Refactoring and Optimizing the Community Atmosphere Model (CAM) on the Sunway TaihuLight Supercomputer

Haohuan Fu
haohuan@tsinghua.edu.cn
CESS, Tsinghua University
National Supercomputing Center in Wuxi

Oct 5th 2016 @ CCDSC

Sunway TaihuLight: an Overview

Homegrown many-core processor: SW26010

- 260 cores per chip
- 3 Tflops

The first system in the world that provides over 100 Pflops performance with over 10 million cores

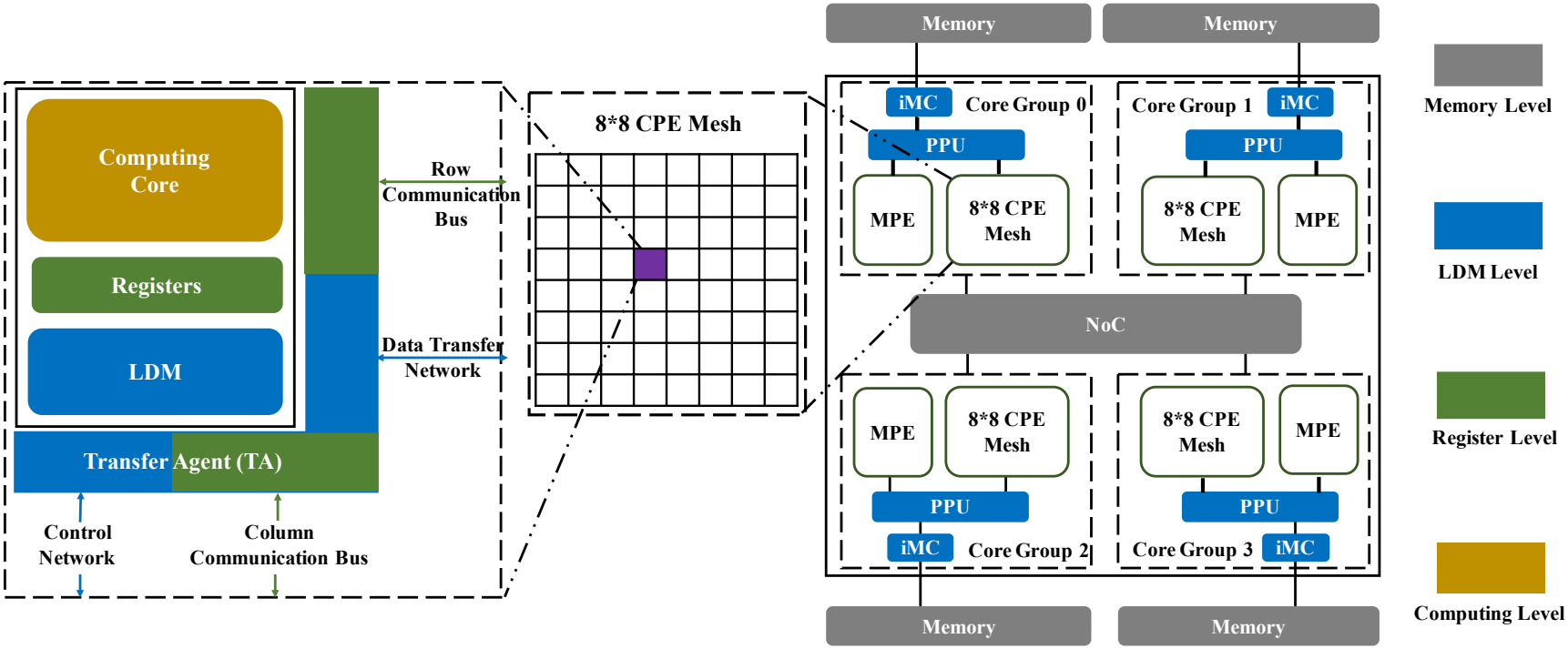
- theoretical peak 125 Pflops, 2.5 times improvement over before
- LINPACK performance 93 Pflops, 3 times improvement over before

High efficiency of the overall system

- 6.05 Gflops/Watt, 3 to 6 times improvement over Tianhe-2, Titan, and K

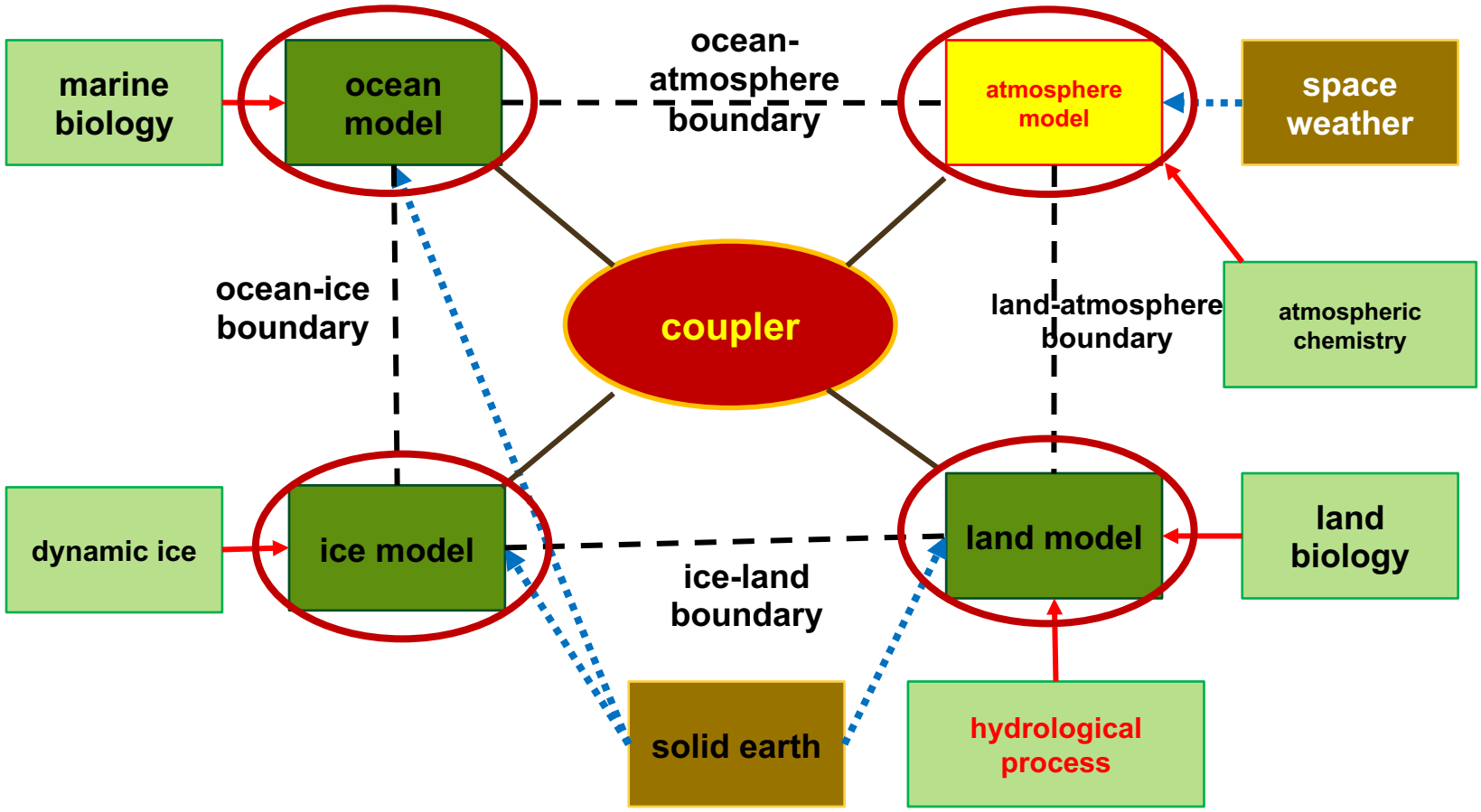
Three full-scale applications elected as 2016 Gordon Bell finalists

SW26010: General Architecture

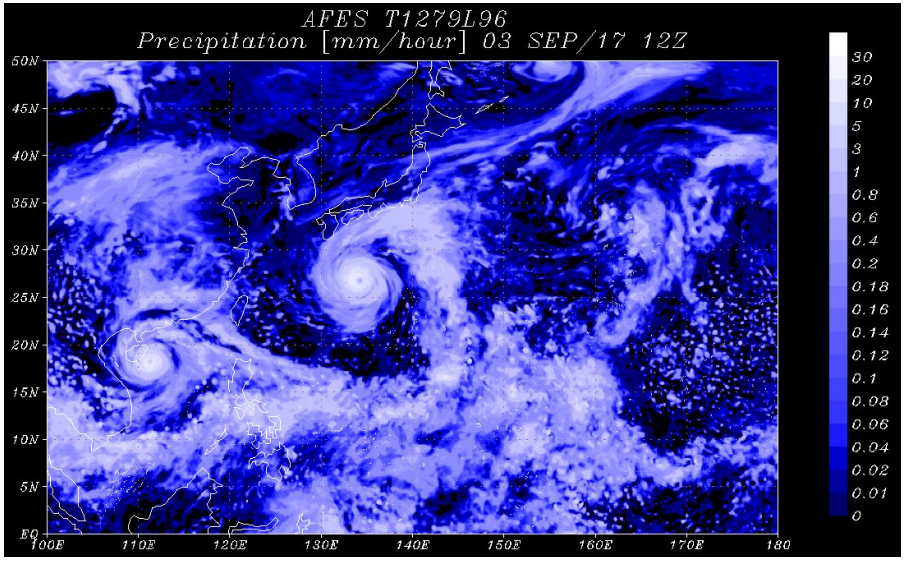


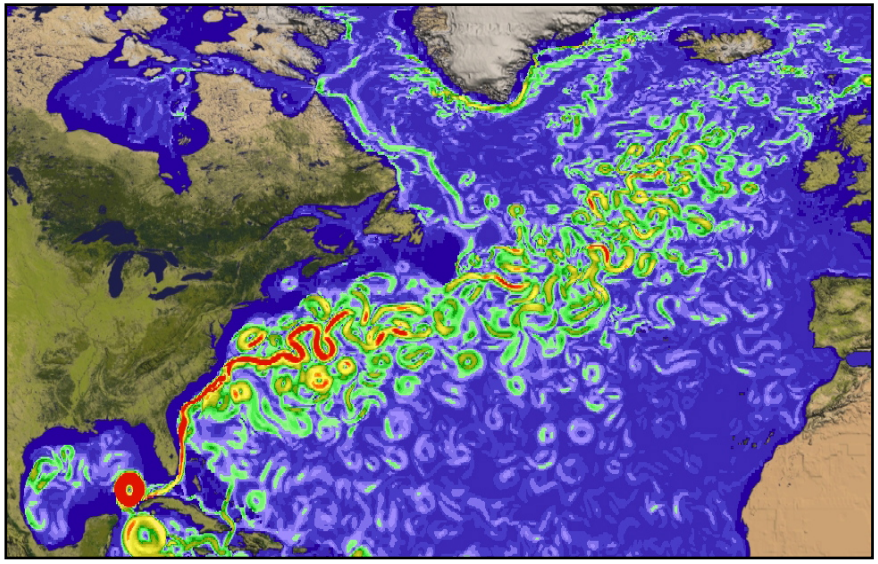
Earth System Modeling and HPC: the Current Computational Challenges

