The 24th International Conference on Parallel and Distributed Systems
IEEE ICPADS 2018, December 11 - 13, Sentosa, Singapore

# AGCM3D: A Highly Scalable Finite-Difference Dynamical Core of Atmospheric General Circulation Model based on 3D Decomposition

**Baodong Wu , Shigang Li, Hang Cao, Yunquan Zhang, Junmin Xiao**
SKL Computer Architectures
Institute of Computing Technology, Chinese Academy of Sciences

**He Zhang , and Minghua Zhang**
Institute of Atmospheric Physics, Chinese Academy of Sciences

中国科学院计算技术研究所
Institute of Computing Technology, Chinese Academy of Sciences

中国科学院大学
University of Chinese Academy of Sciences

中国科学院大气物理研究所
INSTITUTE OF ATMOSPHERIC PHYSICS, CHINESE ACADEMY OF SCIENCES

# CONTENTS

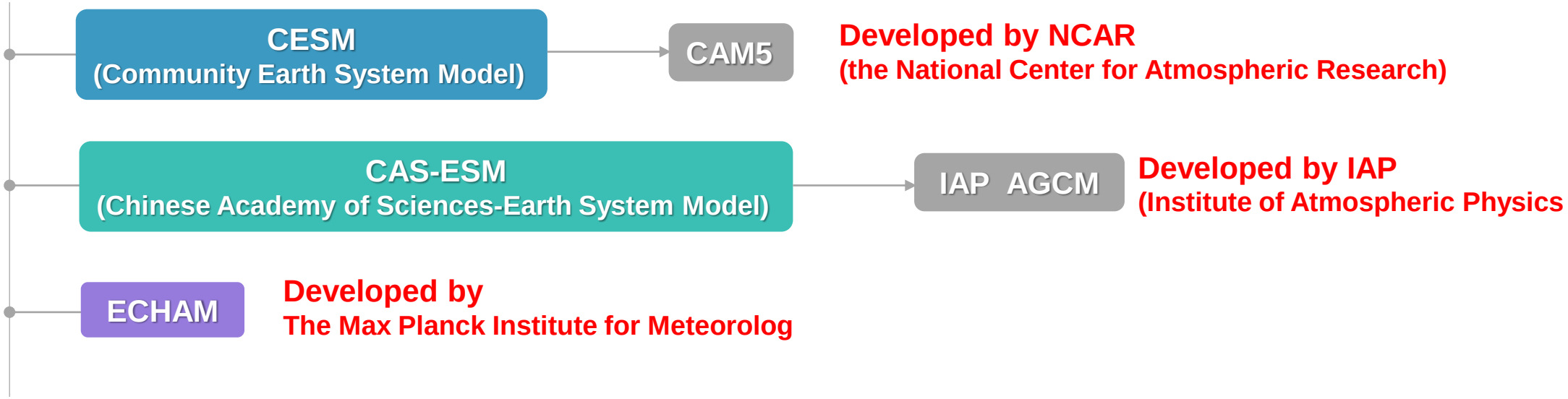
# CONTENTS

# Introduction

## Atmospheric General Circulation Models(AGCM)

1. **Numerical simulation of the global atmospheric circulation is important in climate modeling, and is also a great challenge in scientific computing.**

**Some recently developed atmospheric models:**




**In order to enable high-fidelity simulation of realistic problems,**
**the study of high-performance atmospheric solvers is becoming an urgent demand.**

# Introduction

## Dynamical Core

**2. The dynamical core is one of the most time-consuming modules of Atmospheric General Circulation Models(AGCM).**

**Typically, the dynamical core can be numerically solved two types of mesh:**

### Quasi-uniform polygonal mesh

- ✓ CAM-SE
- ✓ Good parallel scalability
- ✓ Not require the costly polar filtering
- ✓ difficult to preserve the energy conservation
- ✓ difficult to deal with the discontinuous variables

### equal-interval latitude-longitude mesh

- ✓ CAM-FV IAP AGCM
- ✓ Easy to preserve the energy conservation
- ✓ Easy to deal with the discontinuous variables
- ✓ Easy to couple with other component
- ✓ Poor parallel scalability
- ✓ Perform the costly polar or high-latitude filtering

**Our work focuses on improving the parallel scalability for the dynamical cores based on the latitude-longitude mesh, and scales the performance to tens of thousands of CPU cores.**

# Introduction

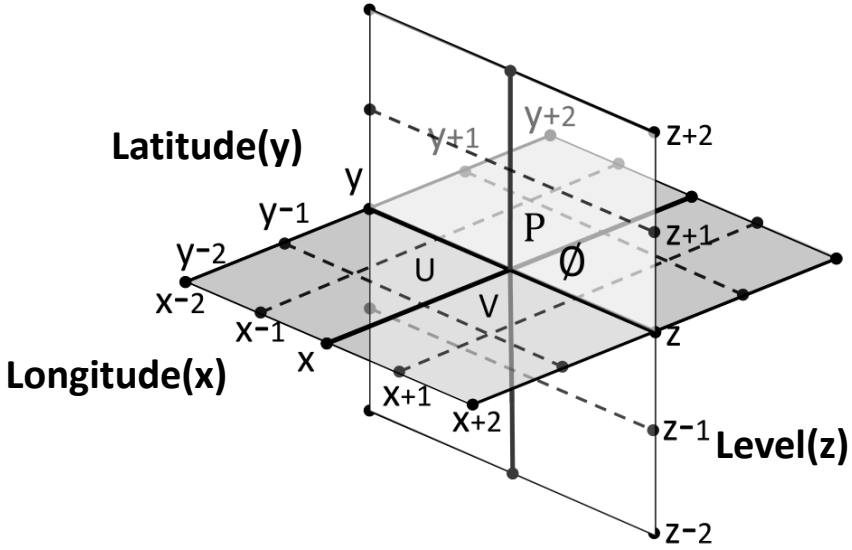
## Dynamical Core

**3. The baseline is the dynamical core of the fourth-generation IAP AGCM. IAP AGCM-4 uses the finite-difference method based on the latitude-longitude mesh to solve the dynamical core.**

