Redesigning CAM-SE for Peta-Scale Climate Modeling Performance and Ultra-High Resolution on Sunway TaihuLight

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

Junfeng Liao
Tsinghua University, China
National Supercomputing Center in Wuxi, China
liaojf08@gmail.com

Nan Ding
Tsinghua University, China
National Supercomputing Center in Wuxi, China
dingnan0701@gmail.com

Xiaohui Duan
Shandong University, China
sunrise.duan@mail.sdu.edu.cn

Lin Gan
Tsinghua University, China
National Supercomputing Center in Wuxi, China
lingan@tsinghua.edu.cn

Yishuang Liang
Tsinghua University, China
National Supercomputing Center in Wuxi, China
liangys@mail.bnu.edu.cn

Xinliang Wang
Tsinghua University, China
National Supercomputing Center in Wuxi, China
clarencewxl@gmail.com

Jinzhe Yang
Imperial College London, UK
National Supercomputing Center in Wuxi, China
jinzheyang@gmail.com

Yan Zheng
National Research Center of Parallel Computer Engineering and Technology, China
zyzhengyanzy@263.net

Weiguo Liu
Shandong University, China
weiguo.liu@sdu.edu.cn

Lanning Wang
Beijing Normal University, China
National Supercomputing Center in Wuxi, China
wangln@bnu.edu.cn

Guangwen Yang
Tsinghua University, China
National Supercomputing Center in Wuxi, China
ygw@tsinghua.edu.cn

ABSTRACT

The Community Atmosphere Model (CAM) is ported, redesigned, and scaled to the full system of the Sunway TaihuLight, and provides peta-scale climate modeling performance. We refactored and optimized the complete code using OpenACC directives at the first stage. A more aggressive and finer-grained redesign is then applied on the CAM, to achieve finer memory control and usage, more efficient vectorization and compute and communication overlapping. We further improve the CAM performance of a 260-core Sunway processor to the range of 28 to 184 Intel CPU cores, and achieve a sustainable double-precision performance of 3.3 PFlops for a 750 m global simulation when using 10,075,000 cores. CAM on Sunway achieves the simulation speed of 3.4 and 21.5 simulation-year-per-day (SYPD) for global 25-km and 100-km resolution respectively; and enables us to perform, to our knowledge, the first simulation of the complete lifecycle of hurricane Katrina, and achieve close-to-observation simulation results for both track and intensity.

SC17, Denver, CO, USA

DOI: 10.1145/3126908.3126909

ACM Reference format:
Haohuan Fu, Junfeng Liao, Nan Ding, Xiaohui Duan, Lin Gan, Yishuang Liang, Xinliang Wang, Jinzhe Yang, Yan Zheng, Weiguo Liu, Lanning Wang, and Guangwen Yang. 2017. Redesigning CAM-SE for Peta-Scale Climate Modeling Performance and Ultra-High Resolution on Sunway TaihuLight. In *Proceedings of SC17, Denver, CO, USA, November 12–17, 2017,* 12 pages.
DOI: 10.1145/3126908.3126909

1 JUSTIFICATION FOR ACM GORDON BELL PRIZE

First refactoring and redesign of the entire Community Atmospheric Model (over half million lines of code) for Sunway TaihuLight, achieving 3.4 SYPD for 25km global atmospheric simulation and close-to-observation simulation of hurricane Katrina lifecycle; a sustainable double-precision performance of over 3.3 PFlops (750-m resolution) of the dynamical-core using 10,075,000 cores.

2 PERFORMANCE ATTRIBUTES

The following table shows the major performance numbers updated in the recent months, and some other highlights of this work.

3 OVERVIEW OF THE PROBLEM

People say, "a storm may arise from a clear sky". Predicting the weather and climate has been one of the most sophisticated problems in the world that relates to so many different factors and disciplines. While ancient scholars mostly built their prediction

Performance Attributes	Content
Sustainable performance	***3.3 PFlops using 10,075,000 cores*** *(improved from previous 1.6 PFlops)*
Simulation year per day (SYPD)	***3.4 SYPD*** *for ne120 (25-km horizontal resolution)* ***21.5 SYPD*** *for ne30 (100-km horizontal resolution)*
Refactoring efforts	*CAM original total is 754,129 LOC* *We modified* ***152,336 LOC*** *We further added* ***57,709 LOC***
Category of achievement	*Time-to-solution, scalability, and peak performance*
Extreme event simulation	*Simulate the lifecycle of hurricane Katrina with close-to-observation accuracy*
Type of method	*Explicit*
Results reported on basis of	*Whole application with I/O*
Precision reported	*Double precision*
System scale	*Measured on full-scale system*
Measurement	*Timers*

systems based on observations, since the 1950s, the numerical simulation approach has taken off, and demonstrated its effectiveness in making predictions, and providing scientific support for various issues ranging from extreme weather and climate disaster mitigation [1] to everyday life planning.

During this long process that originates from the first numerical program that John von Neumann and Charney wrote on ENIAC (often considered as the first computer in history) [2] to today, we see the numerical simulation tool has grown from a simple solver of barotropic equations to one of the world's most complicated software projects (such as Community Atmospheric Model [3], Community Earth System Model [4], etc.) that possess millions lines of code, and involve hundreds or even thousands of contributors from various research institutions and organizations.

Along with the development process of such weather and climate system model, the demands for computing power are also keeping on increasing. With more and more scientific understanding converted into software modules, more and more grids in the model to push the resolution from a few hundred kilometers to a few kilometers, and the increased accuracy requirement that increases the ensemble sizes, the climate models nowadays can easily occupy all available resources of high-end computing facilities.

While the demands from the science side continue to grow, the HPC architecture roadmap somehow has diverted. Most of the leading-edge supercomputers coming out in recent years are equipped with heterogeneous architectures in the form of many-core chips (Tianhe-2 [5], Titan [6], etc.). One extreme example is the Sunway TaihuLight announced in 2016, which provides a peak performance of 125 PFlops with over 10 million cores [7].

This architectural change brings tough challenges for programming. Both the computing architecture and the memory hierarchy are changed radically. For example, each Sunway processor is providing the parallelism at the level of 260 cores in one chip, and each of the cores still has wide vector units to form another level of parallelism. Inside the computing cores, the cache is replaced by an user-controlled scratchpad memory for the purpose of meeting the hardware and power constraints at such a level of computing density. These radical changes require a total different design and programming philosophy when conducting a numerical problem.

Therefore, it is becoming almost impossible for well established numerical codes, such as the millions lines of code in the climate domain, to gain performance benefits, if any, from the newly built supercomputers. The heavy code legacy is one impediment, but the broad spectrum of sciences related to the hundreds of physics, chemistry, or even economic schemes woven into the model have made it even more difficult to achieve sustainable development of both science and computation in climate system models. As a result, most of the existing work still mainly targets at some parts of the models, such as the dynamical core [8–10] or certain specific physics schemes [11]. We only see a few work [12] that manages to port the complete models, which are still far from the level of complexity of global atmospheric models.

To explore the possible solutions for such a difficult situation, and more specifically, to achieve an efficient utilization of the Sunway TaihuLight for climate-kind applications, in our work, we pick CAM [3] as our target application, and perform an extensive refactor or even redesign process for both the dynamical core and the physics schemes in such an extremely complicated software program. With over hundreds of modules in such a complex model, we have at least 20 to 30 kernels that contribute a meaningful portion (although usually only 2% to 5%) to the total run time. Therefore, to gain any meaningful performance improvement, we have to port and optimize the entire model, which brings challenges at a completely different level from partial kernel optimizations.

As the first step, we take OpenACC-based refactoring as the major approach, and apply source-to-source translator tools to exploit the most suitable parallelism for the CPE cluster of SW26010 that powers Sunway TaihuLight, and to fit the intermediate variable into the limited on-chip fast buffer. Combining the OpenACC-refactored code with the projected performance upper bound based on the memory capacities (assuming bandwidth as the major constraint), we then derive a more aggressive fine-grained optimization workflow that maps the original OpenACC directive description of parallelism into a more flexible Athread code, to enable finer memory control and more efficient vectorization. Among the 754,129 lines of code (LOC) from the original CAM model, around 152,336 LOC have been modified, and 57,709 LOC are newly added.

As far as we know, this would be the first reported effort that migrates a scientific application at such a level of complexity to a completely different hardware architecture. Our effort results in a fully ported CAM model on Sunway TaihuLight that supports 25-km resolution using 1,872,000 cores, and provides a simulation speed of 3.4 simulation years per day (SYPD). The further fine-grained optimization for the CAM-SE dynamic core part manages to scale to 10,075,000 cores for a 750-m global simulation, and provides an unprecedented performance of 3.3 PFlops in double-precision.