More and more component models

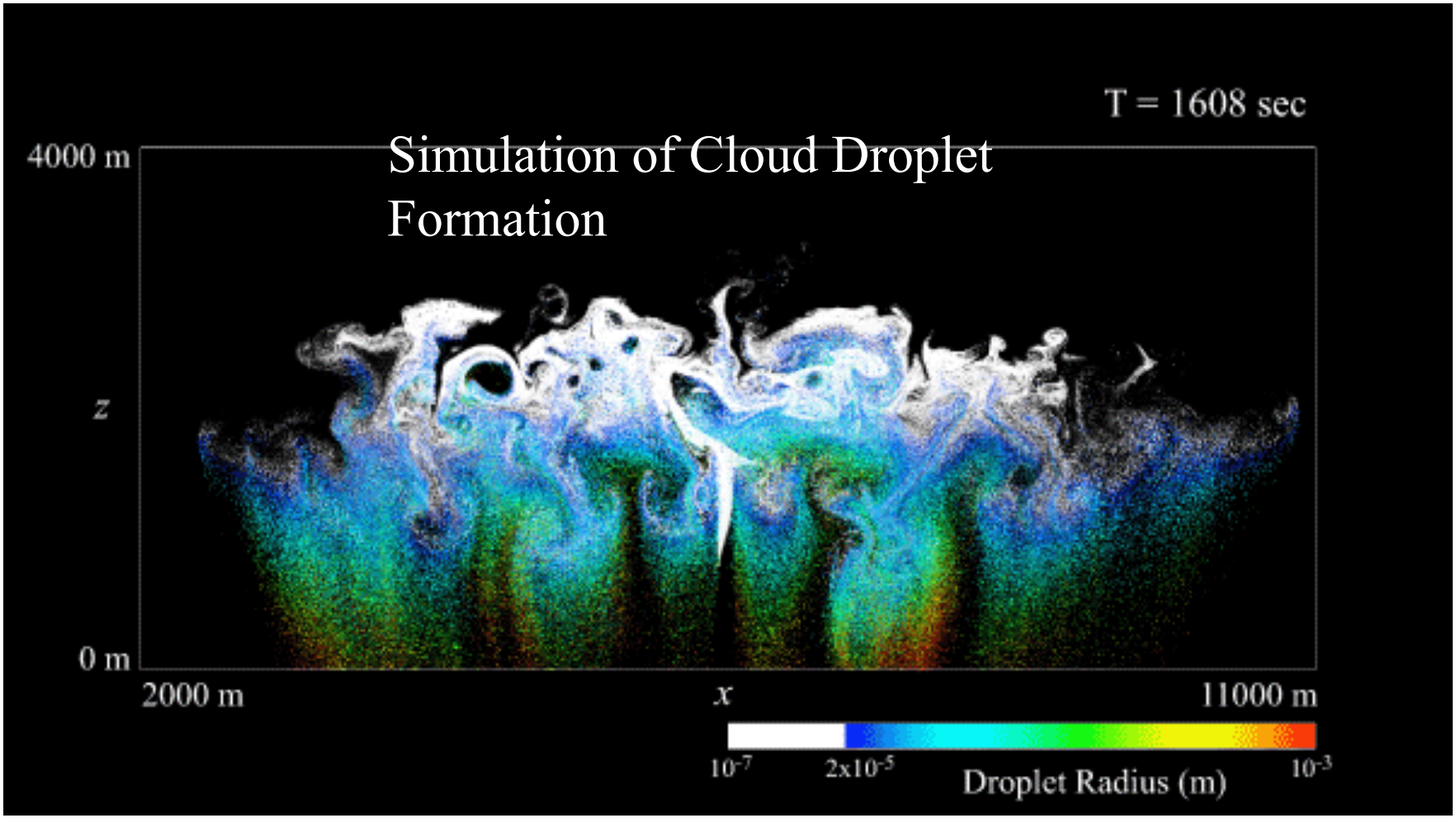


Increase in Spatial and Temporal Resolution to be Cloud-Resolving and Eddy-Resolving



