Preliminary Implementation of GRAPES global model on Sunway Taihu light

Zhiyan JIN
Numerical Weather Prediction Center, China Meteorological Administration

Wei Xue, Ping Xu, Hongsong Meng, Yongbin Jiang, Zhao Liu
Tsinghua University and National Supercomputing Center in Wuxi

18th Workshop on HPC in Meteorology, ECMWF, 24-28 September 2018

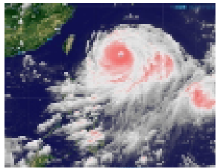
Outline

1. Introduction to GRAPES global model
2. Introduction to Sunway TaihuLight
3. Refactorization of critical computing kernels of GRAPES
   - ✓ Semi-Lagrange interpolation
   - ✓ Helmholtz solver
   - ✓ Halo communication optimization
   - ✓ Stencil kernel optimization
4. Optimizations of GRAPES with OpenACC*
5. Evaluation
6. Summary

Introduction of GRAPES global model

Meteorological Centers and Research Institutions in CMA Campus

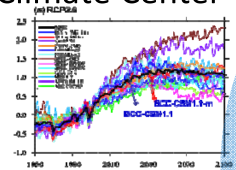
National Satellite Met. Center



National Met. Information Center

National Meteorological Center

National Climate Center



CMA

CMA Met. Observation Center

CMA Public Met. Service Center

CMA Weather Modification Center

CMA Numerical Weather Prediction Center

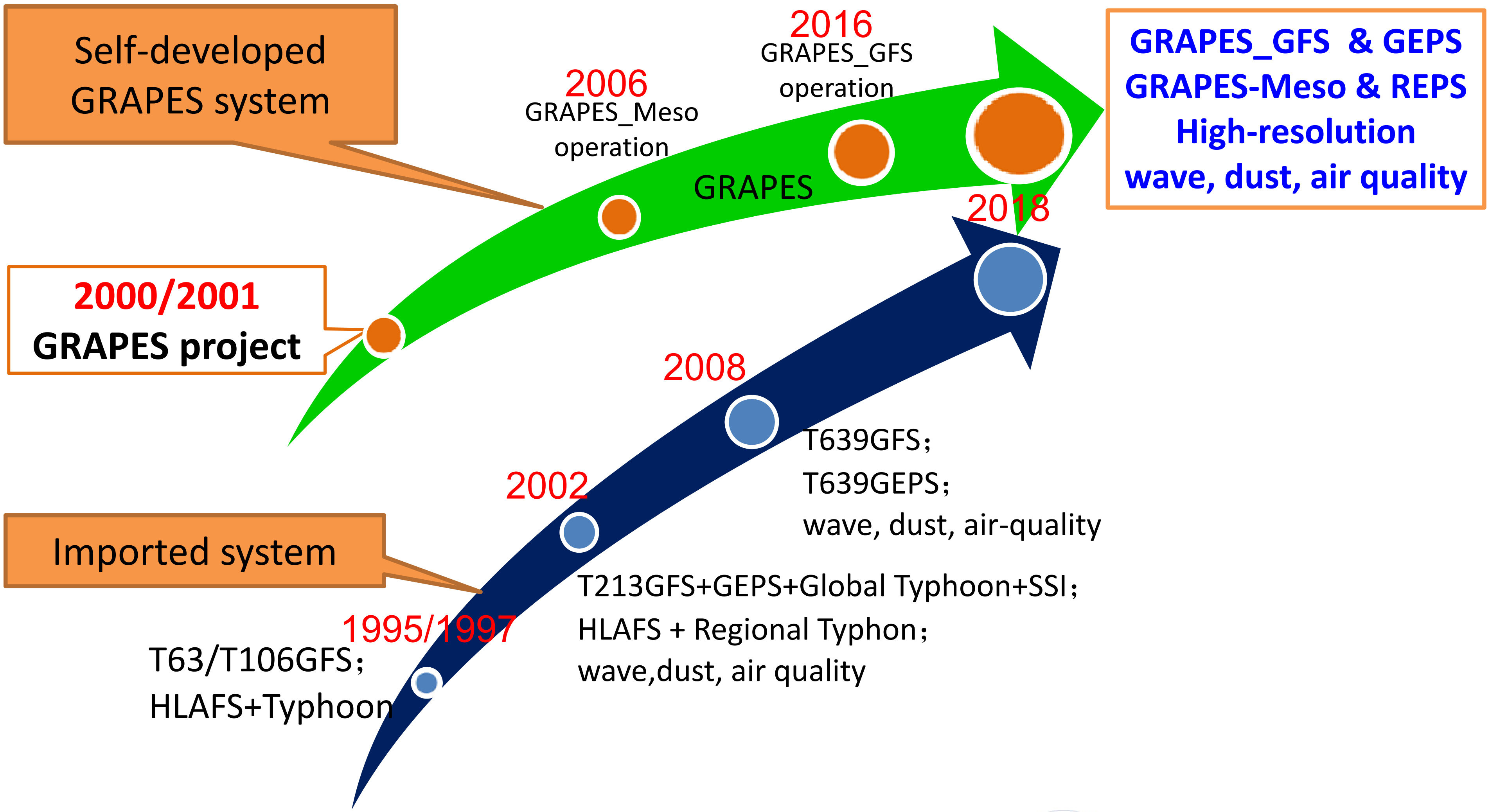
Chinese Academy of Met. Sciences

8 CMA Specialized Research Institutes

Mission: 0-10day weather fcst.

History of CMA NWP Operational systems

—from imported to self-developed core techniques/systems



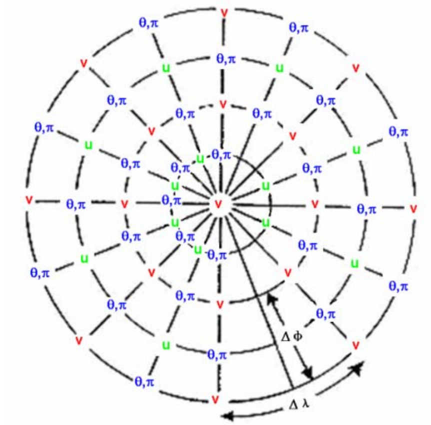
About GRAPES

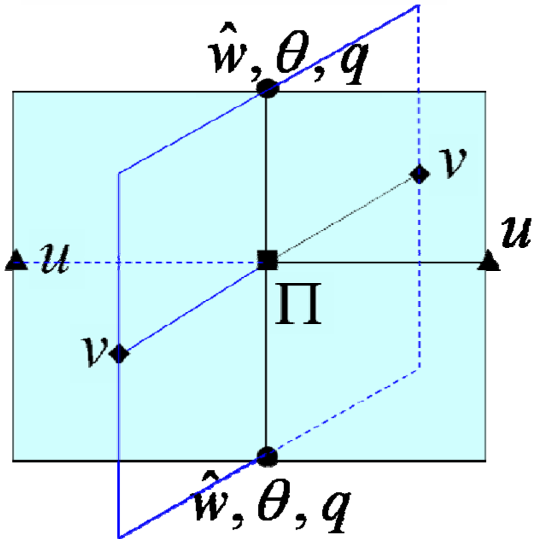
GRAPES
(Global/Regional Assimilation PrEdiction System)

- **Model**
  - Fully compressible equations with shallow atmosphere approximation
  - Regular Lat/lon, Arakawa-C with V at poles
  - Terrain-following Z, Charney-Phillips staggering
  - 2TL-SISL time integration
  - PRM scalar advection (conservation & monotonicity)
- **Assimilation**
  - Unified 3/4DVAR framework
  - Incremental analysis
  - Digital filter, initialization in 3DVAR, weak constraint in 4DVAR

GRAPES Dynamic Core

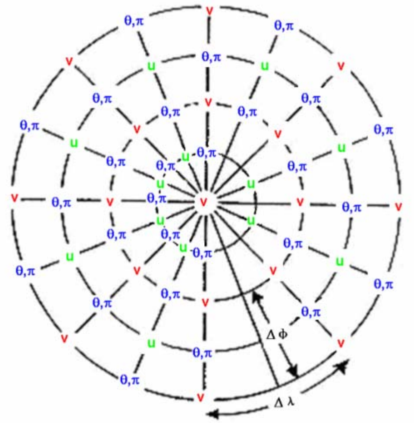
- **Fully compressible equations**
  - Height-based terrain-following coordinate
  - Option of Hydrostatic and Non-hydrostatic
  - Modified Arakawa C lat-lon horizontal grid
  - Charney-Phillips vertical grid
- **Off-centered 2-time-level semi-implicit semi-Lagrangian (SISL) time-stepping**
- **3D vector form of SISL formulation**
- **PRM for scalar advection**
- **Preconditioned GCR for Helmholtz Eq.**
- **Spherical & polar effects of trajectory calculation**
- **Polar filter**

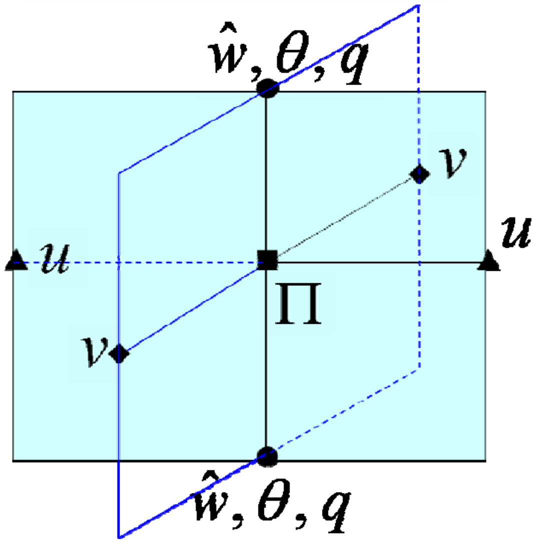




GRAPES Dynamic Core

- **Fully compressible equations**
  - Height-based terrain-following coordinate
  - Option of Hydrostatic and Non-hydrostatic
  - Modified Arakawa C lat-lon horizontal grid
  - Charney-Phillips vertical grid
- **Off-centered 2-time-level semi-implicit semi-Lagrangian (SISL) time-stepping**
- **3D vector form of SISL formulation**
- **PRM for scalar advection**
- **Preconditioned GCR for Helmholtz Eq.**
- **Spherical & polar effects of trajectory calculation**
- **Polar filter**

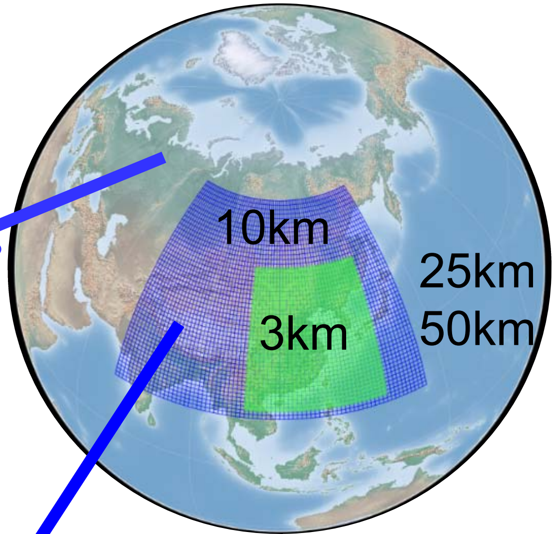




Physics package

- WRF physics for meso-scale application
- Physics for global forecast
  - Radiation:
    - RRTMG LW(v4.71)/SW(v3.61)
  - Cumulus:
    - Simplified Arakawa Schubert
  - Microphysics: CMA two-moment microphysics
  - Cloud: Prognostic
  - Land surface: CoLM
  - Gravity wave drag:
    - Kim & Arakawa 1995; Lott & Miller 1997; Alpert, 2004
  - Small scale orographic form drag : Beljaars, Brown & Wood(2004)

Operational NWP Configs at CMA



Global: GRAPES_GFS/GEPS

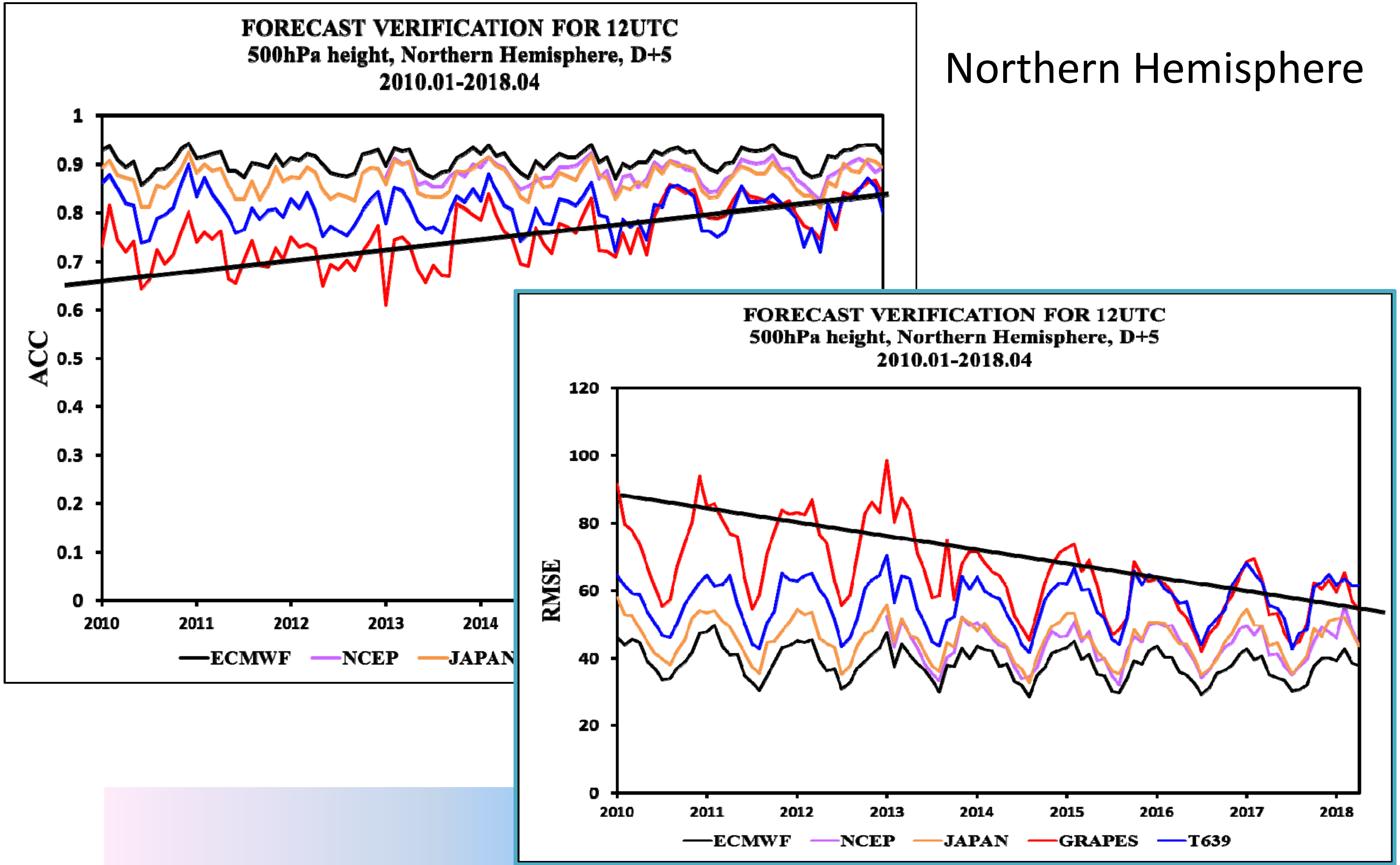
- 25/50km deterministic/ensemble(M30)
- 60 vertical levels (～3hPa top)
- 10-day forecast twice daily
- 4DVAR-100km inner loop

East Asia: GRAPES_Meso/REPS

- 3+10/10km deterministic/ensemble(M15)
- 50 vertical levels (～50hPa top)
- 24-hour(eight times/day)/120-hour forecast (two times/day)
- 3DVAR