Besides the significant performance improvements, CAM on Sunway also enables us to perform, to our knowledge, the first simulation of complete lifecycle of hurricane Katrina, and achieve close-to-observation simulation results for both track and intensity.

With the significantly improved performance of CAM-SE on Sunway TaihuLight as the major contribution of our efforts, we also take this opportunity to investigate the potential programming

challenges and solutions that a lot of other large-scale scientific applications would face during the transition to the Exascale era, which is just another couple of years away from now.

4 CURRENT STATE OF THE ART

4.1 Atmospheric Modeling

Ever since the first numerical atmospheric program running on ENIAC (the first electronic digital computer) [13], atmospheric research has made a rapid progress with the technical revolution of computers, and is no doubt among the major consumers of the world's most powerful supercomputers. Based on the powerful supercomputers, many pioneering works have been made by different research groups. Satoh *et. al* targeted at the Nonhydrostatic Iscosahedral Atmospheric Model (NICAM) and achieved 3.5-km and 7-km global simulations [14] based on the Earth Simulator, and 870-m global resolution [15] on K computer. In US, the CAM-SE dynamics core supports up to 12.5-km resolution with a simulation speed of around 4.6 SYPD across over 172,800 CPU cores on Jaguar [3], while the Weather Research and Forecasting (WRF) model provides a single-precision performance of 285TFlops on 437,760 cores of Blue Waters [16].

In the recent years, as the heterogenous architecture that involves many-core accelerators is becoming the mainstream for modern HPC systems, we see a lot of efforts that intend to overcome the tough climate modeling difficulties by using those new architectures. Early researches are mostly focusing on the standalone physics scheme, and have achieved good speedups over their CPU counterparts. For example, in the study of using GPUs to accelerate the famous Weather Research and Forecast model (WRF), some efforts have been made in porting different modules, such as the chemical kinetics modules [17] and the microphysics scheme [18], and managed to obtain speedups of 8.5× and 70×, respectively. The work in [9] focused on the porting of the shortwave radiation parameterization of CAM and achieved a speedup of 14×; while the work in [19] ported the microphysics module of Global/Regional Assimilation and Prediction System (GRAPES) with the speedup of 140×.

Compared with the physical schemes that usually only contain high density arithmetic kernels, the dynamical core parts generally require interactions and communications across different grids, and are thereby more difficult to deal with. In recent decades, we also see quite a lot of works on accelerating the dynamical cores on GPU devices, such as the efforts on NIM [20], GRAPES [11], CAM-SE [21], HIRLAM [22], and NICAM [23]. The achievable speedups for those works are around 3 to 10 times when compared against the parallel CPU solutions.

Thanks to the performance boosts delivered by the 125 PFlops Sunway TaihuLight Supercomputer, some successes have been made towards fully exploiting the huge computing capacity, such as the work by Yang *et. al,* [10] that scales an implicit solver for the 3D Euler equations to the 10 million cores of Sunway TaihuLight, and achieves a sustained performance of 8 PFlops, and the work by Qiao *et. al,* [24] that simulates the realistic surface wave and achieves 45.43 PFlops in ultra-high resolution of (1/100) degree.

Until now, we only see a few models that is completely ported onto the GPU platform. One example is the GPU-based acceleration of ASUCA, a next-generation high resolution mesoscale atmospheric model developed by the Japan Meteorological Agency (JMA) [25]. The speedup of 80-fold is obtained when compared against a single CPU core, with an excellent scalability for up to a few thousands of nodes. Another example is the complete porting of the Princeton Ocean Model (POM) onto GPU devices [12], which performs a manual porting and optimization onto 4 GPUs, and achieves an equivalent performance to 408 CPU cores.

The COSMO regional weather model used by MeteoSwiss has been rewritten (dynamical core part) and refactored (physics part) through domain and data structure specific tools (STELLA[26], OpenACC), achieving the considerable performance acceleration[27].

Our work differs from the above efforts from mainly two folds: one is the target application, the Community Atmospheric Model, which presents the complexity level of over half million lines of code, and hundreds of major functions or kernels, in contrast to the existing works that mostly focus on kernels or solvers; the other one is the target platform of Sunway TaihuLight, which puts the further requirement of scaling the redesigned code to the parallel level of over 10 million cores (with the totally different architecture of SW26010 being another challenging factor).

4.2 Exploring Solutions for Exascale

With the planned Exascale computing systems coming in another 3 to 5 years, the applications for Exascale have clearly become a focus for HPC-related research in different countries.

Hardware-software co-design used to be considered as a potential solution that resolves the current mis-match between the existing software and the evolving hardware. However, if you do look into the two elements of the problem, they do not yet show a likely trend towards converging. One side is the software in the form of millions lines of legacy code that accumulates the scientists' previous discoveries, and the other side is the future Exascale machines that would most likely adopt completely different architectures to satisfy the performance target, and to resolve the power and heat constraints. In other words, both sides still have their own challenges to resolve, and are unlikely to converge. Meanwhile, considering the broad spectrum of scientific applications that a supercomputer generally needs to support, and the lack of commercial elements in their nature, it is not quite possible to see fully customized large scale supercomputers in supercomputing centers, such as the series of Anton machine [28, 29] (supported by an financial investment institute), or the deep learning domain [30] (backed by hundreds of Internet companies).

Therefore, considering all these different factors, software redesign by itself becomes the only feasible way to scale the science with the emerging HPC technologies. One example is the recently announced project called NWChemEX Exascale Computing Project [31] that aims to develop a new version of the widely used computational chemistry code, NWChem [32]. Led by the Pacific Northwest National Laboratory (PNNL), the large amount of original NWChem code will be completely redesigned and re-implemented in C++, so as to be better optimized on the upcoming Exascale supercomputers. As such, there are at least 15 representative applications, such as the ALPINE project and the ExaSky project, which would be redesigned in algorithms, scalability and overall performance to meet the Exascale computing supercomputers [33].

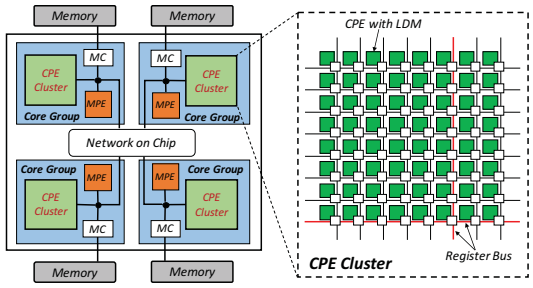

Figure 1: The architecture of the many-core SW26010 CPU

While these projects are still in the initialization stage, in this paper, we report our concrete efforts for CAM-SE on Sunway TaihuLight. We should argue that the challenge in our case is even tougher, as the Sunway processor is using a completely new hardware architecture with the software ecosystem just being built up. Our target is not only to make it work on Sunway TaihuLight, but also to scale the code to the massive number of cores in the system and to achieve peta-scale computing performance.

5 SUNWAY TAIHULIGHT SUPERCOMPUTER

5.1 System Overview

Announced in June 2016 as the fastest supercomputer in the world, the Sunway TaihuLight has kept the record ever since. The peak performance is over 125 PFlops, and the Linpack performance is over 93 Plops. The supercomputer is equipped with China's homegrown many-core processor SW26010 which has 260 cores.

The machine takes a two-level approach to build the network. Inside a supernode with 256 processors, all the processors are fully connected through a customized network board. Above the supernode, the central network switches process the communication packets. Through such a two-level network topology, all the 40,960 nodes (10,649,600 cores) are connected efficiently.

Not only the performance outperforms all the existing supercomputers, the overall power efficiency of TaihuLight is also outstanding, providing a power efficiency of 6.06 GFlops / watt.

5.2 The SW26010 Many-Core Processor

Figure 1 provides the general architecture of the many-core heterogeneous processor. The 260 cores on each SW26010 are grouped into 4 core-groups (CGs), with 65 cores in each CG. Within each CG, there are one management processing element (MPE), one cluster of 8×8 computing processing elements (CPE), and one memory controller (MC).

The MPE is a complete 64-bit RISC core that supports both user and system modes. While also capable of performing computations, the MPE is generally used for handling management and communication functions.