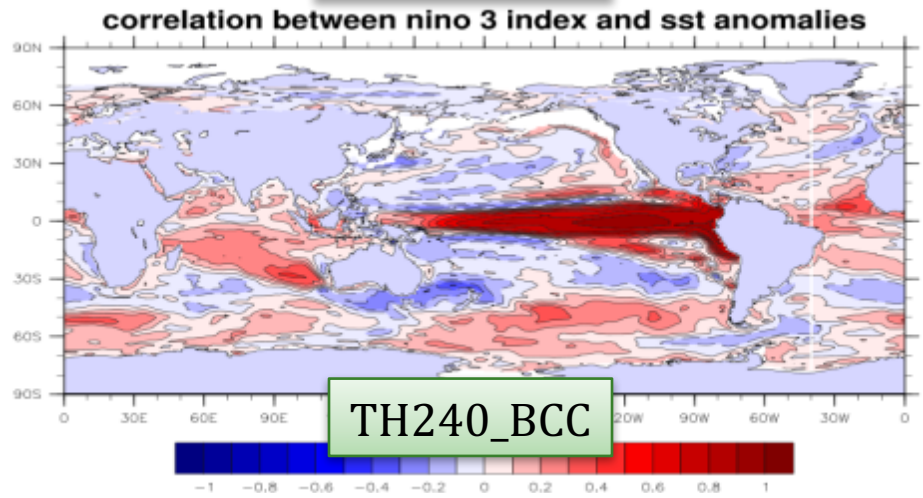


Simulation of more and more detailed physics processes

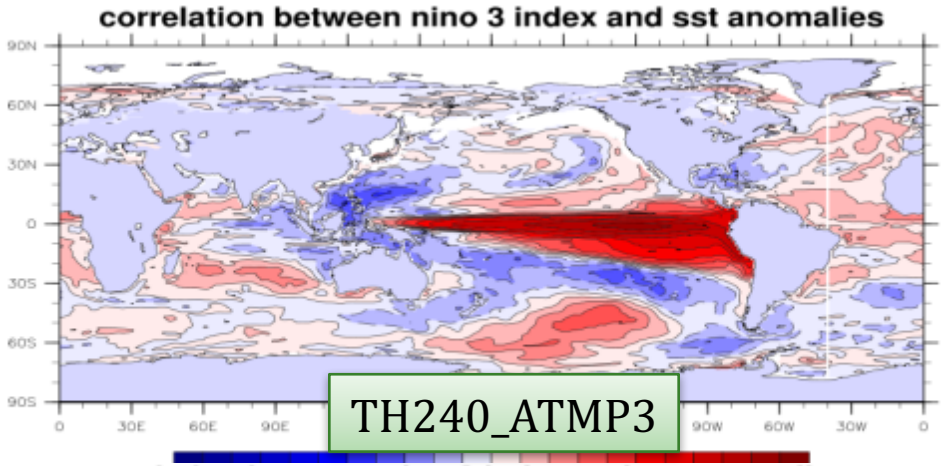


Online Ensembles

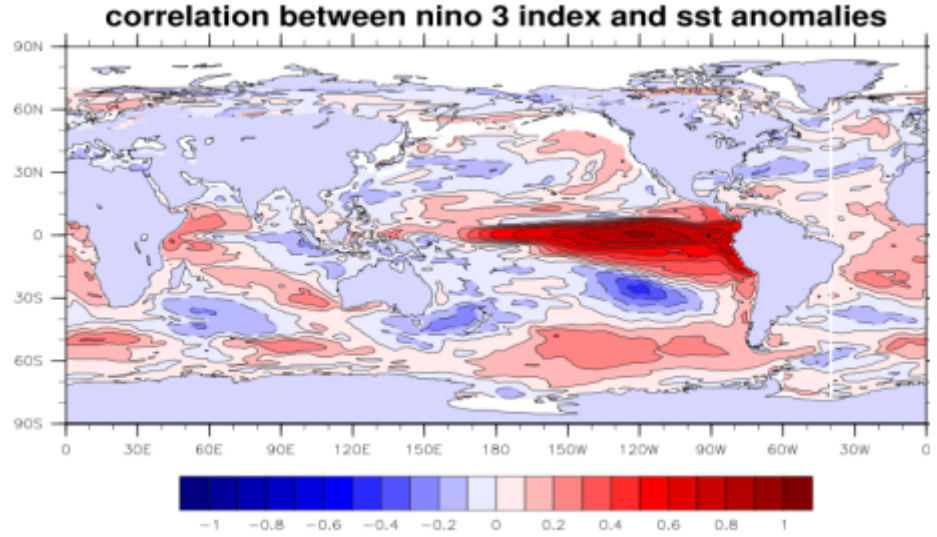
TH240_CAM



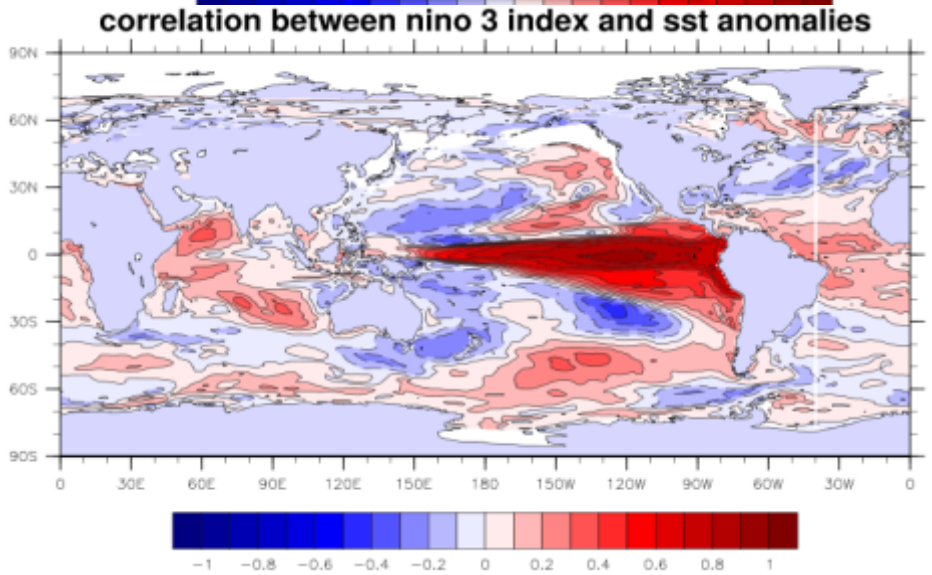
TH240_N_111



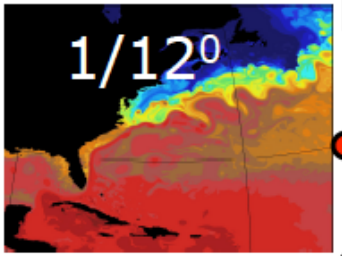
TH240_BCC

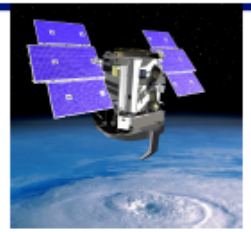


TH240_ATMP3

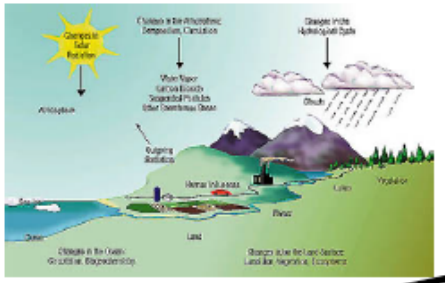


Balancing Science Goals with Computing Power





Data Assimilation

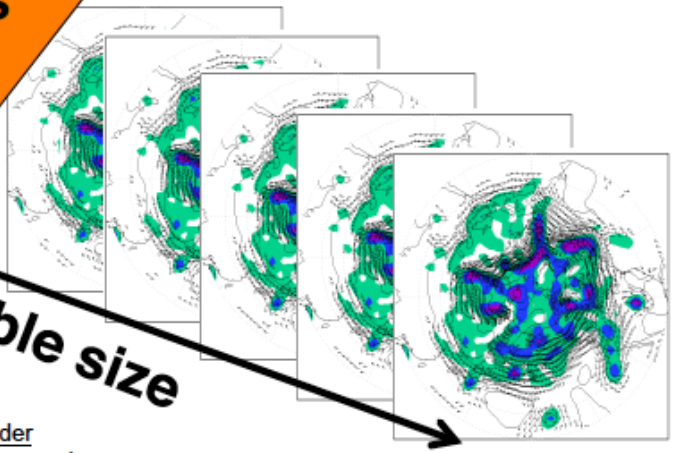


Resolution

Augmented Computing Resources

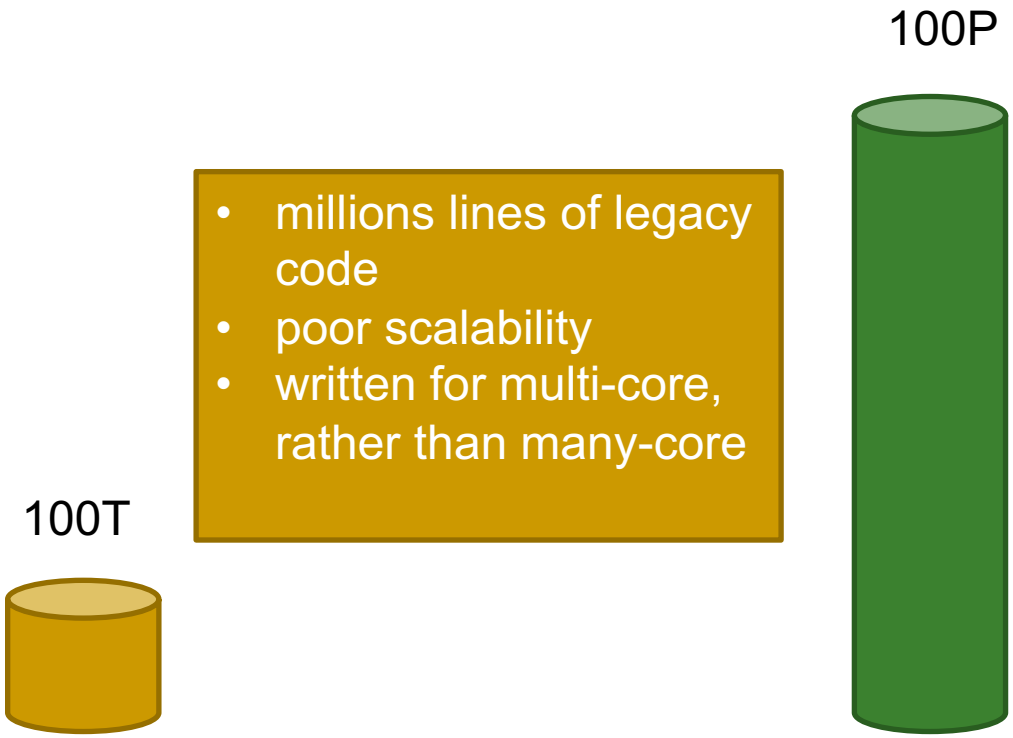
Complexity

Duration and/or Ensemble size



University of Colorado at Boulder
National Center for Atmospheric Research

The Gap between Software and Hardware



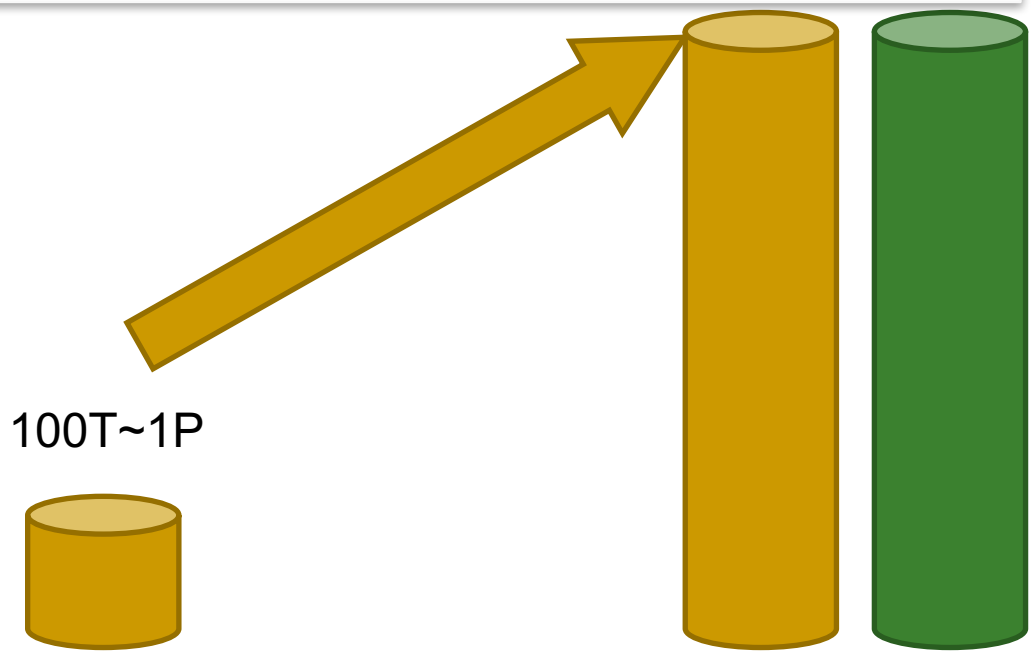
- millions lines of legacy code
- poor scalability
- written for multi-core, rather than many-core

China's models
- pure CPU code
- scaling to hundreds or thousands of cores

China's supercomputers
- heterogeneous systems with many-core chips
- millions of cores

Our Research Goals

- highly scalable framework that can efficiently utilize many-core processors
- automated tools to deal with the legacy code