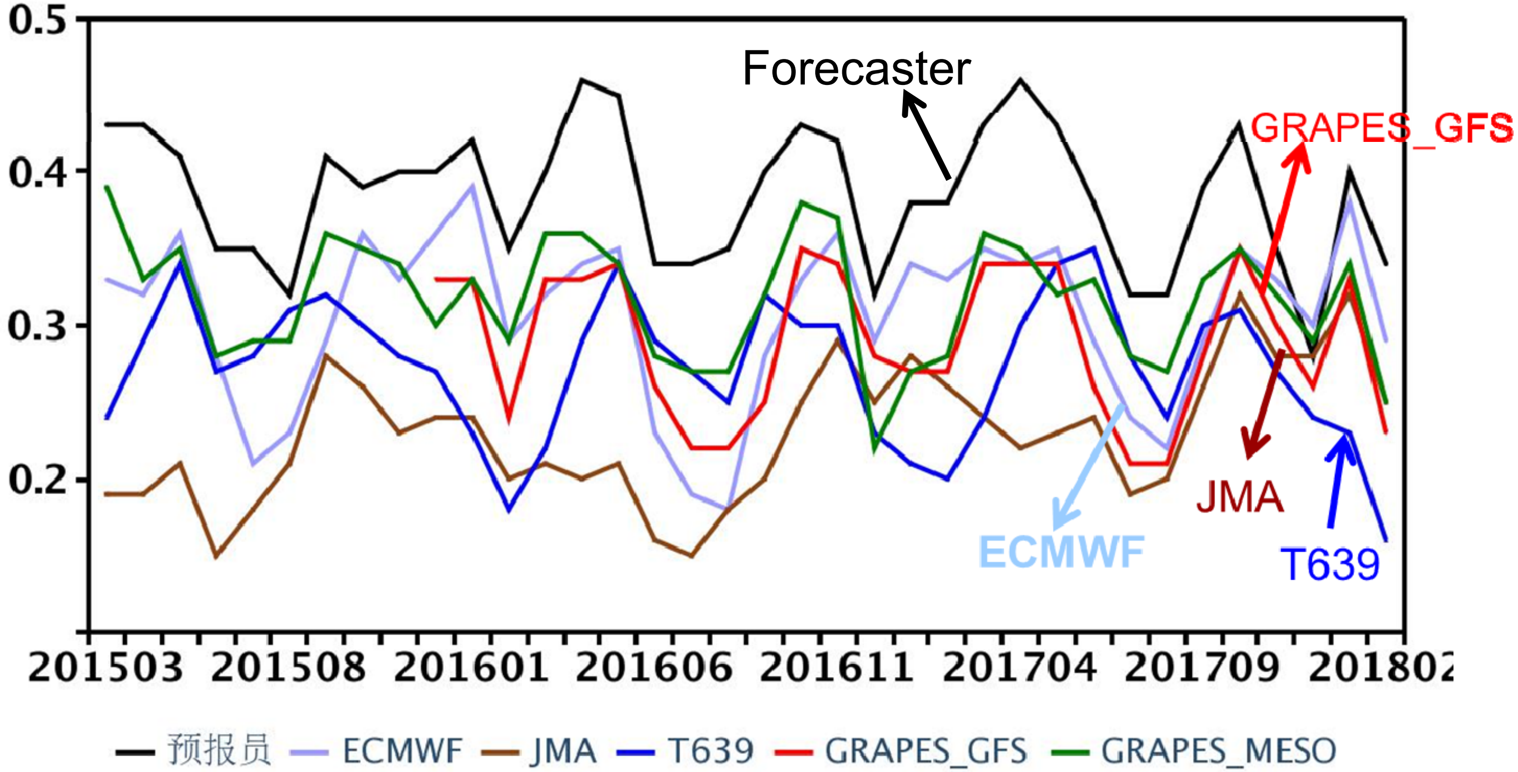
ACC & RMSE of 500hPa height 5-day forecast

Northern Hemisphere



Comparison of precipitation forecast over China among ECMWF, JMA & CMA GRAPES_GFS, Meso

ETS score of 48h forecast of rain belt

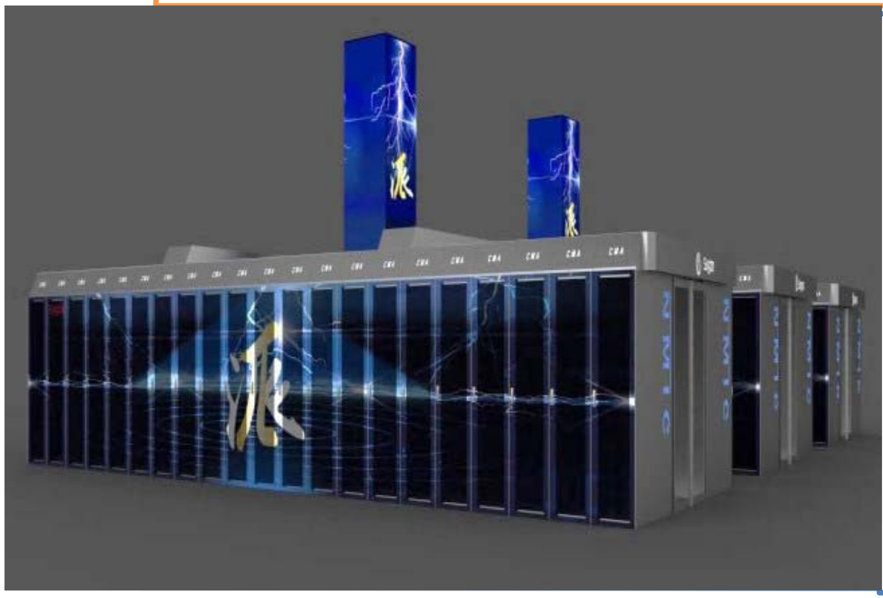


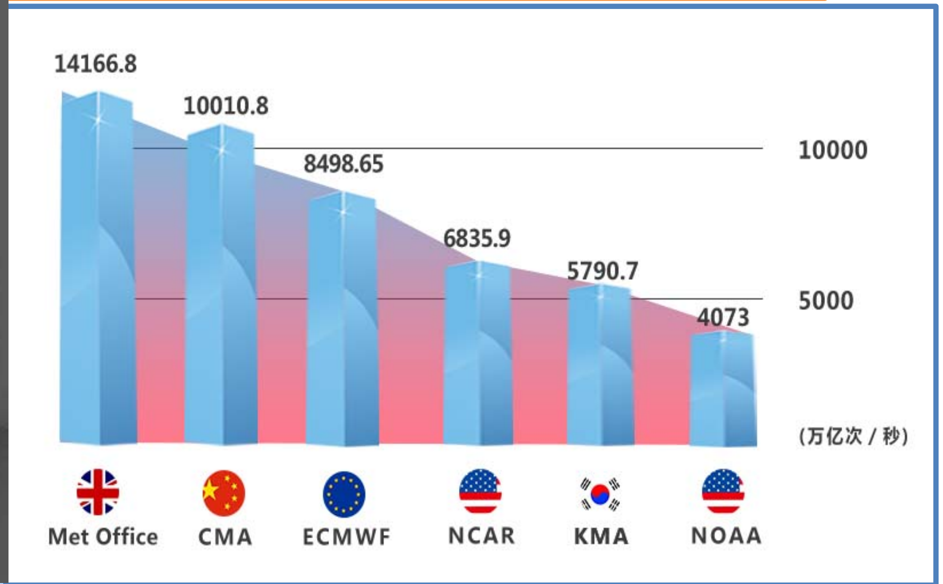
New CMA HPC

- 2 computers, peak performance: 8189.5 TFLOPS
- Parastor 300 storage: 23,088TB
- Node/CPU: 3076 nodes, 98432 cores
  - Based on Intel Xeon Gold 6124 (2.66GHz 16 cores) processor
  - 2 CPUs/node (16 cores/CPU)
- 100Gb/s EDR InfiniBand inter connection
- RedHat Enterprise Linux Server V7.4

and

- 4xIntel KNL 7250(68c 1.4GHz) X 6, 73.1TFlops
- 2xTesla P100 GPU X 24, 289.5TFlops



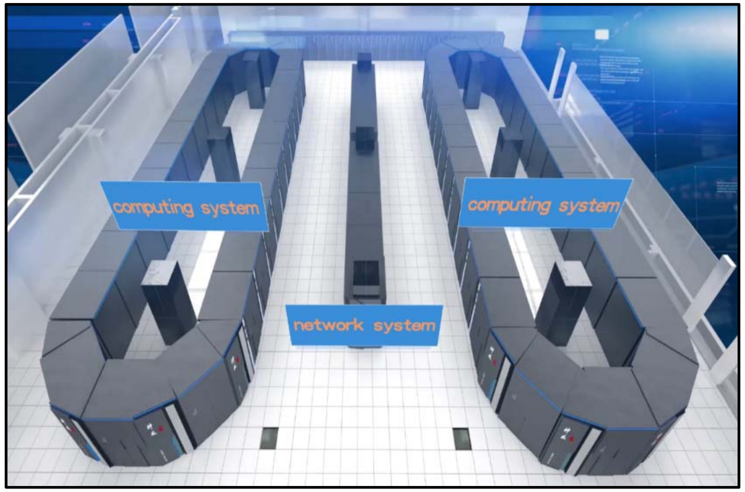


Heterogeneous and Many-core architecture becomes mainstream

System/ Launch date	Rpeak (PFLOPS)	Rmax (PFLOPS)	Power Efficiency (MFLOPS/W)	Cores	Architecture
Summit/ 201806	**187.66**	**122.30**	**13889.05**	**2,282,544**	**GPU Acceleration**
TaihuLight 201606	**125.44**	**93.02**	**6051.13**	**10,649,600**	**Heterogeneous many-core**
Sierra/ 201806	**119.19**	**71.61**	**/**	**1,572,480**	**GPU Acceleration**
Tianhe-2A 201806	**100.68**	**61.44**	**3324.56**	**4,981,760**	**Matrix2000 Acceleration**
ABCI/ 201806	**32.58**	**19.88**	**/**	**391,680**	**GPU Acceleration**
Piz Daint 201706	**25.33**	**19.59**	**8622.36**	**361,760**	**GPU Acceleration**
Titan 201211	**27.11**	**17.59**	**2142.77**	**560,640**	**GPU Acceleration**
Seq 201					

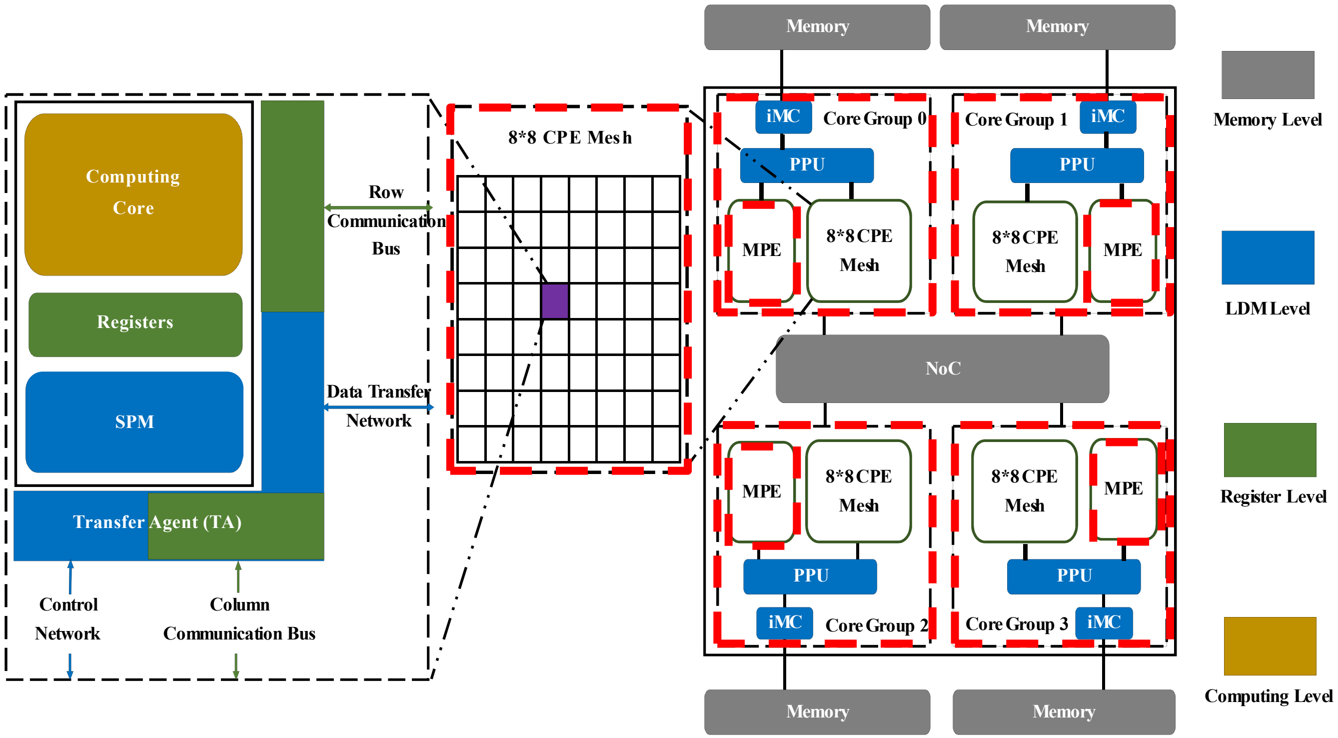
Evaluation of the potential of current operational model refactorization has to be done as early as possible

Target Platform: Sunway TaihuLight



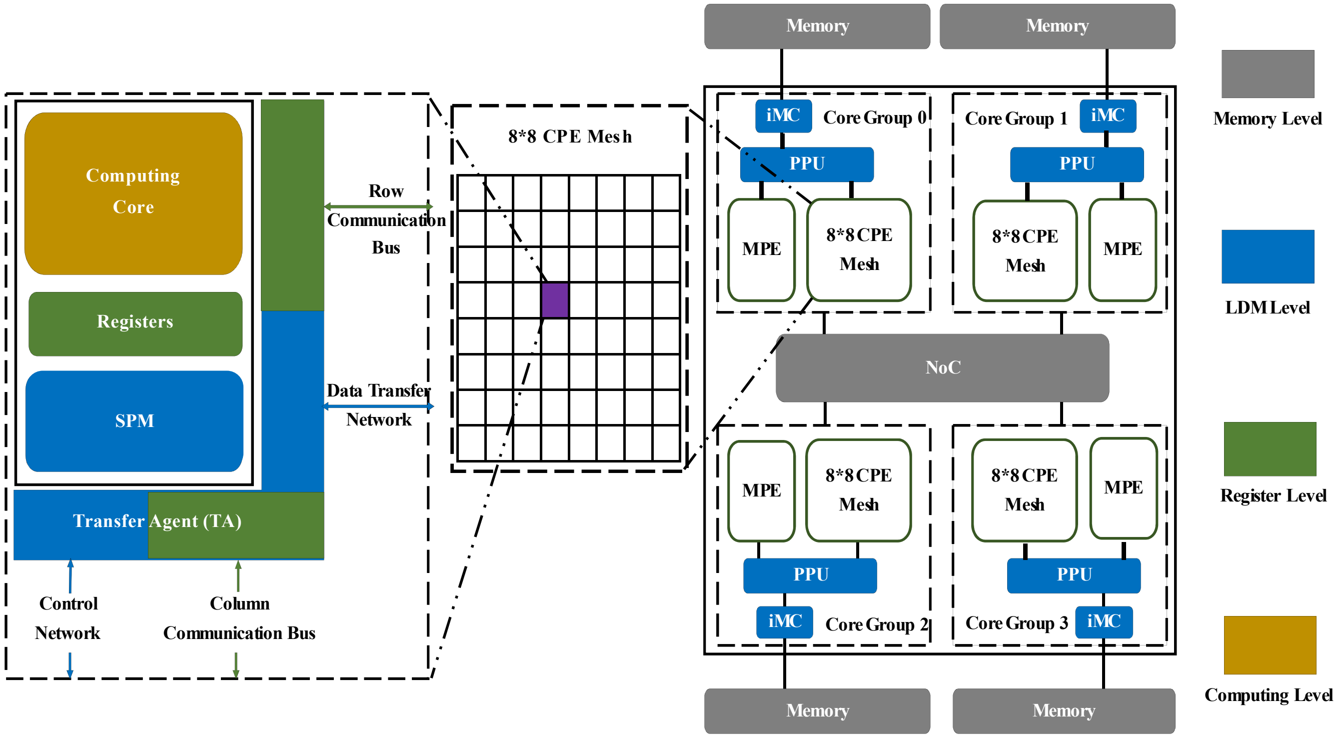
Entire System	
Peak Performance	125 PFlops
Linpack Performance	93 Pflops / 74.4%
Total Memory	1310.72 TB
Total Memory Bandwidth	5591.45 TB/s
# nodes	40,960
# cores	10,649,600

