This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2366754, IEEE Transactions on Computers

Ultra-scalable CPU-MIC Acceleration of Mesoscale Atmospheric Modeling on Tianhe-2

Wei Xue*‡, Chao Yang†§, Haohuan Fu‡, Xinliang Wang*‡, Yangtong Xu*‡,
Junfeng Liao*‡, Lin Gan*‡, Yutong Lu¶, Rajiv Ranjan‖, Lizhe Wang**
*Dept. of Computer Science & Technology, Tsinghua University, China
†Institute of Software, Chinese Academy of Sciences, China
‡Ministry of Education Key Laboratory for Earth System Modeling,
and Center for Earth System Science, Tsinghua University, China
§State Key Laboratory of Computer Science, Chinese Academy of Sciences, China
¶Dept. of Computer Science & Technology, National University of Defense Technology, China
‖Computational Informatics, CSIRO, Australia
**Institute of Remote Sensing & Digital Earth, Chinese Academy of Sciences, China
Corresponding author: Lizhe Wang, lizhe.wang@gmail.com

***Abstract*—In this work an ultra-scalable algorithm is designed and optimized to accelerate a 3D compressible Euler atmospheric model on the CPU-MIC hybrid system of Tianhe-2. We first reformulate the mesocale model to avoid long-latency operations, and then employ carefully designed inter-node and intra-node domain decomposition algorithms to achieve balance utilization of different computing units. Proper communication-computation overlap and concurrent data transfer methods are utilized to reduce the cost of data movement at scale. A variety of optimization techniques on both the CPU side and the accelerator side are exploited to enhance the in-socket performance. The proposed hybrid algorithm successfully scales to 6,144 Tianhe-2 nodes with a nearly ideal weak scaling efficiency, and achieve over 8% of the peak performance in double precision. This ultra-scalable hybrid algorithm may be of interest to the community to accelerating atmospheric models on increasingly dominated heterogeneous supercomputers.**

***Keywords*-Atmospheric modeling; MIC; Stencil; Tianhe-2**

I. Introduction

As global warming has become a pivotal environmental and social issue in the 21st century, large-scale numerical models have become an indispensable tool for climate science. Climate information provided by different models has a great number of user groups, public and private organizations, and its demand is continuously increasing [1]. The challenge of climate simulation is due to that it involves a large number of physical processes interacting over a large range of space and time scales. More accurate models need wider range of scales with a finer mesh resolution and larger number of physical processes with more complicated theories, which leads to significantly increased demand for computing power.

Over the last 40 years, the computing power of the fastest supercomputers has increased from 250 Mflops to 33.9 Pflops, with a trend of increasing by ten times per decade in the future. In addition, heterogeneous systems with high density many-core accelerators, have gradually become a main force in current supercomputers. Examples include Tianhe-1A, Tianhe-2, and Titan.

While climate model calls for an increase of several orders of magnitude in the computing power, most existing global climate models are still struggling with the poor scalability and the inability to use many-core accelerators such as GPUs or MICs in current heterogeneous supercomputers [2][3].

To tackle with the above problem, we propose a highly scalable algorithm framework for atmospheric simulation which can make efficient utilization of both CPU resources and many-core accelerators. Our previous work of a highly-scalable 2D shallow water model has scaled successfully on both Tianhe-1A and Tianhe-2. It sustains a performance of 0.8 Pflops in double precision with 3,750 nodes (45,000 cores and 3,750 GPUs) on Tianhe-1A [4] and a double performance of 1.63 Pflops with 8,664 nodes (nearly 1.7 million cores) on Tianhe-2 [5].

In this work, we further extend our previous work into a mesoscale fully compressible 3D Euler model that provides much more accurate description of the atmospheric dynamics. The 3D Euler model requires more floating-point operations and leads to a more complex communication pattern that challenge the bandwidth in demand. Moreover, the definition of a larger computational area and the different spatial and temporal scales in a wider range will challenge the efficiency of the petascale heterogeneous supercomputer. A stencil-based method is required instead of a particle-based method because of the advantage of its reduction on the coupling of the system. Based on a successful computation and communication overlapping algorithm as well as the communication optimizations, the algorithm achieves a nearly ideal weak scaling efficiency and 73% strong scaling efficiency using up to 6,144 computing nodes of Tianhe-2.