The CPE, on the contrary, is a 64-bit RISC core that only supports the user mode, with the design goal to maximise the aggregated computing power and to minimize the complexity of the micro-architecture. Organized as an 8×8 mesh, the cluster of CPEs supports low-latency register communication among the CPEs, to achieve low-cost data sharing schemes within the cluster.

As for the memory, each MPE has a 32KB L1 instruction cache, a 32KB L1 data cache, and a 256KB L2 cache for both instruction and data. Each CPE has a 16KB L1 instruction cache, and a 64KB SPM, (also called the Local Data Memory (LDM)), which serves the same function as the L1 cache, but in a user-controlled way.

The MPE and the CPEs within the same CG share the same memory which is controlled by the MC. The 4 CGs are connected using network on chip (NoC). Each SW26010 processor provides a peak performance of over 3 Tflops, a power efficiency of 10 GFlops/W, and connects to 32 GB memory with a bandwidth of 132 GB/s. While providing a high computing performance and high power efficiency, the memory size and the memory bandwidth is relatively limited as a compromise.

5.3 The Programming Environment

The parallel programming model of TaihuLight can be described as "MPI + X", where X denotes OpenACC or Athread. In most cases, each CG corresponds to one MPI process. For the MPE and 64 CPEs within each CG, we use Sunway OpenACC or Athread to exploit in-CG parallelism.

The Sunway OpenACC compilation tool supports OpenACC 2.0 syntax and targets the CPE clusters over shared memory. This customised version of OpenACC supports the management of parallel tasks, the abstraction of heterogeneous code, and the different ways of data transfers. Combined with the specific features of the SW26010 architecture, a number of syntax extensions has been made based on the original OpenACC 2.0 standard, such as finer control of multi-dimensional array buffering, multi-dimensional array transpose, etc.

The second option, Athread, is a lightweight threading library specifically designed for the SW26010 processor and its predecessors, which is also the underlying implementation for OpenACC. Similar to the POSIX threads, Athread allows a program to control multiple different flows of work that overlap in time.

While Sunway OpenACC provides an easy-to-use interface for porting programs onto the Sunway architectures, the constraint is that a number of performance-related programming interfaces (vectorization, DMA operation control, etc.) is not yet supported. In contrast, the Athread interface requires much more programming efforts, but also provides the option to fully exploit the performance potential of the Sunway architecture through fine-grained controls.

6 COMMUNITY ATMOSPHERE MODEL (CAM)

Climate models are indispensable and ubiquitous tools for simulating past, present and future climates, understanding the Earth's global climate system and predicting the effects of climate change. In order to support studying the geophysical fluid dynamics of global atmosphere and ocean at the beginning, climate models are integrating a variety of fluid-dynamical, physical, chemical and biological procedures across different time scales and spatial scales.

CAM is the most important and computationally consuming component in CESM, and it has long been a killer application on supercomputers [34]. It is also one of the state-of-the-art and widely used coupled system models, which has been developed and maintained for over 30 years by NCAR with a large code base for more than one million five hundred thousand LOCs. In the trend of

higher resolution and more complex physics parameterizations, CAM is now consuming more computing power than ever. The performance scalability of CAM has been a vital issue for achieving acceptable performance on current or future large-scale computing systems [35].

Generally, CAM is characterized by two phases: the dynamics and the physics. These two phases are executed in turn during each simulation time-step. For the physics, it approximates subgrid phenomena such as clouds, precipitation processes, long/shortwave radiation, and turbulent mixing. For the dynamics, it expresses the evolutionary equations for the atmospheric flow. The dynamical core of CAM-SE uses a second-order accurate N-stage Runge-Kutta method which is implemented as a combination of the RK2 and Leapfrog schemes [3].

We use CAM-SE which includes a spectral element (SE) dynamical core option from NCAR's High-Order Method Modeling Environment (HOMME) [3].

7 INNOVATIONS REALIZED

7.1 Overview

This work manages to migrate the complete CAM-SE (both the dynamical core and the physical scheme parts) to the completely different hardware architecture of Sunway TaihuLight, and manages to scale the performance of the model to 10 million cores.

Our major contributions are as follows:

- For both the dynamical core and the physics parts, we take OpenACC-based refactoring as the major approach, and apply source-to-source translation tools to exploit the most suitable parallelism for the CPE cluster, and to fit the intermediate variable into the limited on-chip fast buffer.
- Based on the OpenACC-refactored code of CAM-SE, we further derive a fine-grained optimization workflow using Athread, to achieve finer memory control and more efficient vectorization.
- For the modules with heavy data dependency and inadequate parallelism, we apply the register communication feature of the Sunway architecture to expose more parallelism for the increased number of cores within each processor.
- For the communication between neighbour elements, we redesign the boundary exchange part to add the feature of computation and communication overlapping, and to avoid unnecessary memory copies, which nearly doubles the performance of CAM-SE in large-scale runs.

For the first step of OpenACC-based refactoring, when comparing the original ported version using only MPEs and the refactored version using both the MPE and CPE clusters, we achieve up to 22x speedup for the compute-intensive kernels. For the 25-km resolution CAM global model, we manage to scale to 28,800 MPEs, and 1,872,000 CPEs, and achieve a simulation speed of 3.4 simulation years per day (SYPD).

Moreover, the fine-grained Athread approach combined with the register communication schemes can further improve the major kernels in CAM-SE by another 10 to 15 times. The completely redesign version of CAM-SE can scale to the full size of Sunway TaihuLight, providing a performance of 3.3 PFlops at the 750 m resolution global simulation.

7.2 OpenACC-based Refactoring of CAM-SE

We use the Sunway OpenACC compiler (a customized version that expands from the OpenACC 2.0 standard) as the major tool to achieve a suitable mapping of CAM onto the new Sunway heterogeneous many-core processors. Due to large code base developed over the last few decades, and the general demand from climate scientists to maintain a same source, we try to minimize the manual refactoring efforts (to only the dynamical core part), and mainly rely on the source-to-source translation tools to achieve an automated and efficient porting.

Besides, compared with GPU and other many-core accelerators, both the on-chip fast buffer and the available memory bandwidth of the Sunway processor are relatively limited (detailed in Section 5), which makes our porting significantly more challenging. A large part of our tools and optimization strategies would focus on minimizing the memory footprints.

Through a careful refactor of the SE dynamical core, we manage to combine the distributed loops in the major computational functions into aggregated multi-level loops, and expose a suitable level of both parallelism and variable storage space for the CPE cluster architecture.

For the physics parts, which includes numerous modules with different code styles by different scientists, we design a loop transformation tool to identify and expose the most suitable level of loop body for the parallelization on the CPE cluster. In addition, we also design a memory footprint analysis and reduction tool, and a number of customized Sunway OpenACC features, to fit the frequently-accessed variables into the local fast buffer of the CPE.

7.3 Athread-based Fine-grained Redesign

While the OpenACC-based refactoring manages to migrate all the major modules of CAM onto the SW26010 architecture, the limitations of the OpenACC approach place performance constraints on quite a few major kernels. As shown in Table 1, for the most time consuming kernel `euler_step`, the OpenACC version is only 1.5x faster than the Intel single-core performance. For the kernel `compute_and_apply_rhs`, with data dependency, the OpenACC version is even 6x slower than Intel single-core version.

Such results clearly demonstrate a few major constraints that come with the OpenACC approach:

- With a focus on minimizing the code modificatoins needed, the OpenACC approach in many cases removes the option to make algorithmic adjustments or to expose certain low-level interfaces for finer control of either computing or memory operations.
- Threading overhead becomes a huge issue for programs with a complicated set of modules and no clear hot spots (e.g. CAM).
- Using directives only, apparently, makes it impossible for us to perform aggressive algorithmic or data structure related optimizations, which are in many cases the only way to make utilization of Sunway's different architecture.

Meanwhile, looking at the CAM software itself, there is also a number of elements that call for redesign rather than simple porting of the code, such as design patterns that work perfectly for previous architectures but could bring disasters for many-core accelerators like Sunway:

Table 1: Key kernels of dynamics and their timing cost (s) of 6,144 processes over different computing platforms..