**In IAP AGCM-4, the dynamic core revolves around the solutions of the baroclinic primitive equations.**

$$
\begin{cases}
\left[\frac{\partial U}{\partial t}\right]_{x',y,z} = \left[-\alpha^*\widetilde{L}(U) - \beta^*\widetilde{P}(\lambda) + \gamma^* f^* V\right]_{x',y,z} \\
\left[\frac{\partial V}{\partial t}\right]_{x,y',z} = \left[-\alpha^*\widetilde{L}(V) - \beta^*\widetilde{P}(\Theta) + \gamma^* f^* U\right]_{x,y',z} \\
\left[\frac{\partial \Phi}{\partial t}\right]_{x,y,z} = \left[-\alpha^*\widetilde{L}(\Phi) + \widetilde{\delta}\cdot\beta^*\widetilde{\Omega}\right]_{x,y,z} \\
\left[\frac{\partial}{\partial t}\left(\frac{p'_{sa}}{p_0}\right)\right]_{x,y} = \left[\beta^*\widetilde{P}(W) - \kappa^*\frac{D_{sa}}{P_0}\right]_{x,y}
\end{cases}
$$

The basic prognostic variables:
the zonal wind(U), meridional wind(V),
the pressure(P), and the temperature(T)
gnostic variables:




Stencil computation for the prognostic variables.

**This is a typical 3D stencil computation model**

# Introduction

## Contribution

### Traditional AGCM2D:

- Two dimensions(latitude and level) is used to parallelize the dynamical core of IAP AGCM-4.
- The dynamical core can only scale up to 1024 MPI processes at the resolution of 0.5° × 0.5°
- The one-dimensional FFT filtering along the longitude (X) dimension in the high-latitude region.
- FFT parallelization leads to expensive all-to-all collective communication

### New AGCM3D:

- **3D decomposition method** releases the parallelism in all three dimensions (latitude, longitude, and level).
- **A novel adaptive Gaussian filtering scheme** replaces the costly parallel FFT filtering.
- **communication avoiding and message aggregation** reduce the communication overhead.

# CONTENTS

# 3D decomposition method(AGCM3D)

## 3D decomposition method

**The 3D decomposition method is implemented by partitioning all the three dimensions of the mesh and the corresponding variable arrays. The mesh points and the variable arrays are then mapped to a three-dimensional process topology.**

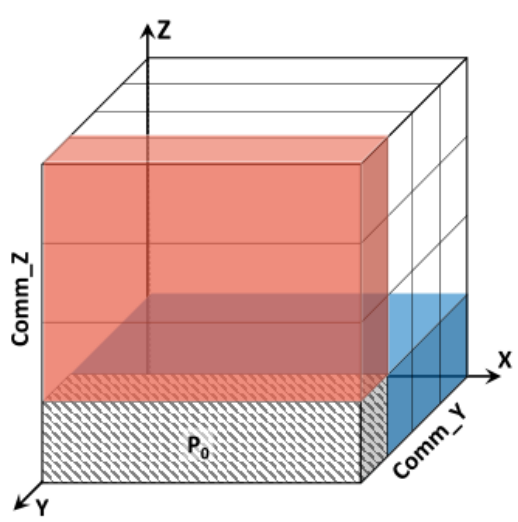
Suppose there are ***M, N, H*** mesh points and $P_x$, $P_y$, $P_z$ processes for X, Y and Z dimensions.

For 2D decomposition, The total number of mesh points in each process has:

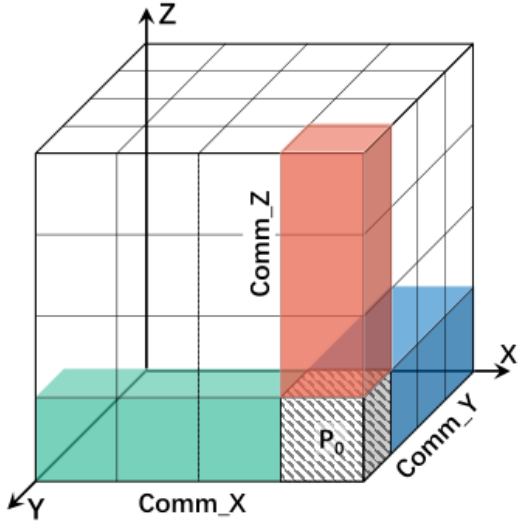
$$\frac{M*N*H}{P_y*p_z}$$

For 3D decomposition, The total number of mesh points in each process has:

$$\frac{M*N*H}{P_x*P_y*p_z}$$




Communication pattern of the 2D decomposition.




Communication pattern of the 3D decomposition.

# 3D decomposition method(AGCM3D)

## 3D decomposition method

**The 3D decomposition not only increases the parallelism, but also decreases the communication overhead.**

| Comparison items | *2D* | *3D* |
|---|---|---|
| Horizontal Resolution | $0.5° \times 0.5°$ | $0.5° \times 0.5°$ |
| Number of mesh points: $M \times N \times H$ | $720 \times 361 \times 30$ | $720 \times 361 \times 30$ |
| Processes number of X dimension | 1 | $P_x$ |
| Processes number of Y dimension | $P_y$ | $P_y$ |
| Processes number of Z dimension | $P_z$ | $P_z$ |
| The theoretical parallelism | $361 \times 30$ | $720 \times 361 \times 30$ |

| Comparison items | *2D* | *3D* |
|---|---|---|
| Per core P2P communication volume along X | 0 | $(23+36\times\frac{30}{P_z})\times\frac{361}{P_y}$ |
| Per core P2P communication volume along Y | $(15+18\times\frac{30}{P_z})\times 720$ | $(15+18\times\frac{30}{P_z})\times\frac{720}{P_x}$ |
| Per core P2P communication volume along Z | $6\times\frac{361}{P_y}\times 720$ | $6\times\frac{361}{P_y}\times\frac{720}{P_x}$ |
| Per core collective communication volume along Z | $\frac{361}{P_y}\times\frac{30}{P_z}\times 720$ | $0\ (if\ P_z=1)$; $\frac{361}{P_y}\times\frac{30}{P_z}\times\frac{720}{P_x}$ $(if\ P_z>1)$ |