To enhance the performance on many-core accelerators, we focus our work on tuning the stencil kernel of the 3D atmospheric model for Intel MIC architecture by model reformulation, data layout optimization, loop splitting and software prefetching. With the above techniques in hands, we successfully push the performance to nearly 10% of the peak

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2366754, IEEE Transactions on Computers

performance on an accelerator. We also carefully tune the domain decomposition among the CPU cores and the multi-MICs as well as the communication across nodes on Tianhe-2, so as to achieve best utilization of various computation resources within the hybrid system.

In summary, the main contributions of this work are:

- A reformulated mesoscale model that avoid calculations of `pow` (x^y). The fully compressible Eulers equations widely adopted in mesoscale atmospheric modeling usually repeated evaluations of `pow`, which may introduce unnecessary overhead on today's many-core/many-core system. We replace the traditional mesoscale model with a reformulated one to avoid the calculation of `pow` and to increase the overall performance.
- A comprehensive tuning progress for complex stencil code on Intel MIC architecture. In this work, we get close to 10% peak performance of single Intel Xeon Phi processor, which is 20% better than that of our previous work on 2D shallow water atmospheric model [5]. The techniques we use may be insightful to similar works on other real-world stencil applications.
- Achieving a nearly ideal weak scaling efficiency of 96% on Tianhe-2 when gradually increasing the number of computing nodes from 64 to 6,144 (nearly 1.2 million cores). A performance of 1.74 Pflops in double precision was sustained in the largest run. The strong scaling efficiency is about 73% when the number of nodes increases from 1,024 to 6,144 for a fixed 53.8 billion mesh with 269 billion unknowns. To the best of our knowledge, this is an unprecedented run of such a large-scale 3D atmospheric simulation as well as a new record of Petaflops atmospheric simulation.
- We present some experiences on detecting and resolving silent errors of device/node during experiments at scale. The work may be of interest to the user community of large scale heterogeneous machine like Tianhe-2 as well as the system software designers and developers of such kind of system.

II. Related Works

As one of the most important applications in the world, atmospheric modeling has drawn much attention in high performance computing [6][7]. Since GPU became a main driven force for heterogeneous computing, techniques on single card acceleration have been studied for several atmospheric models, including the WRF model [8], the NIM model [9], the GEOS-5 model [10], the HOMME model [11], and the GRAPES model [12]. Although significant speedup over a single CPU core has been observed in the above mentioned works, enabling multi-node acceleration is an urgent demand for large-scale simulations.

A pioneering work on speeding up atmospheric modeling on multi-node heterogeneous platforms was done by Shimokawabe *et al.* [13], in which a multi-GPU algorithm was proposed for the ASUCA nonhydrostatic model and demonstrated with up to over 500 GPUs scalability on T-SUBAME 1.2 supercomputer. The work was further extended to nearly 4,000 GPUs on TSUBAME 2.0 with a 145 Tflops performance in single precision [14]. Another nonhydrostatic model, NICAM, was recently accelerated by several times on up to 320 GPUs [15], although only the shallow water code was accelerated. In our previous works, we have enabled both CPU-GPU acceleration on using up to 3750 nodes on Tianhe-1A [4], and CPU-MIC acceleration on using up to 8664 nodes on Tianhe-2 [5] for a 2D shallow water model. In this paper, we extend the previous works to a 3D mesoscale model with more insightful algorithms and optimization techniques and achieve a flops efficiency of over 8% on Tianhe-2. In addition, we present some experiences on detecting and resolving silent errors on Tianhe-2, which may be of interest to the community of heterogeneous computing. Some works [35], [36], [37]focus on about how the ratio of communication-computation degrade the performance. Since the communication will affect the performance a lot in large scale test, in this work we use non-blocking communication to overlap the communication overhead with computation. We also exploit some MPI optimizations and environment variables setting to boost the performance. We think that is the valuable experience for the platform and the extreme large-scale situation.

Stencil computation is an important kernel in scientific applications, a variety of optimization techniques have been well addressed in recent years for CPU, GPGPU, and IBM Cell processor [16]. To resolve issues of data reusing, the cache-oblivious algorithm, and the 3.5-D blocking algorithm have been proposed in [17], [18], [19]. Meanwhile, to relieve the searching burden, auto-tuning framework has been developed for stencil computation, so as to select the optimal blocking factors as well as implementations for different architectures [20]. For stencil optimization on the Intel MIC architecture, [21] discussed the combination of temporal and space blocking as well as block decomposition in task parallelism on MPDATA stencil. In this paper, we employe a reformulated atmospheric model to avoid long-latency operations and take various efforts to tune the complex 3D stencil kernel on the Intel MIC architecture, by trading off between exploiting vectorization with irregular memory accesses and keeping good locality of L1 cache access, and getting a 20% performance improvement compared with our early work in [5].