Key kernels	Description	Intel	MPE	Acc
compute_and_apply_rhs	compute the RHS(right hand side), accumulate into velocity and apply DSS	12.69	92.13	75.11
euler_step	construct strong stability preserving (SSP) second order Runge-Kutta method (RK SSP method)	15.88	175.73	10.18
vertical_remap	compute the vertical flux needed to get back to reference η-coordinate levels	11.38	39.99	16.17
hypervis_dp1	horizontal regular viscosity operator applied to the momentum and temperature equations	4.95	12.71	3.13
hypervis_dp2	horizontal hyper viscosity operator applied to the momentum and temperature equations	3.81	9.05	1.32
biharmonic_dp3d	compute weak biharmonic operator	9.35	36.18	4.43

- Difficulty in data locality: the original CAM code adopts the same data layout for both vertical and horizontal computations, which would work for the traditional cache hierarchy, but do not fit to the 64-KB LDM buffer and its philosophy of requiring users to manage their own cache.
- Computation patterns that are unfriendly to vectorization: with a lot of previous optimizations focus on minimizing the total number of operations, we lose regularity in data and instruction layout, leading to inefficient vectorization.
- Absence of overlapping schemes to hide communication in computation partially or completely. HOMME mainly uses third-order RK methods to solve dynamics equations, leading to 3 subcycles edge packing/unpacking and boundary exchange. Such a pattern could incur serious communication overhead in large-scale runs.

Based on the above considerations, in our work, we decide to take a more aggressive fine-grained redesign approach, and to rewrite CAM. By doing so, we manage to take advantages of Sunway's architecture rather than just making use of the parallelism itself. By taking careful algorithmic adjustment into considerations and adopting the Athread programming interface, with the OpenACC-refactored version as the starting point, we then perform two steps to achieve an optimized Athread version.

The first step is to rewrite the OpenACC Fortran code to the Athread C version. A typical example is shown in Algorithm 1 and 2. Algorithm 1 is the original parallelization method using OpenACC, while Algorithm 2 illustrates our Athread rewrite version of Algorithm 1. The customized OpenACC compiler only supports single collapse for multiple levels of loops, and we cannot insert code between two loops once it is collapsed. Therefore, in Algorithm 1, the *copyin* instruction can only take place in the cycle of q. As a result, every time when the q loop is executed, it reads all related arrays. Nevertheless, even if the next loop reuses the same array, it reads the data again, which increases the bandwidth usage so significantly that it becomes the inevitable bottleneck.

On the contrary, the Athread provides flexible customization on memory reuse or other behaviours. If the next loop requires the same piece of data as the current one, user defines the memory to keep the data for the next loop. In our case, by applying such scheme, total data transfer size has been decreased to 10% compared with the OpenACC solution, and it is no longer a bottleneck.

The second step is to perform manual vectorizations for certain code regions. We can specifically declare vectorized data types and write vectorized arithmetic operations, so as to improve the vectorization efficiency of the corresponding code regions.

Algorithm 1 Euler Step Acc

```
Require:
  elements : elements
  nets, nete, col, row : the row and column id of that core

  !$ACC parallel loop collapse(2)
  for ie ← nets + col to nete do
    for q ← 1 to num_tracers do
      t ← s + 32
      !$ACC copyin(elements(ie).derived_dp(s .. t)
      ...//And other non q related arrays
      !$ACC copyin(elements(ie).qdp(q, s .. t)
      for s ← 1 to vlayers, step 32 do
        for k ← s to t do
          ..//computation
```

Algorithm 2 Euler Step Athread on CPEs

```
Require:
  elements : elements
  nets, nete, col, row : the row and column id of that core

  for ie ← nets + col to nete, step 8 do
    s ← row * vlayer_per_core
    t ← (row + 1) * vlayer_per_core
    DMA-get(elements(ie).derived_dp(s..t))
    ..//Same for other non q related arrays
    for q ← 1 to num_tracers do
      DMA-get(elements(ie).qdp(q, s..t)))
      for k =← s to t do
        ..//vectorized computation
      DMA-put(elements(ie).qdp(q, s..t)))
```

7.4 Register Communication-based Parallelization Scheme

The SW26010 has no coherent cache to exchange data among the CPEs, instead, it supports the register communication operations. From Figure 1, by using register communications, data can be directly exchanged between the LDMs of the two CPEs that belongs to the same row or the same column within tens of cycles.

Such data exchange mechanism among different CPEs provides an important option for data reuse among different CPE threads, and, in many cases, provides more parallelism to fit to the increased number of cores.

To make an efficient utilization of the register communication feature, we apply a column-wise decomposition to assign different portions to different CPEs. In the 3D mesh of CAM-SE, each column is called an element (Figure 2), which is further divided into a 4 by 4 grid at each level, and multiple layers (128 in our case) to simulate the atmospheric dynamics. The resulting configuration of CPEs is also shown in Figure 2.

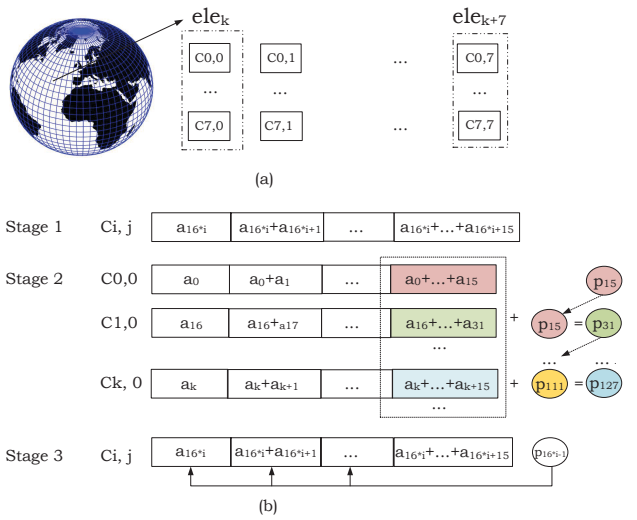

Figure 2: (a) The partition of 3D mesh of CAM. In our Athread parallelization, in order to support convenient register communication among CPEs, we divide the 128 layers into 8 groups. Each group of 16 levels is processed by one CPE. (b) The parallel scan and sum operation on CG for 3D mesh of CAM. CPE $C_{i,j}$ corresponds to the computation of layer range $[16 \times i, 16 \times i + 15]$. p_k is the geopotential height of the kth layer, a_k is the pressure differences between two neighbouring layers k and $k-1$.

In a lot of cases, we need to scan the element and compute the summation results along the column axis. We take the scan and summation case in the *compute_and_apply_rhs* kernel as an example of how the data dependency is minimized using our parallelization scheme and the register communcation feature. Since the layers are equally partitioned, each CPE processes a group of 16 layers. p_k is originally computed sequentially along the column: $p_k = p_{k-1} + a_k$, and p_0 is the initial geopotential height. If we only calculates the accumulative pressure difference for current layers within the CPE, it is parallel friendly. Figure 2 illustrates our three-stage accumulation algorithm.

Stage 1, Local Accumulation: Each CPE computes the local accumulation sum of its own layers.

Stage 2, Partial Sum Exchange: Each CPE $C_{i,j}$ which is not in the first row waits for the $p_{16\times i-1}$ from $C_{i-1,j}$ by blocking register reading, once the sum is get, it computes $p_{16\times i-1} + \sum_{j=0}^{15} a_{i+j}$ and sends it to $C_{i+1,j}$, if $C_{i,j}$ is not in the last row.

Stage 3, Global Accumulation: Computes all the atmospheric pressures within each CPE.

7.5 Shuffle and Register Communication-based Array Transposition

Another major difficulty for achieving efficiency for the kernels on the Sunway architecture is the switch of array axis during the different loops in a same function. As we do not have a large-size cache, accessing the data elements in a transposed manner can bring significant performance penalties in such cases.

To achieve an efficient solution for array transposition in Athread, we propose a shuffle and register communication-based approach, which achieves fast transposition on small matrices using shuffle instructions of registers, and then uses register communication scheme to achieve larger matrix transposition.

Fig. 3 provides an example for the transposition of a 4 by 4 matrix using progressive register level shuffle operations.

The instruction is *Shuffle(a,b,mask)*, as is shown in the top left of the figure. a and b are two 256-bit registers which contain 4 double-precision floating numbers. The *mask* provides the information of which 2 numbers from each register are being taken to the new register, and the first two numbers come from a, while the other two numbers are from b. In our example, it's position 0 and 2 of A (corresponding value $a0$ and $a2$), and position 0 and 1 of b (corresponding value $b0$ and $b1$).

The bottom left part of the figure shows a 4 by 4 matrix transposition by using 8 shuffle operations, colored arrows end in a same row of that matrix denotes one shuffle operation.

The right part shows how we schedule larger matrix transposition by using register communication, each C_{ij} denotes a 4 by 4 sub matrix in LDM. We transpose matrix among n CPEs by $n-1$ phases, 1 to $n-1$. In Phase k, CPE i exchange a sub matrix with CPE $i \oplus k$, thus each phase contains a set collision free exchange among the all these CPEs, the arcs connecting two CPEs denotes a exchange between the two CPEs.