**The volume of point-to-point communications along Y and Z dimensions are reduced by $P_x$ times.**

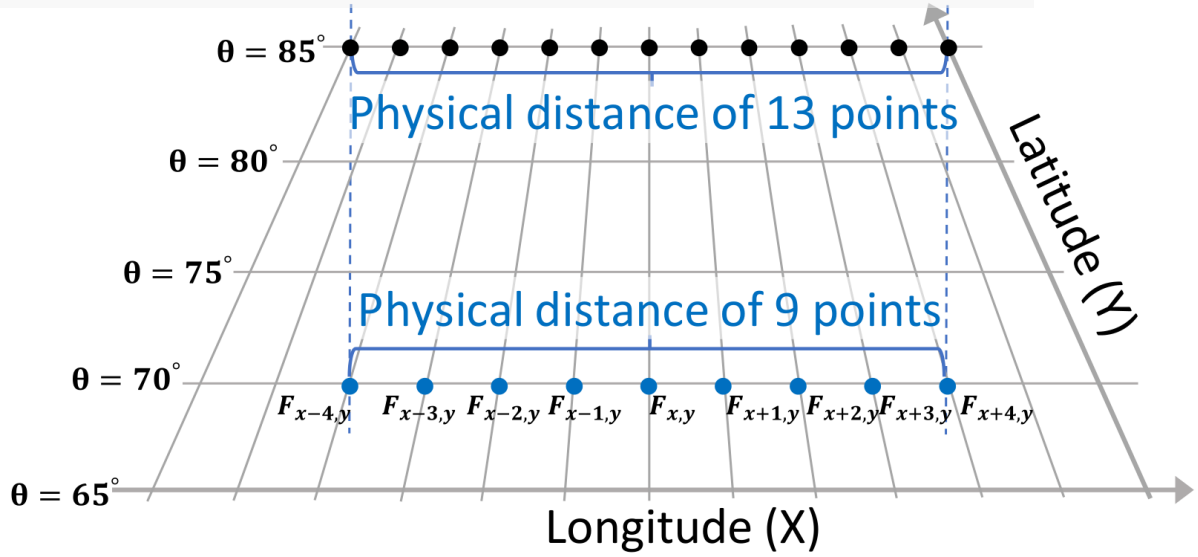
# 3D decomposition method(AGCM3D)

## Adaptive Gaussian filtering scheme

**The physical distance of 9 mesh points at 70° is equal to the physical distance of 13 mesh points at 85°. The time step of dynamical core must be small enough to meet the stability requirements of the governing equations, which result in high computational cost.**

To alleviate the problem caused by the mesh lines clustering along the X dimension, the filtering module is applied in the finite-difference dynamical core.

**Poleward of ±70°, FFT filtering along longitude (X) dimension is used on the tendencies of U,V,P,T to dump out the short-wave modes.**




**The latitude mesh lines cluster at the high-latitude region**

**For AGCM3D, The all-to-all communication of parallel FFT incurs at least $\log_2 P_x$ number of communications and total M communication size for each process , which is too high to be amortized by the benefit of the 3D decomposition**

# 3D decomposition method(AGCM3D)

## Adaptive Gaussian filtering scheme

**If the latitude θ =±70° , the filtering width B $_\theta$=4K $_\theta$+1, K $_\theta$=2, the Gaussian filtering is:**

$$\sum_{n=-2K_\theta}^{2K_\theta} F_{(x+n),y} * W_{x,y=\pm 70°;x+n} \qquad \text{Where } W: W_{x,y=\pm 70°;x+n} = \frac{e^{-\frac{n^2}{K_\theta{}^2}}}{\sum_{k=-2K_\theta}^{2K_\theta}(e^{-\frac{K^2}{K_\theta{}^2}})} \tag{1}$$

**If ±70° < θ<±87° , the filtering width B $_\theta$=4K $_\theta$+1, K $_\theta$=2, the Gaussian filtering is:**

$$\sum_{n=-2K_\theta}^{2K_\theta} F_{(x+n),y} * W_{x,y;x+n} \qquad \text{Where } W: W_{x,y;x+n} = W_{x,y=\pm 70°;x+n} L_\theta + \frac{1}{1+2K_{70°}}(1-L_\theta) \ , L_\theta = \frac{\sin(90°-70°)}{\sin(90°-|\theta|°)} \tag{2}$$

**If ±87° ≤ θ ≤ ±90° , the filtering width B $_\theta$=4K $_\theta$+1, K $_\theta$=3, the Gaussian filtering is the same as above formula, the number of filtering calls is N $_\theta$ .**

$$N_\theta = \left\lfloor \frac{\sin(90°-87°)}{\sin(90°-|\theta|)} \right\rfloor, \pm 87° \le \theta \le \pm 90° \tag{3}$$

| Filtering scheme | Iteration times | Latitude |
|---|---|---|
| $\sum_{n=-4}^{4} F_{(x+n),y} * W_{x,y=\pm 70°;x+n}$ | 1 | θ = ±70° |
| $\sum_{n=-4}^{4} F_{(x+n),y} * W_{x,y;x+n}$ | 1 | ±70° < θ < ±87° |
| $\sum_{n=-6}^{6} F_{(x+n),y} * W_{x,y;x+n}$ | $N_\theta$ | ±87° ≤ θ ≤ ±90° |