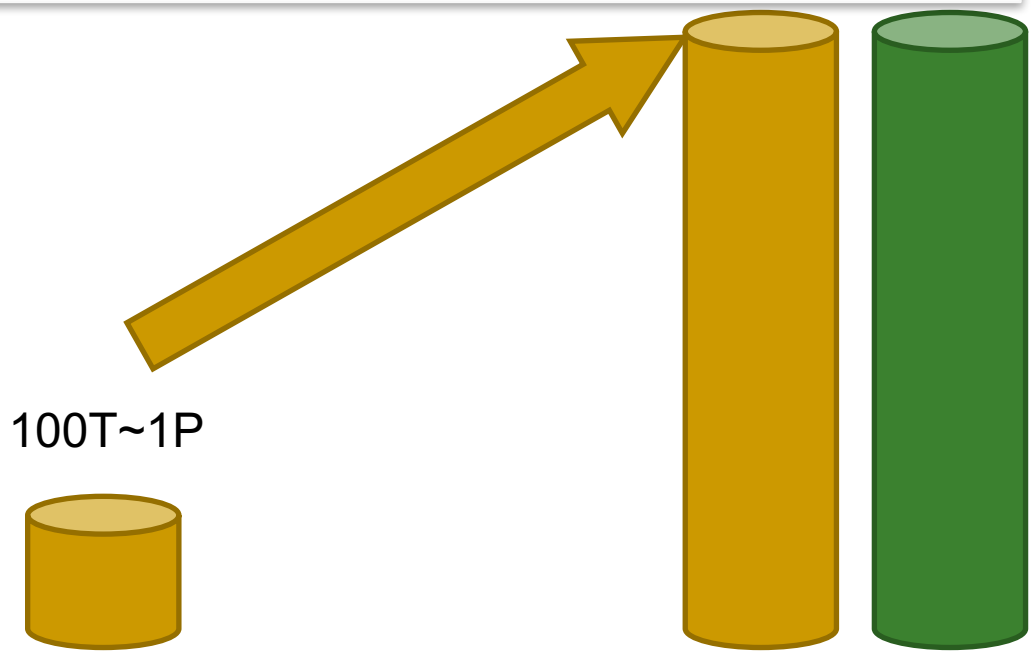


China's models
- pure CPU code
- scaling to hundreds or thousands of cores

China's supercomputers
- heterogeneous systems with many-core chips
- millions of cores

Our Research Goals

- highly scalable framework that can efficiently utilize many-core processors
- automated tools to deal with the legacy code



China's models
- pure CPU code
- scaling to hundreds or thousands of cores

China's supercomputers
- heterogeneous systems with many-core chips
- millions of cores

Example: Highly-Scalable Atmospheric Simulation Framework

Yang, Chao
Institute of Software, CAS
computational mathematics

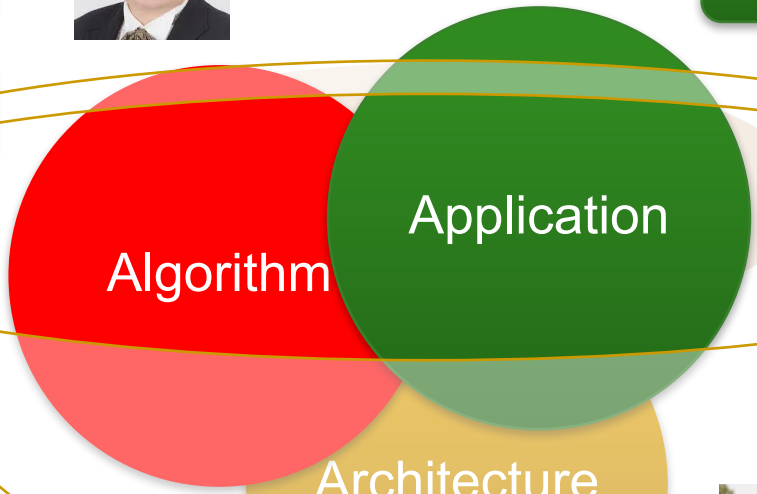
cube-sphere grid or other grid

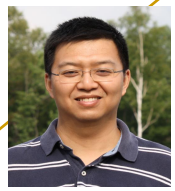
explicit, implicit, or semi-implicit method

cloud resolving

Wang, Lanning
Beijing Normal University
climate modeling

Xue, Wei
Tsinghua University
computer science



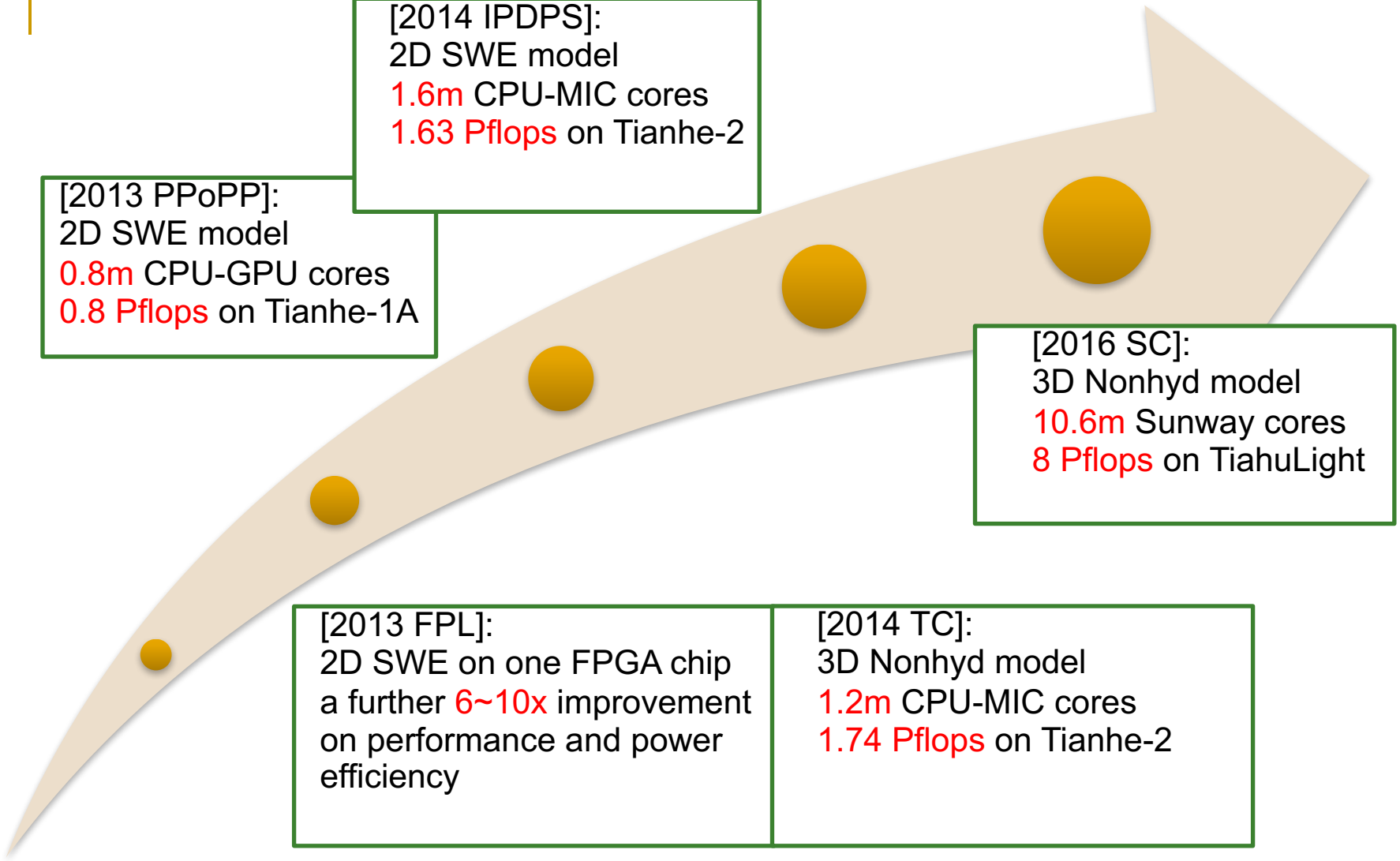


Fu, Haohuan
Tsinghua University
geo-computing

Sunway, GPU, MIC, FPGA

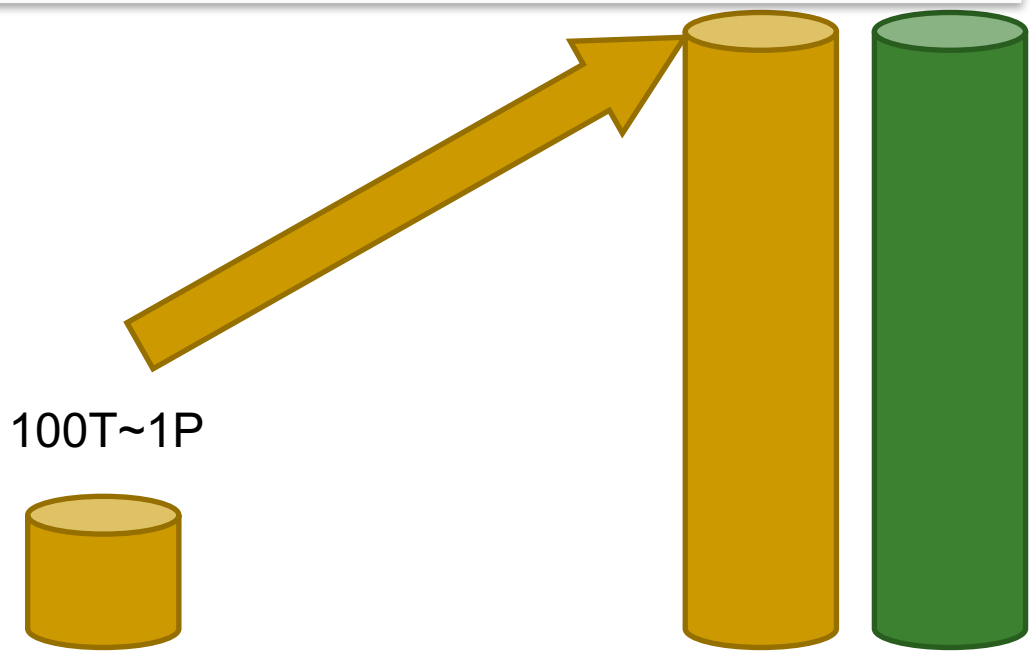
C/C++, Fortran, MPI, CUDA, Java, …

The “Best” Computational Solution


[2013 PPoPP]:
2D SWE model
0.8m CPU-GPU cores
0.8 Pflops on Tianhe-1A
[2014 IPDPS]:
2D SWE model
1.6m CPU-MIC cores
1.63 Pflops on Tianhe-2
[2016 SC]:
3D Nonhyd model
10.6m Sunway cores
8 Pflops on TiahuLight
[2013 FPL]:
2D SWE on one FPGA chip
a further 6~10x improvement
on performance and power
efficiency
[2014 TC]:
3D Nonhyd model
1.2m CPU-MIC cores
1.74 Pflops on Tianhe-2

Our Research Goals

- highly scalable framework that can efficiently utilize many-core accelerators
- automated tools to deal with the legacy code