SW26010 Processor



Name	MPE	CPE
Frequency (GHz)	1.45	1.45
Cores	1	64
SIMD width	4 double / 8 integer	4 double / 8 integer
Peak Gflop/s in D.P.	23.2	742.4
Data L1 / L2 cache	32KB / 256KB	64KB (SPM) / -
Instruction L1 / L2 cache	32KB / +	16KB / 64KB
Memory capacity (GB)	32 (shared)	

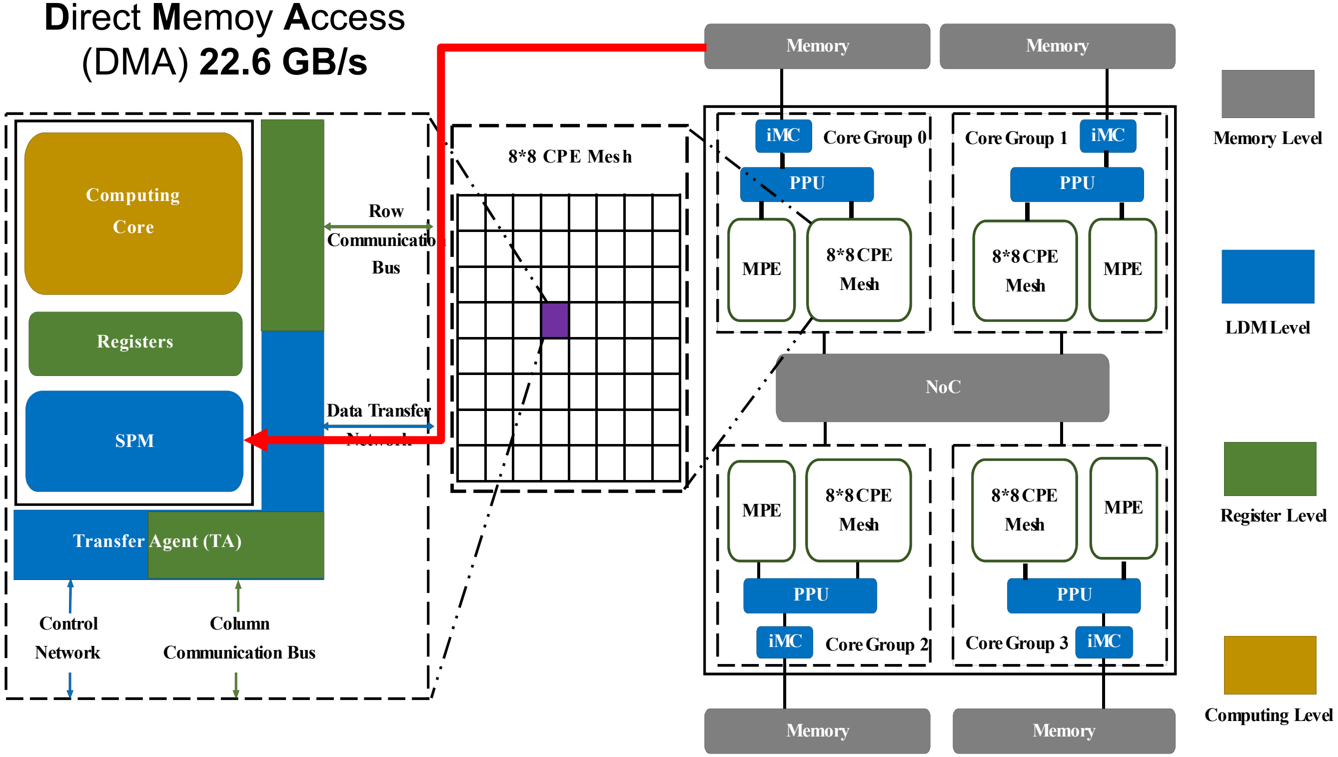
SW26010 Processor



Name	MPE	CPE
Frequency (GHz)	1.45	1.45
Cores	1	64
SIMD width	4 double / 8 integer	4 double / 8 integer
Peak Gflop/s in D.P.	23.2	742.4
Data L1 / L2 cache	32KB / 256KB	64KB (SPM) / -
Instruction L1 / L2 cache	32KB / +	16KB / 64KB
Memory capacity (GB)	32 (shared)	

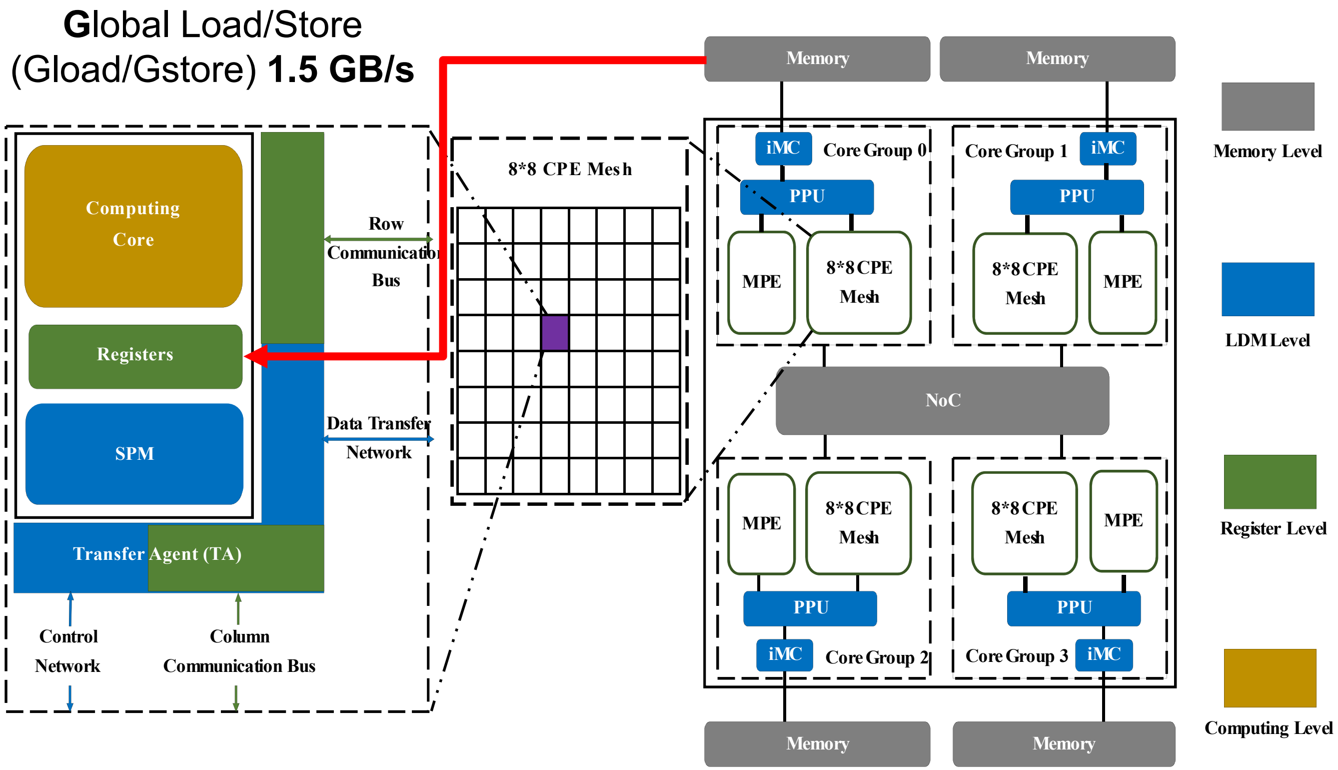
SW26010 Processor

Direct **M**emoy **A**ccess (DMA) **22.6 GB/s**



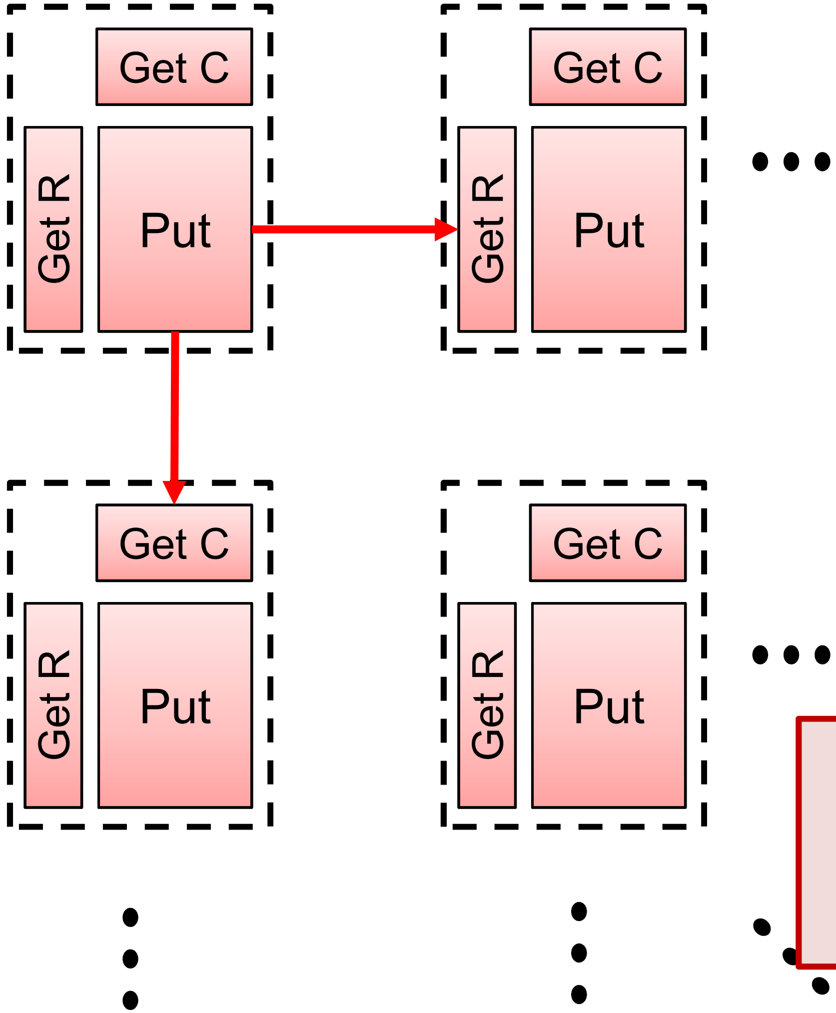
Name	MPE	CPE
Frequency (GHz)	1.45	1.45
Cores	1	64
SIMD width	4 double / 8 integer	4 double / 8 integer
Peak Gflop/s in D.P.	23.2	742.4
Data L1 / L2 cache	32KB / 256KB	64KB (SPM) / -
Instruction L1 / L2 cache	32KB / +	16KB / 64KB
Memory capacity (GB)	32 (shared)	

SW26010 Processor



Name	MPE	CPE
Frequency (GHz)	1.45	1.45
Cores	1	64
SIMD width	4 double / 8 integer	4 double / 8 integer
Peak Gflop/s in D.P.	23.2	742.4
Data L1 / L2 cache	32KB / 256KB	64KB (SPM) / -
Instruction L1 / L2 cache	32KB / +	16KB / 64KB
Memory capacity (GB)	32 (shared)	

Register Communication of SW26010




```
//P2P Test
if (id%2 == 0)
    while(1)
        putr(data,
id+1);
else
    while(1)

getr(&data);
```

Latency: less than **11** cycles
Integrated Bandwidth: **637** GB/s

Xu, Zhigeng, James Lin, and Satoshi Matsuoka. "Benchmarking SW26010 Many-Core Processor." *Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2017 IEEE International*. IEEE, 2017.

Summary of SW26010 Processor

- Heterogeneous architecture
- Manual cache system (SPM)
- Direct memory access (DMA)
- Limited register communication

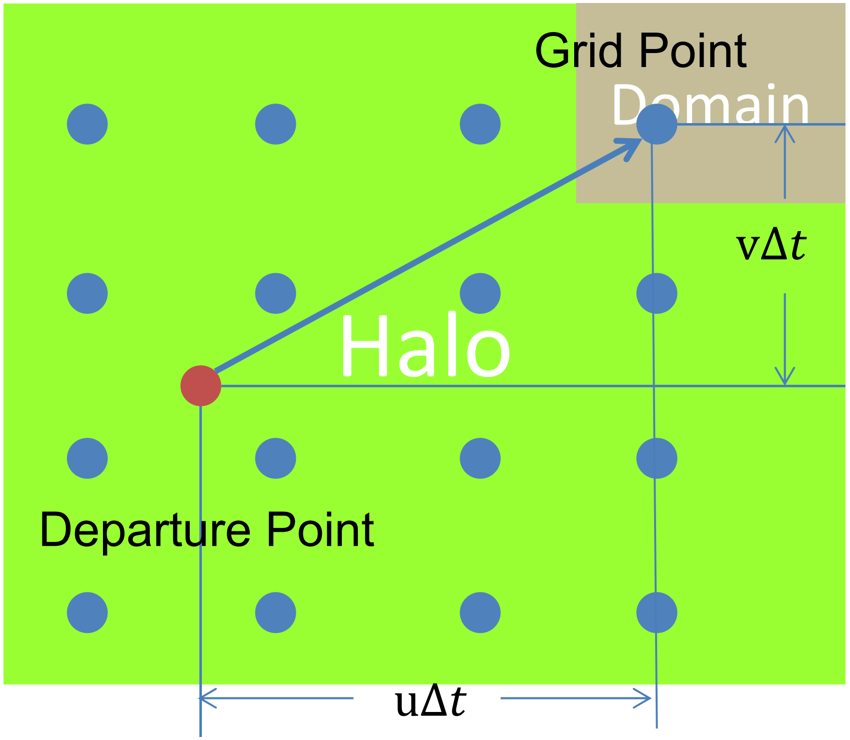
Different computing resources have been fully controlled by developers along with high development efforts

Principal Programming Model on TaihuLight

- MPI+X
  - X : OpenACC* / Athread
  - One MPI process manages to run on one management core (MPE)
  - OpenACC* is directive-based programming tool for SW26010
    - OpenACC2.0 based
    - Extensions for the architecture of SW26010
    - Supported by SWACC/SWAFORT compiler
    - OpenACC* conducts data transfer between main memory and on-chip memory (SPM), and distributes the kernel workload across compute cores (CPEs)
  - Athread is the threading library to manage thread on compute core (CPE), which is used in OpenACC* implementation

