10M-Core Scalable Fully-Implicit Solver for Nonhydrostatic Atmospheric Dynamics

Chao Yang*§, Wei Xue†‡**, Haohuan Fu‡**, Hongtao You¶, Xinliang Wang†‡, Yulong Ao*§, Fangfang Liu*§, Lin Gan†‡**, Ping Xu†‡, Lanning Wang‖, Guangwen Yang†‡**, Weimin Zheng†

*Institute of Software and State Key Laboratory of Computer Science, Chinese Academy of Sciences, China

†Department of Computer Science and Technology, Tsinghua University, China

‡MOE Key Lab for Earth System Modeling, and Center for Earth System Science, Tsinghua University, China

§University of Chinese Academy of Sciences, China

¶National Research Center of Parallel Computer Engineering and Technology, China

‖College of Global Change and Earth System Science, Beijing Normal University, China

**National Supercomputing Center in Wuxi, China

***Abstract*—An ultra-scalable fully-implicit solver is developed for stiff time-dependent problems arising from the hyperbolic conservation laws in nonhydrostatic atmospheric dynamics. In the solver, we propose a highly efficient hybrid domain-decomposed multigrid preconditioner that can greatly accelerate the convergence rate at the extreme scale. For solving the overlapped subdomain problems, a geometry-based pipelined incomplete LU factorization method is designed to further exploit the on-chip fine-grained concurrency. We perform systematic optimizations on different hardware levels to achieve best utilization of the heterogeneous computing units and substantial reduction of data movement cost. The fully-implicit solver successfully scales to the entire system of the Sunway TaihuLight supercomputer with over 10.5M heterogeneous cores, sustaining an aggregate performance of 7.95 PFLOPS in double-precision, and enables fast and accurate atmospheric simulations at the 488-m horizontal resolution (over 770 billion unknowns) with 0.07 simulated-years-per-day. This is, to our knowledge, the largest fully-implicit simulation to date.**

***Index Terms*—atmospheric modeling; fully implicit solver; Sunway TaihuLight supercomputer; heterogeneous many-core architecture.**

I. Justification for ACM Gordon Bell Prize

An important attempt is made to design an ultra-scalable fully-implicit solver for nonhydrostatic atmospheric simulations. With both algorithmic and optimization innovations, the solver scales to 10.5-million heterogeneous cores on Sunway TaihuLight at an unprecedented 488-m resolution with 770-billion unknowns, sustaining 7.95 PFLOPS performance in double-precision with 0.07 simulated-years-per-day (SYPD).

Performance Attributes	Content
Category of achievement	*Time-to-solution*
Type of method used	*Fully implicit*
Results reported on basis of	*Whole application including I/O*
Precision reported	*Double precision*
System scale	*Measured on full-scale system*
Measurement mechanism	*Timers*

II. Simulation of Atmospheric Dynamics

Every year, extreme weather/climate events may bring economic loss in hundreds of billion dollars [1] and sometimes cause catastrophic disasters to the living condition of human beings [2]. Ever since the ENIAC system in 1950s, generations of scientists have been continuously working on improving the simulation and prediction capability of atmosphere models by developing innovative numerical algorithms on state-of-the-art computing platforms [2]. With six decades passed, the continuous advance of the scientific understanding about the climate system, the computing methods, and the computing capabilities have finally pushed us to the edge of seamless weather-climate simulations/predictions at the resolution of the km-level and beyond.

On the road to the seamless weather-climate prediction, a major obstacle is the difficulty of dealing with various spatial and temporal scales [3]. The atmosphere contains time-dependent multi-scale dynamics that support a variety of wave motions. For example, the seasonal Asian summer monsoon usually comes at the planetary length scale of the earth with the order of 10^3-10^4 km, but thunderstorms and tornadoes often develop in minutes with an horizontal scale range of 10 km to under a few hundred meters. It is therefore important for atmosphere models to deliver accurate simulation results at the ultra-high horizontal resolutions of kilometer or even hundreds of meters, along which great efforts in both the numerical algorithms and the high-performance computing should be made [4].

The fastest traveling waves of the atmosphere, such as the acoustic and inertia-gravity waves, are usually not of interest to the scientists. They often impose restrictive time step constraints for explicit time-stepping methods; and these restrictions are the major limiting factor of explicit methods in ultra-high-resolution atmospheric modeling. By using a simplified equation set such as the hydrostatic or anelastic (Boussinesq or sound-proof) equations, the fast waves are filtered out, but these simplifications are usually not accurate when the grid resolution tends to the km-level [5], [6]. Another way to stabilize fast waves is to make use of semi-implicit [7], [8] or split-explicit methods [9], [10], which relax the dependency between the time step length and the horizontal grid resolution. However, the relaxed dependency can still become a bottleneck of the time-to-solution performance at the extreme scale [11]. Fully implicit methods, on the other hand, are free of the stability limitation, and are therefore potentially desirable. The price of using a fully implicit method is that one need to solve one or a few large linear/nonlinear equation system at each time step, which requires innovative design to achieve high efficiency on state-of-the-art supercomputing platforms.

With many-core accelerators becoming the major provider of computing power in various supercomputers, we see a huge demand to migrate the weather/climate models to heterogeneous supercomputers. One challenge is to make an efficient utilization of the increasingly popular many-core accelerators or processors, which can be extremely difficult for implicit solvers on heterogeneous supercomputers. The Sunway TaihuLight supercomputer [12], released in 2016, pushes the parallelism to the level of over ten million cores, which poses another great scalability challenge to the current numerical algorithms and optimization paradigms.

In this work, we present a highly scalable fully implicit solver for three-dimensional nonhydrostatic atmospheric simulations governed by the fully compressible Euler equations. Unlike simplified equations with the hydrostatic or anelastic assumptions, the fully compressible Euler equations are accurate to the mesoscale with almost no assumption made [3]. In particular, we consider atmospheric flows in a regional domain above a rotating sphere with possibly nonsmooth bottom topography [13]:

$$\frac{\partial Q}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z} + S = 0, \tag{1}$$

$$\begin{aligned}
Q &= (\rho', \rho u, \rho v, \rho w, (\rho e_T)', (\rho q)')^T, \\
F &= (\rho u, \rho u u + p', \rho u v, \rho u w, (\rho e_T + p)\, u, \rho u q)^T, \\
G &= (\rho v, \rho v u, \rho v v + p', \rho v w, (\rho e_T + p)\, v, \rho v q)^T, \\
H &= (\rho w, \rho w u, \rho w v, \rho w w + p', (\rho e_T + p)\, w, \rho w q)^T, \\
S &= (0, \partial \bar{p}/\partial x - f \rho v, \partial \bar{p}/\partial y + f \rho u, \rho' g, 0, 0)^T,
\end{aligned}$$

where ρ, $\mathbf{v} = (u, v, w)$, p, θ and q are the density, velocity, pressure, total energy and moisture of the atmosphere, respectively. The Coriolis parameter is provided in f and all other variables such as g, γ are given constants. The values of $\rho' = \rho - \bar{\rho}$, $(\rho e_T)' = \rho e_T - \bar{\rho}\bar{e_T}$ and $p' = p - \bar{p}$ have been shifted according to a hydrostatic state that satisfies $\partial \overline{p}/\partial z = -\overline{\rho} g$. The system is closed with the equation of state $p = (\gamma - 1)\rho(e_T - gz - ||\mathbf{v}||^2/2)$. Note that we choose the total energy density instead of the traditional pressure- or temperature-based values as a prognostic variable to fully recover the energy conservation law and avoid the repeated calculation of powers.