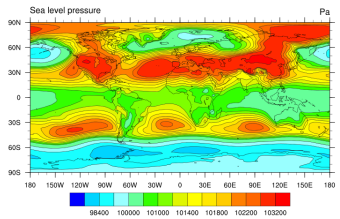
China's models
- pure CPU code
- scaling to hundreds or thousands of cores

China's supercomputers
- heterogeneous systems with many-core chips
- millions of cores

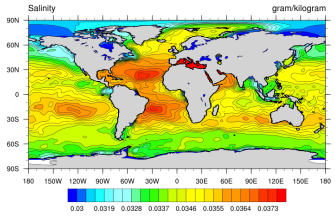
Earth System Modeling and HPC: Our Efforts on Refactoring CAM

THE CESM PROJECT

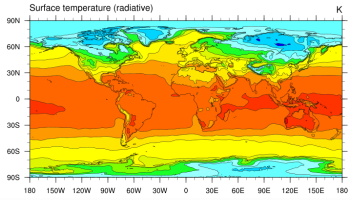
Tsinghua + BNU



F算例（大气+陆面）



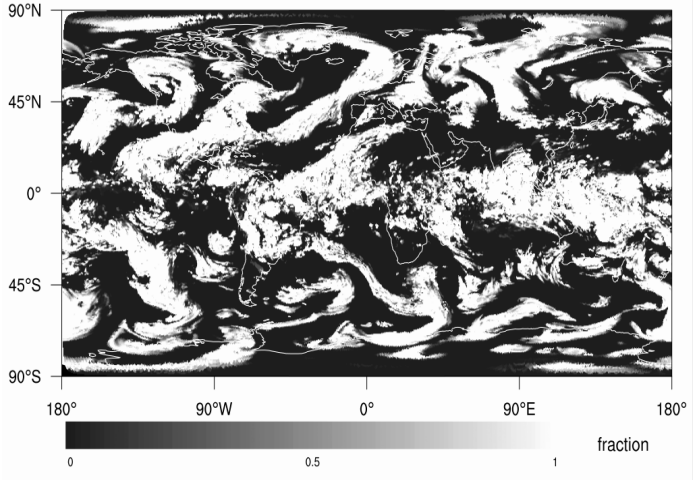
G算例（海洋+海冰）



B算例（全耦合）

- Four component models, millions lines of code

- Large-scale run on Sunway TaihuLight
- 24,000 MPI processes
- Over one million cores
- 10-20x speedup for kernels
- 2-3x speedup for the entire model



Major Challenges

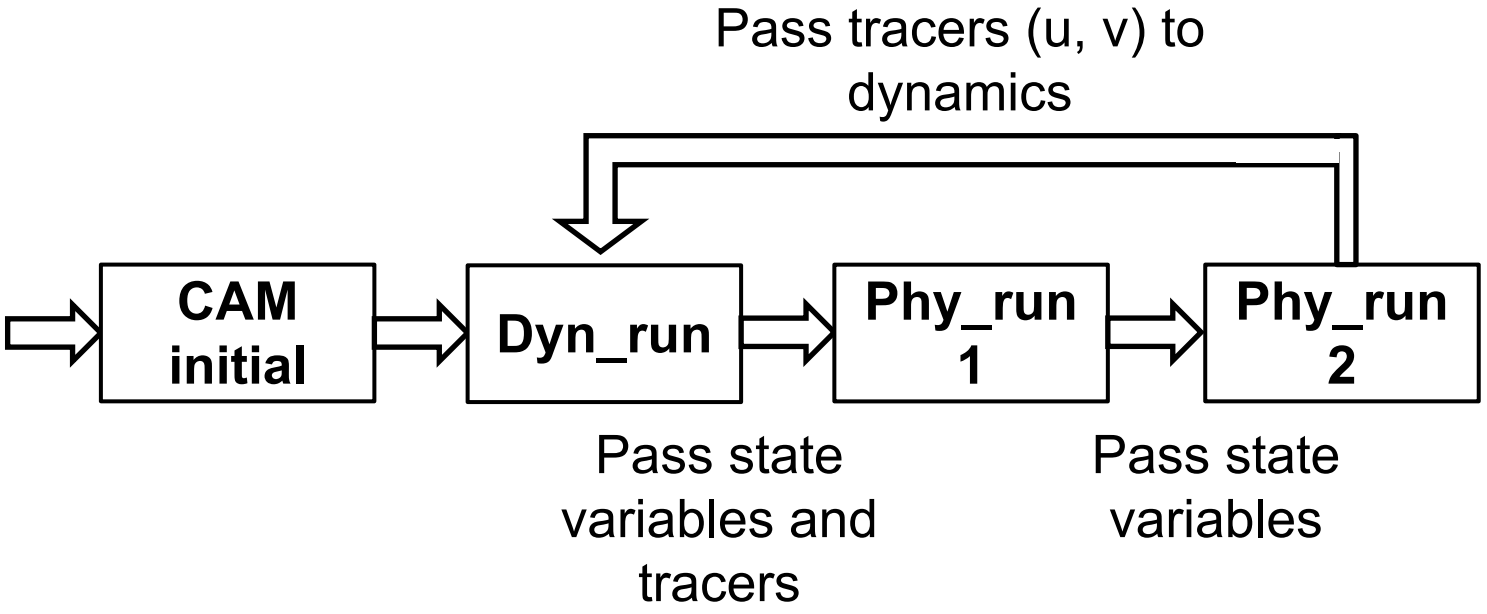
a high complexity in application, and a heavy legacy in the code base (millions lines of code)

an extremely complicated MPMD program with no hotspots (or hundreds of hotspots)

misfit between the in-place design philosophy and the new architecture

lack of people with interdisciplinary knowledge and experience

Workflow of CAM

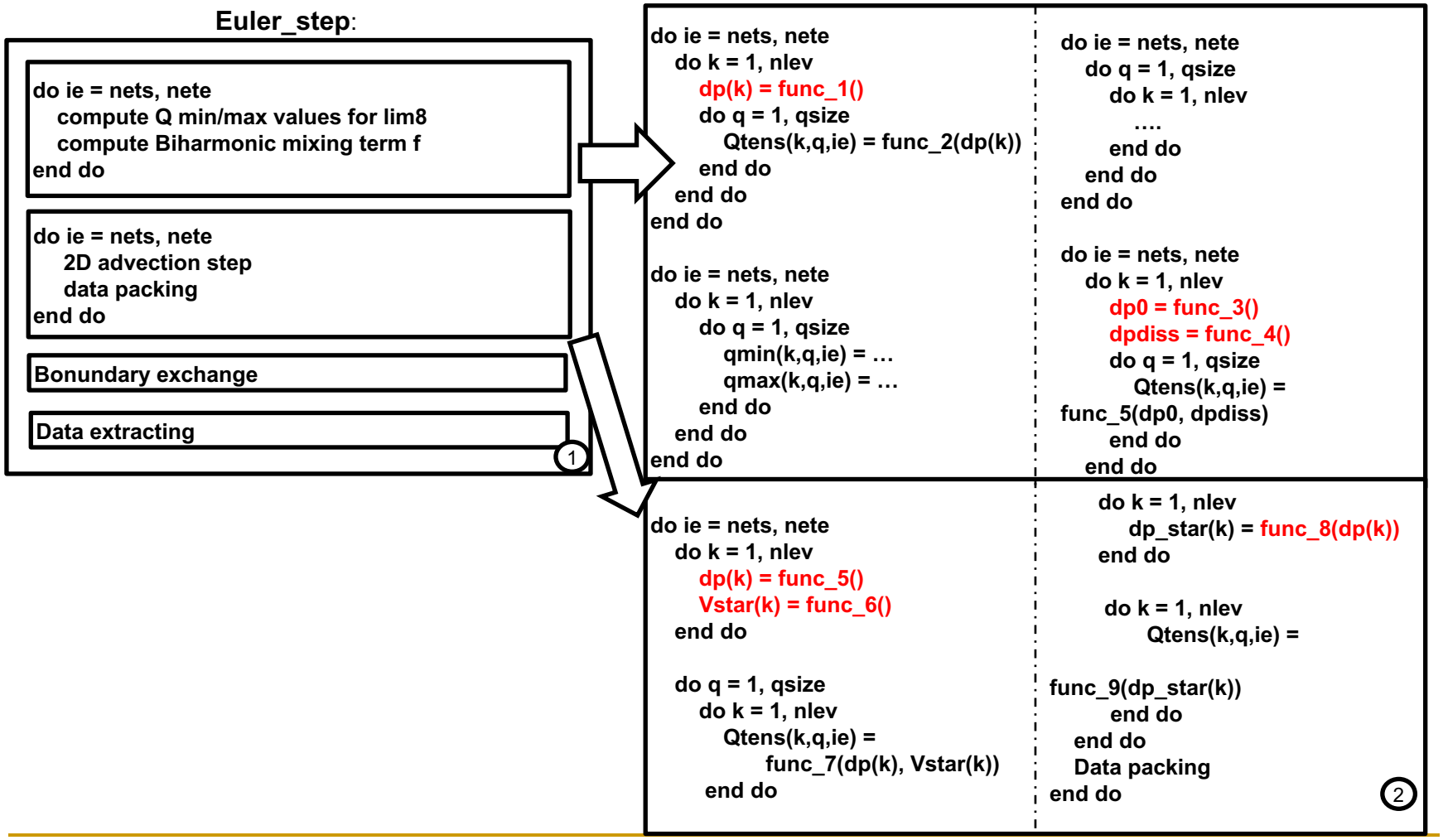


After initialization, the physics and the dynamics are executed in turn during each simulation time-step.