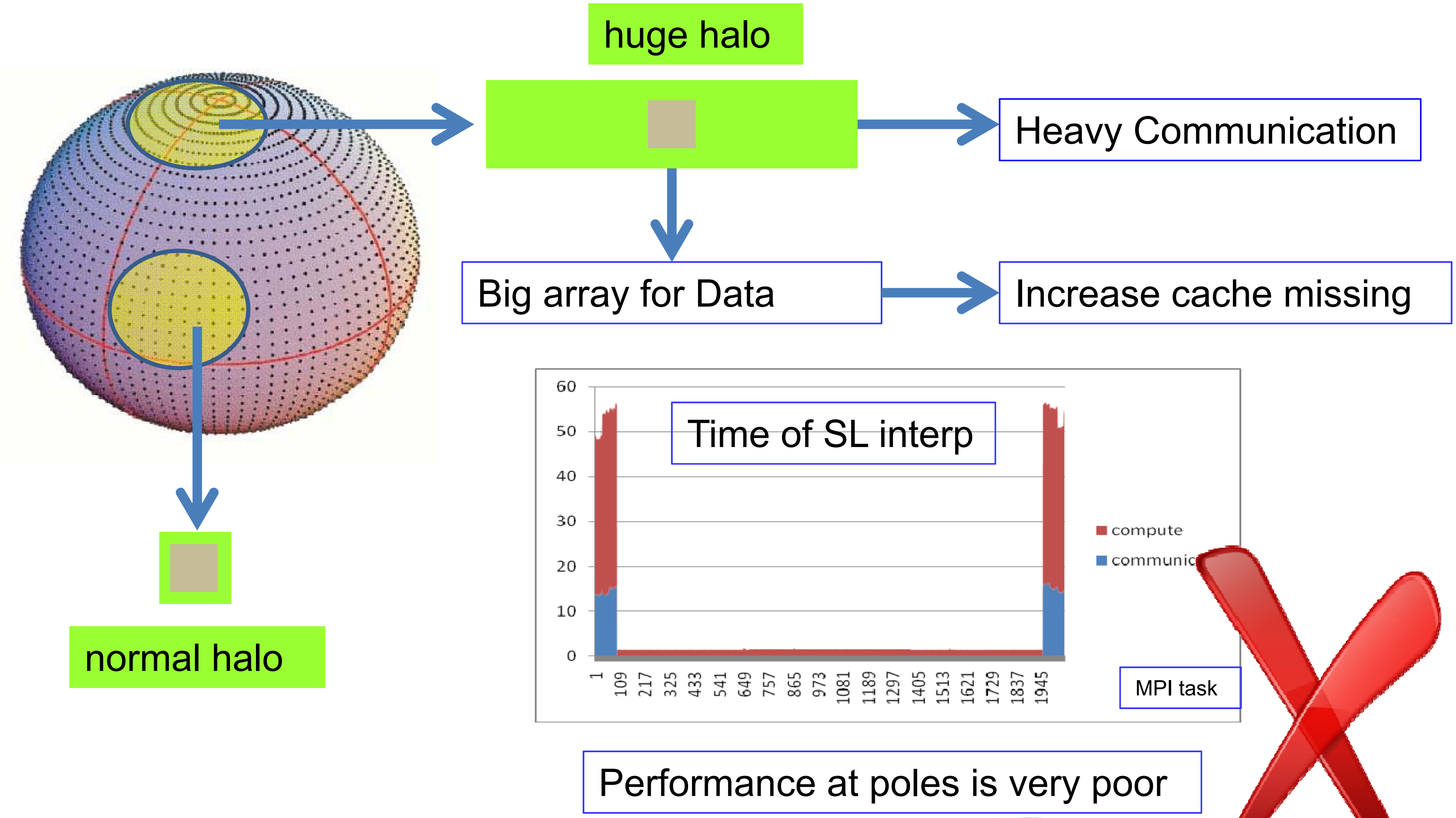
Semi-Lagrange Interpolation

Semi-Lagrange Interpolation

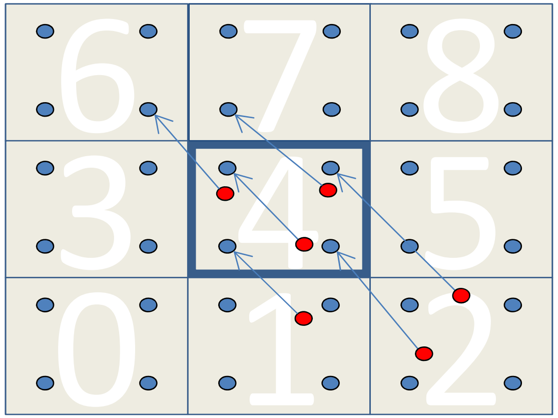


If the departure point is outside the domain,
halo is needed for interpolation

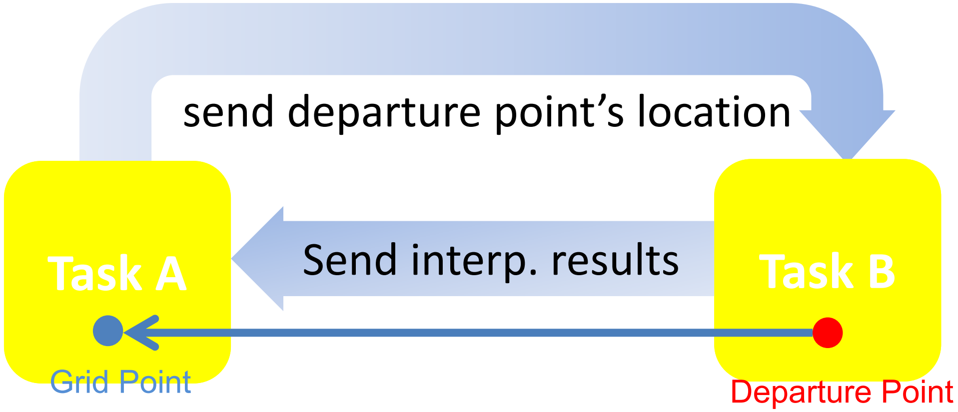
Semi-Lagrange interpolation

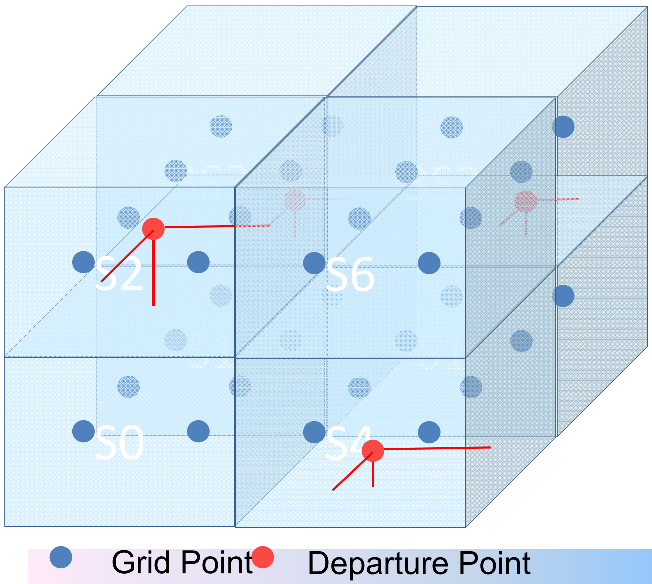


Semi Lagrange departure point interpolation



MPI task partition for grid and departure points

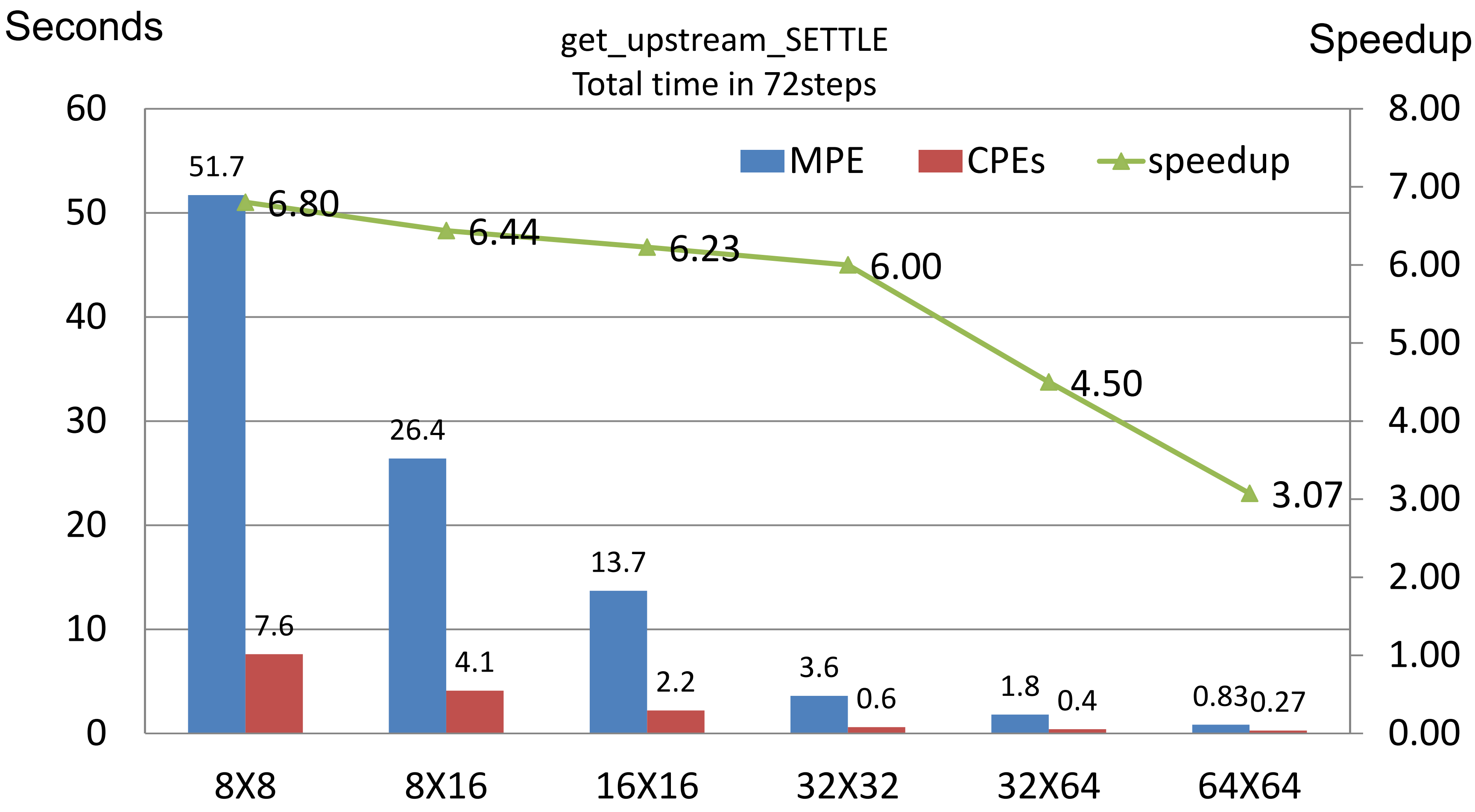




thread partition in a MPI task

The grid points and departure points in a MPI task is partitioned both in horizontal & vertical dimensions, small enough to fit into the 64k bytes SPM of a CPE. Subdomain have small halo needed by interpolation.

Semi-Lagrange Interpolation

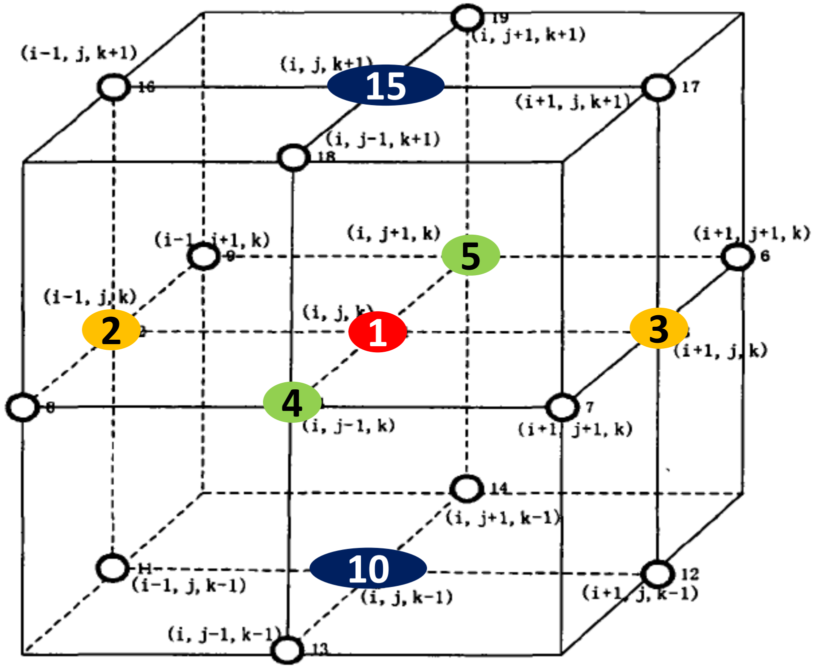


GRAPES global, 0.5° ,DP,Taihu light, subroutine “get_upstream_SETTLE”, total time in 72 steps

Helmholtz Solver

Characteristics of Helmholtz Eq.

- Math. Model & Matrix Characteristics
  - Large Scale non-symmetrical Linear Equations for Globe
    - 25km H-resolution, 1440x720x36=37,324,800
  - 19-diagnal Coefficient Matrix
    After scaling with diagonal elem.
    - C1 = 1.0
    - C10/C15 ~ 10^{-1}
    - C2/C3 ~ 10^{-2}
    - C4/C5 ~ 10^{-3}
    - Others <= 10^{-5}
  - Not good distribution of eigenvalues
    - 100km H-resolution, max/min ~ 3×10^{4}



Improved pre-GCR algorithm (IGCR)

- Preconditioning with Restricted Additive-Schwarz domain decomposition scheme
  - Only one overlapping layer is enough
  - Additional halo update is introduced
- Improved GCR algorithm for strength reduction and communication reduction

```
Compute R_0 = b − A x_0, R̂_0 = M^{-1} R_0, p_0 = R̂_0
Do i = 1, 2, ..., until convergence
    α_{i-1} = ⟨R_{i-1}, Ap_{i-1}⟩ / ⟨Ap_{i-1}, Ap_{i-1}⟩
    x_i = x_{i-1} + α_{i-1} p_{i-1}
    R_i = R_{i-1} − α_{i-1} Ap_{i-1}
    R̂_i = M^{-1} R_i
    Do j = int[(i − 1)/k]k, ..., i − 1
        β_{ij} = −⟨AR̂_i, Ap_j⟩ / ⟨Ap_j, Ap_j⟩
    EndDo
    p_i = R̂_i + Σ_{j=int[(i−1)/k]k}^{i−1} β_{ij} p_j
    Ap_i = AR̂_i + Σ_{j=int[(i−1)/k]k}^{i−1} β_{ij} Ap_j
EndDo
```

$$(r_{j+1}, Ap_i) = (r_j, Ap_i) - \alpha_j (Ap_j, Ap_i) =$$

$$(r_{j-1}, Ap_i) - \alpha_{j-1} (Ap_{j-1}, Ap_i) =$$

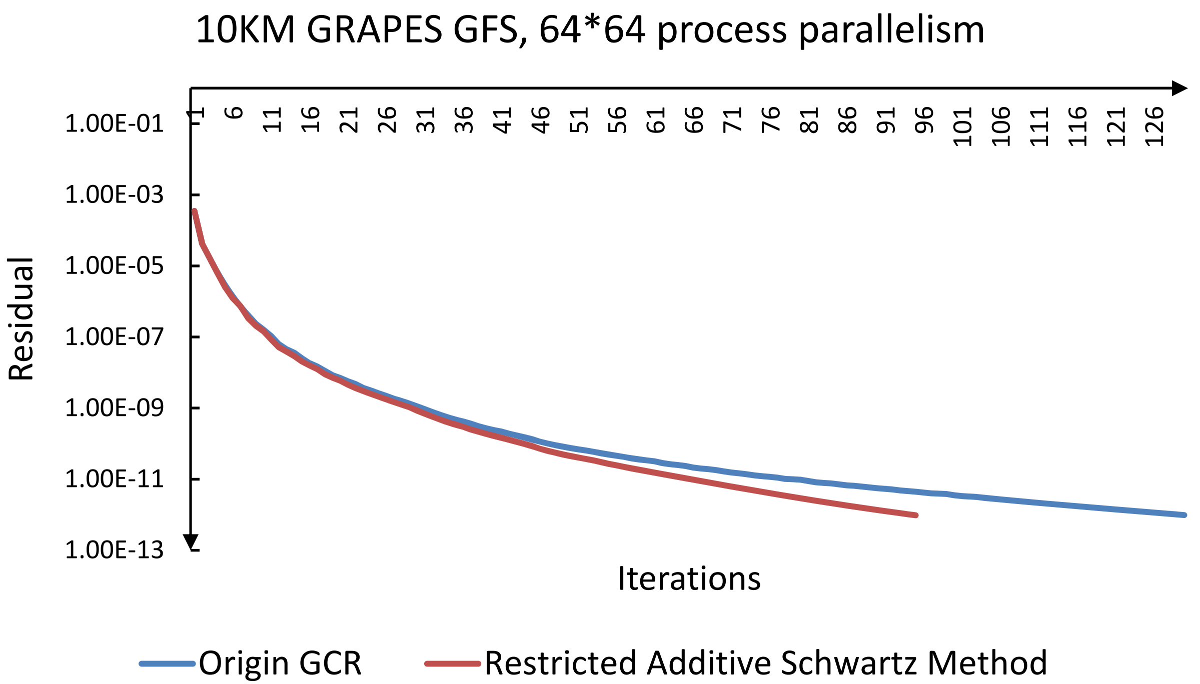
$$(r_i, Ap_i) - \alpha_i (Ap_i, Ap_i) = 0 \qquad i \leqslant j$$