As a newly launched many-core system, there are several studies conducted with respect to analyzing performance results, designing algorithms on the Intel MIC architecture as well as tuning kernels. Tuning an image reconstruction code on Intel Xeon Phi was demonstrated in [22]. A high performance multi-node MIC-based Linpack implementation was presented in [23]. The discussions concerning the L1 cache pressure issue and high latency for memory access of current Intel Xeon Phi acclerators in [24], [23], [25], [26] help us better understand the performance results of the atmospheric model. The gather-scatter performance bottlenecks were evaluated in [27] during vectorizing molecular dynamics code on different architectures, including the Intel MIC architecture. It was also

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2366754, IEEE Transactions on Computers

reported in [28] that current system software MPSS is lack of protecting the offloads and managing processes across Intel Xeon Phi processors and CPU cores, which helps us detect silent errors in large scale tests on Tianhe-2.

III. Model Description

Several equation sets are available to model the complex dynamics of the atmosphere. Compared with other simplified models such as the shallow water model, the hydrostatic model, or the incompressible model, the fully compressible Euler equations are accurate at mesoscale with almost no assumption made [29]. Considering a non-rotating 3D channel with possibly nonsmooth bottom topography, a widely adopted form of the Euler equations is as follows [30]:

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

$$\begin{aligned} Q &= (\rho', \rho u, \rho v, \rho w, (\rho\theta)')^T, \\ F &= (\rho u, \rho uu + p', \rho uv, \rho uw, \rho u\theta)^T, \\ G &= (\rho v, \rho vu, \rho vv + p', \rho vw, \rho v\theta)^T, \\ H &= (\rho w, \rho wu, \rho wv, \rho ww + p', \rho w\theta)^T, \\ S &= (0, 0, 0, \rho' g, 0)^T, \end{aligned} \tag{2}$$

Here ρ, $\mathbf{v} = (u, v, w)$, p, and θ are the density, the velocity, the pressure and the potential temperature of the atmosphere, respectively. The system is closed with the equation of state

$$p = p_{00}\left(\frac{\rho R\theta}{p_{00}}\right)^{\gamma}, \tag{3}$$

where $p_{00} = 1013.25\text{hPa}$ is the ground level pressure, $R = 287.04\text{J}/(\text{kg}\cdot\text{K})$ is the gas constant for dry air and $\gamma = 1.4$. To minimize roundoff errors, values of $\rho' = \rho - \overline{\rho}$, $(\rho\theta)' = \rho\theta - \overline{\rho}\overline{\theta}$ and $p' = p - \overline{p}$ have been shifted according to the hydrostatic state that satisfies $\frac{\partial \overline{p}}{\partial z} = -\overline{\rho}g$.

To solve the Euler equations (1)-(3), we make use of a terrain-following mesh and employ a cell-centered finite volume scheme for spatial discretization together with a second-order TVD Runge-Kutta method for time stepping; a similar work has been done in [31] for the Euler equations in 2D. At each time step, two stencil sweeps are applied consecutively at all mesh elements. For each mesh element, the stencil evaluation consists of three steps: (i) compute coordinates of the mesh element; (ii) reconstruct intermediate states from inside and outside of the element by using a linear interpolation; (iii) calculate numerical fluxes on mesh boundaries by employing a AUSM+up Riemann solver and update the mesh state. The pattern of the resulting computation is a 3D 25-point stencil, as shown in Fig. 1.

IV. Tianhe-2 Supercomputer

The Tianhe-2 supercomputer, developed by the National University of Defense Technology (NUDT), is the current top-ranked supercomputer in the TOP 500 list since June 2013. Tianhe-2 has already been installed in Guangzhou Supercomputing Center and is currently in early-use phase.

The peak performance of Tianhe-2 is 54.9 Pflops. The sustained LINPACK performance is 33.9 Pflops, leading to

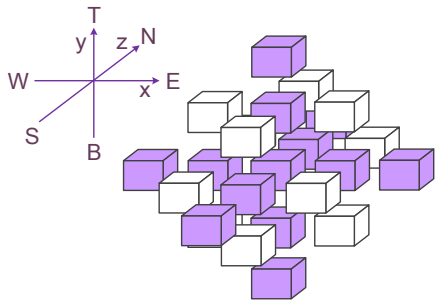

Fig. 1. A 25-point stencil for the 3D Euler equations.

a performance-per-watt of 1.9 Gflops/W. Tianhe-2 consists of 16,000 computing nodes, each of which is equipped with two Intel Xeon E5-2692 processors, three Intel Xeon Phi 31S1P accelerators, and 64GB memory. There are in total 1.4PB of memory and 12.4PB of storage with a power consumption of 17.8MW at its peak. The system software includes a 64-bit Kylin OS, an Intel 14.0 compiler, a customized MPICH-3.1 for TH Express-2, and a self-designed hybrid hierarchy file system H2FS.