Porting of CAM: General Idea

- Entire code base: 530, 000 lines of code
- Components with regular code patterns
  - e.g. the CAM-SE dynamic core
  - manual OpenACC parallelization and optimization on code and data structures
- Components with irregular and complex code patterns
  - e.g. the CAM physics schemes
  - loop transformation tool to expose the right level of parallelism and code size
  - memory footprint analysis and reduction tool

Refactoring the Euler Step



Refactoring the Euler Step

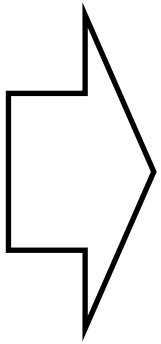
```
do ie = nets, nete
   do k = 1, nlev
      dp(k) = func_1()
      do q = 1, qsize
         Qtens(k,q,ie) =
func_2(dp(k))
      end do
   end do
end do

do ie = nets, nete
   do k = 1, nlev
      do q = 1, qsize
         qmin(k,q,ie) = …
         qmax(k,q,ie) = …
      end do
   end do
end do
```

```
do ie = nets, nete
   do q = 1, qsize
      do k = 1, nlev
         ….
      end do
   end do
end do

do ie = nets, nete
   do k = 1, nlev
      dp0 = func_3()
      dpdiss =
func_4()
      do q = 1, qsize
         Qtens(k,q,ie)
= func_5(dp0, dpdiss)
      end do
   end do
end do
```

```
do ie = nets, nete
   do k = 1, nlev
      dp(k) = func_5()
      Vstar(k) = func_6()
   end do

   do q = 1, qsize
      do k = 1, nlev
         Qtens(k,q,ie) =
    func_7(dp(k), Vstar(k))
      end do
```

```
      do k = 1, nlev
         dp_star(k) =
func_8(dp(k))
      end do

      do k = 1, nlev
         Qtens(k,q,ie)
=

func_9(dp_star(k))
      end do
   end do
   Data packing
end do
```

2



```
do ie = nets, nete
   do k = 1, nlev
      do q = 1, qsize
         Qtens(k,q,ie) =
        func_2(func_1())
      end do
   end do
end do

do ie = nets, nete
   do k = 1, nlev
      do q = 1, qsize
         qmin(k,q,ie) = …
         qmax(k,q,ie) = …
      end do
   end do
end do
```

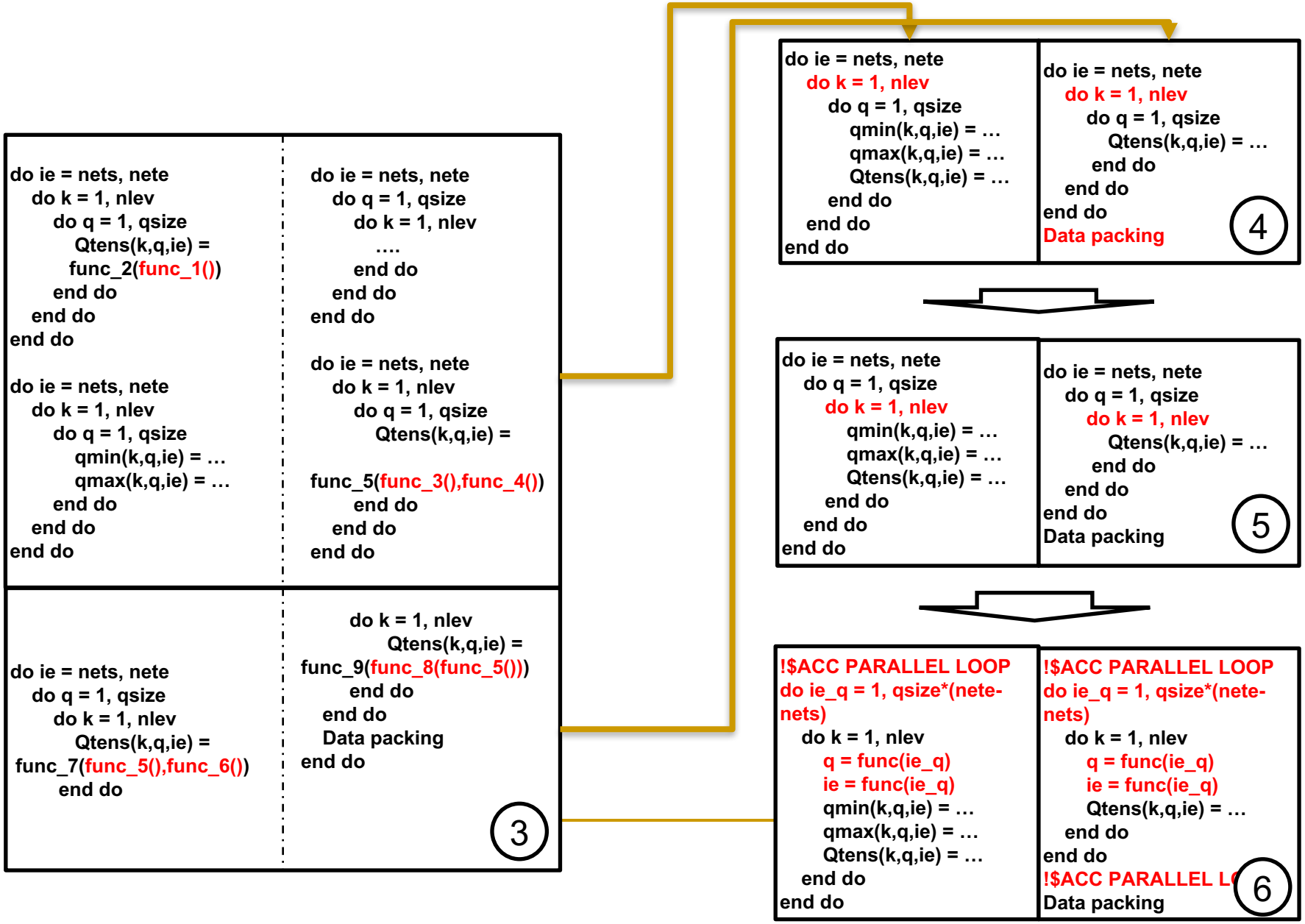
```
do ie = nets, nete
   do q = 1, qsize
      do k = 1, nlev
         ….
      end do
   end do
end do

do ie = nets, nete
   do k = 1, nlev
      do q = 1, qsize
         Qtens(k,q,ie) =

func_5(func_3(),func_4())
      end do
   end do
end do
```

```
do ie = nets, nete
   do q = 1, qsize
      do k = 1, nlev
         Qtens(k,q,ie) =
 func_7(func_5(),func_6())
      end do
```

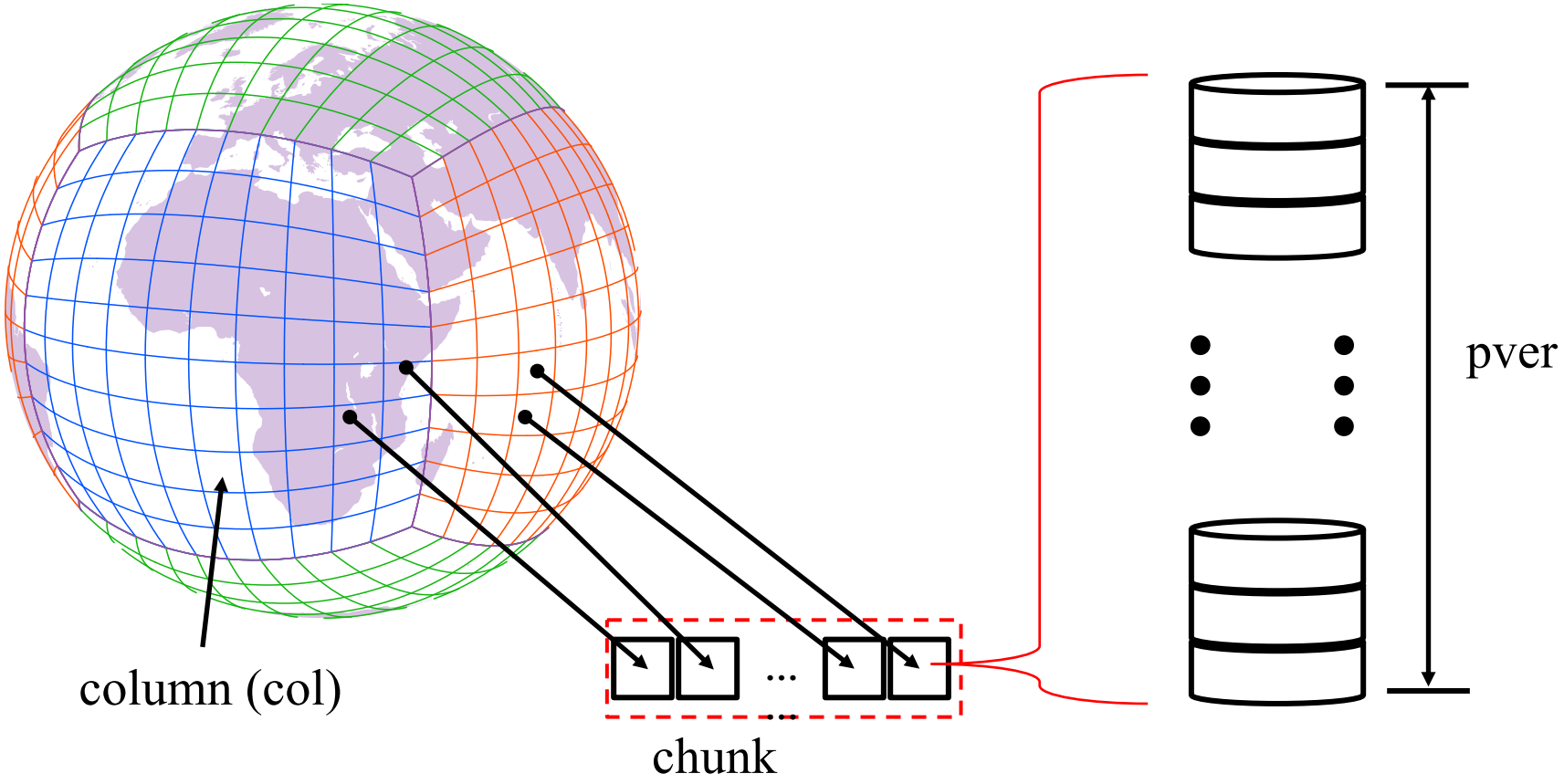
```
      do k = 1, nlev
         Qtens(k,q,ie) =
func_9(func_8(func_5()))
      end do
   end do
   Data packing
end do
```