Compare to Baseline Algorithm (k=10)

- ✓ (-) 1 Allreduce
- ✓ (-) 1 SpMV
- ✓ (+) k BLAS1

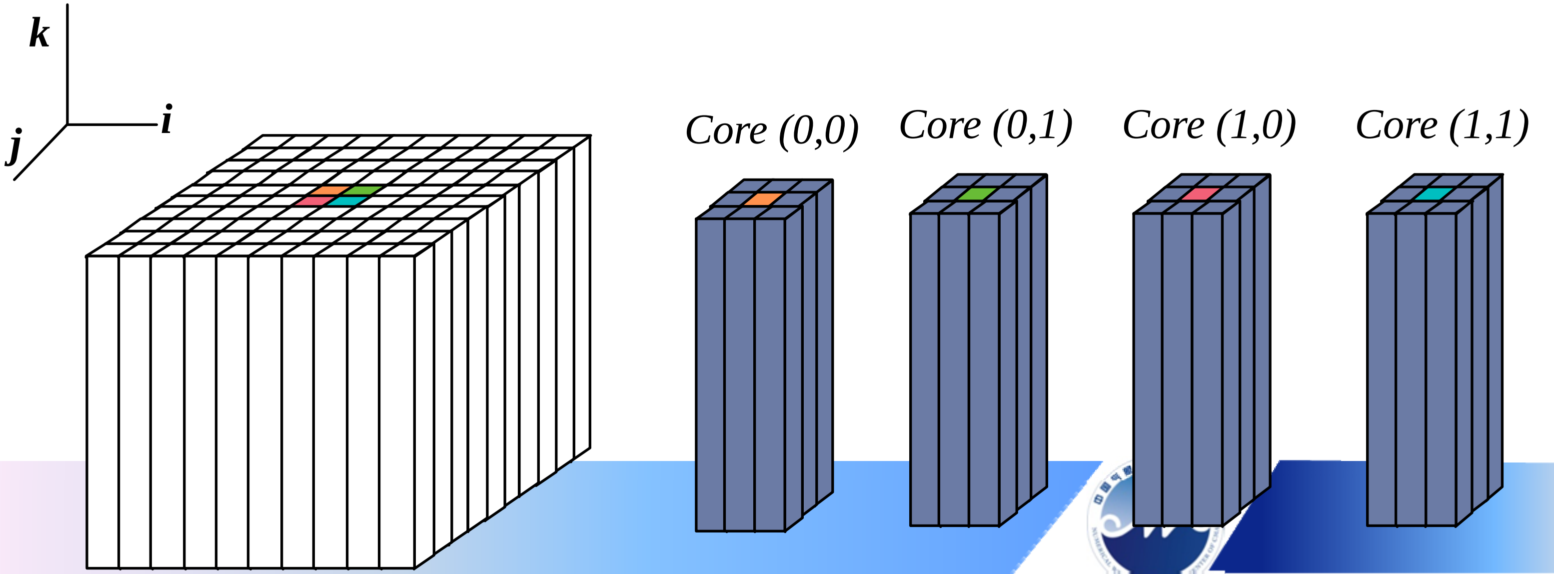
Convergence improvement

- Convergence of RAS method and origin method



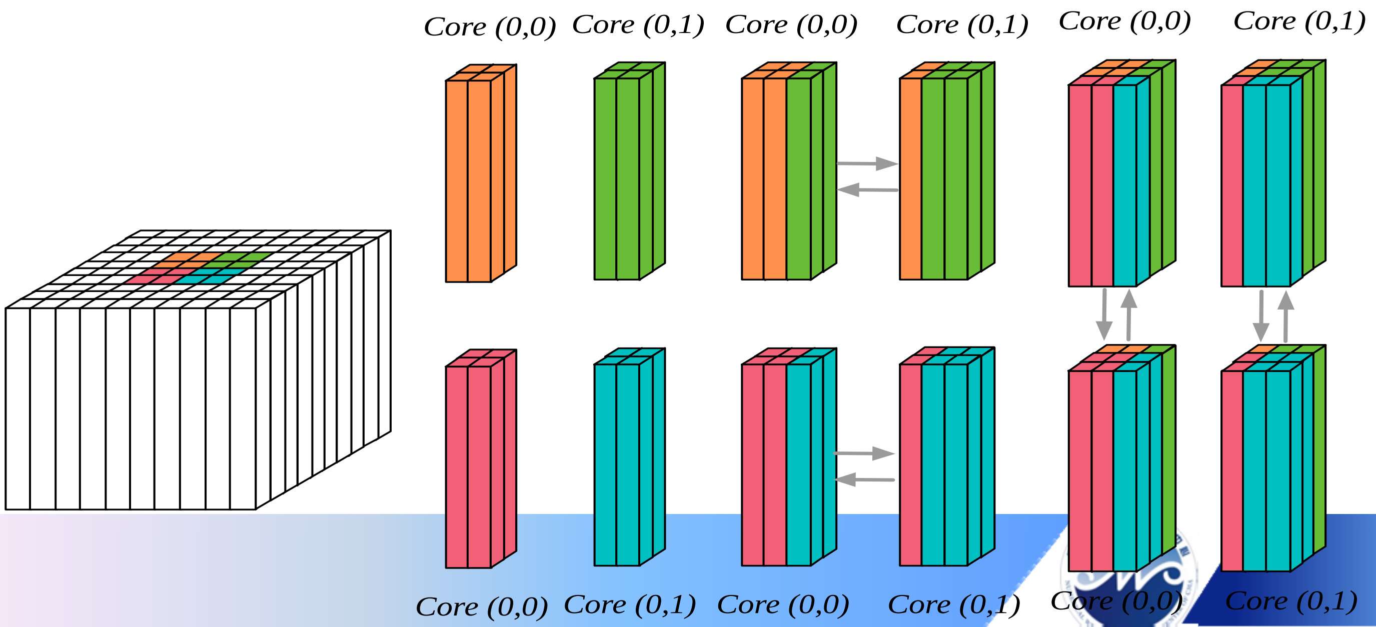
Sparse Matrix-vector Multiplication for SW26010 processor

- Chunk access of a k column
  - Matrix: continuous storage with (19, k, i, j) order and bulk access of 19 nonzeros of each raw of matrix
  - Vector: stride access following the geometry
- For vector side, each core should read 9 column data to compute one column



Sparse Matrix-vector Multiplication for SW26010 processor

- For computation of each column, 9 column data has to be access for each core
- Collaborated data access by multiple cores reduces the data access volume in total but introduces more synchronizations
- Good trade-off: 2*2 Cores share data with each other by register communication, each core only need to read 4 column data



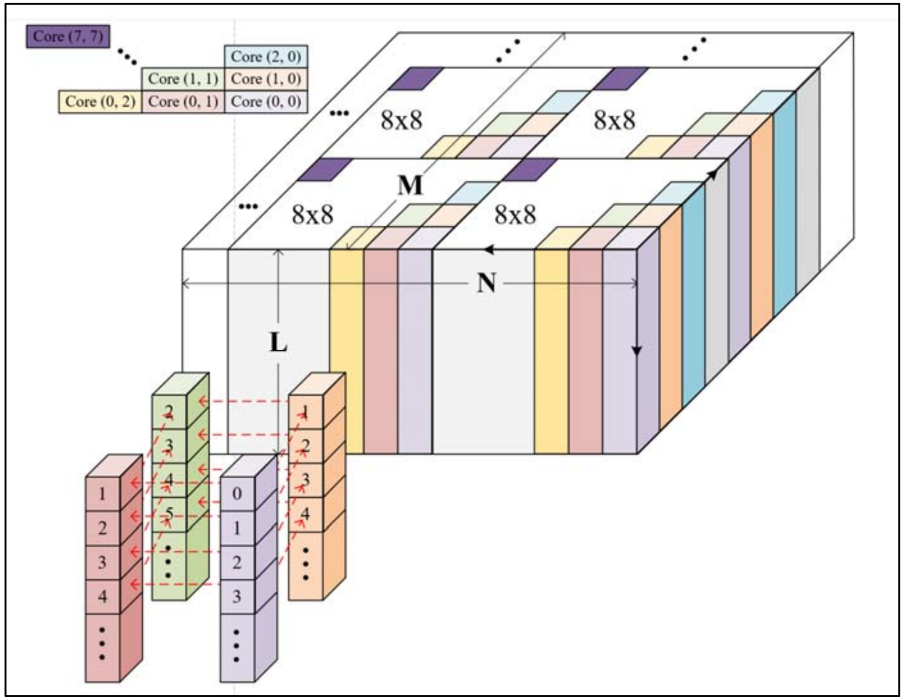
Sparse Matrix-vector Multiplication for SW26010 processor

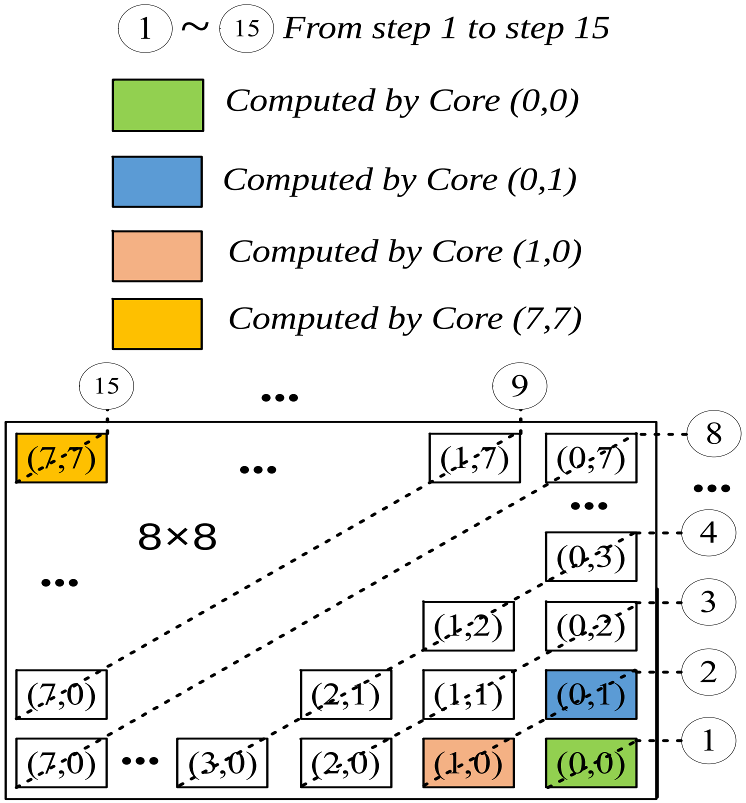
- More Cores share data with each other, less x need to read.
- Cores which share data with each other need synchronization, more Cores, more expensive for synchronization.

Groups	Average Columns of data to read per Core	Cores to synchronization
1*1	9	0
2*2	4	4
4*4	9/4	16
8*8	25/16	64

Fine-grained parallel Incomplete LU factorization for SW26010 processor

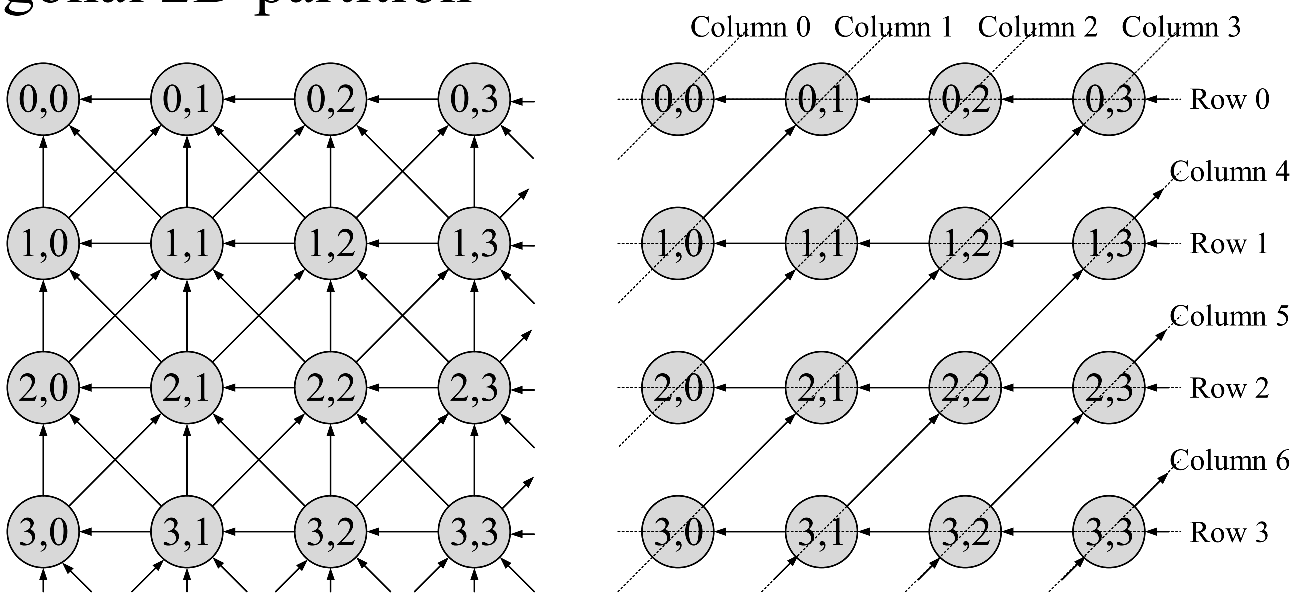
- Example of 7 points ILU





Fine-grained parallel Incomplete LU factorization for SW26010 processor

- How to handle diagonal diagonal communication on SW26010?
  - Reduction in communication path graph
  - Diagonal 2D partition



Communication-Avoiding GCR for Sunway TaihuLight

Algorithm： communication avoiding preconditioned generalized conjugate residual method

$x_0, r_0 = b - Ax_0, z_0 = M^{-1}r_0, p_0 = z_0$

while not converged do

1, $Calculate\ \boldsymbol{V} = [\boldsymbol{p_0}, \boldsymbol{M^{-1}Ap_0}, (\boldsymbol{M^{-1}A})^2 \boldsymbol{p_0}, ..., (\boldsymbol{M^{-1}A})^s \boldsymbol{p_0}, \boldsymbol{z_0}, \boldsymbol{M^{-1}Az_0}, ..., (\boldsymbol{M^{-1}A})^{s-1} \boldsymbol{z_0}]$

2, $Let\ \boldsymbol{G} = \boldsymbol{V}^T \boldsymbol{A}^T \boldsymbol{AV}, \boldsymbol{G_m} = \boldsymbol{V}^T \boldsymbol{M}^T \boldsymbol{AV}$

3, $\boldsymbol{m}_0 = [0_{s+1}, 1, 0_{s-1}]^T, \boldsymbol{n}_0 = [1, 0_{2s}]^T, \boldsymbol{l}_0 = [0_{2s+1}]^T$