All computing nodes are connected with a customized network named TH Express-2, which uses a fat tree topology including thirteen 576-port switches at the top level. Every 32 nodes in Tianhe-2 are collected into one frame and most of the frames have 16 links connected to the top level switches. The bi-directional bandwidth of TH Express-2 can achieve 20GB/s in theory, and offloaded collective operations are supported.

The Intel Xeon Phi accelerator is manufactured based on the Intel Many Integrated Core (MIC) architecture. The Intel Xeon Phi 31S1P accelerator installed in Tianhe-2 has a theoretical peak double precision performance of 1.003 TreaFLOPS. Each MIC coprocessor equipped Tianhe-2 has 57 cores running at 1.1GHz and 8GB on-board GDDR5 memory all connected by a high performance bidirectional ring. Each core is an in-order, dual-issue core having 4-way hyper-threading support to help hide memory and multi-cycle instruction latency. Each core also has a proprietary L1 cache and a shared L2 cache. The L1 cache consists of 8-way set associative 32 KB L1 instruction and 32 KB L1 data cache. L1 data cache has two ports: one for read and the other for write. Issuing a prefetch instruction will occupy both of the two ports at the same time, and might delay L1 data accesses for computation. In the worst case, it will stall the core pipeline. The L2 cache is also 8-way set associative and 512 KB in size with a 14-15 cycles latency, which is fully coherent using a set of tag directories. The latency of the L2 cache miss on MIC can be an order of magnitude larger than that on multi-core CPUs. The Intel MIC architecture provides a completely new 512-bit SIMD instruction set. Among the new features provided by the new instruction set, the gather instruction supports loading sparse memory locations into a vector register and the scatter instruction performs its reverse operation. The two instructions may help enhance the performance of a simple code when

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2366754, IEEE Transactions on Computers

using the data layout of Array of Structure (AoS), which usually degrade the performance on many-core platforms.

The current Intel Xeon Phi accelerator features a shared memory model across all threads and supports traditional multiprocessor programming models such as Pthreads and OpenMP. Offloading is the most efficient way to program Intel MIC architecture as an accelerator and is used for hybridizing our code on Tianhe-2. The Intel Many Integrated Core Platform Software Stack (MPSS) provides the capability to operate Intel Xeon Phi accelerator properly and inter-operate with other hardware components in a computing node, which has a symmetric software architecture both on host side and accelerator side. The MPSS version used in Tianhe-2 is 3.1.2. It has to be noted that MPSS introduces some overhead on the host side and offloading involves one hardware core of the accelerator for initialization, marshaling and transferring data, and invocation. Moreover, as mentioned in [28], the MPSS does not protect against thread and memory oversubscription and isolate the execution of offloads on accelerators and also does not manage the computing resources across accelerators and host efficiently, which will introduce difficulty in running applications when one accelerator is temporally offline in large-scale multi-accelerator systems like Tianhe-2.

V. Parallel Algorithms

A. Reformulated model to avoid long-latency operations

In the compressible Euler model (1) and (2), the pressure p in (2) is calculated based on the state equation (3), which requires computations of `pow` (x^y). For today's many-core platform, the evaluation of `pow` may introduce a long latency and substantially degrade the overall performance. Therefore it is necessary to seek for a new formulation that does not require this long-latency operation. To that end, we follow [39] by replacing the energy equation with the total energy ρe_T as a prognostic variable and rewrite (2) as

$$\begin{aligned} Q &= \left(\rho', \rho u, \rho v, \rho w, (\rho e_T)'\right)^T, \\ F &= \left(\rho u, \rho uu + p', \rho uv, \rho uw, (\rho e_T + p)\, u\right)^T, \\ G &= \left(\rho v, \rho vu, \rho vv + p', \rho vw, (\rho e_T + p)\, v\right)^T, \\ H &= \left(\rho w, \rho wu, \rho wv, \rho ww + p', (\rho e_T + p)\, w\right)^T, \\ S &= \left(0, 0, 0, \rho' g, 0\right)^T, \end{aligned} \tag{4}$$

where p is calculated from

$$p = (\gamma - 1)\rho(e_T - \frac{1}{2}||\mathbf{v}||^2 - gz). \tag{5}$$

Although the mathematical models have changed, the discretization described in Section III remains essentially intact with only minor modifications.

