-scale tests

- ❖ Validation of the model (cont'd)
 - Real topography of the Earth, zonal flow
Day 15, 256*256*6 mesh (40km res)



Large-scale tests

- ❖ Validation of the model (cont'd 2)
 - Real topography of the Earth, zonal flow
Day 15, 10,240*10,240*6 mesh (1km res)



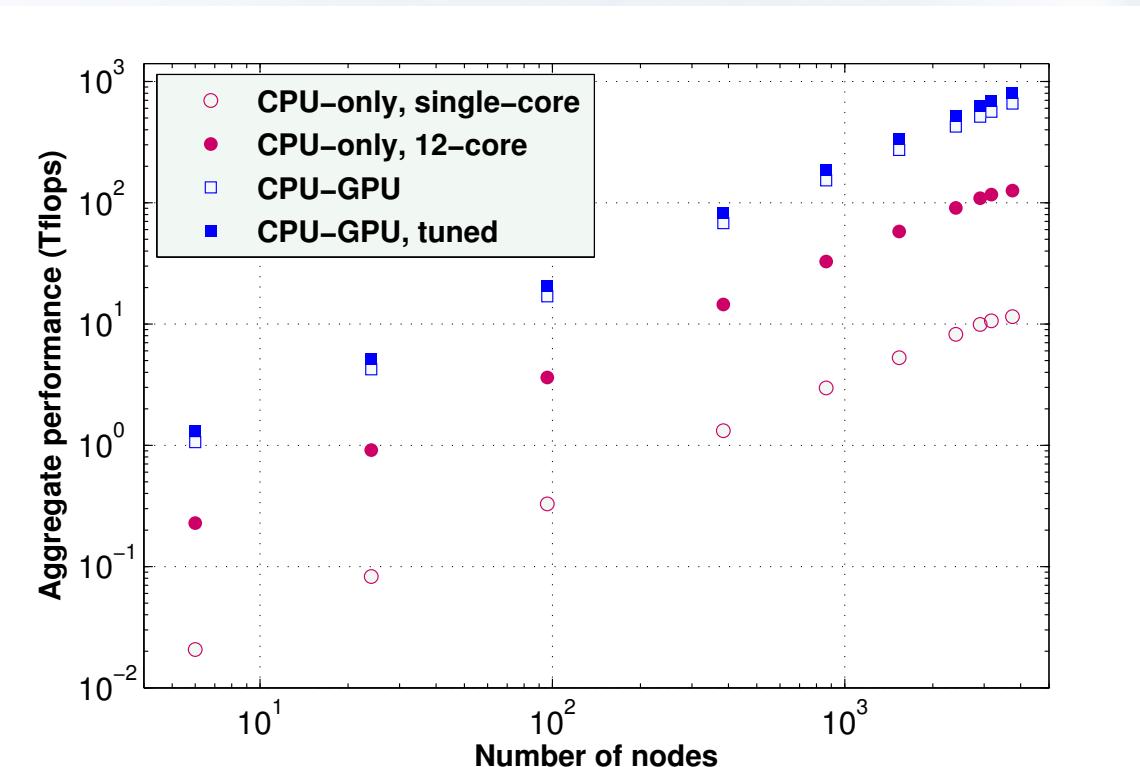
Large-scale tests

❖ Weak-scaling tests: configurations

Number of nodes	Mesh size	Peak of (CPU, GPU)
$6 = 6 \times 1 \times 1$	$6 \times 1024 \times 1024$	(0.8, 3.1) Tflops
$24 = 6 \times 2 \times 2$	$6 \times 2048 \times 2048$	(3.3, 12) Tflops
$96 = 6 \times 4 \times 4$	$6 \times 4096 \times 4096$	(14, 49) Tflops
$384 = 6 \times 8 \times 8$	$6 \times 8192 \times 8192$	(54, 198) Tflops
$864 = 6 \times 12 \times 12$	$6 \times 12288 \times 12288$	(121, 445) Tflops
$1536 = 6 \times 16 \times 16$	$6 \times 16384 \times 16384$	(216, 791) Tflops
$2400 = 6 \times 20 \times 20$	$6 \times 20480 \times 20480$	(337, 1236) Tflops
$2904 = 6 \times 22 \times 22$	$6 \times 22528 \times 22528$	(408, 1496) Tflops
$3750 = 6 \times 25 \times 25$	$6 \times 25600 \times 25600$	(527, 1931) Tflops

Large-scale tests

Weak-scaling results



- Largest run: 3750 nodes, $25,600 \times 25,600 \times 6$ mesh (11.8 B unknowns)