7.6 A Redesigned *bndry_exchangev*

The *bndry_exchangev* subroutine is the major part of communication within CAM. The subroutine comes with a good abstraction design to hide the complexity of communication from normal users, and has achieved good scalability to a large scale. However, the option to computation and communication overlapping is not considered. While in a normal setup, the communication time is not the biggest part in the total run time, for the cases with millions of cores, the communication time contributes to around 23% of the total run time of *prim_run* (the dynamic part).

Therefore, in our redesigned *bndry_exchangev* subroutine, we add the feature of computation and communication overlapping. We divide the elements of one process to two parts, the inner part and the boundary part. The boundary part consists of elements that are involved in *bndry_exchangev*. We compute the boundary first, and then start the asynchronous MPI communcation on the MPE with an *MPI_wait* in the end. We apply the same strategy to all the three halo exchanges in the Euler step, and reduces the run time of HOMME by 23% in the best cases.

Moreover, the abstraction design of the communication part also leads to unnecessary memory copies in certain cases. In the original design of HOMME, the exchanged data items from both MPI messages and intra-node memory copies are all copied into the same pack and unpack memory buffer, to achieve a unified interface for all related halo-exchange operations. However, such a design also sacrifices performance by performing redundant memory copies, as data items would always aggregate to the pack and unpack buffers, and then forwarded to the corresponding elements for the computation afterwards. In our case, we keep most of the unified interfaces for the convenient programming for future code development. For

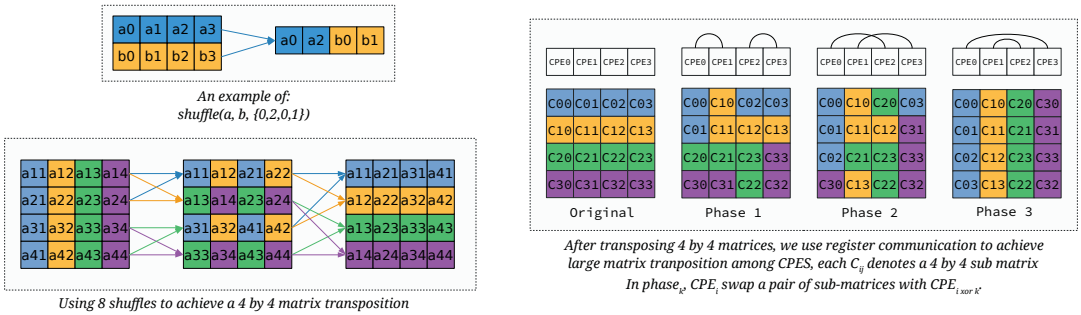

Figure 3: Two levels of our transposing scheme among CPEs, and it contains intra CPE and inter CPE transposing.

the unpack procedure, instead of the original data flow from receive buffer to pack buffer, and then to the corresponding elements on the boundary that needs the data, we identify the boundary-related data in the very beginning, and fetch the data directly from receive buffer to the corresponding elements. Such a optimization to the memory copy would again reduce the run time of the dynamical core of CAM by another 30%.

8 PERFORMANCE RESULTS

8.1 Metrics and Experiment Configurations

8.1.1 Flops Measurement. We analyze and collect the total number of double-precision (Flops) arithmetic operations using three different ways (1) manually counting all double-precision arithmetic instructions in the assembly code, or the original code if necessary; (2) using hardware performance monitor of the Sunway TaihuLight supercomputer, PERF, to collect the retired double-precision arithmetic instructions on the CPE cluster; (3) running the same MPE-only version of the very original code on an Intel platform, and using Performance API (PAPI) for total amount of double-precision arithmetic operations.

The result from the third method is higher, while the other two methods are almost identical with each other. Platform difference is probably the main explanation to the difference, therefore, we apply the second method (PERF) for counting the double-precision arithmetic operations.

8.1.2 Simulation Year per Day (SYPD). SYPD refers to Simulated Year per Day, which is commonly used for measuring simulation speed. In our experiments, we use the simulation time for every simulation day t_D as the basic measurement unit. By analyzing t_D of several runs with simulated period over years, we find that t_D varies with negligible differences. Therefore, we compute t_D as the mean of all the measurements of the simulations, and compute the value of SYPD based on t_D.

8.1.3 Experiment Configuration. For the cubed-sphere mesh used in CAM-SE, the computational domain consists of six patches, each of which contains different amount of mesh cells. For experiments in this work, we choose six different dimension sizes, as shown in Table 2.

In the CAM-SE, the resolution is defined as the number of spectral elements *ne* along the edge of each cube face. For example, $ne256 = 256 \times 256 \times 6$ represents the configuration of 256 spectral elements along the edge of each cube face (6 cube faces in total), which corresponds to the 12.5 km horizontal resolution. We conduct one-week simulation for each case and use the average time-to-solution of 3 runs as the reported performance.

Table 2: Meshsize Configuration of Different Problems

problem size	horizontal	vertical	# elements
ne64	$64 \times 64 \times 6$	128	24,576
ne256	$256 \times 256 \times 6$	128	393,216
ne512	$512 \times 512 \times 6$	128	1,572,864
ne1024	$1024 \times 1024 \times 6$	128	6,291,456
ne2048	$2048 \times 2048 \times 6$	128	25,165,824
ne4096	$4096 \times 4096 \times 6$	128	100,663,296

8.2 Simulation Validation

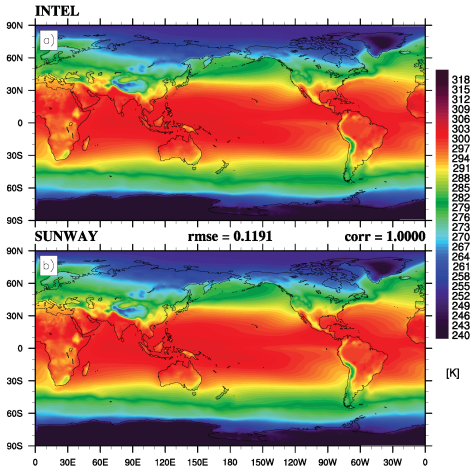

Figure 4: 30-year climatological atmospheric surface temperature simulated by (a) CESM on Intel (control run) and (b) CESM on Sunway TaihuLight (test run).

Our validation experiment uses the CESM1.2 F_PRESTENT_DAY compset with activated CAM, which applies the CAM-SE dynamical core and CAM5 physics suite, with the horizontal resolution NE30 (48,602 grid points) and 30 vertical levels. The land component uses activated CLM4. The ocean and sea-ice components use climatological sea surface temperature and sea ice data.

Figure 4 shows the 30-year climatological atmospheric surface temperature, simulated on Intel cluster (control run) and on Sunway TaihuLight (test run), respectively. We observe almost identical patterns for the simulations on these two completely different hardware architectures.

8.3 Performance Speedups

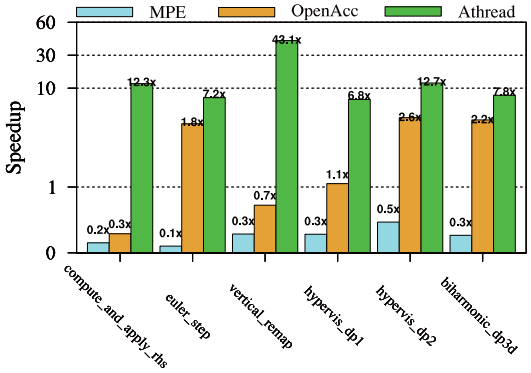

Figure 5: Speedups of different kernels based on 6144 MPI processes for Intel Xeon E5 2680 V3 and SW26010.

Figure 5 shows the speedup of different kernels (as shown in Table 1) for using 6,144 MPI processes, based on Intel CPUs (Xeon E5 2680 V3) and Sunway processors respectively. The performance speedups are sitting on top of each bar, with the Intel CPU performance being the basic reference. For each kernel, we demonstrate the results using one Intel CPU process, using one MPE, using both the MPE and the 64 CPEs with OpenACC directives, and the Athread version that further applies all the optimizations.

The figure shows that compared with the performance using one Intel process, the performance of using one MPE is around 2-10 times slower. Applying the OpenACC directives would improve the performance by 3 to 22 times, only scaling the performance of 64 CPEs to the range close to a single Intel core.