In the 3D Euler model, there are some other operations might also bring out long latency, for example, `div` and `sqrt` in calculating fluxes and sound speed. These operations are widely used in today's scientific applications and are usually not easy to be replaced. Despite some efforts done in [32][33], we did not use those techniques in our design. It is worth pointing out that further investigations and analysis are necessary on these operations, especially on today's many-core accelerators.

B. Inter-node domain decomposition

The whole 3D channel is partitioned into small subdomains with each assigned to a node of Tianhe-2. In practice, we use a 2D decomposition strategy although the whole domain is 3D (as shown in Fig. 2). The reason of using a 2D method instead of 3D is two fold. First, the vertical scale of the atmosphere is relatively small compared to the horizontal scale. Second, a 2D method helps reduce the complexity of boundary computation and the number of immediate neighbors for communication. Based on the 2D partition, before doing the stencil computing for each subdomain, it is required to communicate with four neighbors for updating the halo area needed by the stencil sweep of the subdomain. The whole procedure of a stencil sweep is illustrated in Fig. 3.

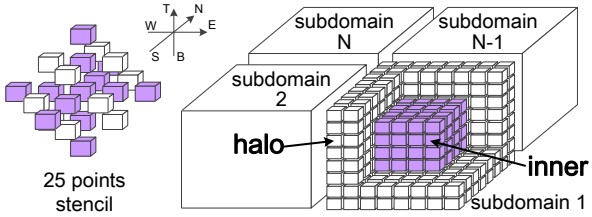

Fig. 2. A 2D domain decomposition scheme for the 3D domain. The decomposition is done only in the horizontal plane.

C. Intra-node subdomain partition

On each hybrid computing node, we need to perform the stencil computation of one subdomain (as shown in Fig. 2) and have to divide the subdomain among the CPU cores and accelerators. There are a number of options available for the subdomain partition.

One is the adjustable inner-outer partition proposed in [4]. The subdomain is divided into an inner part owned by accelerators and an outer part by CPUs. The size of the inner- and outer-part is adjustable to achieve a balanced utilization of both computing units. However, the method does have some disadvantages. First, the performance on the CPU side is not satisfactory due to irregular memory access pattern when processing the outer part. Second, it is not easy to efficiently communicate between multi-accelerators in a same node. Currently, the best way is to use the CPU as an agency. Third, it is difficult to conduct communication to both CPUs and other accelerators concurrently from an accelerator.

Another choice for partition is the process-level partition used in NICAM [15] and phase-field simulations [34]. The process-level partition performs an uniform partition of the cube and assigns one process to one accelerator correspondingly in the system. Even though this method is easy to implement, it might bring performance issues due to larger amount of MPI communication and inefficient utilization of CPUs.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2366754, IEEE Transactions on Computers

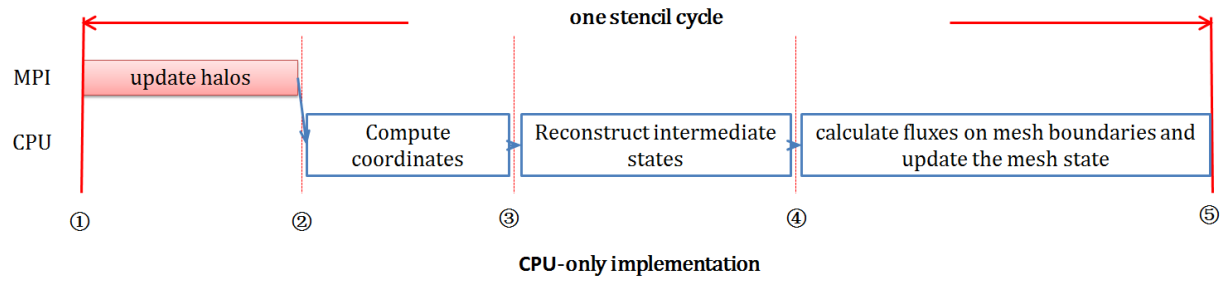

Fig. 3. A CPU-only stencil algorithm for a stencil sweep in the 3D Euler solver.