# 3D decomposition method(AGCM3D)

Communication optimizations

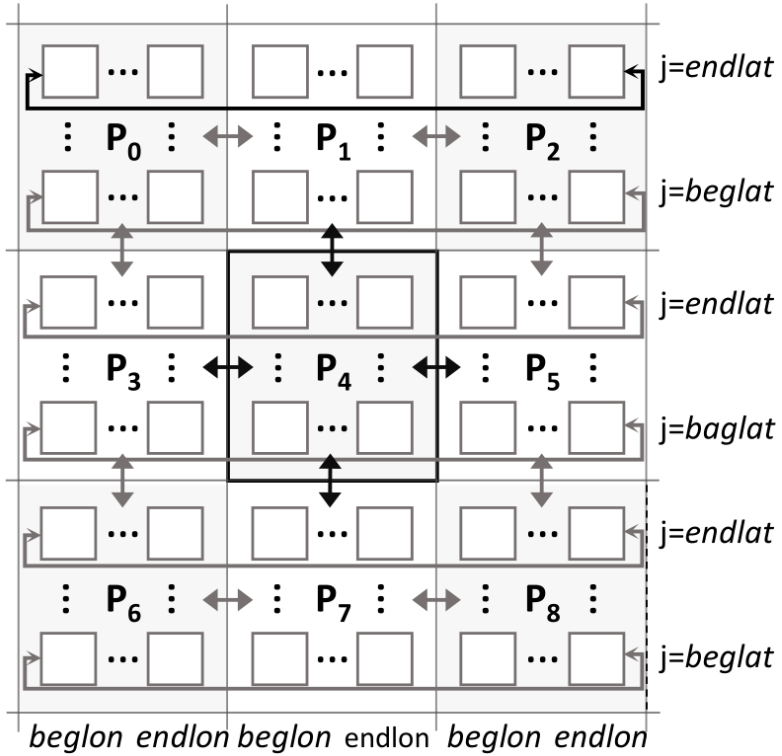
**We use the techniques of message aggregation and communication avoiding used to reduce the communication overhead of the 3D decomposition method.**

The 3D decomposition adds point-to-point communication between the direct neighbor processes along the X dimension, and periodic border communication between the first process and the last process along the X dimension.

The same communication pattern is used by calculations of multiple variables, and the messages are very short.

For 4096 processes, the size of each message is 500 bytes. However, messages more than 32 KB can achieve good bandwidth utilization for MPI over InfiniBand network.

Therefore, we package all the short messages with the same destination as a long message, and send it by one communication to improve bandwidth utilization.



# CONTENTS

# Experiment results

## Experimental environment

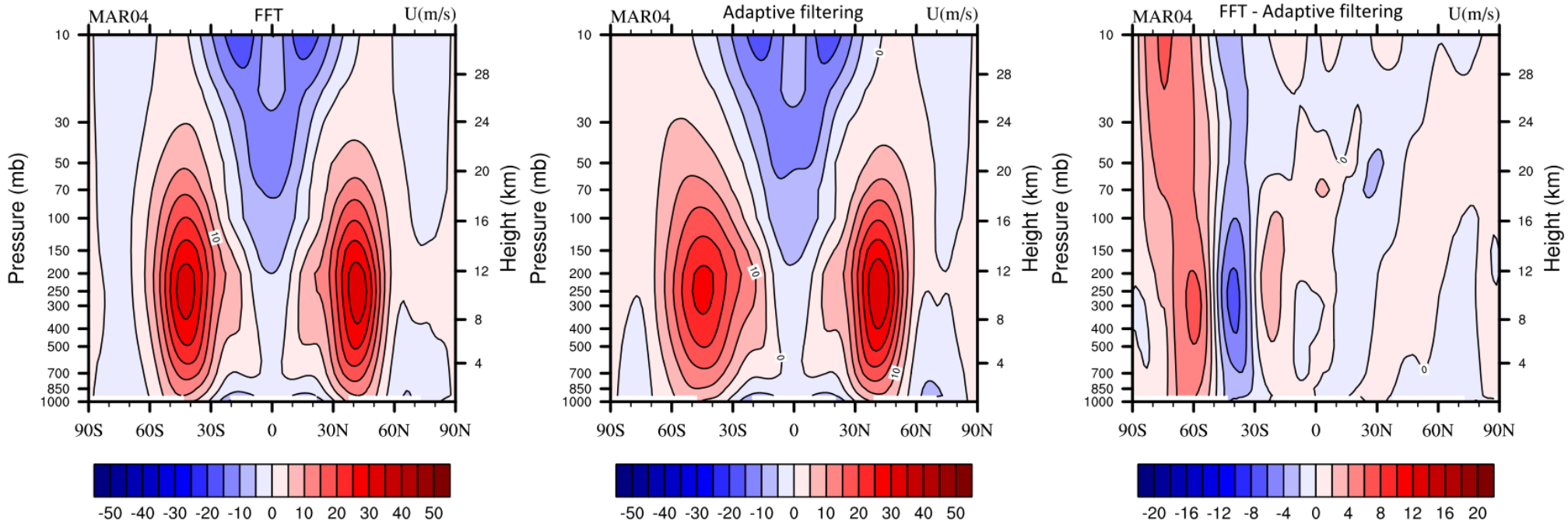
| | |
|---|---|
| **Machine name** | Tianhe-2 supercomputer |
| **Processers** | Intel Xeon E5-2692 processor |
| **CPU cores** | 24 cores in each node |
| **Network** | TH Express-2 interconnected network |
| **MPI version** | mpi3-dynamic (MPI 3.0 standard) |
| **Case model** | The idealized dry-model experiments |
| **horizontal resolution** | 0.5° × 0.5° |

| **Number of processes** | ***2D*** $(P_y \times P_z)$ | ***3D*** $(P_x \times P_y \times P_z)$ |
|---|---|---|
| 128 | $32 \times 4$ | $32 \times 4 \times 1$ |
| 256 | $32 \times 8$ | $32 \times 8 \times 1$ |
| 512 | $32 \times 16$ | $32 \times 16 \times 1$ |
| 1024 | $64 \times 16$ | $32 \times 32 \times 1$ |
| 2048 | – | $32 \times 64 \times 1$ |
| 4096 | – | $32 \times 64 \times 2$ |
| 8192 | – | $32 \times 64 \times 4$ |
| 16384 | – | $32 \times 64 \times 8$ |
| 32768 | – | $32 \times 64 \times 16$ |
| 65536 | – | $64 \times 64 \times 16$ |

# Experiment results

## The Correctness of the Adaptive Gaussian Filtering

- ✓ Through the Held-Suarez test of FFT and adaptive filtering, the results show that both the FFT filtering and our adaptive Gaussian filtering can produce a reasonably realistic zonal mean circulation with westerly jet cores located near 250 hPa over the middle-latitudes of both hemispheres.