Only through Athread redesigns, we can expose all the performance-sensitive codelines, and achieve significant performance improvements through fine-grained description of the compute and memory access operations. Even compared with the OpenACC version, the Athread optimization can further improve the performance by up to 50×. Compared with the a single Intel core, the performance of 64 CPEs is also multiplied by another 7x to 46x. These results demonstrate that, for the migration to completely different architecture of the current 100-Pflops and future Exascale supercomputers, redesign at a fine-grained level is a meaningful step to go.

Figure 6 shows the performance improvements for the entire CAM model, described in SYPD. The left panel shows the results of ne30 with three different versions. As a complex model that

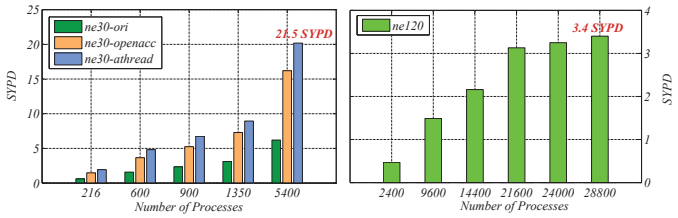

Figure 6: The performance improvements for the entire CAM model in ne30 and ne120. *ori* refers to the original version based on MPE, *openacc* refers to the usage of OpenACC directive, and *athread* refers to the further usage of Athread.

involves kernels accelerated as well as parts that are inherently serial, the speedup is not as good as in standalone kernels. However, with the number of processes increasing from 216 to 5400, we can still observe reasonable performance benefits from OpenACC refactoring and Athread redesign. For different process numbers, OpenACC can generally bring 1.4 to 1.5 times performance benefits, while Athread can further improve by another factor of 1.1 to 1.4 times. The fastest simulation speed achieved for ne30 (100-km resolution) is 21.5 SYPD when using 5400 processes. In the right panel of Figure 6, we show the performance results of ne120 (the OpenACC version). 3.4 SYPD is observed at 28800 processes.

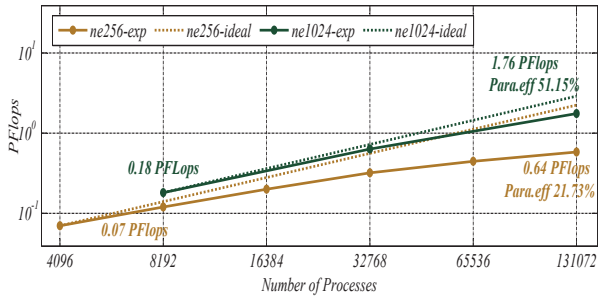

Figure 7: Strong scalability simulation using 'ne256' and 'ne1024' configurations of HOMME. *exp* refers to the measured result, while *ideal* refers to the theoretical scaling.

8.4 Scaling Results for HOMME

Figure 7 shows the results of strong scalability for HOMME model based on the Sunway TaihuLight supercomputer. We perform the experiments for two different problem sizes, i.e., *ne*256 (total number of elements is 393,216) and *ne*1024 (total number of elements is 6,291,456). The number of Sunway processes (or CGs) increases from 4,096 all the way to 131,072 for *ne*256 (the number of cores ranging from 266,240 to 8,519,680). The sustainable performance is increased from 0.07 PFLops to 0.64 PFlops, with the parallel efficiency of over 21.7% at 131,072 processes. For *ne*1024, due to the surge of total elements and memory limitation, the number of processes start from 8,192 (532,480 cores). We can observe that the sustainable performance is increased from 0.18 to 1.76, with the parallel efficiency of around 51%. The drop of efficiency from

*ne*1024 to *ne*256 is mainly due to the decreased number of elements, as well as the decreased portion of computing in the total time.

Figure 8 shows four different groups of experiments for the weak scaling simulation. The number of elements per process is fixed to be 48, 192, 650, and 768 for the four groups respectively, and the number of processes is gradually increased to the full machine.

In Figure 8, the four experiments (element size of 48, 192, and 650, 768 in each process) demonstrate very good scaling efficiencies. For the element size of 48, 192, 768, the number of processes are increased from 512 all the way to 131,072 (corresponding to 33,280 to 8,519,680 cores), and the parallel efficiencies at the largest run are 88.3%, 92.3% and 92.2%, respectively. Between different problem sizes, the increase of the elements in each process generally leads to less overhead caused by the communication, and thus obtains better parallel efficiency. For the problem size of 650, however, we are able to further extend the largest number of processes to be 155000, which corresponds to 10,075,000. The parallel efficiency at this poit is 98.5%, and the sustainable performance is 3.3 PFlops.

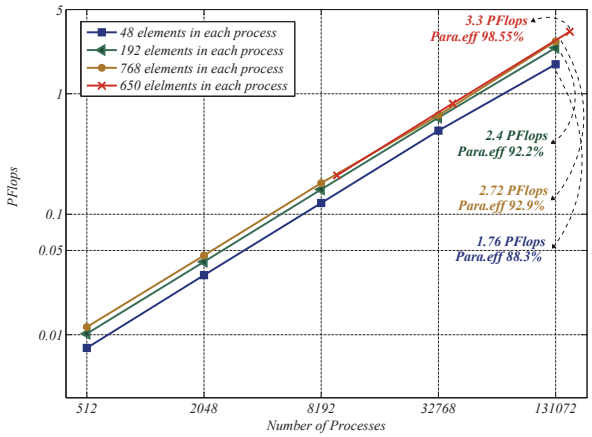

Figure 8: Weak scalability, elements in each process are fixed to be 48, 192, 650, 768, respectively.

8.5 Performance over Other Dynamical Cores

Table 3 shows our simulation speed compared with other dynamical cores for the experiment configuration in the benchmarks for the Next Generation Global Prediction System (NGGPS) evaluation [35] in US. Our redesigned HOMME outperforms the speed of FV3, and MPAS for the 12.5 km scenario. For the extreme case of 3 km simulation, the performance advantage is even better, and is 2.1 times and 4.5 times better than the FV3 and MPAS, respectively.

The result also echoes what we see in the scaling experiments, *i.e.* the efficiency in high-resolution case is generally better than that in low-resolution cases, due to the limited number of elements assigned to each CG. In high-resolution cases, we have enough compute to assign to the 65 cores in each CG, and can therefore achieve a better performance.

9 SIMULATION OF HURRICANE KATRINA

Using the redesigned CAM model on Sunway, we manage to simulate the hurricane Katrina with fine accuracy in both the track and the intensity. This is, to the best of our knowledge, the first work to simulate the life cycle of Katrina using a global climate model.

Table 3: The simulation time of our redesigned HOMME over other dynamical cores reported in the evaluation for the next generation global prediction system (NGGPS) [35].

dynamical core	our work	FV3	MPAS
12.5 km simulation for 2-hour prediction workload			
Number of MPI Processes	131,072	110,592	96,000
run time	**2.712 s**	3.56 s	7.56 s
3 km simulation for 30-min prediction workload			
Number of MPI Processes	131,072	110,592	131,072
run time	**14.379 s**	30.31 s	64.80 s

Katrina was an extraordinarily powerful and destructive hurricane that devastated Gulf Coast of the United States in August 2005, causing thousands of deaths and damages estimated as nearly 200 billion US dollars [36]. It was the costliest and one of the five deadliest hurricanes that ever stuck the United States.

Tropical cyclones such as Katrina are significantly under-resolved at traditional GCM grid spacings of 50–300 km [37]. Research indicates that approximately 50 km or smaller horizontal grid spacing is necessary to simulate tropical cyclones observed in the climate system [38]. Due to the high requirement on resolution, most existing efforts are using regional weather prediction models for tropic cyclones simulation, instead of the global climate model that are too complicated to guarantee good time-to-solution. However, regional weather prediction models have to face the challenges such as the required lateral boundary conditions (LBCs) which may introduce additional errors through the use of a non-global domain. In this work, by using Sunway TaihuLight, we manage to simulate Katrina achieving both fine accuracy and good time-to-solution.

As shown in Figure 9 (b), the *ne*120 test with a horizontal resolution of 25 km perfectly captured the horizontal structure of Katrina, showing distinctive spatial distribution features of the cyclone through both upwelling flux and wind fields, while the *ne*30 (100 km) test failed to simulate hurricane Katrina (Figure 9 (a)). The *ne*120 test produced an excellent track with the positions nearly identical to observation, during the entire duration of hurricane Katrina (Figure 9 (c)) with the final landfall over Texas coast, except for slightly deviation towards east during the initial 24-hour period. The time series of the simulated cyclone maximum sustained wind (MSW) also agrees well with the observed evolution of Hurricane Katrina during the period from 1800 UTC of 23 August to 1200 UTC of 31 August, 2005 (Figure 9 (d)).