According to the issues mentioned above, we extend the 2D subdomain partition method proposed previously in [5] to the 3D subdomain. As shown in Fig. 4, when taking one Tianhe-2 node as an example, we partition the 3D subdomain in the horizontal plane into a halo area, three accelerator areas, three exchange areas, and one CPU area. The inner part of the 3D subdomain is sliced only along the y dimension to minimize memory stride and at the same time help fit for cache blocking on the accelerator. The exchange areas and the halo area are processed by the CPU and sent to the corresponding accelerators after packing the result. Here the size of the exchange areas and halo area is determined by the size of optimal cache blocking on the CPU, so as to enhance flops efficiency while not hindering the efficiency of inter-node communications and intra-node data transfers. Compared to the other two approaches, the CPU area is more regular for memory accesses and more cache friendly.

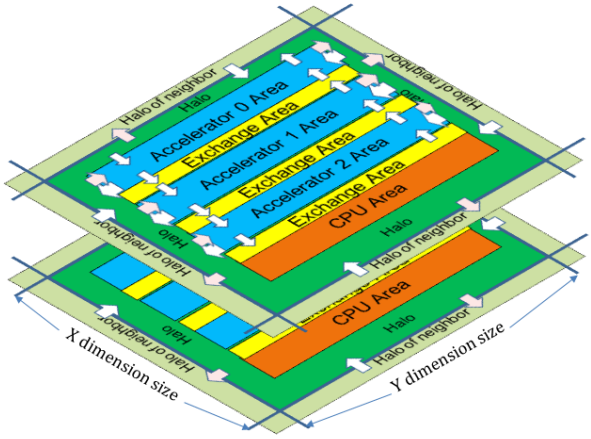

Fig. 4. A flexible partition scheme in which we divide the 3D subdomain into halo area, exchange areas, accelerator areas, and CPU area.

D. Computation-communication overlap

Inspired by previous works [35] [36] [37] [38], a carefully designed computation-communication overlapping algorithm is presented for the 3D Euler solver to incorporate with both the inter-node domain decomposition and the intra-node subdomain partition. In the algorithm, the computations for the inner part processed by the CPU and the accelerator are used to hide the cost of inter-node communication and reduce the effect of communicating fluctuation. The overall work flow of the hybrid algorithm for each stencil sweep on Tianhe-2 is shown in Fig. 5. In the hybrid algorithm, except updating halo which is overlapped with other operations, the CPU is in charge of: ① copying data for the halo area; ② computing stencils in the halo area; ③ computing stencils in the exchange area; and then ④-⑨ exchanging data with the three MIC cards. simultaneously, each MIC card only focuses on: ① computing stencils in the accelerator area; and ⑤-⑧ exchanging data with the CPU.

VI. Implementations and Optimizations

A. Accelerator level optimizations

Several optimization techniques are employed on MIC, include threading/vectorization, loop splitting, AoS to SoA, prefetching and intermediate variable reusing. The performance increase of each method is shown side-by-side in Fig.6.

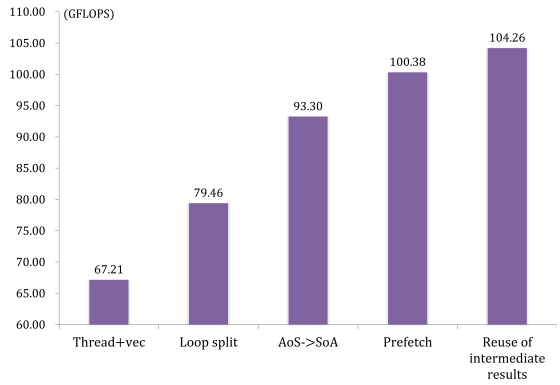

Fig. 6. Performance tuning by using different optimization techniques on MIC. The problem size on MIC is $256 \times 116 \times 224$.

1) Threading and vectorization: The CPU processor and the MIC accelerator share a similar computing architecture that consists of multiple cores and vector processing units. Therefore, as a starting point of the optimization process, we utilize threading and vectorization to achieve an efficient utilization of all available processing resources.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2366754, IEEE Transactions on Computers