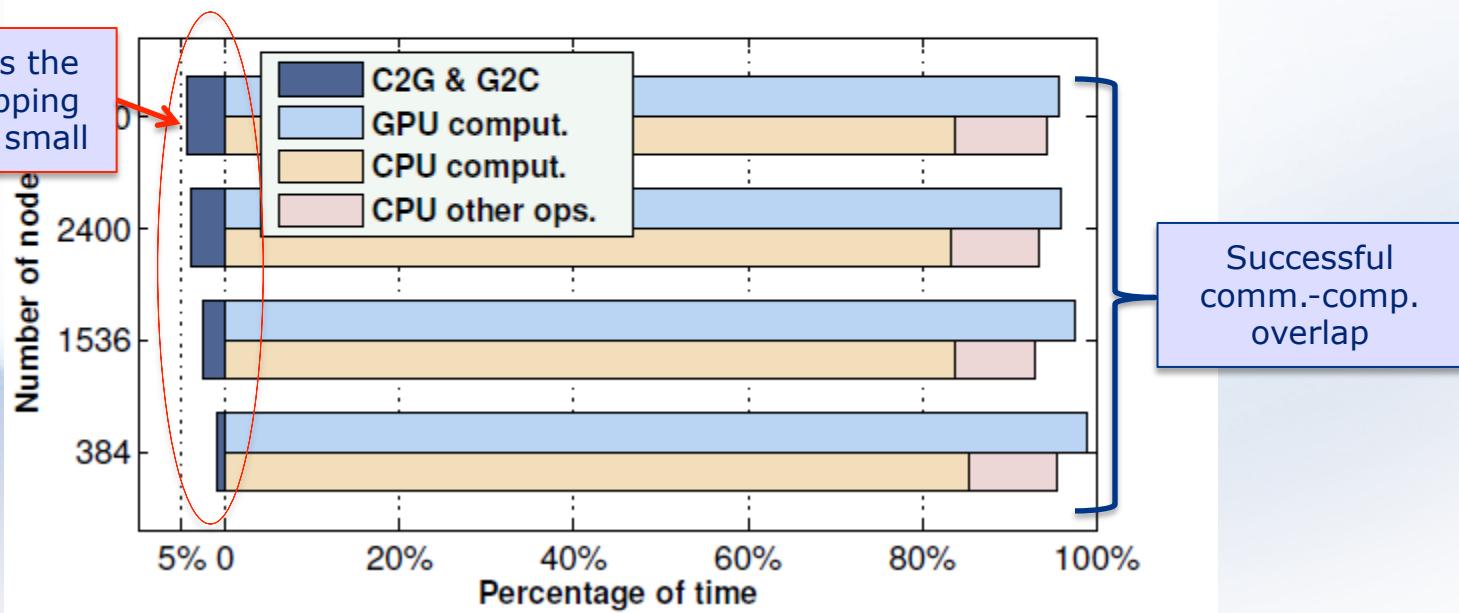
CPU-only (1-core)	CPU-only (12-core)	CPU-GPU	CPU-GPU tuned
11.5 Tflops	126 Tflops	658 Tflops	809 Tflops

32.8%
of peak!

Large-scale tests

✧ Strong-scaling results ($N=25,200$, 100 time steps)

Number of nodes	384	1536	2400	3750
Time (s)	56.9	14.4	9.4	5.9
Efficiency	1.00	0.99	0.97	0.98
Agg. Tflops	84.5	335.1	513.4	809.6



Contents



Background



Mesh & Equations



Algorithm & Implementations



Large-scale tests

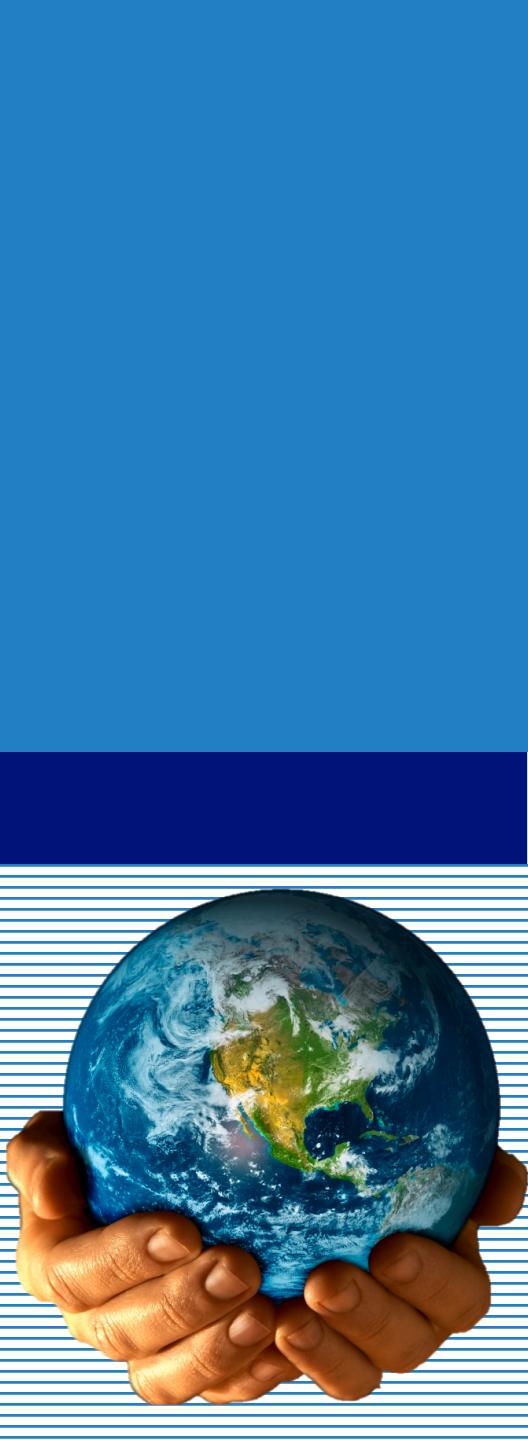


Conclusions



Conclusions

- ✧ A scalable approach for global SWEs on CPU-GPU clusters
 - Adjustable partition between CPUs/GPUs
 - Effective comm.-comp. overlap to hide comm. cost
 - “Pipe-flow” scheme to arrange message-passing
 - Systematic optimizations on CPU and CUDA code
- ✧ Ideal weak/strong scaling on the Tianhe-1A
 - Up to 3750 nodes (45,000 CPU cores + 52,500 GPU cores)
 - Sustaining 809 Tflops (32.8% of peak) on 3750 nodes
- ✧ SWEs on FPGA
 - Using Maxeler Xilinx FPGA clusters
 - Estimate a further improvement



Thank You !

Presenter: Lin Gan

Computer Science & Technology
Center for Earth System Science
Tsinghua University, China

Email: l-gan11@mails.tsinghua.edu.cn

Tel: +86-15810537953