The fully compressible Euler equations (1) are discretized with a conservative cell-centered finite volume scheme of second-order accuracy on a height-based terrain-following 3-D grid. A fully implicit second-order Rosenbrock method is employed for time integration, which supports adaptive time-stepping (turned off in this work to simplify the discussion on the performance).

III. State of the Art

Due to the long development history, existing weather and climate models are mainly designed for CPU-based platforms. Related HPC efforts are mainly focused on improving the scalability and efficiency to support increasingly higher resolutions. For example, thanks to the huge performance boosts delivered by the Earth Simulator and the K computer, Japanese groups have done a series of pioneering works, such as the 3.5-km and 7-km global simulations on the earth simulator [14] that successfully captured the lifecycles of two real tropical cyclones [15], and the 870-m global resolution simulation on K computer with 230 TFLOPS double-precision performance for 68 billion grid cells. In US, the CAM-SE dynamic core of NCAR supports up to 12.5-km resolution, and can provide a simulation speed of around 4.6 SYPD when using 172,800 CPU cores on Jaguar [16]. The Weather Research and Forecasting (WRF) model has been employed to simulate the landfall of Hurricane Sandy, providing a single-precision performance of 285 TFLOPS on 437,760 cores of Blue Waters [17]. In the recent initiative to build next generation global prediction system (NGGPS) of US [18], we see a number of candidates that can already support seamless weather-climate simulation at the scale of a few kilometers. Examples include the Model for Prediction Across Scales (MPAS) and the Finite Volume Model version 3 (FV3), which scales to 110,592 CPU cores on the Edison system with a simulation speed of around 0.16 SYPD for the 3-km resolution in double precision.

Due to the heavy legacy and the distributed computation pattern of atmospheric models, it involves both design challenges as well as huge programming efforts to port weather/climate models onto many-core accelerators. Early studies often focused on the many-core acceleration of standalone physics parameterization schemes ([19], [20]). In recent years, more efforts were made to migrate the dynamic cores or even complete atmospheric models to accelerator-based platforms, also pioneered by Japanese researchers. For example, on the TSUBAME 1.2 and 2.0 systems, T. Shimokawabe et al. conducted successful multi-node GPU-based acceleration of the ASUCA nonhydrostatic model [21], with a single-precision performance of 145 TFLOPS. More recently, a GPU-based acceleration of the NICAM model [22] on TSUBAME 2.5 sustained a double-precision performance of 60 TFLOPS using 2,560 GPUs. In China, our group have enabled both CPU-GPU and CPU-MIC accelerations of an explicit time-stepping global shallow water model on Tianhe-1A and Tianhe-2, both scaling to half-system levels with sustained double-precision performance of 800 TFLOPS [23] and 1.63 PFLOPS [24], respectively. Further, the work was extended to the 3-D nonhydrostatic case on Tianhe-2, scaling also to the half-system scale with over 8% FLOPS efficiency in double-precision Tianhe-2 [25]. The previous work mentioned above, though mostly focuses on explicit methods, may serve as guidances for us to develop highly efficient implicit solvers.

Many complex partial differential equation (PDE) based problems often require implicit solvers that allow for large time-step size but require to solve nonlinear equations. For homogeneous supercomputers, a most recent work is by Rudi et al. [26], in which a fully implicit solver based on an innovative AMG method scaled to 1.57 million homogeneous cores on the IBM Sequoia supercomputer with 96% and 33% parallel efficiency in terms of weak and strong scalability, respectively, sustaining a FLOPS efficiency of around 3.41% of in double-precision. Some previous efforts on designing highly efficient implicit solvers include [27], [28], [29], all on homogeneous CPU-based systems.

Due to the intrinsic "divide-and-conquer" nature, domain decomposition methods (DDMs) were recognized as a good iterative solver or preconditioner for solving large-scale linear or nonlinear equation systems resulted from the discretization of PDE-based problems on massively parallel cluster systems. In the past three decades, tremendous efforts have been made on both the theoretical analysis and the application techniques of DDMs for different types of PDEs. For elliptic PDEs, classical DDMs such as the additive and the multiplicative Schwarz methods have optimal convergence rate in terms of both strong and weak scalability, as long as certain coarse-level corrections are added [30]. But similar theoretical analysis does not apply to time-dependent hyperbolic PDEs such as the the fully compressible Euler equations arising from multi-physics conservation laws. It was observed in our previous work [31], [32] that, for time-dependent hyperbolic PDEs, coarse-level corrected DDMs are also a promising approach. We remark that multigrid based approaches, such as the AMG work by [26], are also potentially valuable to apply. But we prefer to keep a uniform data partition strategy on all mesh levels to achieve balanced load across different parallel computing units, which is easier to achieve when DDMs are used as the basic design on each level. In particular, we combine the DDMs within a multigrid cycle and propose a low-cost DD-MG method for preconditioning the solution of the discretized Euler equations.

A homogeneous domain partition strategy is usually preferred in traditional DDMs for the consideration of load balance. But this is no longer suitable for heterogeneous architectures, which provide another level of parallelism inside each compute node. This means that the subdomain solver of a DDM should be able to exploit the on-chip many-core resources and provide robust approximations to the subdomain solution. Unfortunately, classical subdomain solvers such as the incomplete LU factorizations are difficult to parallelize due to the sequential nature and the irregular behavior of the method. Considering a general many-core processor, the newly proposed PILU method [33], [34] is a promising approach. But it usually requires a few sweeps to achieve the similar convergence rate as the sequential ILU does, because the asynchronization introduced in the method breaks the data dependency. Therefore the parallel speedup of the PILU method is sub-optimal. By taking architecture advantage of Suway TaihuLight, we design a highly parallel ILU method that provides high speedup without sacrificing the convergence rate. Based on the newly design ILU method and the DD-MG algorithm, our proposed fully implicit solver can efficiently scale to the full-system scale on Sunway TaihuLight for solving nonhydrostatic atmospheric problems at ultra-high resolutions.

IV. The Sunway TaihuLight Supercomputer

A. System Overview

Released in June 2016, the Sunway TaihuLight supercomputer [12] claims the top place in the latest TOP500 list with a peak performance of 125 PFLOPS and a sustained Linpack performance of 93 PFLOPS. There are 40,960 compute nodes in the system, spanning across 40 cabinets, with each cabinet containing 4 supernodes. Each supernode includes 256 SW26010 processors that are fully connected by a customized network switch board, and 8 TB DDR3 memory. The network topology across supernodes is a two-level fat-tree. The global file system manages both SSD storage and HDD storage with the aggregation bandwidth of over 250 GB/s and the capacity exceeding 10 PB. An I/O forwarding architecture is integrated to handle the stability issue of the Lustre file system due to massive connections between clients and I/O servers.

The software environment of the system includes a customized 64-bit linux OS kernel and a customized compiler supporting C/C++, Fortran and mainstream parallel programing languages such as MPI, OpenMP and OpenACC. The message passing library on the Sunway TaihuLight supports MPI 3.0 specification and has been tuned for massively parallel run. A high-performance and light-weight thread library named *Athread* is also provided to exploit fine-grained parallelism within the socket.