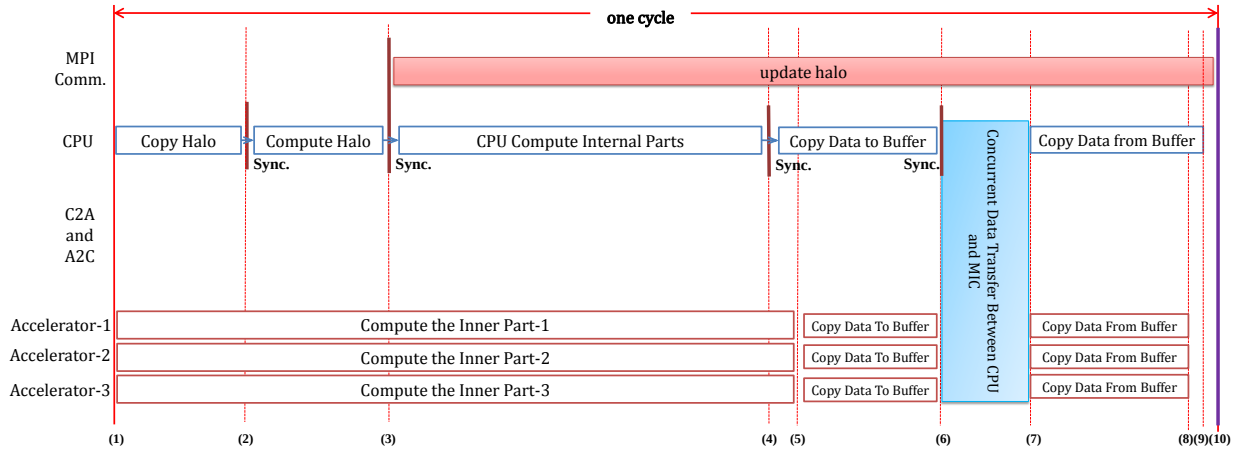

Fig. 5. The work flow of the hybrid algorithm for each stencil sweep.

Taking the MIC accelerator as an example, we run 224 threads on the 56 cores to process different planes in the 3D domain. In terms of vectorization, we move all conditional statements to the CPU side, so as to assure a better vectorization on the MIC side.

As the baseline version of the entire optimization process, our design with threading and vectorization can achieve a performance of 67.21 GFlops.

2) Loop splitting: After removing all the `pow` operations, the performance of CPU increases significantly from 62.19 GFlops to 116.67 GFlops. However, the same technique has little effect on MIC. Instead an apparent performance gap between the CPU and the MIC accelerator may occur. Therefore it is necessary to further enhance the performance on MIC to avoid imbalanced utilization of different computing units.

According to the profiling results in Intel VTune, the L1 hit ratio on the MIC side is only 69.4%. By analyzing the algorithm, we find out that the low hit ratio is mainly due to the complicated data structure used in the 3D model. We have 5 variables (40 bytes) for each mesh element, and around $33 \times 5 = 165$ variables (1320 bytes) for the stencil computation of each mesh element. Considering the vectorization on MIC, we need 6920 bytes for calculating 8 elements in parallel, which is already very close to the L1 data cache size allocated for each thread (8KB). Therefore it is important to find a way to relieve the data requirement during the computing.

An important feature of the 25-stencil algorithm is that the stencil computation along the three different dimensions are all independent. Thus, we can split the entire loop into three individual loops. By splitting the loop, the size of data required for computing 8 elements in parallel is reduced to 3160 bytes, 4720 bytes, 4720 bytes for the x, y, and z directions respectively, and the L1 cache hit ratio is increased to 92.6%. The performance on the MIC side also increases from 67.21 GFlops to 79.46 GFlops accordingly.

3) AoS to SoA: According to VTune, the vectorization intensity on the MIC side is only 4.252, which is relatively low when compared with the CPU side. We analyze that the main obstacle to vectorization on MIC is mainly due to the original AoS (Array of Structure) data layout in the kernel. To solve this issue, we reformat the data layout on the MIC side from AoS to SoA (Structure of Array). The data layout on the CPU side remains unchanged, so as to avoid rewriting the data transfer and communication interfaces. For the data exchange between CPU and MIC, the CPU would perform the data layout transform in the data exchange buffer. The change of data layout from AoS to SoA on MIC leads to a significant improvement of the vectorization intensity from 4.252 to 8.878, with the performance increased from 79.46 GFlops to 93.30 GFlops accordingly.

4) Prefetching: Compared with the AoS data layout, the SoA data layout enables a better vectorization but also leads to a worse data locality. After the data layout transform, the L1 cache hit ratio reduces from 92.6% to 72.9%.

To compensate the cache miss penalty, we include manual cache prefetching instructions to increase the L1 cache hit ratio. By exploring all possible prefetching offsets, we identify that the optimal strategy is to prefetch the 8 elements required for the current vectorized operation to L1, and to prefetch the 8 elements required for the next vectorized operation to L2. The strategy for L1 cache prefetching is shown in Fig.7. The colored elements are required in the calculation while among them the ones with dark color are manually prefetched.