3

Refactoring the Euler Step



Refactoring of the Physics Schemes



A Loop Transformation Tool: Typical Scenario

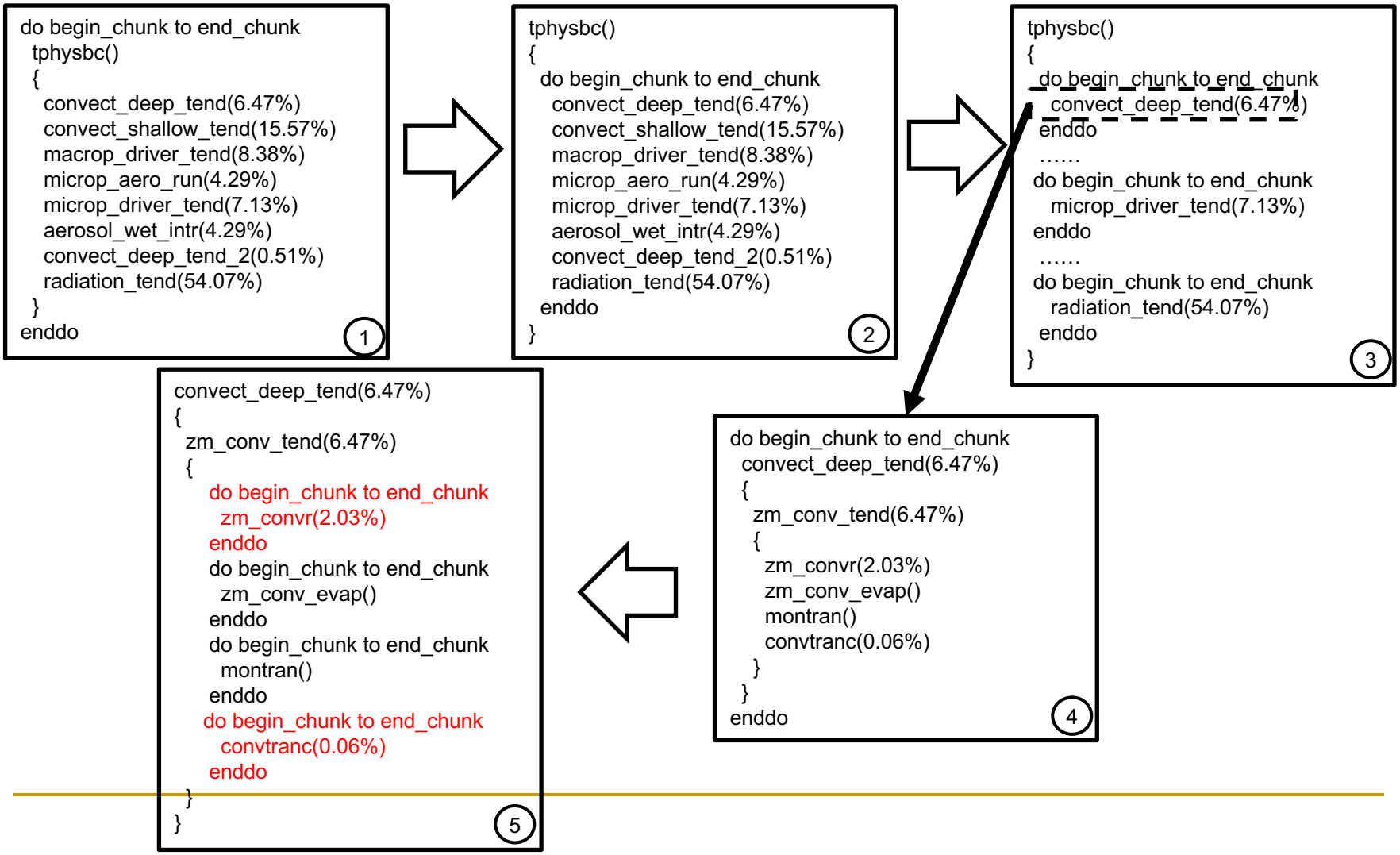
```
Do i = 1, m
   call F(A, B(i), C(i,:))
End do
```

```
Subroutine F(A, B, C)
   !parameter declaration
   real :: A, B
   real, dimension(:) :: C

   !local variable declaration
   real :: X, Y

   !execution
   X = 1
   Y = 1
   call lower1(X, C)
   call lower2(Y, C)
   B = X+Y
   C(:) = C(:) + X*Y
End
```

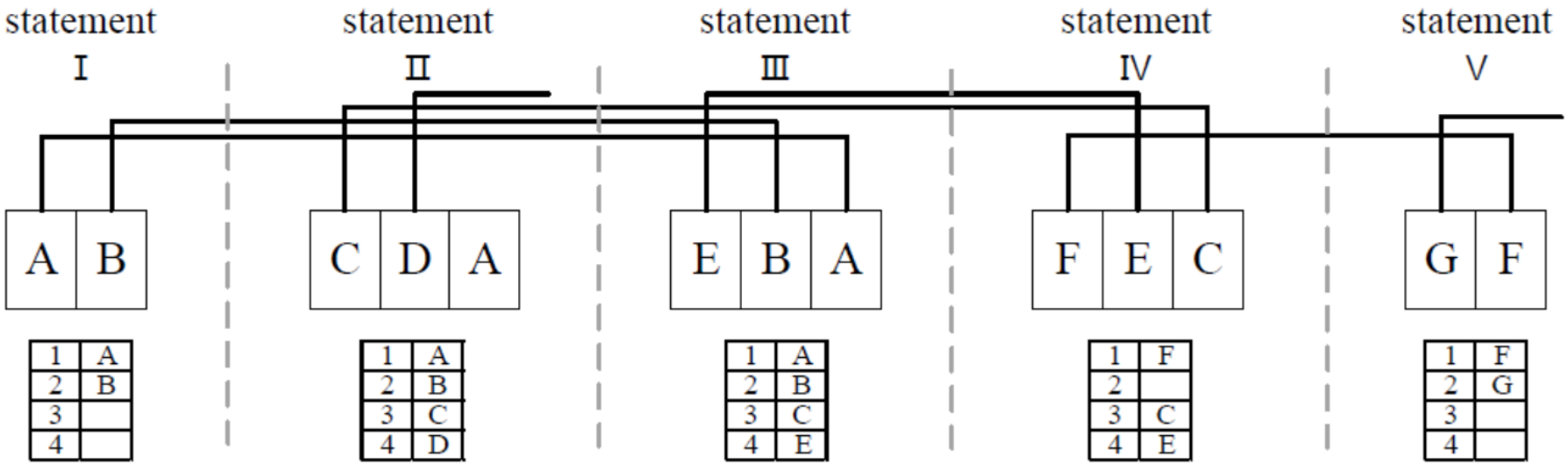
```
call F(A, B(:), C(:,:), m)
```

```
Subroutine F(A, B, C, m)
   !parameter declaration
   real :: A
   real, dimension(:) :: B
   real, dimension(:,:) :: C
   integer :: m

   !local variable declaration
   real, dimension(m) :: X, Y

   !execution
   do i = 1, m
      X(i) = 1
      Y(i) = 2
      call lower1(X(i), C(i, :))
   end do
   do i = 1, m
      call lower2(Y(i), C(i, :))
      B(i) = X(i)+Y(i)
      C(i, :) = X(i)*Y(i)
   end do
End
```

Loop Transformation for Phys_run1



Variable Storage Space Analysis and Reduction Tool



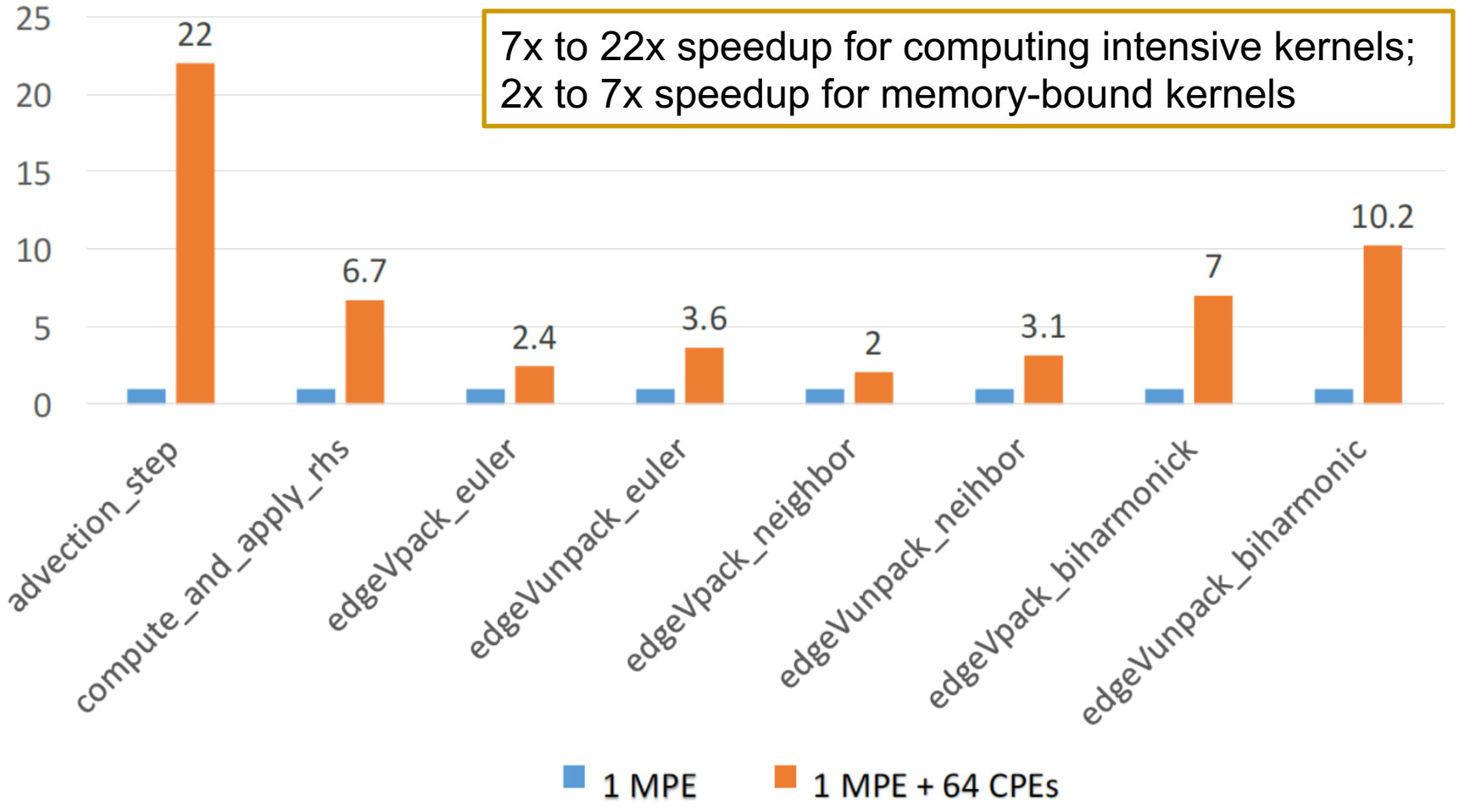
Basic functions

- Estimate the storage requirements of the variable and arrays
- Identify the lifespan of the variables and arrays
- Determine whether the variables and arrays of each CPE thread can fit into the 64KB SPM.