B. The SW26010 Many-core Processor

The SW26010 processor works at the frequency of 1.45 GHz with an aggregated peak performance of 3.06 TFLOPS in double precision and an aggregated memory bandwidth of 130 GB/s. The general architecture of the processor is shown in Fig. 1. Each SW26010 processor comes with 4 core groups (CGs), with each including one management processing element (MPE) and one computing processing element (CPE) cluster of 64 CPEs, in total 260 cores in each processor. The MPE and CPE are both complete 64-bit RISC cores but serve different roles during the computation. The MPE, supporting the complete interrupt functions, memory management,

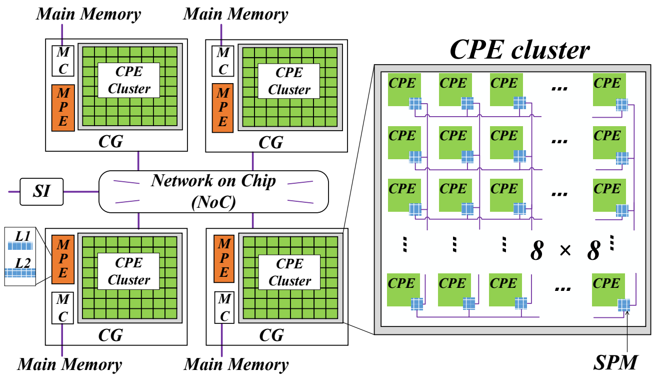

Fig. 1. The general architecture of SW26010 processor [12]. Each CG includes one MPE, one CPE cluster with 8×8 CPEs, and one memory controller (MC). These 4 CGs are connected via the network on chip (NoC). Each CG has shared memory space, connected to the MPE and the CPE cluster through the MC. All processors are connected with each other through a system interface (SI).

superscalar, and out-of-order issue/execution, is good at handling the management, task schedule, and data communications. In terms of the memory hierarchy, each MPE has a 32 KB L1 data cache, and a 256 KB L2 cache for both instruction and data. The CPE is designed for the purpose of maximizing the aggregated computing throughput while minimizing the complexity of the micro-architecture. The CPE cluster is organized as an 8×8 mesh, with a mesh network to achieve low-latency register data communication (P2P and collective communications) among the CPEs in one CG. Unlike the MPE, the CPE does not support interrupt functions. And each CPE has its own 16 KB L1 instruction cache and a 64 KB Scratch Pad Memory (SPM) that can be configured as either a Local Data Memory (LDM) that serves as user-controlled buffer (for performance-oriented programming) or a software-emulated cache for automatic data buffering (for more convenient porting of the program). Through the memory controller, Direct Memory Access (DMA) is supported for data transportation across the SPM and the main memory, and normal load/store instructions are also available for registers to transfer data with the main memory.

V. Major Innovative Contributions

A. Summary of Contributions

Our major contribution is a highly scalable fully implicit solver for the nonhydrostatic atmospheric dynamics governed by hyperbolic conservation laws, which enables fast and accurate atmospheric simulations at ultra-high resolutions. Our solver is developed based on a hybrid domain-decomposed multigrid (DD-MG) preconditioner to achieve robust convergence rate on distributed parallel computers at the extreme scale, and a geometry-based pipelined incomplete LU factorization (GP-ILU) method to efficiently solve the overlapping subdomain problems by fully exploiting the on-chip many-core parallelism.

We have implemented the fully implicit solver in an experimental atmospheric dynamic core and deployed it on the Sunway TaihuLight supercomputer. The fully implicit solver scales well to the entire system with over 10.5 million heterogeneous cores in both strong and weak scaling cases. In particular, our implicit solver is free of the time step constraint, and can provide a simulation speed of around 1.0 SYPD at the horizontal resolution of 3-km, which is substantially superior to our explicit counterpart developed from our previous work on Tianhe-2. The fully implicit solver is able to conduct simulations at the unprecedented 488-m resolution (total DOFs: over 770 billion) with 0.07 SYPD, sustaining an aggregate double-precision performance of nearly 8 PFLOPS with over 50% parallel efficiency. This is, to the best of our knowledge, the largest fully implicit simulation in terms of total DOFs, total number of cores and aggregate performance, to date.

B. Algorithm

1) The DD-MG preconditioner: For the fully compressible Euler equations, the linear Jacobian system is especially difficult to solve due to the hyperbolic and stiff nature of the problem. We propose a hybrid preconditioner DD-MG that combines both geometric multigrid and algebraic domain decomposition methods to accelerate the convergence of the linear solver. In the DD-MG method, the MG component is defined as $M^{-1} = M_f^{-1} + M_c^{-1} - M_f^{-1} A_f M_c^{-1}$, where M_f^{-1} is the one-level DD preconditioner, M_c^{-1} is the projected coarse-level correction that can be defined recursively, and A_f is the Matrix-free Jacobian. In particular, we use the cascade κ-cycle MG with low-order pre- and post-smoothers and the left restricted additive Schwarz (RAS) [35] DD component built based on a low-order finite volume scheme in the DD-MG preconditioner, as illustrated in Fig. 2.

2) The GP-ILU factorization: On a given overlapping subdomain, we construct an approximated Jacobian matrix based on a low-order 7-point spatial discretization and order the unknowns without breaking the coupling of all physical components on each mesh cell. The subdomain matrix then carries the mesh information that

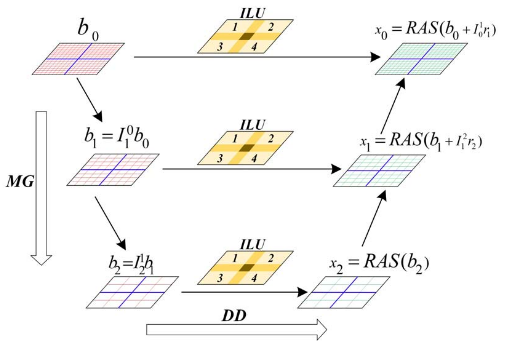

Fig. 2. The DD-MG preconditioner of three levels, which is a hybrid composition of the algebraic DD and a geometric κ-cycle MG. In particular, on each MG level, we use the one-level RAS method for the DD preconditioning to exploit the same degree of parallelism on the process level.

can be used further in the subdomain solver, which has been found helpful [29] to improve not only the convergence but also the parallel performance. In the DD-MG framework, the subdomain solver can be solved inexactly by an incomplete factorization method. However, classical ILU-based methods are difficult to be parallelized due to the sequential nature and the irregular behavior of the method. Considering a general many-core processor, the newly proposed PILU method [33], [34] is a promising approach. Based on it, we can design a geometric ILU method for solving the subdomain problems. But the parallel speedup of the PILU method is sub-optimal due to the break of the data dependency. Using the fast register communication mechanism (detailed in Section V-C) supported by the SW26010 CPU, we are able to design a new parallel ILU method, the geometry-based piplined ILU (GP-ILU) method, which faithfully maintains the data dependency of the original ILU method, but exploits the on-chip parallelism more efficiently.

All major operations in our solvers are summarized in Table I. For the explicit solver, only the *FX* kernel is required, along with a few vector update operations.

C. Implementation and Optimization

We implement the proposed fully implicit solver as well as an explicit one based on the PETSc (Portable Extensible Toolkit for Scientific computation [36]) library, by which we set the in-memory data layout as the array-of-structures in the z-x-y order. Then we perform a systematic optimization across the process, the thread, as well as the instruction level, and achieve substantial speedups in all performance-critical kernels.