4, $for\ k = 0 : s-1,\ do$

5, $\alpha_k = \boldsymbol{m_k^T G_m n_k} / \boldsymbol{n_k^T G n_k}$

6, $\boldsymbol{l_{k+1}} = \boldsymbol{l_k} + \alpha_k \boldsymbol{n_k}$

7, $\boldsymbol{m_{k+1}} = \boldsymbol{m_k} - \alpha_k \boldsymbol{T n_k}$

8, $\beta_j^k = -\boldsymbol{m_{k+1}^T G n_j} / \boldsymbol{n_j^T G n_j}$

9, $\boldsymbol{n_{k+1}} = \boldsymbol{m_{k+1}} + \sum_{j=0}^{k} \beta_j^k \boldsymbol{n_j}$

10, $end\ for$

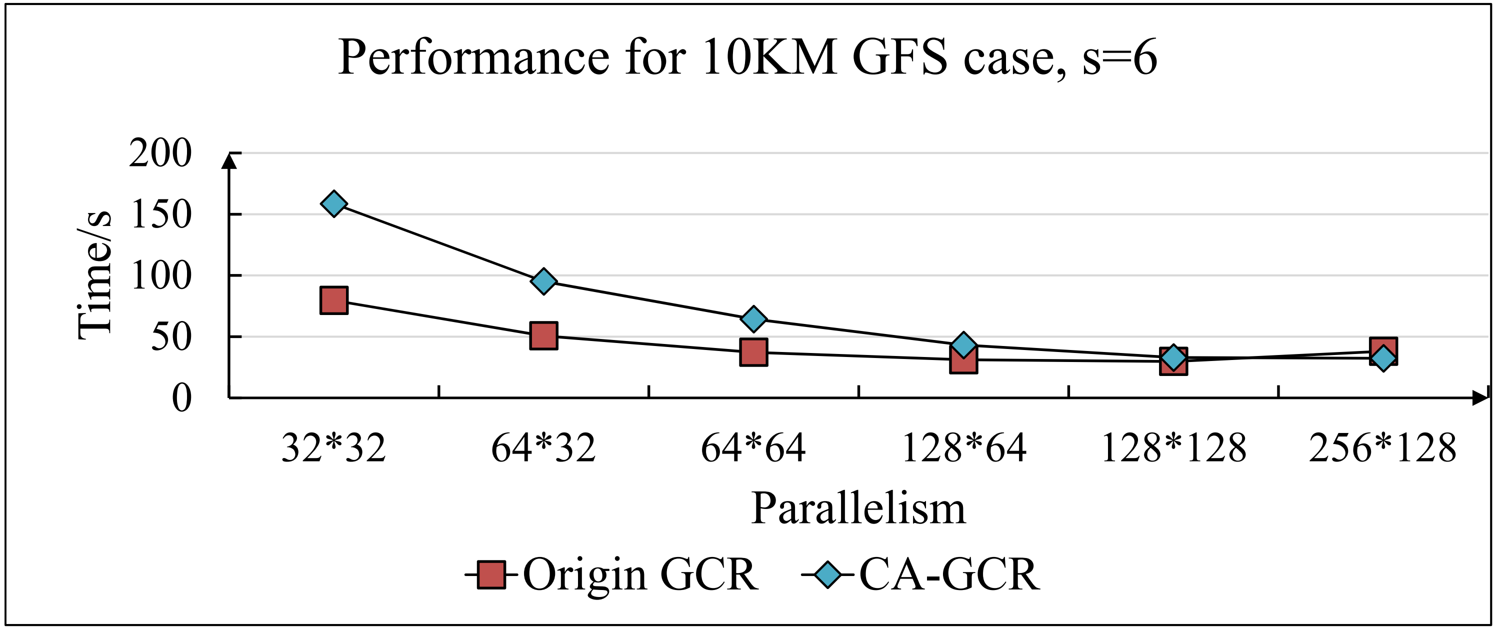
11, $\boldsymbol{z_s} = \boldsymbol{V m_s}, \boldsymbol{p_s} = \boldsymbol{V n_s}, \boldsymbol{x_s} = \boldsymbol{x_0} + \boldsymbol{V l_s}$

12, $\boldsymbol{z_0} = \boldsymbol{z_s}, \boldsymbol{p_0} = \boldsymbol{p_s}$

end while

Communication-Avoiding GCR for Sunway TaihuLight

- Some computing kernels still have some room for optimization
- If the scale continues to expand, CA-GCR may beat origin GCR.



Halo Communication Optimization

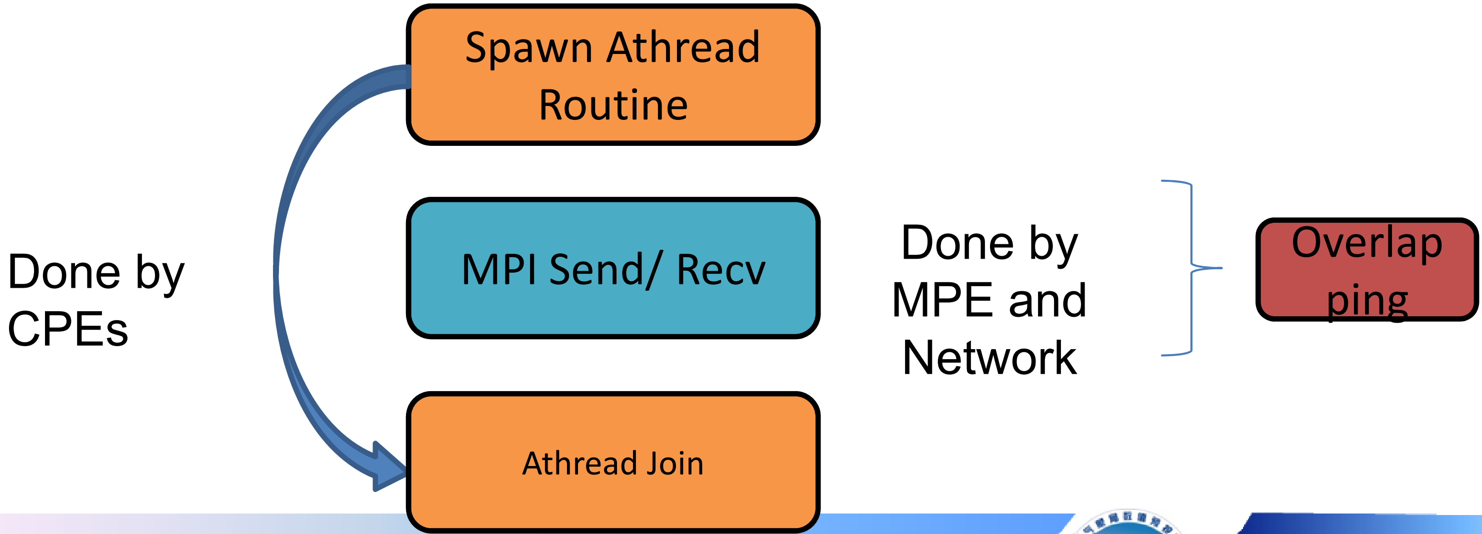
Halo communication optimization

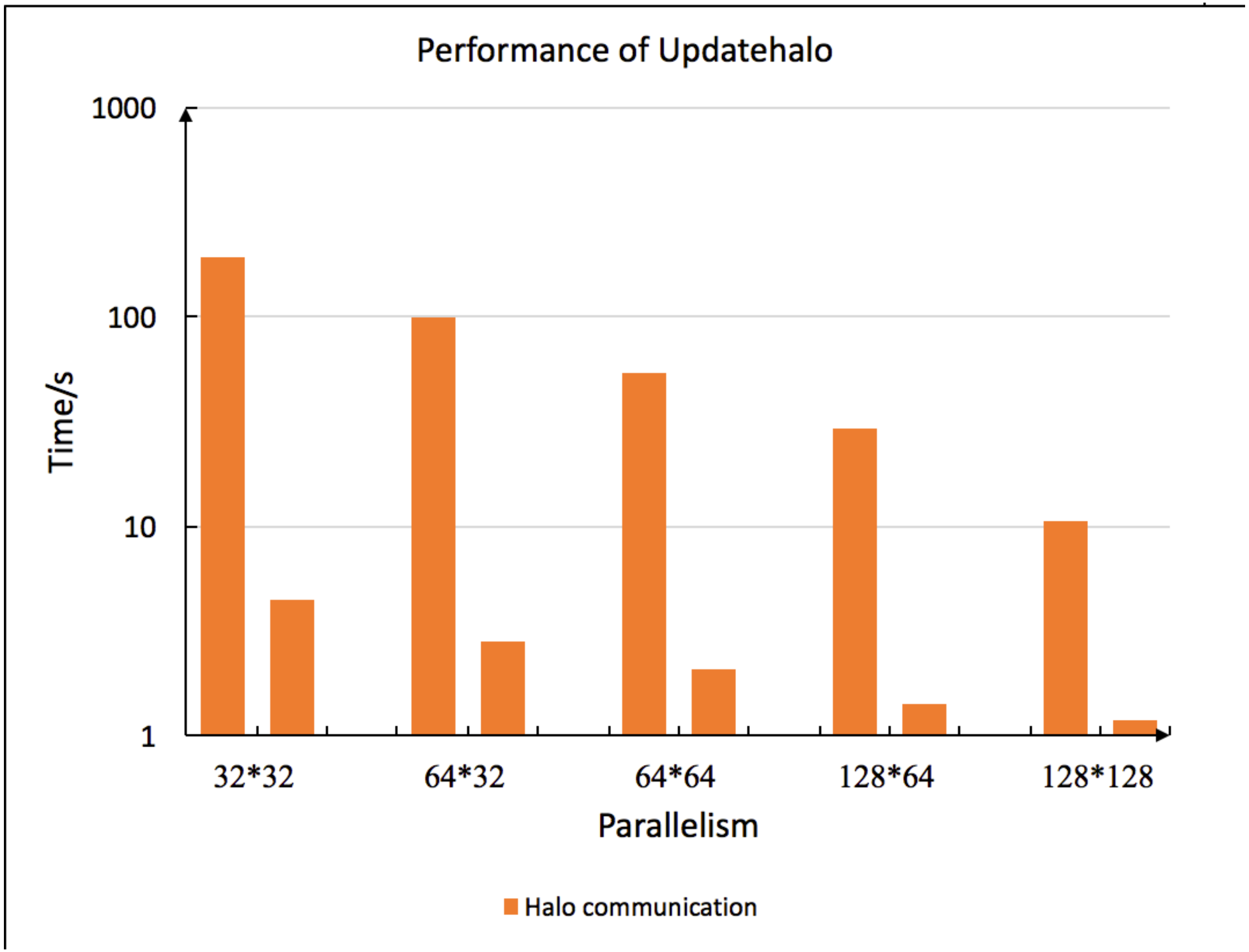
- Array transpose
  - Domain partition on i and j dimensions
  - (i , k, j) order -> (k, i, j) order for DMA friendly data access
  - The conversion can be conducted at the beginning and at the end of loop code to minimize memory access overhead.

- Assign task to partial CPEs by column
  - 64-core simultaneous access is over-provisioning for memory subsystem of SW26010 if good access pattern
  - Fewer core access makes larger chunk data access, thus better bandwidth utilization

Halo communication optimization

- Quite a few halo communications may hurt performance
- Solution:
  - Offloading the data package to CPE cluster
  - Performing data package and communication overlapping



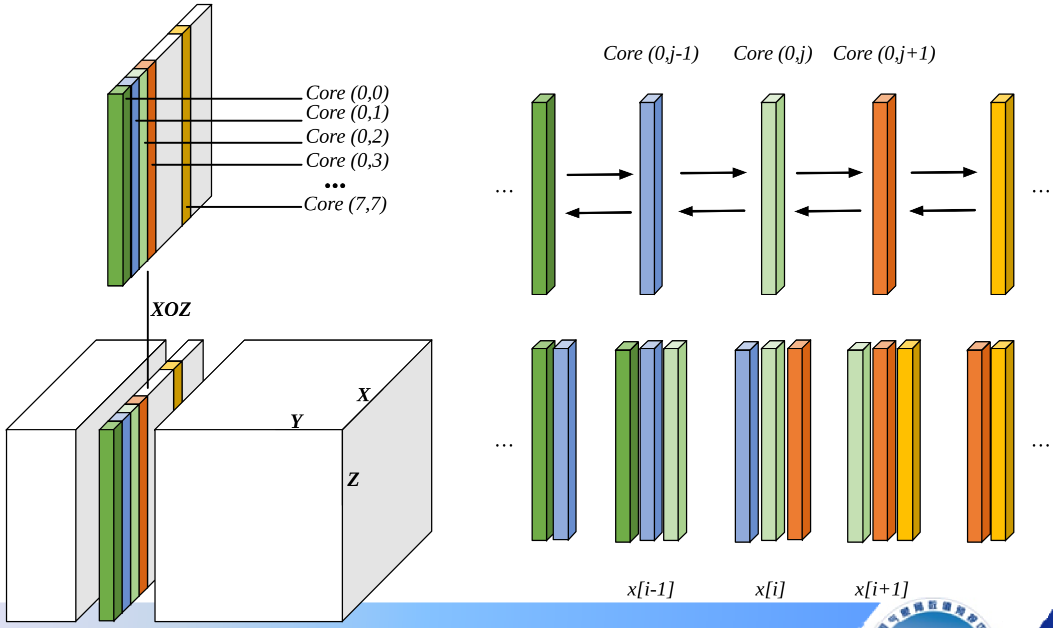


Performance of Updatehalo
1000
100
10
1
Time/s
32*32
64*32
64*64
128*64
128*128
Parallelism
Halo communication

Stencil-like kernel Optimization

Stencil kernel optimization (x-axis stencil case)

- Neighbour CPEs can share the dependent data by register communication rather than reading from memory



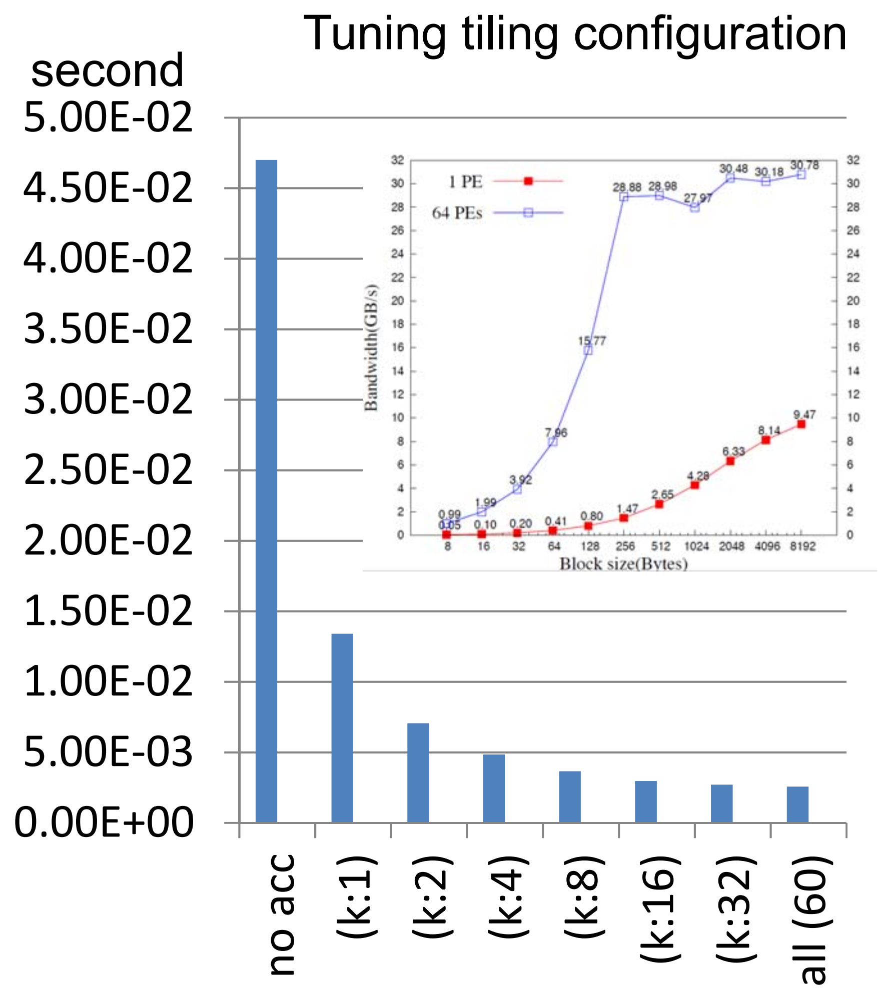
Exploring thread-level parallelism of GRAPES with OpenACC*