With our efforts of CAM over Sunway TaihuLight, high resolution atmospheric simulation can be achieved both efficiently and effectively, which might be a possible and potential way to conduct seamless weather and climate simulations and predictions.

10 IMPLICATIONS

People also say, “the only constant is change itself”. From vector machine, to IBM/Intel clusters, from single core, multi-core, to many-core, computer architecture, as well as the programming model, changes constantly. Researchers can choose either to adapt

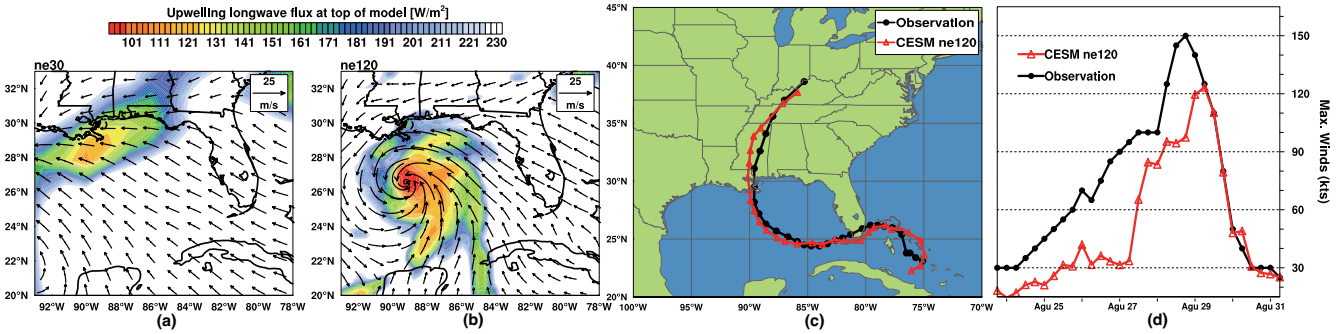

Figure 9: (a) Model simulated upwelling longwave flux and wind field at 1800 UTC of 28 August, 2005 from CESM *ne*30 (100 km); (b) Model simulated upwelling longwave flux and wind field at 1800 UTC of 28 August, 2005 from CESM *ne*120 (25 km); (c) Observed track of Hurricane Katrina from US National Hurricane Center accompanied by simulated tracks from CESM *ne*120; (d) Time series of observed and CESM ne120 predicted maximum wind (kts).

their problems to the newest general-purpose supercomputers, or, in rare cases, to customize a supercomputer for a specific kind of problems. The first option is usually the majority. As the supercomputers supported by national research organizations generally need to serve a wide spectrum of scientific problems, with different requirements for architectures, the new systems can only be built based on a general thinking to improve the computing performance and power efficiency. Therefore, we see very few special customized systems (usually with private or commercial sources of funding support), such as Anton [28] and Anton 2 [29] for molecular dynamics, and various deep learning processors [39].

In recent years, with the newly announced supercomputers almost all taking a heterogeneous many-core architecture, we see a huge challenge for existing scientific applications to achieve convincing performance improvements on these latest supercomputers. Take the CAM model as an example. With over three decades of development, the code has already gone through architectural transitions from Cray Vector Machines to IBM, and to most recent Intel clusters, and has accumulated million lines of legacy code that were possibly written in different decades. Another difficult is that for such a complicated model with so many different modules, we see no hotspots, or hundreds of hotspots. Therefore, we need to perform redesign of the entire software package to achieve any meaningful performance benefits.

In this work, we perform extensive optimizations for porting the CAM to the Sunway TaihuLight supercomputer. Porting an entire atmospheric model differs significantly from migrating just modules or kernels. Specifically, for the case of CAM, profiling results show only very few kernels above the threshold of occupying over 4% of the total execution time. For example, the most time-consuming kernel *euler_step* from Table 1 only accounts for 3.1% and 7.4% of the total run time for *ne*256 and *ne*4096 respectively. The implication is that we have to either redesign the entire model, or the performance benefits would only marginal. Besides, considering the distributed and complicated communication pattern in such a complex model, the speedup for the entire model, which might not be as exciting as the speedups reported for single kernels, does demonstrate the tremendous efforts we put into the redesign process. A large part of our redesign and optimization ideas can also be applied to other platforms, e.g. using shuffle or transposing to increase locality, using fused memory operation to achieve better bandwidth, using vertical layer division to increase parallelization, and using hybrid parallelization to hide communication overhead.

The OpenACC-based refactoring manages to scale the CAM model to 1,560,000 cores, with a simulation speed of 3.4 SYPD at the resolution of 25 km. With Athread-based fine-grained re-implementation of the HOMME dynamical core, we can further improve the performance of a CG (1 MPE + 64 CPEs) to the range that is equivalent to 7 to 46 Intel CPU cores, and to achieve a sustainable double-precision performance of 3.3 Pflops for a 750-m global simulation when using 10,075,000 cores .

To the best of our knowledge, this would be the first reported effort that migrate a scientific application at such a level of complexity to a completely different hardware architecture, and scale to over 10 million cores. This was also the first reported effort that manage to simulate the life cycle of a real-world Tropical cyclones, the hurricane Katrina, using a global climate model.

Experimental results show that our optimization reduces major bottlenecks from the original legacy of over million lines of codes, for example, data dependency, network bandwidth, memory control, etc. By refactoring and redesigning the whole framework, from top to bottom, the limitations for CAM are largely removed. Porting CAM, or other numerical applications with similar legacy problem, to the TaihuLight, or the soon-arriving Exa-scale supercomputers in the future, is applicable and feasible.

ACKNOWLEDGEMENT

Wei Xue and Yuxuan Li from Tsinghua University have made great contributions, and not listed as co-authors due to the limit on the number of authors for Gordon Bell Prize submissions. We also appreciate Rich Loft and John Dennis from NCAR, and Zhenya Song from the First Institute of Oceanography, SOA, for their suggestions. This work was supported in part by the National Key R&D Program of China (grant no. 2016YFA0602200), the National Natural Science Foundation of China (grant no. 41374113, 91530323), China Postdoctoral Science Foundation (no. 2016M601031), Tsinghua University Initiative Scientific Research Program (no. 20131089356).

The corresponding authors are L. Gan (lingan@tsinghua.edu.cn), L. Wang (wangln@bnu.edu.cn), X. Duan (sunrise.duan@mail.sdu.edu.cn), and N. Ding (dingnan0701@gmail.com).

REFERENCES

[1] Jeffrey K Lazo, Megan Lawson, Peter H Larsen, and Donald M Waldman. US economic sensitivity to weather variability. *Bulletin of the American Meteorological Society*, 92(6):709–720, 2011.

[2] Peter Lynch. The ENIAC forecasts: A re-creation. *Bulletin of the American Meteorological Society*, 89(1):45–55, 2008.

[3] John M Dennis, Jim Edwards, Katherine J Evans, Oksana Guba, Peter H Lauritzen, Arthur A Mirin, Amik St-Cyr, Mark A Taylor, and Patrick H Worley. CAM-SE: A scalable spectral element dynamical core for the Community Atmosphere Model. *The International Journal of High Performance Computing Applications*, 26(1):74–89, 2012.

[4] JE Kay, C Deser, A Phillips, A Mai, C Hannay, G Strand, JM Arblaster, SC Bates, G Danabasoglu, J Edwards, et al. The Community Earth System Model (CESM) large ensemble project: A community resource for studying climate change in the presence of internal climate variability. *Bulletin of the American Meteorological Society*, 96(8):1333–1349, 2015.

[5] Xiangke Liao, Liquan Xiao, and et al. MilkyWay-2 supercomputer: system and application. *Frontiers of Computer Science*, 8(3):345–356, 2014.

[6] Devesh Tiwari, Saurabh Gupta, George Gallarno, Jim Rogers, and Don Maxwell. Reliability lessons learned from GPU experience with the Titan supercomputer at Oak Ridge leadership computing facility. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, page 38. ACM, 2015.

[7] Haohuan Fu, Junfeng Liao, Jinzhe Yang, Lanning Wang, Zhenya Song, Xiaomeng Huang, Chao Yang, Wei Xue, Fangfang Liu, Fangli Qiao, et al. The Sunway TaihuLight supercomputer: system and applications. *Science China Information Sciences*, 59(7):072001, 2016.