TABLE I
LIST OF MAJOR KERNELS IN THE FULLY IMPLICIT SOLVER.

Kernel	Input	Output
FX	x	$F(x)$
AX	x, $\tilde{x}$	$Ax \equiv (\partial F(\tilde{x})/\partial \tilde{x})x$
MAT	$\tilde{x}$	$A_p \equiv \partial F^{low}(\tilde{x})/\partial \tilde{x}\big\|_{\Omega_p}$
RAS	b	$\sum_{p=1}^{np}(R_p^0)^T(L_pU_p)^{-1}R_p^\delta b$
ILU	$b_p = R_p^\delta b$	$(L_pU_p)^{-1}b_p$ where $L_pU_p \approx A_p$
GCR	b	One GCR iteration applied on b
MG	b	One MG κ-cycle applied on b

Due to the massive parallel computing capabilities and limited memory bandwidth of SW26010, how to exploit as much parallelism as possible and how to best utilize the limited memory resources are crucial for performance. In particular, we focus on the effective use of the small but fast LDM on each CPE of SW26010 processor. In this section, we highlight two major categories of optimization techniques: the LDM-oriented partitioning that identifies the most suitable form of parallelism for multi-threading and vectorization; and the memory-related optimization for maximizing the data reuse and coalescing memory accesses.

1) LDM-oriented Partitioning: On the SW26010 processor, three partitioning schemes are employed, corresponding to different types of kernels, as shown in Fig. 3. For the *AX* kernel, the computation domain is decomposed into the halo and the inner parts to do communication-computation overlapping. For the inner part, a 2.5D blocking is combined with a double-buffering scheme to hide the memory access latency. The same partition and scheduling strategy is also used in the dominated *FX* kernel of our explicit solver. Compared with the *AX* kernel, the *MAT* kernel has a similar computation pattern but does not require halo exchange, and involves fewer inputs and more outputs. Therefore, we use a column-wise blocking/pipelining along the z-axis in the z-x plane, as shown in Fig.3(b).

With the support of inter-thread communication and synchronization, we implement the GP-ILU method as a two-level pipeline (inter/intra-thread levels). This method provides a better solver performance when compared to the blocked PILU method since only one sweep is needed. Details are shown in Fig.3 (c). We partition the data domain of each process into several 8$\times$8 cell columns, which exactly maps to the layout of the 8$\times$8 CPE cluster. With this fine-grained partition, the overhead of imbalance during startup and finalization of the pipelines can be minimized. Note that the factorization, forward (solving the lower triangular part), and backward (solving the upper triangular part) processes, can be performed in a similar manner.

2) Memory-related Optimization:

a) A customized data sharing scheme through register communication: In 2.5-D blocking, each CPE has to directly access the data in a strided way from memory, which leads to inefficient memory usage. To resolve this issue, an on-line data sharing method is implemented to maximize data locality via the fast register communication feature, as shown in Fig. 4. More CPEs in a group lead to more continuous memory access and a better data reuse, but the overhead of the higher on-line process and synchronization is also higher. Based on our experiments, the optimal choice is to use 4 CPEs in a same CG together.

b) On-the-fly array transposition: In the *FX*, *AX* and *MAT* kernels, there are both AOS-friendly and SOA-friendly computation parts. We conduct the on-the-fly array transposition to achieve highly efficient transformations between AOS and SOA, and better vectorization. The shuffle instruction supported by SW26010 is used to implement this feature. In normal scenarios, the shuffle of two vectors can be finished in one operation. Using the shuffle instruction, we reduce the latency of conversion to only 12 instruction cycles on SW26010 when converting four cell structures with six double-precision members into six 256-bit vectors and vice versa.

The partitioning method, the GP-ILU method, the on-the-fly array transposition method, and the on-cache data sharing technique, can also be applied to other many-core processors, such as MIC and GPU.

3) xMath: There are several other operations that need to be optimized on the Sunway supercomputer. The operations include BLAS-1 vector updates as well as halo exchange. We have developed a high-performance extened math library called *xMath* that supports highly optimized BLAS, LAPACK, FFT operations on the Sunway TaihuLight platform. We call the BLAS-1 operations in the *xMath* library to improve the performance of vector updates. In addition, we also optimize some other vector operators using many-core parallelization and fuse some kernels when it is possible. By calling the *xMath* library and conduct manual optimizations, most BLAS-1 vector operations are boosted by a factor of around 20$\times$ as compared to MPE-only versions.

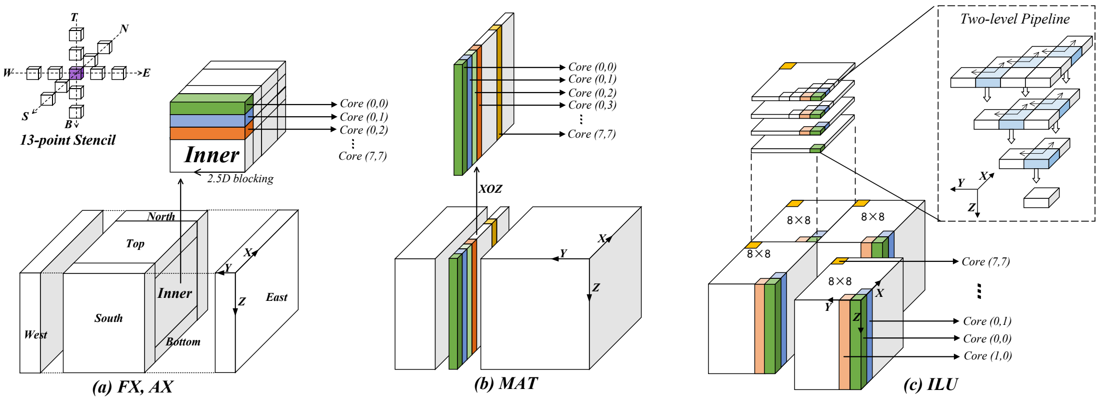

Fig. 3. The data partitioning and task scheduling of different kernels in our solver. (a). the AX and FX kernels are partitioned into inner and halo parts. Following the 2.5-D blocking for inner part, the proper block size for one CPE can be determined by the consideration on the LDM size, vectorization, double-buffering footprint and DMA efficiency, which is 4x4 on SW26010. (b) a column-wise blocking/pipelining methodology is presented for the MAT kernel and the block size is a multiple of 4 for vectorization. (c). the data domain is partitioned into several 8x8 blocks, to perform a two-level pipelining method. Take the forward process of ILU(0) as an example, the inter-thread level pipeline can be exploited on the x-y plane by taking the advantage of the fast register communication across the CPE cluster, as one CPE only needs the results from its east and south neighbors to start the calculation. The inner-thread level pipeline is performed along with z direction within each CPE according to the limited size of LDM.