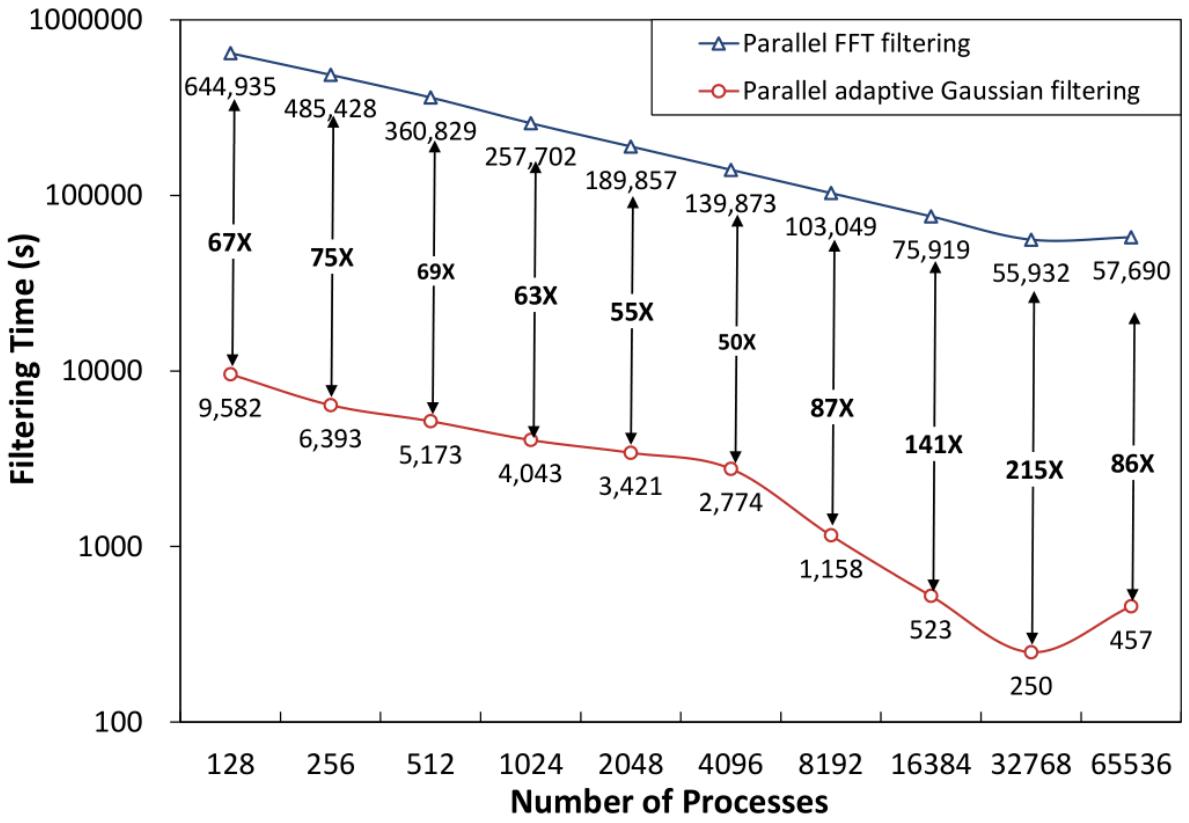



Distribution of zonal wind from the Held-Suarez tests

# Experiment results

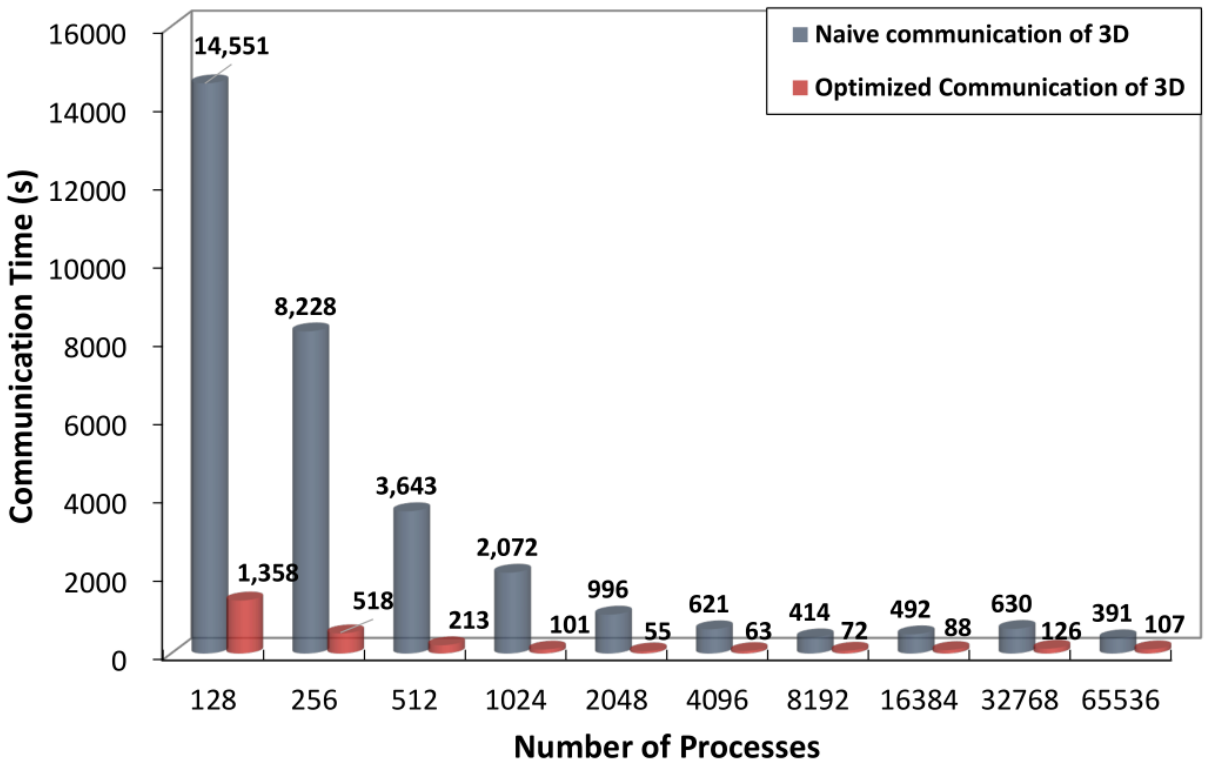
## The Performance of the Adaptive Gaussian Filtering