[8] John Michalakes and Manish Vachharajani. GPU acceleration of numerical weather prediction. *Parallel Processing Letters*, 18(04):531–548, 2008.

[9] Rory Kelly. GPU Computing for Atmospheric Modeling. *Computing in Science and Engineering*, 12(4):26–33, 2010.

[10] Chao Yang, Wei Xue, Haohuan Fu, Hongtao You, Xinliang Wang, Yulong Ao, Fangfang Liu, Lin Gan, Ping Xu, Lanning Wang, et al. 10M-core scalable fully-implicit solver for nonhydrostatic atmospheric dynamics. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 6. IEEE Press, 2016.

[11] Zhuowei Wang, Xianbin Xu, Naixue Xiong, Laurence T Yang, and Wuqing Zhao. GPU Acceleration for GRAPES Meteorological Model. In *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*, pages 365–372. IEEE.

[12] S. Xu, X. Huang, Y. Zhang, H. Fu, L.-Y. Oey, F. Xu, and et. al. gpuPOM: a GPU-based Princeton Ocean Model. *Geoscientific Model Development Discussions*, 7(6):7651–7691, 2014.

[13] Jule G Charney and Arnt Eliassen. A numerical method for predicting the perturbations of the middle latitude westerlies. *Tellus*, 1(2):38–54, 1949.

[14] Masaki Satoh, Taro Matsuno, Hirofumi Tomita, and et. al. Nonhydrostatic icosahedral atmospheric model (NICAM) for global cloud resolving simulations. *Journal of Computational Physics*, 227(7):3486–3514, 2008.

[15] Hironori Fudeyasu, Yuqing Wang, Masaki Satoh, Tomoe Nasuno, Hiroaki Miura, and Wataru Yanase. Global cloud-system-resolving model NICAM successfully simulated the lifecycles of two real tropical cyclones. *Geophysical Research Letters*, 35(22), 2008.

[16] Peter Johnsen, Mark Straka, Melvyn Shapiro, Alan Norton, and Thomas Galarneau. Petascale WRF simulation of hurricane sandy: Deployment of NCSA's cray XE6 blue waters. In *High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for*, pages 1–7. IEEE, 2013.

[17] J.C. 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*, pages 1–11, Nov 2009.

[18] Jarno Mielikainen, Bormin Huang, Jun Wang, H.-L. Allen Huang, and Mitchell D. Goldberg. Compute unified device architecture (CUDA)-based parallelization of WRF Kessler cloud microphysics scheme . *Computers and Geosciences*, 52(0):292 – 299, 2013.

[19] Huadong Xiao, Jing Sun, Xiaofeng Bian, and Zhijun Dai. GPU acceleration of the WSM6 cloud microphysics scheme in GRAPES model. *Computers and Geosciences*, 59(0):156 – 162, 2013.

[20] Mark W Govett, Jacques Middlecoff, and Tom Henderson. Running the NIM next-generation weather model on GPUs. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 792–796. IEEE Computer Society.

[21] Ilene Carpenter, RK Archibald, Katherine J Evans, Jeff Larkin, Paulius Micikevicius, Matt Norman, Jim Rosinski, Jim Schwarzmeier, and Mark A Taylor. Progress towards accelerating HOMME on hybrid multi-core systems. *The International Journal of High Performance Computing Applications*, 27(3):335–347, 2013.

[22] V.T. Vu, G. Cats, and L. Wolters. Graphics processing unit optimizations for the dynamics of the HIRLAM weather forecast model. *Concurrency and Computation: Practice and Experience*, 25(10):1376–1393, 2013.

[23] Irina Demeshko, Naoya Maruyama, Hirofumi Tomita, and Satoshi Matsuoka. Multi-GPU implementation of the NICAM atmospheric model. In *European Conference on Parallel Processing*, pages 175–184. Springer, 2012.

[24] Fangli Qiao, Wei Zhao, Xunqiang Yin, Xiaomeng Huang, Xin Liu, Qi Shu, Guansuo Wang, Zhenya Song, Xinfang Li, Haixing Liu, et al. A highly effective global surface wave numerical simulation with ultra-high resolution. In *High Performance Computing, Networking, Storage and Analysis, SC16: International Conference for*, pages 46–56. IEEE.

[25] Takashi Shimokawabe, Takayuki Aoki, Chiashi Muroi, Junichi Ishida, Kohei Kawano, Toshio Endo, Akira Nukada, Naoya Maruyama, and Satoshi Matsuoka. An 80-fold speedup, 15.0 TFlops full GPU acceleration of non-hydrostatic weather model ASUCA production code. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE Computer Society.

[26] Tobias Gysi, Carlos Osuna, Oliver Fuhrer, Mauro Bianco, and Thomas C. Schulthess. Stella: A domain-specific tool for structured grid methods in weather and climate models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 41:1–41:12, New York, NY, USA, 2015. ACM.

[27] Oliver Fuhrer, Carlos Osuna, Xavier Lapillonne, Tobias Gysi, Ben Cumming, Mauro Bianco, Andrea Arteaga, and Thomas Christoph Schulthess. Towards a performance portable, architecture agnostic implementation strategy for weather and climate models. *Supercomputing frontiers and innovations*, 1(1):45–62, 2014.

[28] David E Shaw, Ron O Dror, John K Salmon, JP Grossman, Kenneth M Mackenzie, Joseph A Bank, Cliff Young, Martin M Deneroff, Brannon Batson, Kevin J Bowers, et al. Millisecond-scale molecular dynamics simulations on Anton. In *High performance computing networking, storage and analysis, proceedings of the conference on*, pages 1–11. IEEE, 2009.

[29] David E Shaw, JP Grossman, Joseph A Bank, Brannon Batson, J Adam Butts, Jack C Chao, Martin M Deneroff, Ron O Dror, Amos Even, Christopher H Fenton, et al. Anton 2: raising the bar for performance and programmability in a special-purpose molecular dynamics supercomputer. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 41–53. IEEE Press, 2014.

[30] Andrew S Cassidy, Rodrigo Alvarez-Icaza, Filipp Akopyan, Jun Sawada, John V Arthur, Paul A Merolla, Pallab Datta, Marc Gonzalez Tallada, Brian Taba, Alexander Andreopoulos, et al. Real-time scalable cortical computing at 46 giga-synaptic OPS/watt with. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 27–38. IEEE Press, 2014.

[31] Ready for the Dawn of Aurora: NWChemEx granted time on new LCF supercomputer to advance popular computational chemistry code. https://www.pnnl.gov/science/highlights/highlight.asp?id=4411. Accessed: 2017-02.

[32] Marat Valiev, Eric J Bylaska, Niranjan Govind, Karol Kowalski, Tjerk P Straatsma, Hubertus JJ Van Dam, Dunyou Wang, Jarek Nieplocha, Edoardo Apra, Theresa L Windus, et al. NWChem: a comprehensive and scalable open-source solution for large scale molecular simulations. *Computer Physics Communications*, 181(9):1477–1489, 2010.

[33] Paul Messina et. al. Revolutionizing High-Performance Computing. Technical report, Exa-Scale Computing Project.

[34] Niall Ferguson. *Civilization: the six killer apps of western power*. Penguin UK, 2012.

[35] John Michalakes, Rusty Benson, Tom Black, Michael Duda, Mark Govett, T Henderson, P Madden, George Mozdzynski, Alex Reinecke, William Skamarock, et al. Evaluating Performance and Scalability of Candidate Dynamical Cores for the Next Generation Global Prediction System. 2015.

[36] Eric S Blake, Edward N Rappaport, Jerry D Jarrell, Chris Landsea, and Tropical Prediction Center. *The deadliest, costliest, and most intense United States tropical cyclones from 1851 to 2006 (and other frequently requested hurricane facts)*. NOAA/National Weather Service, National Centers for Environmental Prediction, National Hurricane Center Miami, 2007.

[37] Colin M Zarzycki, Christiane Jablonowski, and Mark A Taylor. Using variable-resolution meshes to model tropical cyclones in the community atmosphere model. *Monthly Weather Review*, 142(3):1221–1239, 2014.

[38] Lennart Bengtsson, Kevin I Hodges, Monika Esch, Noel Keenlyside, Luis Kornblueh, JING-JIA LUO, and Toshio Yamagata. How may tropical cyclones change in a warmer climate? *Tellus a*, 59(4):539–561, 2007.

[39] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, et al. Dadiannao: A machine-learning supercomputer. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 609–622. IEEE Computer Society, 2014.