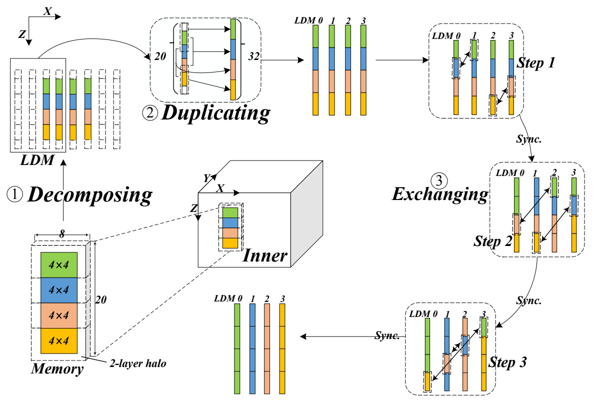

Fig. 4. The customized data sharing method used in stencil-like kernels including *FX* and *AX*. Here, each block contains 4×4 cells and 2 halo layers. ① decomposing: 4 cores are grouped together, each of which loads the data of $4\times 4+2\times 2 = 20$ cells continuously. ② duplicating: certain data on each core is duplicated to construct the data domain with 4×8 (i.e. 32) cells due to 2-layer halos. ③ exchanging: the resulting data is exchanged between different cores along with pairs in group via register communication, and finally each core obtains their required data.

VI. Experiment Setup

A. Design of Experiments

As a fundamental atmospheric process, the baroclinic instability is responsible for the generations of mid-latitude cyclones and storm systems that may result in severe weather/climate disasters. It is therefore of crucial importance for an atmospheric model to accurately reproduce this dynamical phenomenon with high efficiency. We employ the moist baroclinic instability test in a β-plane 3D channel [6] to validate the correctness and examine the performance of the proposed fully implicit solver in our experimental dynamic core. In the setup, the test is initiated by adding a confined perturbation in the zonal wind field to a geostrophically balanced background zonal flow at the earth troposphere. The computational domain in the baroclinic instability test is a 3D channel spanning a $40,000$ km $\times$ $6,000$ km $\times$ 30 km range, with periodic boundary conditions along the flow direction and free-slip, non-penetrating boundaries everywhere else. Although designed in a plane channel, this configuration retains the triggering mechanism of the baroclinic jet with great details, resembling the north semisphere with the latitude range of 18°N to 72°N.

We run the test using our optimized fully implicit solver with a horizontal resolution of 10 km and a vertical resolution of 500 m for the purpose of comparison with referenced others provided in other atmospheric models. With the fully implicit method, the time step for the simulation can be set to as large as 1200 s, which is substantially greater than the explicit time step (usually a few seconds or less). By using 16,000 CGs on the Sunway TaihuLight supercomputer, we are able to conduct the simulation at the speed of around 4.1 SYPD. The simulation results of the 500 m level profile at day 10 to 16 are presented in Fig. 5. It is observed from the figure that our fully implicit solver can successfully capture the baroclinic jet where distinct low and high temperature regions are generated with sharp fronts. The simulated results at day 10 agree well with reference results such as those in [6]. After day 12, the wave starts to break due to the increasingly strong interaction of large eddies; and our fully implicit solver is able to continue the simulation with unreduced computing efficiency (c.f., the time step size should be quickly reduced when using an explicit solver instead).

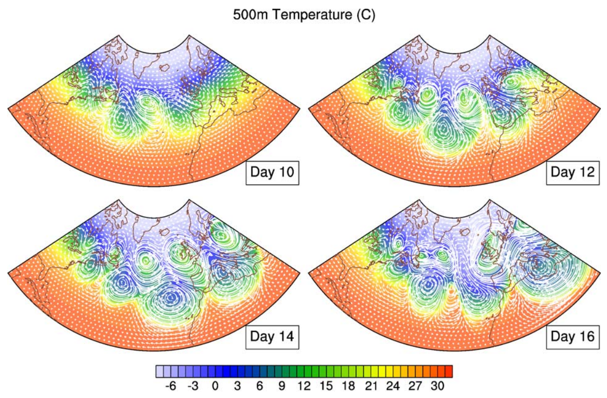

Fig. 5. The 500 m level simulation results at day 10, 12, 14 and 16 for the baroclinic instability test. Shown here are the temperature contours with the overlaid horizontal wind field, for which only around 1/3 of the whole computational domain is drawn to see more details of the baroclinc jets and eddies.

B. FLOPS Measurement

To conduct an accurate performance measurement for both our implicit and explicit solvers, we collect the number of double-precision arithmetic operations using three different methods summarized as follows.

- Manually counting all double-precision arithmetic instructions in the assembly code.
- Analysis by using hardware performance monitor, PERF, provided by the vendor of the Sunway TaihuLight supercomputer, to collect the number of double-precision arithmetic instructions retired on the CPE cluster.
- Measuring the double-precision arithmetic operations by running the same MPE-only versions of our solvers instrumented by Performance API (PAPI) on an Intel Xeon E5-2697v2 platform.

The first and the second methods provide almost identical double-precision arithmetic operation counts, while the result with PAPI is 3% higher. This is probably due to the difference between the Intel Xeon platform and the SW26010 platform. In our study, we employ the second method (the PERF result) to count the exact total number of double-precision arithmetic operations. The FLOPS results can then be calculated in PETSc by utilizing its profiling functionality.

VII. Performance Results

A. Many-core Acceleration on SW26010

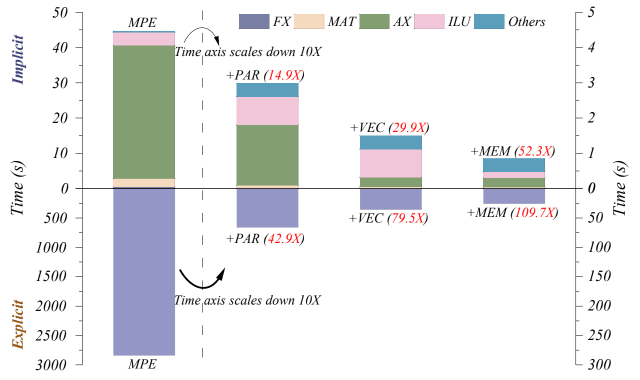

Fig. 6. Average runtime measured by per-minute-long simulation. Both the implicit and explicit solvers are using 7,260 MPI processes, with a horizontal resolution of 2.8km. Each process is dealing with a 3D block in the size of $64 \times 64 \times 128$. *MPE*, *+PAR*, *+VEC* and *+MEM* represent the MPE-only version, the threaded version with the LDM-oriented partition, the vectorized version, and the final version, respectively. The substantial difference of the run time between the fully-implicit and the explicit solvers are partially due to the difference of the time step size, which will be discussed later in detail.

As far as we know, our work in this paper is the first atmospheric dynamic solver that scales over a heterogeneous supercomputer with over 10 million cores. Therefore, the first key effort in improving the overall performance is to make an efficient utilization of the 260 cores within the SW26010 many-core processor, in terms of both the computing and the memory resources.

Figure 6 demonstrates the performance evaluation of both our implicit and explicit solvers on the SW26010 many-core processor, when applying three different optimization steps covered in Section V (the LDM-oriented partition, vectorization, and memory-related optimization). We use the MPE version (using only the 4 MPEs within the processor) as the base line.

In our implicit solver, the *AX* kernel is the most time-consuming kernel, taking over 84.5% of the entire execution time. By applying the LDM-oriented partition scheme, we manage to cut the execution time of the *AX* kernel by 22 times in the multi-threading step, and further by 6.5 times in the vectorization step (with benefits of both vectorization and strength reduction), which demonstrates the effectiveness of the identified level of parallelism (over 140 times for the two steps combined). We observe a similar performance boost for the the *MAT* kernel (53.5 times for the two steps combined).

The only exception is the *ILU* kernel. Due to the fine-grained and unaligned memory operations introduced by the proposed blocked PILU method, the *ILU* kernel only achieves 4.6 times, and becomes the most time-consuming part after the first two steps. While this is considered a major performance issue in almost all the existing many-core architectures, by mainly combining the geometry-based pipelined ILU scheme and our proposed register communication based data sharing, we can achieve both a better data locality and fewer iterations, leading to a substantial speedup of 4.8 times.