- ✓ We compare the performance of the parallel FFT filtering and the parallel adaptive Gaussian filtering used in the 3D decomposition.
- ✓ Compared with the parallel FFT filtering, our parallel adaptive Gaussian filtering improves the performance by **an average of 90x**



# Experiment results

## Communication Optimizations

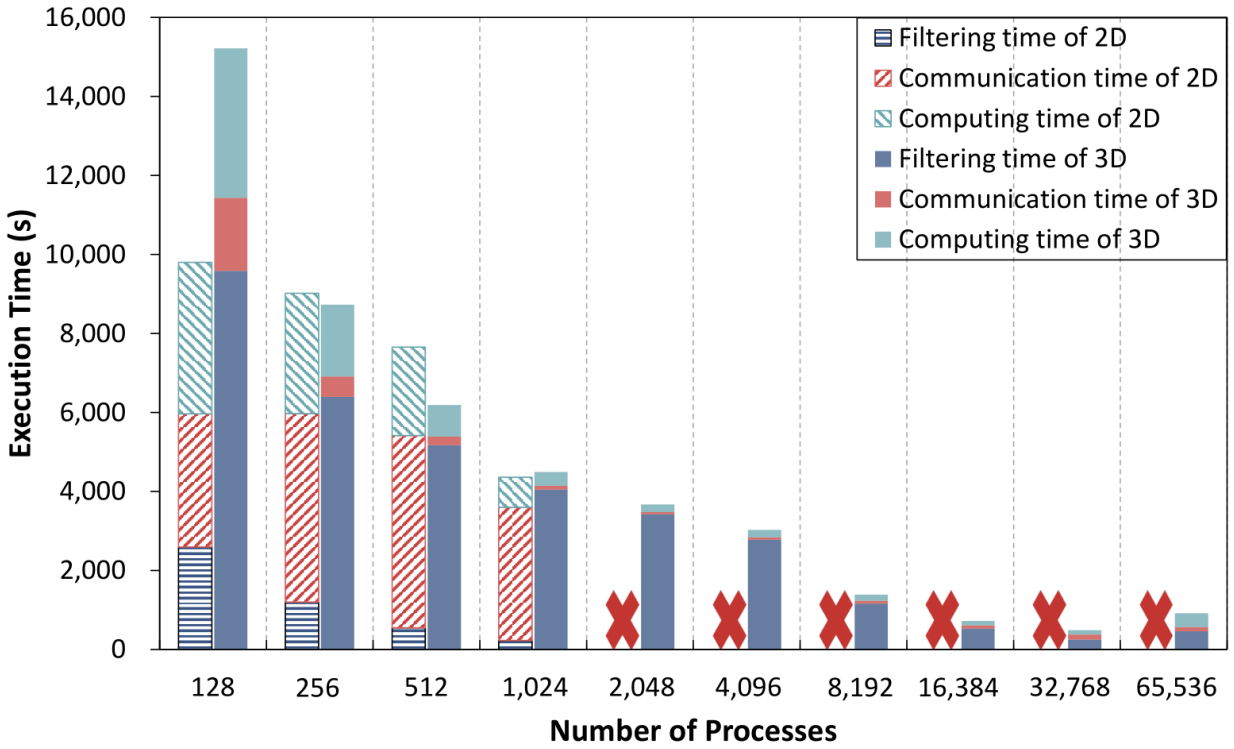



- ✓ We compare the performance of the naive communication and the optimized communication by message aggregation of the 3D decomposition.
- ✓ The optimized communication improves the performance by **10x on average.**
- ✓ The minimum communication overhead is **55s at 2048 cores** for the optimized communication.
- ✓ The decomposition along the Z dimension is added for more than 2048 cores, which leads to extra point-to-point communication and collective communication along the Z dimension.

# Experiment results

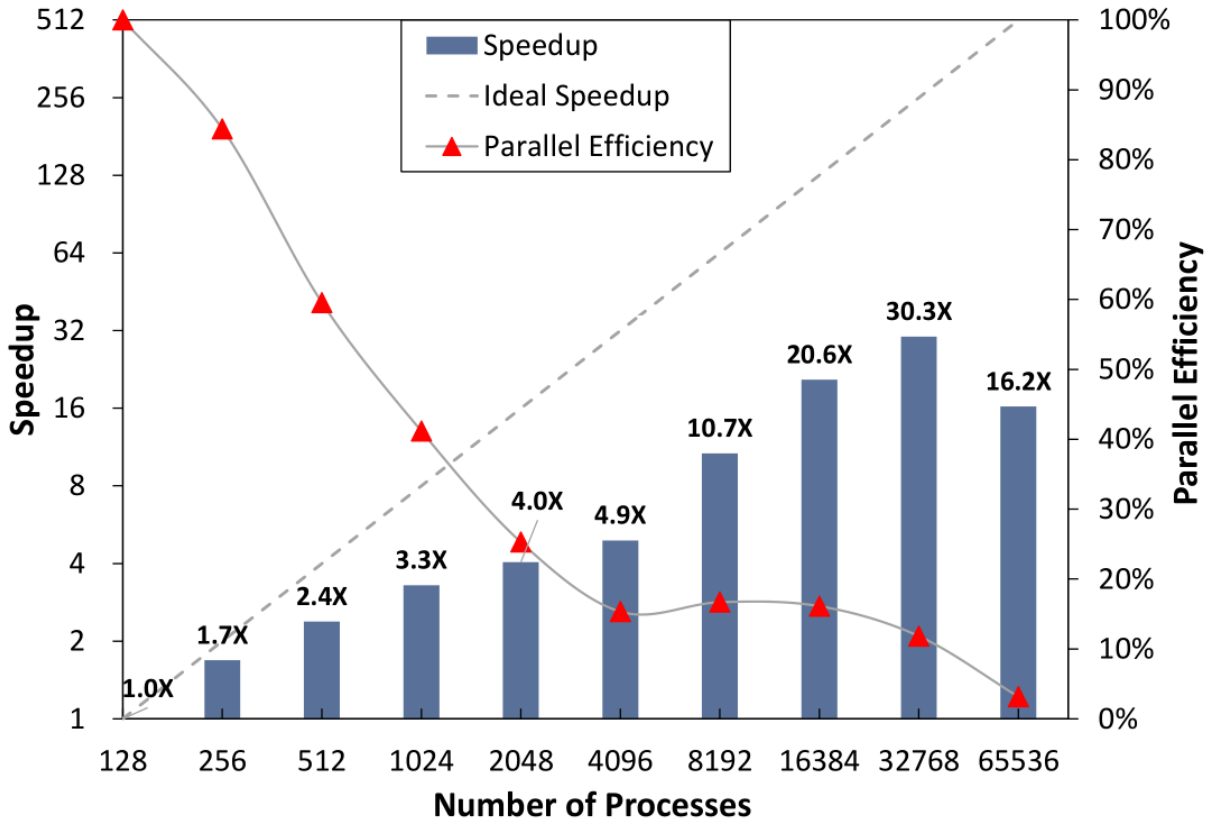
## Scalability and Overall Performance Test