OpenACC* optimization on TaihuLight

```
!$acc parallel loop &
!$acc copyout(c)
!$acc collapse(2) tile(k:16)
DO j = 1, 45
    DO k= 1,60
        DO i =1, 90
            c(i,k,j) = 0.
         END DO
    END DO
END DO
!$acc end parallel loop
….
!$acc parallel loop copyin(A) &
!$acc copyout(b)
!$acc collapse(2) tile(k:16)
DO j = 1, 45
    DO k= 1,60
        DO i =1, 90
            A(i,k,j) = B(i,k,j)
         END DO
    END DO
END DO
!$acc end parallel loop
….
```

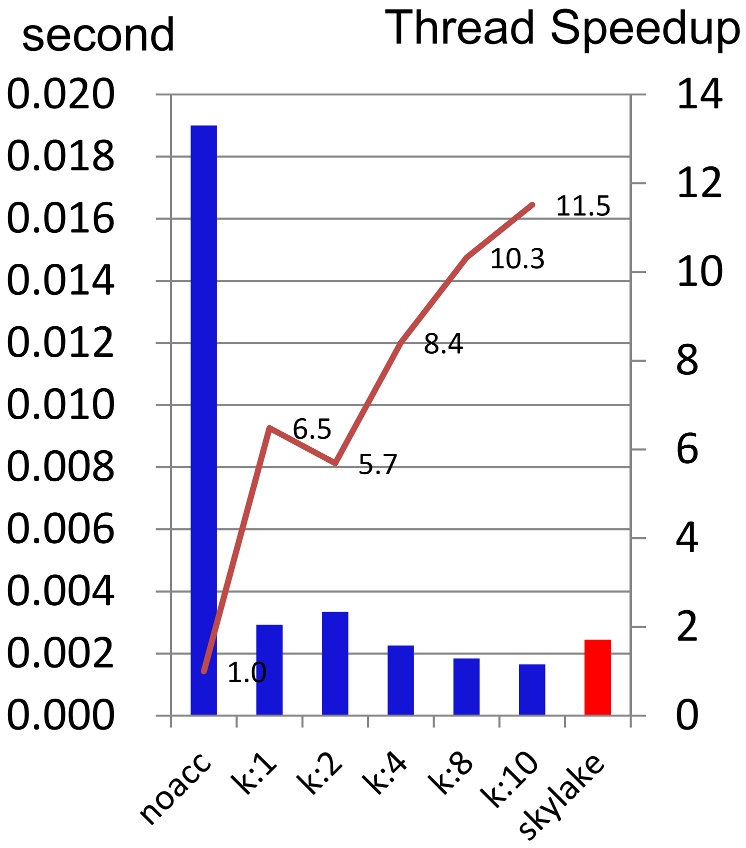
X7

X13



OpenACC* optimization on Taihu Light

```
Start_time = mpi_wtime()
!$acc parallel loop  packin(its,ite,jts,jte,kts,kte,dc025,cp25)&
!$acc copyin(u,v,pi,d2k,th,thref,thv,fv,zsy,rkrf,dy)&
!$acc copyout(vl,vn) local(i,k,j,t,zdpdz,uv) &
!$acc  collapse(2) tile(k:10) annotate(entire(dy,d2k,rkrf,fv))
 DO j=jts,jte
     DO k=kts,kte
       DO i=its,ite
        uv=dc025*(u(i,k,j-1)+u(i-1,k,j-1)+u(i,k,j)+u(i-1,k,j))
        t=(pi(i,k,j)-pi(i,k,j-1))/dy(j)
        zdpdz=t+zsy(i,k,j)*((((pi(i,k+1,j)-pi(i,k,j))/d2k(k)+(pi(i,k,j)-pi(i,k-1,j))/d2k(k-1))+   &
                  ((pi(i,k+1,j-1)-pi(i,k,j-1))/d2k(k)+(pi(i,k,j-1)-pi(i,k-1,j-1))/d2k(k-1)))*dc025)
         vl(i,k,j)=-fv(j)*uv-cp25*(thref(i,k,j-1)+thref(i,k+1,j-1)+thref(i,k,j)+thref(i,k+1,j))*zdpdz
         vn(i,k,j)=-(th(i,k,j-1)+th(i,k+1,j-1)+th(i,k,j) +th(i,k+1,j))*cp25*zdpdz     &
                  -cp25*(thv(i,k,j-1)+thv(i,k+1,j-1)+thv(i,k,j)+thv(i,k+1,j))*zdpdz-krf(k)*v(i,k,j)*3.
        ENDDO
     ENDDO
 ENDDO
!$acc end parallel loop
end_time = mpi_wtime()
if ( myprcid == 0 ) write(*,*) 'vl,vn use ', end_time - start_time
```



OpenACC* optimization on TaihuLight

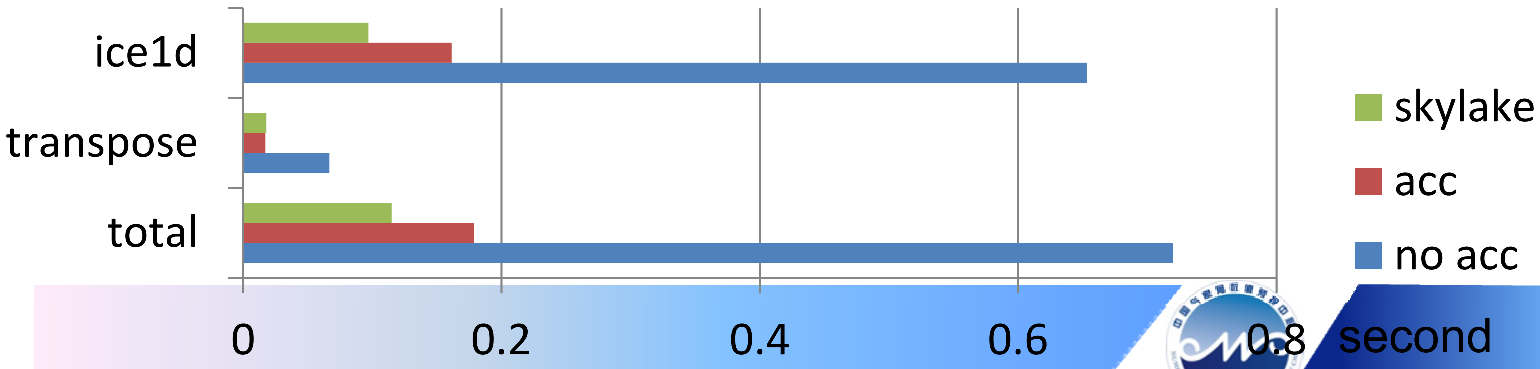
A(i,k,j) → Transpose → A_c(k,i,j)

```
!$acc parallel loop &
!$acc copy(A_c,...14 3D arrays) &
!$acc copyin(20 3D arrays) &
!$acc collapse(2) tile(i:1)
    DO j=1, 45
        DO i=1, 90
            CALL ice1D(A_c(:,i,j)... 34 1D arrays and some others)
        ENDDO
    ENDDO
!$acc end parallel loop
```

A_c(k,i,j) → Transpose → A(i,k,j)

```
subroutine ice1d(A…)
real :: A(1:60),…
…
do k=kts, kte
…
end do
do k=kts,kte
…
…
end do
…
end
```

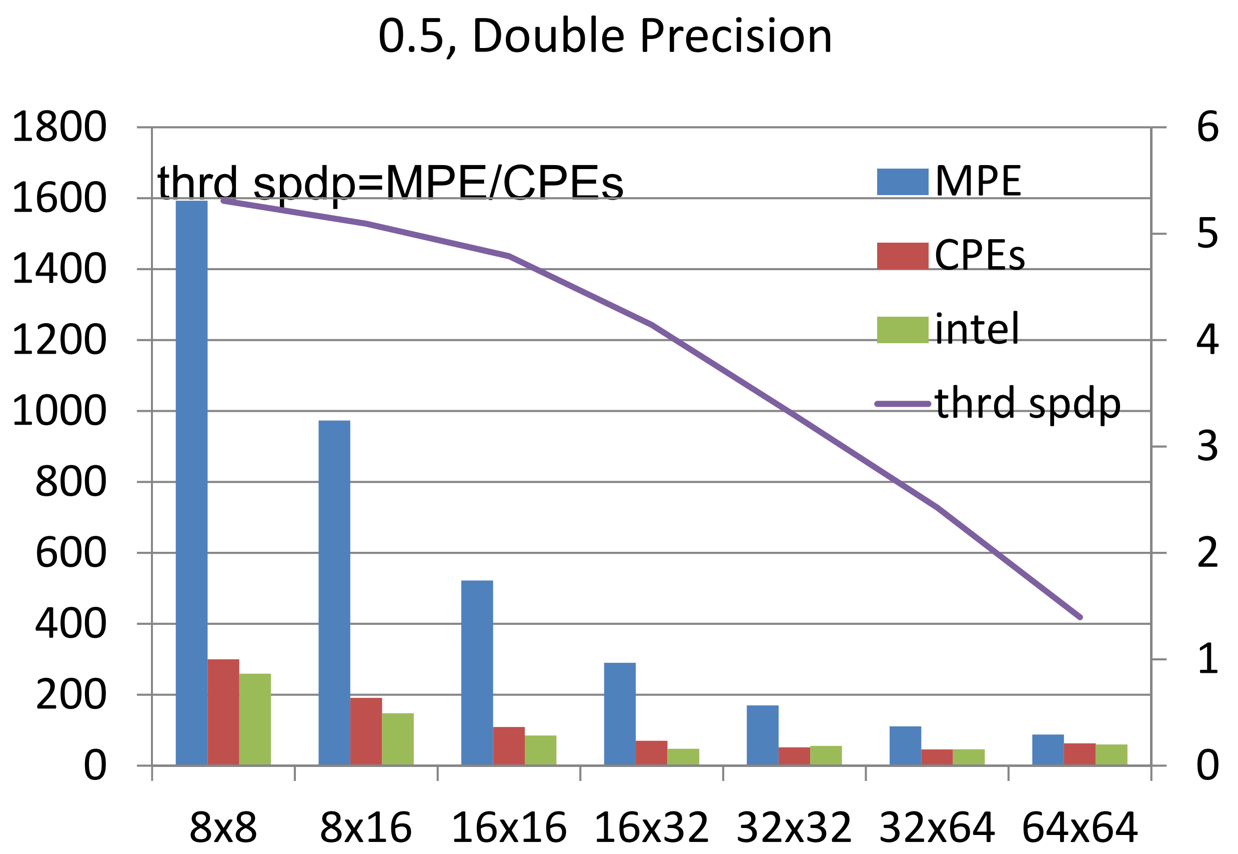
1500 lines



Test Results

- Dynamic core of GRAPES global model in Double precision
  - O.5° resolusion
  - 0.25° resolusion
- Physics of of GRAPES in single precision
  - 0.5° resolusion

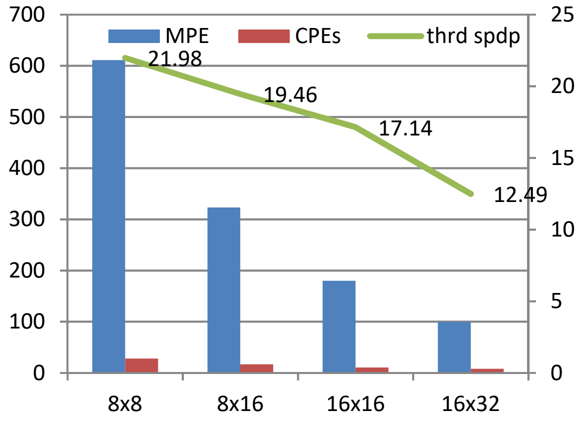
GRAPES-GLB Dynamic core



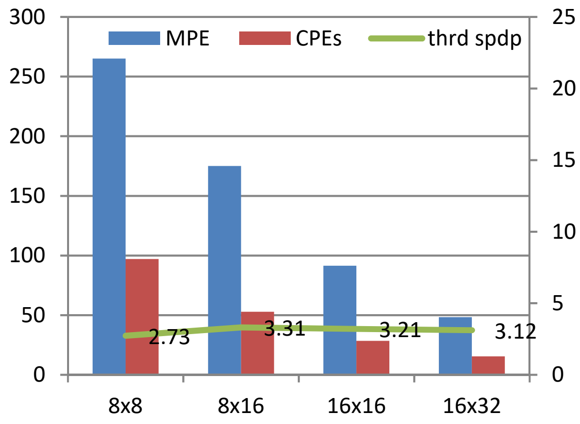
GRAPES global model，0.5°，no physics，72 steps
MPE: “Taihu Light”,no thread parallel processing
CPEs: “Taihu Light” with CPEs parallel processing
Intel: Node/CPU: Intel Xeon Gold 6124 (2.66GHz 16 cores) processor, 2 CPUs/node (16 cores/CPU) 100Gb/s EDR InfiniBand inter connection