In our explicit solver, the *FX* kernel consumes almost all the computation time. Similarly, a carefully-designed LDM-partition scheme manages to achieves 43 times speedup at the multi-threading step, and a further 1.9 times speedup at the vectorization step. The memory-related optimization provides another 1.4 times performance improvement.

In general, our optimization strategies proposed in Section V-C enable a good scaling of performance within the 260 cores of SW26010 processor. The execution times of our implicit and explicit solvers are reduced by 52 and 110 times respectively. The significant performance improvements demonstrate that our optimization strategies are capable of identifying the right mapping between our atmospheric simulation algorithm and the underlying SW26010 many-core architecture, which form the basis for the tremendous simulation capability of our solver in a large-scale scenario.

B. *Strong Scaling Results*

The strong scaling tests are carried out with two configurations. One is on a 20352×3072×192 mesh of 72.0 billion unknowns and the other is on a 13632×2016×192 mesh of 31.8 billion unknowns, corresponding to the 2-km and the 3-km horizontal resolutions, respectively. We set the number of vertical levels to 192, which is relatively large than normal, so that we can examine the effects of the domain decomposition in all three directions. The strong scaling results are shown in Fig. 7, in which the computing throughput is measured in SYPD. In both cases, we start with the smallest possible number of processes (i.e., CGs) that the memory capacity allows and increase the number of processes gradually to the full-system scale. The fully implicit solver scales well to the whole machine in both cases. In particular, a

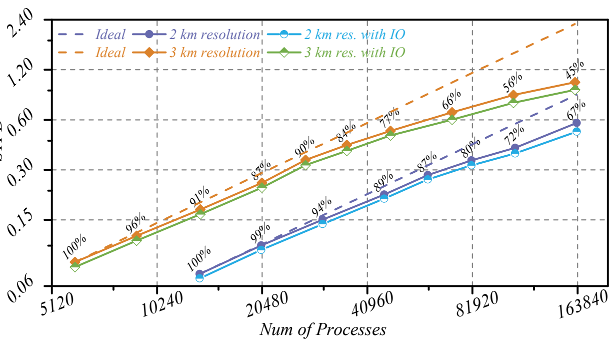

Fig. 7. Strong scaling results on the Sunway TaihuLight supercomputer. For the 2-km run, the solver scales from 13,568 processes to the whole machine with a parallel efficiency of 67%, leading to a 8.1× increase of the SYPD from 0.07 to 0.57. For the 3-km run, the code scales from 5,964 processes to the entire machine with a parallel efficiency of 45%, leading to a 12.2× increase of the SYPD from 0.083 to 1.01.

1.01 SYPD in double-precision is achieved for the 3-km run at the full-system scale (10.46 million cores in total).

In addition, we also evaluate the performance of the entire dynamic core including the I/O and initialization and provide the results in the same figure. Compared to the solver-only results, the overhead of I/O and initialization is within the 5% range, depending on how frequent I/O is required. This is due to the overlapping of computation and I/O efficiently supported by the asynchronous and non-blocking I/O facility and high performance SSD storage of the Sunway TaihuLight supercomputer.

C. *Weak Scaling Results*

TABLE II
CASE CONFIGURATIONS FOR WEAK-SCALING TESTS.

Number of processes	Mesh size #1 (Implicit/Explicit)	Mesh size #2 (Explicit)
168 × 38 × 1	16128 × 2432 × 128	32256 × 4864 × 128
200 × 45 × 1	19200 × 2880 × 128	38400 × 5760 × 128
250 × 56 × 1	24000 × 3584 × 128	48000 × 7168 × 128
300 × 67 × 1	28800 × 4288 × 128	57600 × 8576 × 128
367 × 82 × 1	35232 × 5248 × 128	70464 × 10496 × 128
408 × 91 × 1	39168 × 5824 × 128	78336 × 11648 × 128
453 × 101 × 1	43488 × 6464 × 128	86976 × 12928 × 128
504 × 113 × 1	48384 × 7232 × 128	96768 × 14464 × 128
610 × 137 × 1	58560 × 8768 × 128	117120 × 17536 × 128
672 × 151 × 1	64512 × 9664 × 128	129024 × 19328 × 128
756 × 170 × 1	72576 × 10880 × 128	145152 × 21760 × 128
853 × 192 × 1	81888 × 12288 × 128	163776 × 24576 × 128

In the weak scaling tests we focus on examining the performance of both implicit and explicit solvers. The detailed configurations of the weak scaling tests can be found in Table II. For the implicit run, the size of the subdomain is 96×64×128 and the number of processes is increased from 6,384 to the full-system scale (25.6-fold increase), corresponding to 2.48-km and 488-m horizontal resolutions, respectively. The weak scaling results in terms of sustained PFLOPS are shown in Fig. 8, from which we observe that both the fully implicit and the explicit codes scales well to the whole machine, sustaining up to 7.95 and 23.66 PFLOPS, respectively. In addition, we also provide the scaling results of the explicit code with a larger subdomain size and the sustained performance of the explicit code at the full-system scale is increased to 25.96 PFLOPS. Compared with state-of-the-art work of scalable implicit solver [26] (3.41% of peak performance on 1.57 million homogeneous cores of IBM Sequoia) , we manage to get 2-fold FLOP efficiency on a much larger system (6.45% of peak performance with 10.65 million heterogeneous cores of the Sunway TaihuLight supercomputer).

We remark that performance measured only in PFLOPS could be misleading. When considering the overhead due to the increase of number iterations, the parallel efficiency of the fully implicit solver is 52% as indicated in the figure. For the explicit solver, the time step size is required to be decreased as the number of processes increases (c.f., the fully implicit solver uses a fixed time step size of 240s). But it is usually not considered when calculating the weak scaling parallel efficiency of explicit codes; therefore we follow the tradition and show the parallel efficiency of the explicit solver in the picture, which is 99.5% for the test case with mesh size #1 and 100% for the case with mesh size #2, suggesting that the cost of halo exchange is successfully hidden by the computation of the inner part of each subdomain.

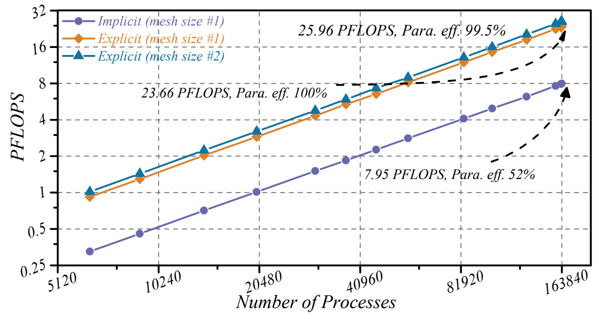

Fig. 8. Weak scaling results on Sunway TaihuLight.