- ✓ In the strong scaling tests, the number of processes is increased from 128 to 65,536.
- ✓ The dynamical core using 2D decomposition only **scales up to 1024 processes.**
- ✓ The 3D decomposition method can scale the performance **up to 32,768 processes.**
- ✓ The communication time for the 3D decomposition is reduced by more than **50% on average** over the process number from 128 to 1024.



# Experiment results

## Scalability and Overall Performance Test




- ✓ Speedup and parallel efficiency of the 3D decomposition method.
- ✓ The 3D decomposition method scales from 128 processes to **32,768 processes**, and achieves **30.3x speedup and 13% parallel efficiency.**

# CONTENTS

# Conclusion and Future work

## 1 Conclusion

- ✓ AGCM3D increases the parallelism of dynamical core significantly by adding decomposition on the longitude dimension.
- ✓ High-latitude FFT filtering is replaced by the new adaptive Gaussian filtering, which has the same filtering effect as FFT.
- ✓ Using message aggregation and communication avoiding, the overhead of communication is significantly reduced.

## 2 Future work

- ✓ We foresee that our method will achieve even better scalability for the higher-resolution simulation.
- ✓ We will couple AGCM3D with the physical process, and utilize many-core architectures to further speedup the simulation.

# Conclusion and Future work

Recent optimization progress in dynamical core

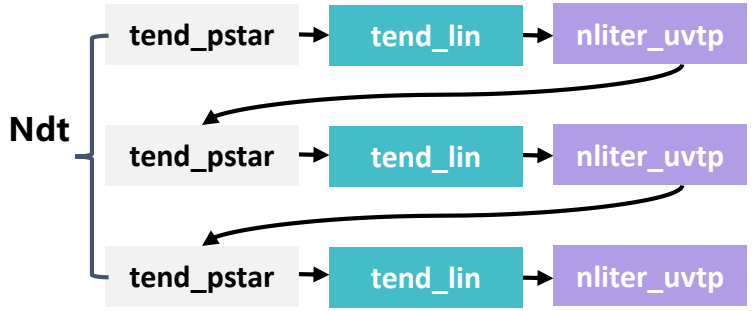
## New time integration scheme

In the dynamical core, we know the main overheads are concentrated in the tendencies of the adaptation (*tend_lin* function) and advection computation (*tend_adv* function). Normally, The *tend_lin* function and *tend_adv* function are called 3*Ndt (Ndt=2 or 3) times and 3 times respectively.

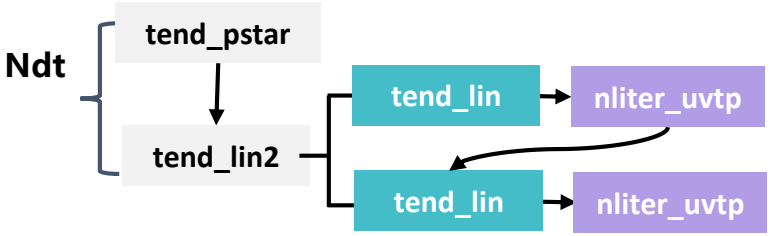
We have improved **the time integration scheme**. By updating the calculation methods of *tend_lin* and *tend_adv* functions, we can call fewer *times tend_lin* and *tend_adv* functions. On average, the call times of *tend_lin* and *tend_adv* can be reduced by **1/3**.

| Function | Call times in normal version | Call times in optimized version |
|---|---|---|
| DYFRAM | 2833 | 2833 |
| tend_lin | 84990 | 56660 |
| nliter_uvtp | 84990 | 56660 |
| tend_adv | 42495 | 28330 |
| nliter_uvt | 42495 | 28330 |

**Call Tend_lin in normal version**




**Call Tend_lin in development version**



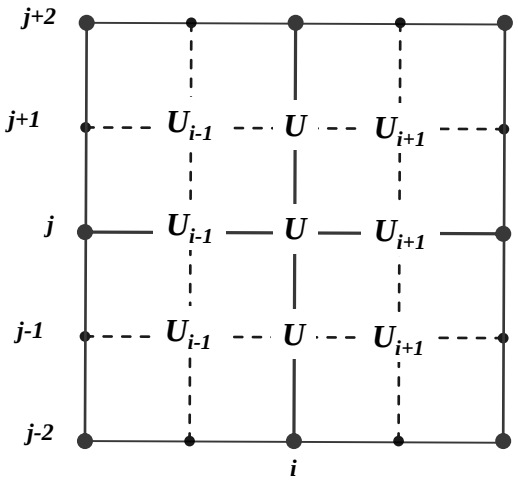
# Conclusion and Future work

## Recent optimization progress in dynamical core

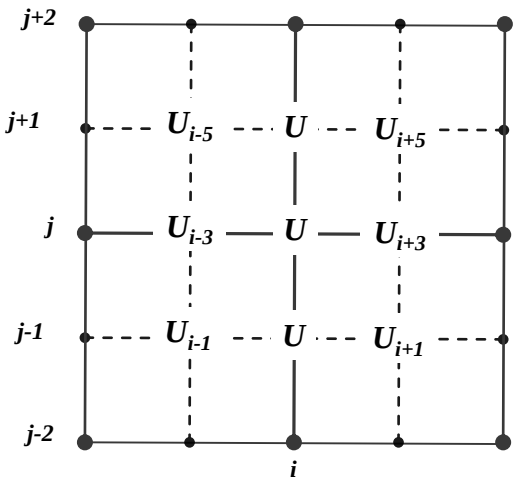
### Leap format optimization

After the *tend_lin* and *tend_adv* computation, the filtering is called to keep the stability. We have tried to use adaptive filtering method instead of FFT filtering for the high latitude. Our new work shows the filtering can be completely removed in the high latitude using leap format calculation.