Trace substance

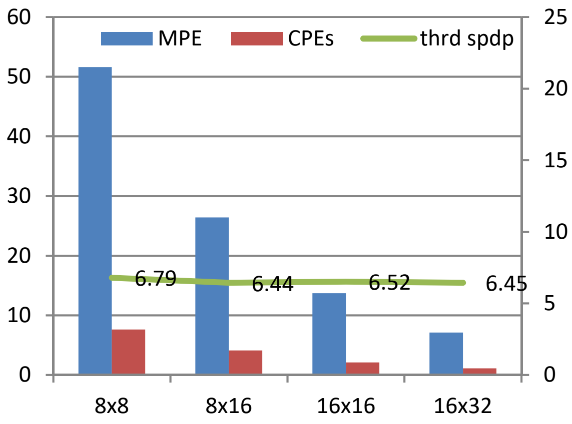
6 routines of dynamic core



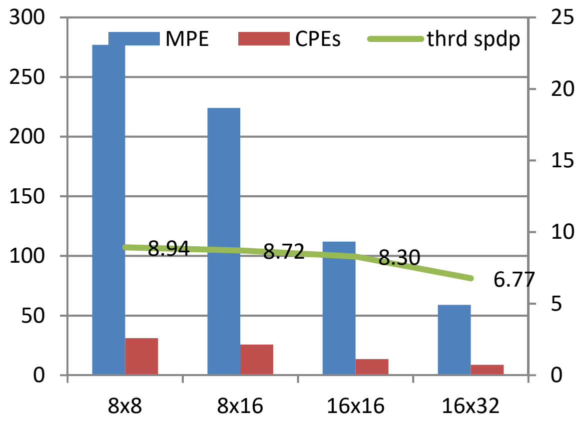
Helmholtz solver



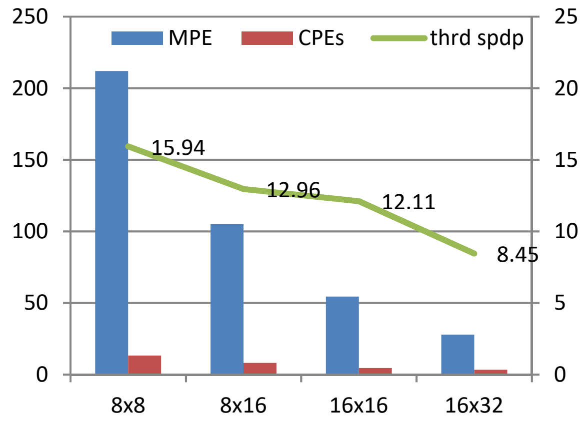
Departure points



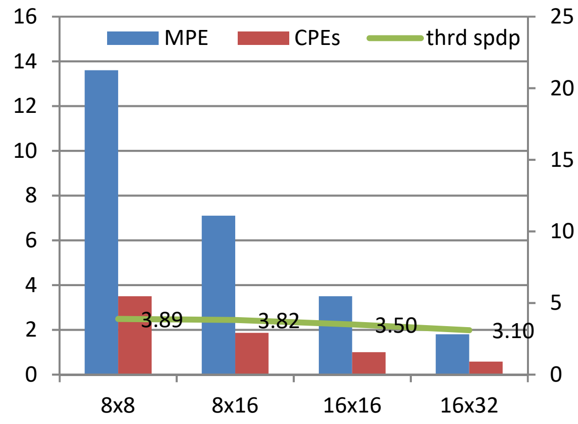
SL interp uvwthpi



SL interp trace substance



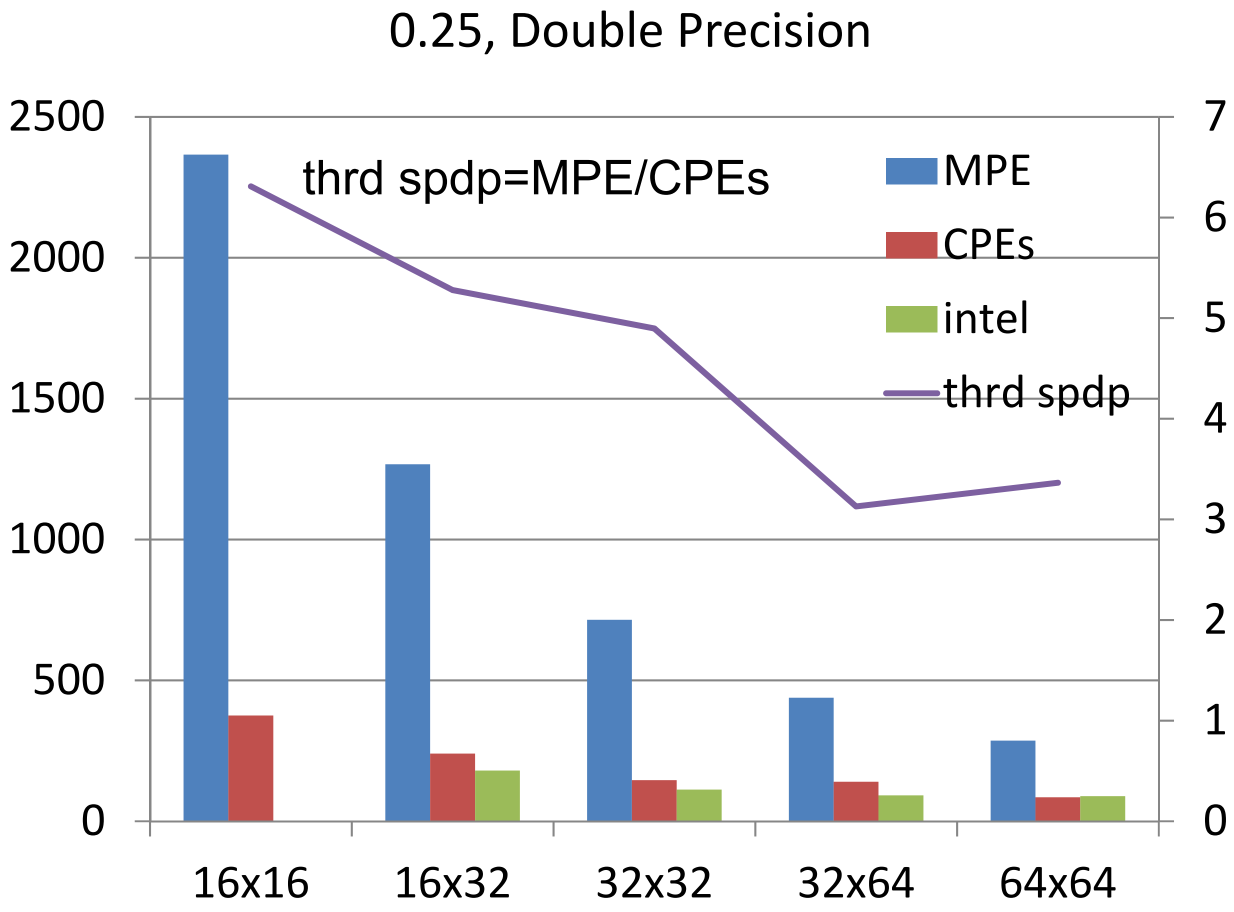
PRM



Linear & none linear

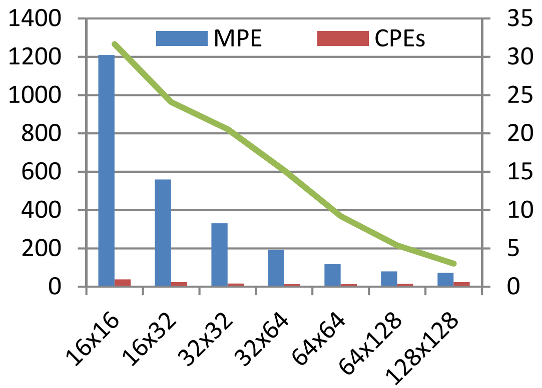
GRAPES global model，0.5°，no physics，72 steps
MPE: TaihuLight, only on MPE
CPEs: TaihuLight, on MPE& CPEs

GRAPES-GLB Dynamic core

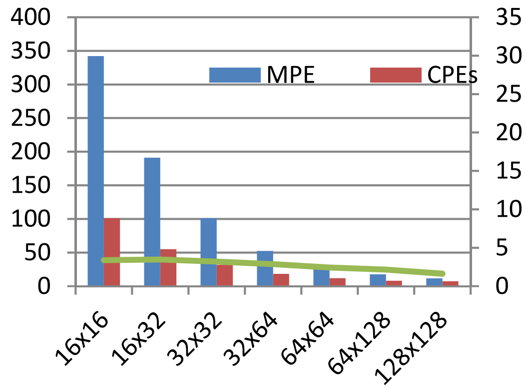


GRAPES global model，0.25°，no physics，72 steps
MPE: “Taihu Light”,no thread parallel processing
CPEs: “Taihu Light” with CPEs parallel processing
Intel: Node/CPU: Intel Xeon Gold 6124 (2.66GHz 16 cores) processor, 2 CPUs/node (16 cores/CPU) 100Gb/s EDR InfiniBand inter connection

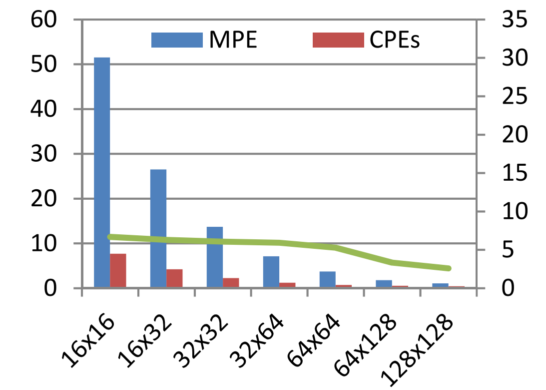
6 routines of dynamic core



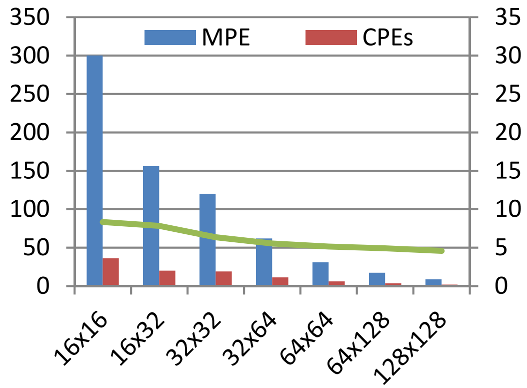
Helmholtz solver



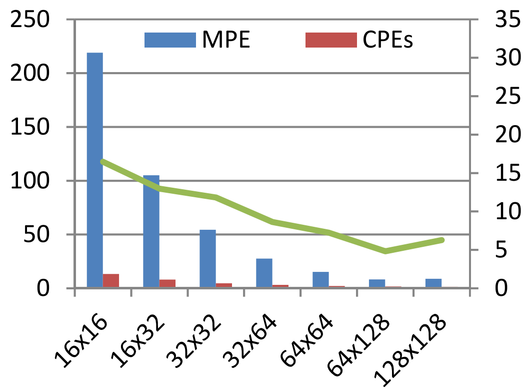
Departure points



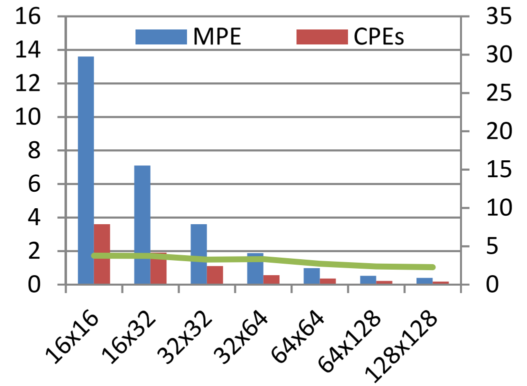
SL interp uvwthpi



SL interp trace substances



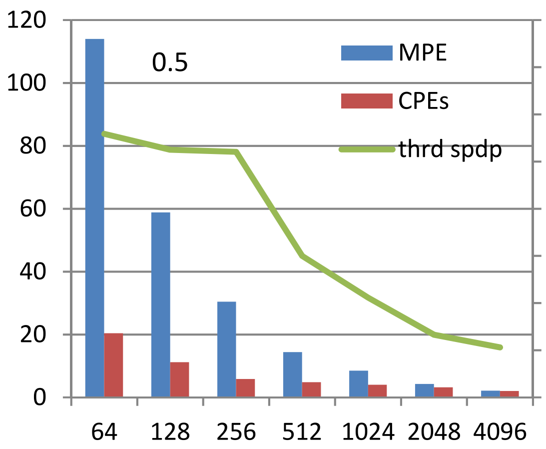
PRM

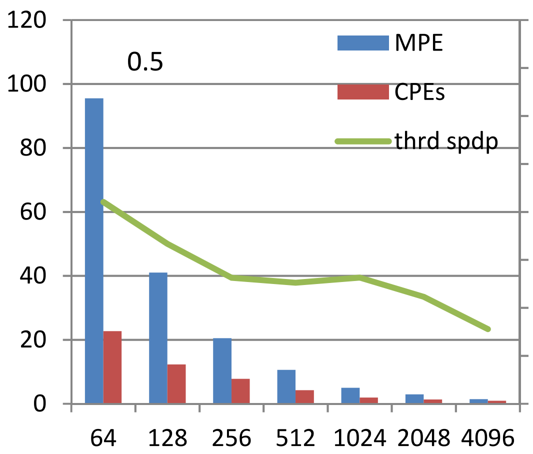


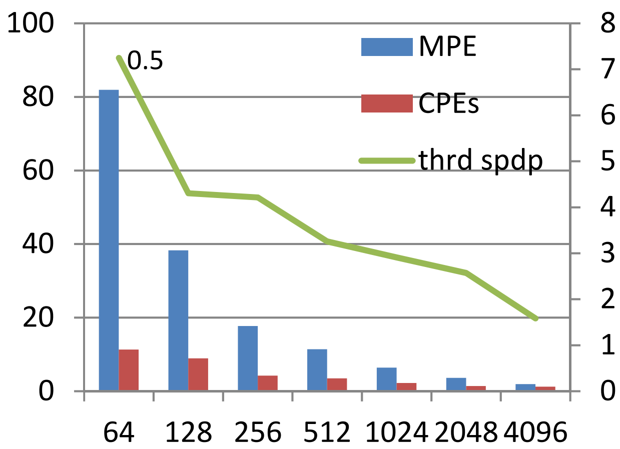
Linear & none linear

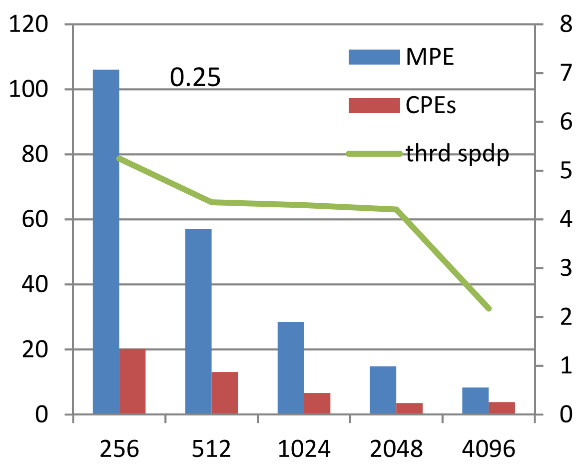
GRAPES global model，0.25°，no physics，72 steps
MPE: "Taihu Light",no thread parallel processing
CPEs: "Taihu Light" with CPEs parallel processing

3 routines in physics package

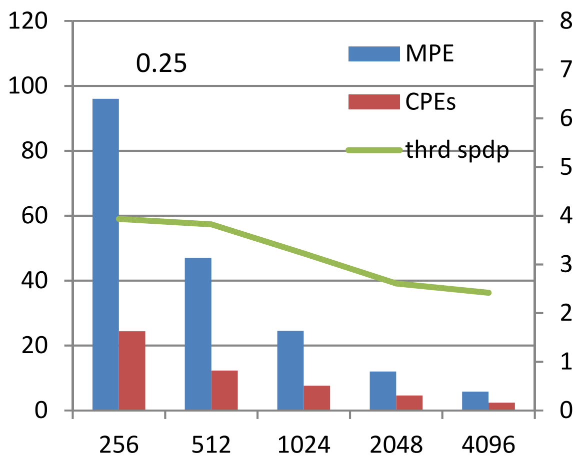




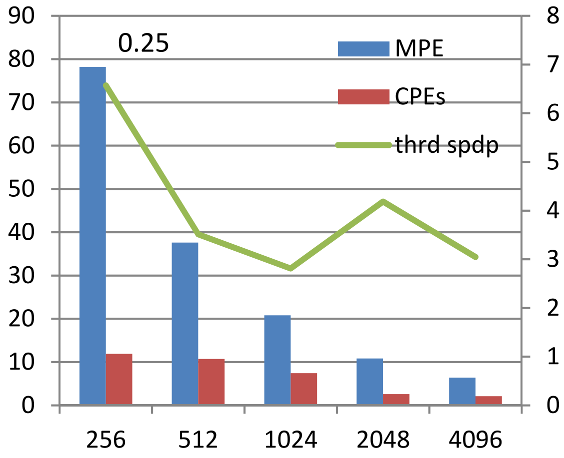




Phy_prep



Microphysics



Phy_post_back

GRAPES global model，no physics，72 steps
MPE: “Taihu Light”, only on MPE
CPEs: “Taihu Light” on MPE & CPEs

Summary

- Current swGRAPES dynamic core has comparable performance with Pi system (intel skylake processors system), suggesting that the refactorization of GRAPES having potential to use heterogeneous many-core platforms
- Fine-grained parallel algorithm for SpMV and ILU can better utilization of SW26010 processor, encouraging future many-core oriented algorithm design.
- Computation and communication overlapping can well fit to the heterogeneous supercomputing system
- OpenACC* has moderate performance improvement and better portability for regular stencil-like computation loops, The key point of the performance is the best utilization of memory bandwidth
- Programming with Athread can get better performance due to full control of cache, accessing data, communication while it is lack of portability
- The work is far from end and further improvements is required

Future Work

- GCR algorithm
  - Geometric Multigrid Preconditioning
  - Two communication optimization algorithms: Pipelined GCR and Chebyshev Methods

- Further Optimization on Sunway TaihuLight

- Improving portability by using OpenACC* enhancement, high performance libraries, and code generation frameworks

THANK YOU
and questions?