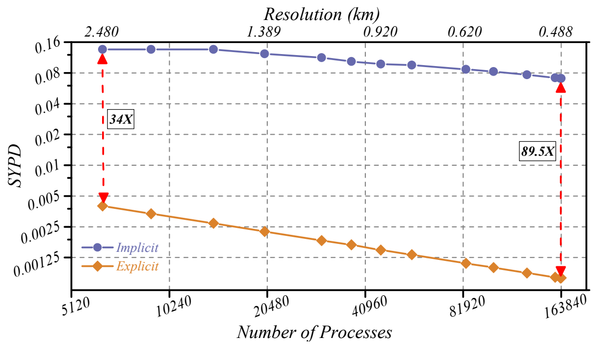

Fig. 9. Time-to-solution results on the Sunway TaihuLight supercomputer. The performance is measured in SYPD, which is the simulation/prediction capability of atmospheric models.

D. Analysis of the Time-to-solution

To conduct a more fair comparison of the fully-implicit and the explicit solvers, we further examine the weak scaling performance in terms of SYPD, which is the time-to-solution measured with respect to the simulation/prediction capability of atmospheric models. We use the same configuration (#1 in Table II) of the weak scaling tests and show the results in Fig. 9. In the ideal case, it is expected that a same SYPD is kept as more processes are used. However, for the stiff time-dependent problems governed by hyperbolic conservation laws, it is often hard to achieve. For the explicit solver, as the number of processes increases, the resolution becomes finer, leading to the decrease of the time step size due to stability reasons. For the implicit solver, a uniform time step size can be used (240s here), but there is a mild increase (1.9×) of the number of iterations as the problem size gets 25.6-fold larger. Observed from Fig. 9, the fully implicit solver is able to deliver 34× more SYPD compared to the explicit solver when using 6,384 processes at the 2.48-km resolution. And the SYPD increase is further improved to 89.5× at the ultra-fine 488-m resolution using the whole system, leading to the simulation/prediction capability of 0.07 SYPD.

VIII. Implications

In this work, we design a highly scalable fully implicit solver and apply it in an experimental dynamic core for nonhydrostatic atmospheric simulations. The solver scales to the full-system scale of 10.5 million cores on the Sunway TaihuLight, providing a simulation speed of 1 SYPD at the horizontal resolution of 3-km and sustaining an aggregate performance of 7.95 PFLOPS in double precision. We summarize the measured simulation speed of our fully implicit solver in Table III, which show competitive results as compared to the current state-of-the-art. The scalable, efficient and robust experimental dynamic core may open the opportunity to revisit the possibility of practical use and the potential for the fully-implicit method in next-generation atmospheric modeling, especially for ultra-high-resolution simulations and large scale computing systems.

TABLE III
SIMULATION SPEED OF OUR FULLY IMPLICIT DYNAMIC CORE.

Resolution	488-m	3-km	12.5-km	25-km
SYPD	0.07	1.0	4.9	14.4
# Processes	163,776	161,028	16,000	16,000

In terms of the numerical methodology and application advancement, our work has demonstrated that fully-implicit methods could be an important option for performing numerical weather and climate simulations, especially for ultra high resolution scenarios. Our comparison between the implicit method and the explicit method (with a similar level of optimizations on the Sunway TaihuLight system) demonstrates an 89.5X time-to-solution advantage for the ultra high resolution of 488 m. Although our explicit solver provides a significantly higher arithmetic performance of 25.96 PFLOPS, our implicit method, with an arithmetic performance of only 7.95 PFLOPS, can cut the computing time-to-solution by almost two orders of magnitude and achieve a simulation speed that is equivalent to our explicit method running on an Exascale system. While one may argue that the explicit method can be further improved by, e.g., time-splitting methods, we can also improve the performance of the implicit solver by applying techniques such as adaptive time-stepping. Therefore, we consider this as a reasonable comparison that demonstrates the benefits and constraints of explicit and implicit methods in atmospheric modeling. The success of the method depends strongly on the design of the solver, in which the convergence, scalability, utilization of underlying hardware architecture are of great importance. The proposed fully implicit solver, especially the hybrid DD-MG preconditioner and the GP-ILU factorization are not only suitable to atmospheric modeling, but also applicable to many important applications governed by time-dependent hyperbolic conservation laws.

On the aspect of the hardware system, our work has demonstrated that many-core architectures could be a suitable candidate for running large-scale PDE solvers with innovative algorithms and well-tuned implementations. While the SW26010 processor is a brand new CPU based on many-core architecture, the corresponding parallel programming interfaces make it possible for us to customize our numerical algorithms and carefully tune the parallelization, buffering, and communication schemes to achieve a highly-efficient implementation of our fully-implicit solver on the Sunway TaihuLight supercomputer. Our work may serve as a base and guidance for domain experts, to get inspiring knowledge on designing scalable and efficient algorithms for large-scale PDE solvers, especially in geophysical fluid dynamics on the cutting-edge heterogeneous systems.

In the future, we would continue our efforts on the new-generation Sunway system, and collaborate with climate scientists to extend our current dynamic solver framework into an atmospheric model that can perform ultra-high resolution atmospheric simulations/predictions for seamless weather-climate modeling.

ACKNOWLEDGMENTS

We would like to acknowledge the contributions of Prof. Jiachang Sun and Prof. Xiao-Chuan Cai for insightful suggestions on algorithm design. We wish to offer our deepest thanks to NRCPC and NSCC-Wuxi for building the Sunway TaihuLight supercomputer and supporting us with the computing resources as well as the technical guidance to bring our ideas into reality. This work was partially supported by the National Key Research & Development Plan of China under grant# 2016YFB0200600, 2016YFA0602200 and 2016YFA0602103, Natural Science Foundation of China under grant# 91530103, 91530323, 61361120098, 61303003 and 41374113, and the 863 Program of China under grant# 2015AA01A302. The corresponding authors are C. Yang (yangchao@iscas.ac.cn), W. Xue (xuewei@tsinghua.edu.cn), H. Fu (haohuan@tsinghua.edu.cn) and L. Gan (lin.gan27@gmail.com).

REFERENCES

[1] J. K. Lazo, M. Lawson, P. H. Larsen, and D. M. Waldman, "US economic sensitivity to weather variability," *Bulletin of the American Meteorological Society*, p. 709, 2011.
[2] P. Bauer, A. Thorpe, and G. Brunet, "The quiet revolution of numerical weather prediction," *Nature*, vol. 525, no. 7567, pp. 47–55, 2015.
[3] P. H. Lauritzen, C. Jablonowski, M. A. Taylor, and R. D. Nair, Eds., *Numerical Techniques for Global Atmospheric Models*. Springer, 2011.
[4] C. Bretherton, "A National Strategy for Advancing Climate Modeling," 2012.
[5] R. Klein, U. Achatz, D. Bresch, O. Knio, and P. Smolarkiewicz, "Regime of validity of soundproof atmospheric flow models," *J. Atmos. Sci.*, vol. 67, pp. 3226–3237, 2010.
[6] P. Ullrich and C. Jablonowski, "Operator-split Runge-Kutta-Rosenbrock methods for nonhydrostatic atmospheric models," *Monthly Weather Review*, vol. 140, no. 4, pp. 1257–1284, 2012.

[7] M. Kwizak and A. J. Robert, “A semi-implicit scheme for grid point atmospheric models of the primitive equations,” *Mon. Wea. Rev.*, vol. 99, pp. 32–36, 1971.

[8] C. Temperton and A. Staniforth, “An efficient two-time-level semi-Lagrangian semi-implicit integration scheme,” *Q. J. R. Meteorol. Soc.*, vol. 113, pp. 1025–1039, 1987.

[9] J. Klemp and R. B. Wilhelmson, “The simulation of three dimensional convective storm dynamics,” *J. Atmos. Sci.*, vol. 35, pp. 1070–1096, 1978.