**Original central difference format**




**New central difference with leap format**




**Original central difference format:**

$$\left(\frac{\partial F}{a\sin\theta\,\partial\lambda}\right)_{i,j}=\frac{F_{i+\frac{1}{2},j}-F_{i-\frac{1}{2},j}}{a\sin\theta_j\Delta\lambda}$$

**New central difference with leap format :**

$$\left(\frac{\partial F}{\mathrm{asin}\theta\,\partial\lambda}\right)_{i,j}=\frac{F_{leap1,j}-F_{leap2,j}}{\mathrm{asin}\theta_j\Delta\lambda^{*}kleap}$$

$$kleap=\frac{\arcsin(\cos 45^\circ\times\sin 0.5^\circ)}{(N\times\arcsin(\cos lat\times\sin 0.5^\circ))}$$

$$leap1=mod(I+kleap,NX)+\frac{I+kleap}{NX\times(IB+2)}$$

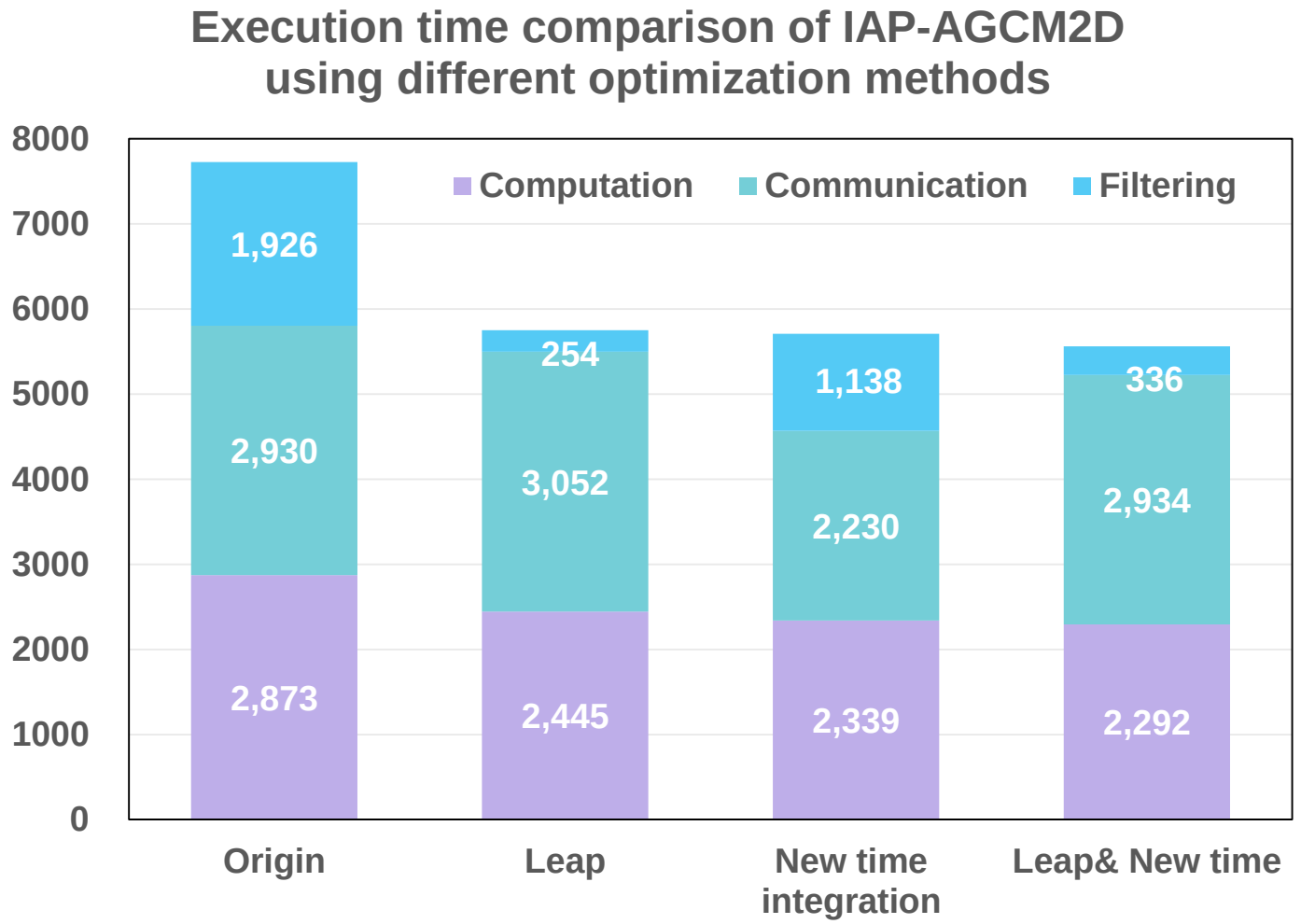
$$leap2=\frac{kleap-1+NLON}{I+NLON}\times NLON+(I-kleap+1)$$

# Conclusion and Future work

We experimented with several optimization methods on Tianhe-2 supercomputer.

As shown on the right figure, we simulate 2months for atmosphere model with 50km resolution. The max time step of origin model and leap format model are 90s, while the time step of new time integration optimization model and hybrid optimization model are 60s.

The results show the execution time of new time integration method is reduced by 1/3, the leap format greatly reduces filtering time. The hybrid method superimposes the performance advantages of both new time integration and leap format.



# THANK YOU