Example Explanation

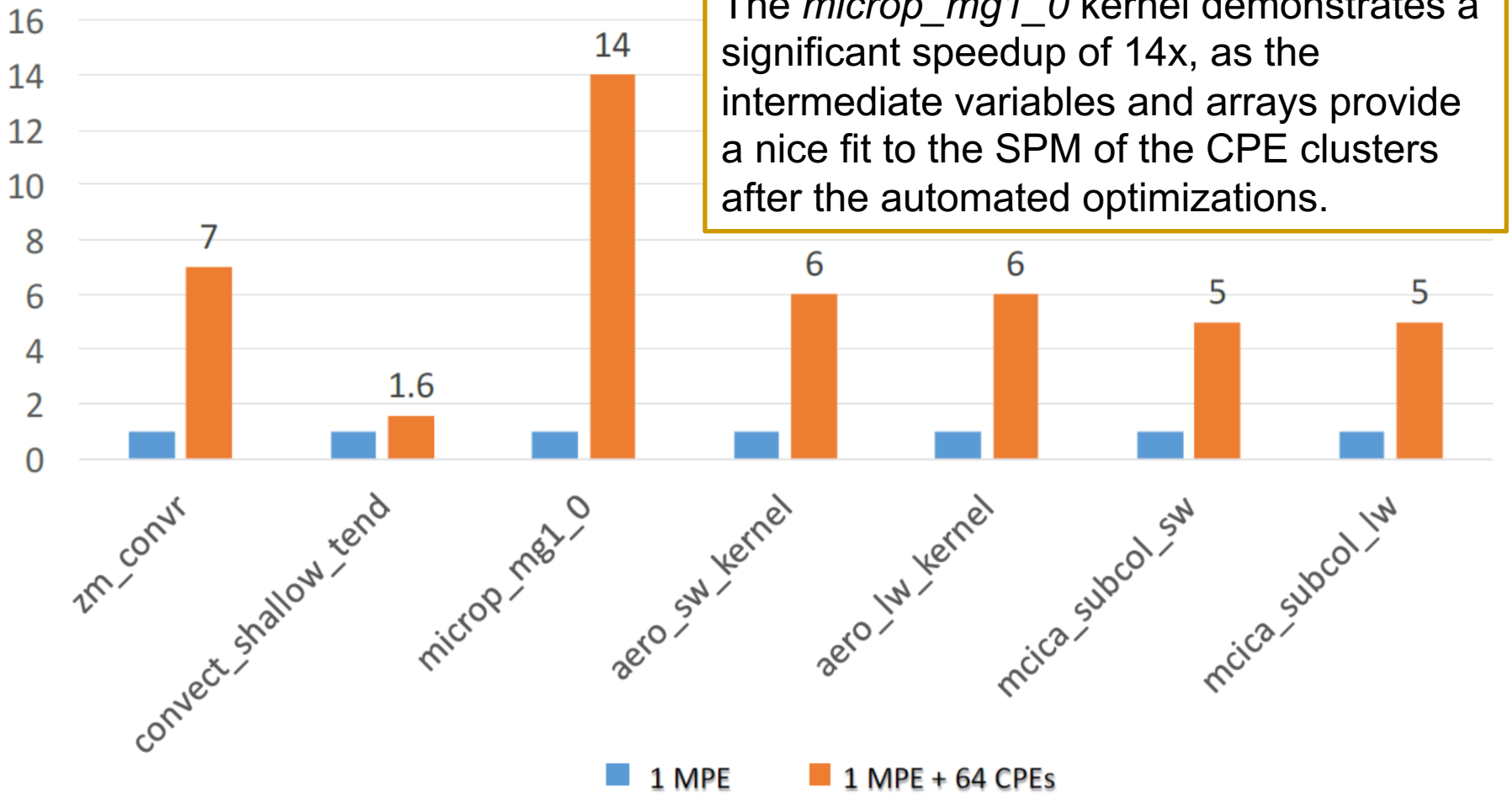
- The original Fortran function accesses 7 intermediate arrays (A to G) during the computation process. By analyzing the lifespan of these 7 arrays, which are annotated by the lines above these arrays, we can determine that 4 arrays would provide sufficient space to store these 7 arrays in different stages of the execution process.

Speedup of Major Kernels in CAM-SE



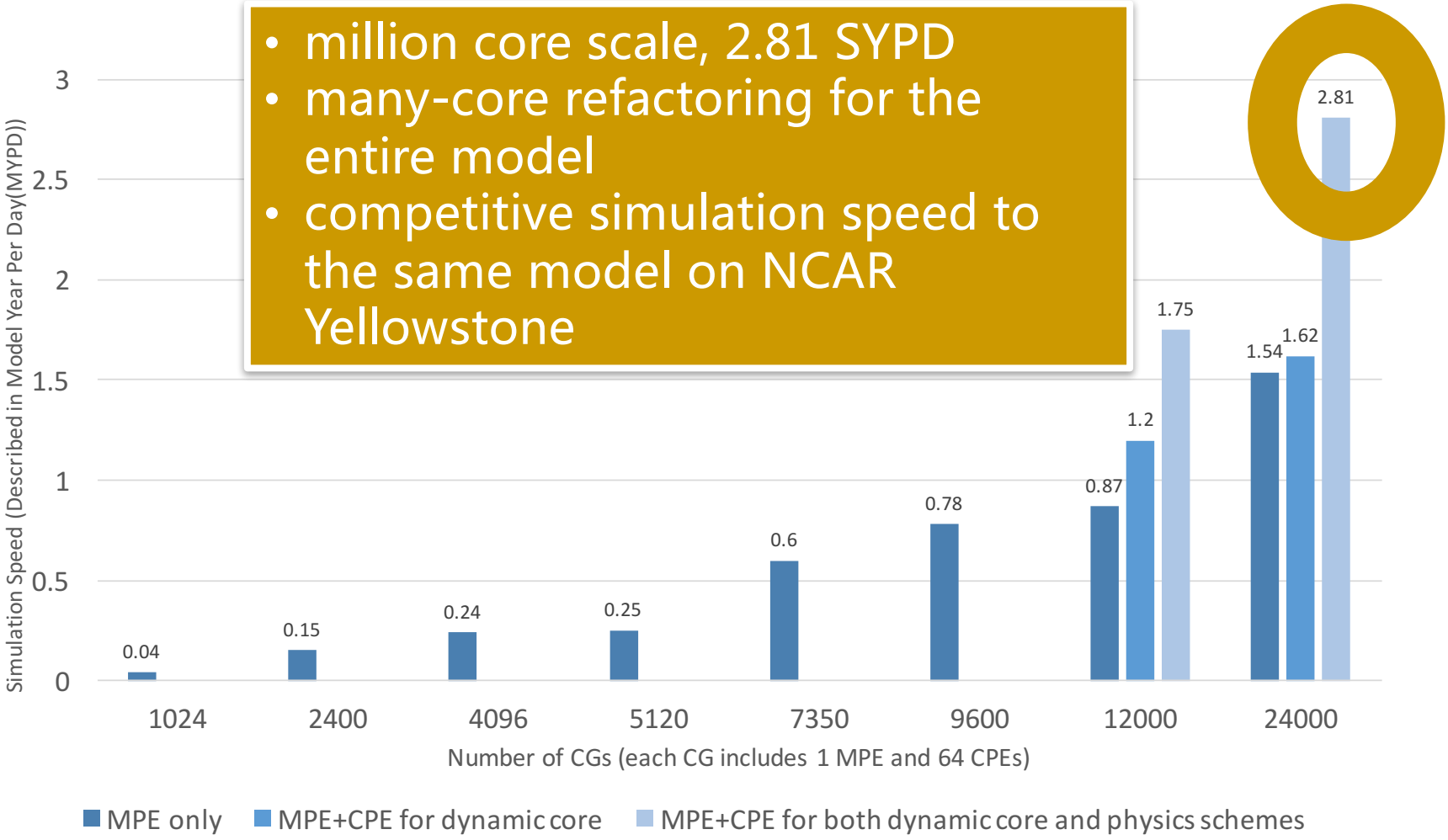
Speedup of Major Kernels in CAM-PHY

The *microp_mg1_0* kernel demonstrates a significant speedup of 14x, as the intermediate variables and arrays provide a nice fit to the SPM of the CPE clusters after the automated optimizations.



CAM model: scalability and speedup

- million core scale, 2.81 SYPD
- many-core refactoring for the entire model
- competitive simulation speed to the same model on NCAR Yellowstone



Future Work

Further improvement from 2.81 SYPD to 5~8 SYPD

dynamic core: by another factor of 2

- computation-communication overlapping
- data sharing among CPEs by register communication

physics schemes: by another factor of 2~4

- further improvement on the loop transformation and variable storage space reduction tool
- 20x speedup for most physics schemes

Sunway TaihuLight

Thank you.

haohuan@tsinghua.edu.cn