[10] M. Satoh, “Conservative scheme for the compressible nonhydrostatic models with the horizontally explicit and vertically implicit time integration scheme,” *Mon. Wea. Rev.*, vol. 130, pp. 1227–1245, 2002.

[11] V. A. Mousseau, D. A. Knoll, and J. M. Reisner, “An implicit nonlinearly consistent method for the two-dimensional shallow-water equations with Coriolis force,” *Mon. Wea. Rev.*, vol. 130, pp. 2611–2625, 2002.

[12] H. Fu, J. Liao, J. Yang, L. Wang, Z. Song, X. Huang, C. Yang, W. Xue, F. Liu, F. Qiao *et al.*, “The Sunway TaihuLight supercomputer: system and applications,” *Science China Information Sciences*, vol. 59, no. 7, p. 072001, 2016.

[13] Y. Ogura and N. A. Phillips, “Scale analysis of deep and shallow convection in the atmosphere,” *Journal of the atmospheric sciences*, vol. 19, no. 2, pp. 173–179, 1962.

[14] M. Satoh, T. Matsuno, H. Tomita, H. Miura, T. Nasuno, and S.-i. Iga, “Nonhydrostatic icosahedral atmospheric model (NICAM) for global cloud resolving simulations,” *Journal of Computational Physics*, vol. 227, no. 7, pp. 3486–3514, 2008.

[15] H. Fudeyasu, Y. Wang, M. Satoh, T. Nasuno, H. Miura, and W. Yanase, “Global cloud-system-resolving model NICAM successfully simulated the lifecycles of two real tropical cyclones,” *Geophysical Research Letters*, vol. 35, no. 22, 2008.

[16] J. M. Dennis, J. Edwards, K. J. Evans, O. Guba, P. H. Lauritzen, A. A. Mirin, and et. al., “CAM-SE: A scalable spectral element dynamical core for the Community Atmosphere Model,” *International Journal of High Performance Computing Applications*, vol. 26, no. 1, pp. 74–89, 2012.

[17] J. Peter, M. Straka *et al.*, “Petascale wrf simulation of hurricane sandy deployment of ncsa’s cray xe6 blue waters,” in *Proceedings of ACM SC’13 Conference*, vol. 63, 2013.

[18] J. Michalakes, R. Benson, T. Black, M. Duda, M. Govett, T. Henderson, P. Madden, G. Mozdzynski, A. Reinecke, and W. Skamarock, “Evaluating Performance and Scalability of Candidate Dynamical Cores for the Next Generation Global Prediction System,” 2015.

[19] R. Kelly, “GPU Computing for Atmospheric Modeling,” *Computing in Science and Engineering*, vol. 12, no. 4, pp. 26–33, 2010.

[20] J. Linford, J. Michalakes, M. Vachharajani, and A. Sandu, “Multi-core acceleration of chemical kinetics for simulation and prediction,” in *High Performance Computing Networking, Storage and Analysis, Proceedings of the Conference on*, Nov 2009, pp. 1–11.

[21] T. Shimokawabe, T. Aoki, J. Ishida, K. Kawano, and C. Muroi, “145 TFlops performance on 3990 GPUs of TSUBAME 2.0 supercomputer for an operational weather prediction,” *Procedia Computer Science*, vol. 4, pp. 1535–1544, 2011.

[22] H. Yashiro, M. Terai, R. Yoshida, S.-i. Iga, K. Minami, and H. Tomita, “Performance Analysis and Optimization of Nonhydrostatic ICosahedral Atmospheric Model (NICAM) on the K Computer and TSUBAME2. 5,” in *Proceedings of the Platform for Advanced Scientific Computing Conference*. ACM, 2016, p. 3.

[23] C. Yang, W. Xue, H. Fu, L. Gan, L. Li, Y. Xu, Y. Lu, J. Sun, G. Yang, and W. Zheng, “A Peta-scalable CPU-GPU algorithm for global atmospheric simulations,” in *ACM SIGPLAN Notices*, vol. 48, no. 8. ACM, 2013, pp. 1–12.

[24] W. Xue, C. Yang, H. Fu, X. Wang, Y. Xu, L. Gan, Y. Lu, and X. Zhu, “Enabling and scaling a global shallow-water atmospheric model on Tianhe-2,” in *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*. IEEE, 2014, pp. 745–754.

[25] W. Xue, C. Yang, H. Fu, X. Wang, Y. Xu, J. Liao, L. Gan, Y. Lu, R. Ranjan, and L. Wang, “Ultra-Scalable CPU-MIC Acceleration of Mesoscale Atmospheric Modeling on Tianhe-2,” *IEEE Trans. Computers*, vol. 64, no. 8, pp. 2382–2393, 2015.

[26] J. Rudi, A. C. I. Malossi, T. Isaac, G. Stadler, M. Gurnis, P. W. Staar, Y. Ineichen, C. Bekas, A. Curioni, and O. Ghattas, “An extreme-scale implicit solver for complex PDEs: highly heterogeneous flow in earth’s mantle,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2015, p. 5.

[27] T. Ichimura, K. Fujita, P. E. B. Quinay, L. Maddegedara, M. Hori, S. Tanaka, Y. Shizawa, H. Kobayashi, and K. Minami, “Implicit nonlinear wave simulation with 1.08t dof and 0.270t unstructured finite elements to enhance comprehensive earthquake simulation,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC ’15. New York, NY, USA: ACM, 2015, pp. 4:1–4:12.

[28] A. Rahimian, I. Lashuk, S. Veerapaneni, A. Chandramowlishwaran, D. Malhotra, L. Moon, R. Sampath, A. Shringarpure, J. Vetter, R. Vuduc, D. Zorin, and G. Biros, “Petascale direct numerical simulation of blood flow on 200k cores and heterogeneous architectures,” in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC ’10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 1–11.

[29] W. D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith, “High-performacne parallel implicit CFD,” *Parallel Computing*, vol. 27, pp. 337–362, 2001.

[30] O. B. Widlund, “The development of coarse spaces for domain decomposition algorithms,” in *Domain Decomposition methods in science and engineering XVIII*. Springer, 2009, pp. 241–248.

[31] C. Yang and X.-C. Cai, “Parallel multilevel methods for implicit solution of shallow water equations with nonsmooth topography on the cubed-sphere,” *J. Comput. Phys.*, vol. 230, no. 7, pp. 2523–2539, 2011.

[32] ——, “A scalable fully implicit compressible Euler solver for mesoscale nonhydrostatic simulation of atmospheric flows,” *SIAM Journal on Scientific Computing*, vol. 36, no. 5, pp. S23–S47, 2014.

[33] E. Chow and A. Patel, “Fine-grained parallel incomplete LU factorization,” *SIAM Journal on Scientific Computing*, vol. 37, no. 2, pp. C169–C193, 2015.

[34] H. Anzt, E. Chow, and J. Dongarra, “Iterative sparse triangular solves for preconditioning,” in *Euro-Par 2015: Parallel Processing*. Springer, 2015, pp. 650–661.

[35] X.-C. Cai and M. Sarkis, “A restricted additive Schwarz preconditioner for general sparse linear systems,” *SIAM J. Sci. Comput.*, vol. 21, pp. 792–797, 1999.

[36] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, “PETSc Users Manual,” Argonne National Laboratory, Tech. Rep. ANL-95/11 – Revision 3.4, July 2013.