By applying the manually prefetching strategy, we increase the L1 cache hit ratio from 72.9% to 81.8%, and improve the performance from 93.3 GFlops to 100.38 GFlops.

We have also explored various spatial and temporary blocking strategies to improve the cache hit ratio. However, none of them seem to work due to the large number of variables related to each mesh element.

5) Intermediate variable reusing: According to our profiling results, the reconstruction step is the most time-consuming part in the stencil sweep. To further improve the performance, we explore the possibility to reuse some of the intermediate variables in the reconstruction step.

When computing the result of mesh element (k, j, i), we

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2366754, IEEE Transactions on Computers

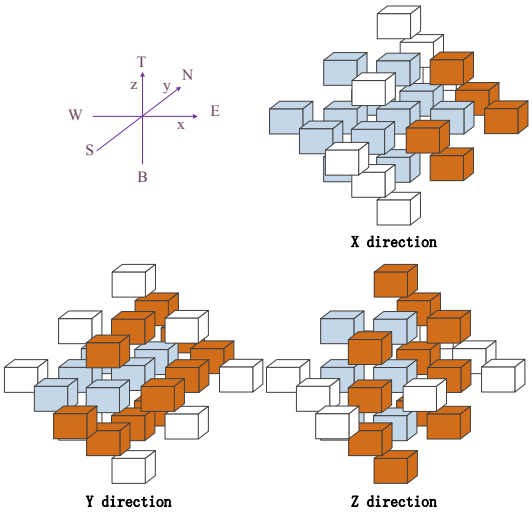

Fig. 7. Prefetching strategy in each direction. Elements in colors are dependent elements in the computing. The dark color ones are manually prefetched to enhance performance.

need to firstly compute the rightward reconstruction value of mesh element $(k, j-1, i)$, the leftward and rightward reconstruction value of mesh element (k, j, i), and the leftward reconstruction value of mesh element $(k, j+1, i)$. Similarly, when computing the result on mesh element $(k, j+1, i)$, we need to compute the rightward reconstruction value of mesh element (k, j, i), the leftward and rightward reconstruction value of mesh element $(k, j+1, i)$, and the leftward reconstruction value of mesh element $(k, j+2, i)$. Therefore, the rightward reconstruction value of mesh element (k, j, i) and the leftward reconstruction value of mesh element $(k, j+1, i)$ can be reused for the computation of mesh elements (k, j, i) and $(k, j+1, i)$.

We only apply this reusing technique along the y direction, as reusing along the x direction would cause vectorization conflicts, and reusing along the z direction leads to thread conflicts. By employing the reusing strategy, we further increase the performance from 100.38 GFlops to 104.26 GFlops.

B. Intra-node level optimizations

For intra-node level computation, we try to solve two important performance issues. One is data transfer between CPUs and MICs, and the other is thread affinity of CPU cores.

The data transfer between CPUs and accelerators needs to be designed carefully to take advantage of the duplex channel of the PCI-E bus. We propose an asynchronous and concurrent data transfer between CPU and accelerators, as shown in Fig. 8. Taking a Tianhe-2 node as an example, the CPU starts to pack and transfer the data to different MICs as early as possible after the computations for corresponding exchange area are done. And the MIC executes two offloading operations, rather than one, for bidirectional data transfers. With this scheme, we can exploit the asynchronous and concurrent communication from CPU to MICs and perform the bidirectional communication in every CPU-MIC pair at the same time.

The thread affinity of CPUs can be a performance bottleneck, especially when using complex offloading mechanism in the algorithm. We find that without explicitly setting the thread affinity of CPUs, the performance of our algorithm will degrade and jitter largely. As the table I shown, the thread affinity does have a strong influence on the performance. We believe that the asynchronous offloads and data transfers might lead to frequent and unexpected thread switching when occupying all the CPU cores during stencil computations for better utilization.

TABLE I
PERFORMANCE COMPARISON OF SETTING THE THREAD AFFINITY.

process number	affinity	max(s)	min(s)	average(s)
$2 \times 2 \times 1$	No affinity	0.4299	0.1775	0.2718
$2 \times 2 \times 1$	Compact	0.1550	0.1523	0.1538
$4 \times 4 \times 2$	No affinity	0.3797	0.1779	0.2623
$4 \times 4 \times 2$	Compact	0.1585	0.1561	0.1567

C. Inter-node level optimizations

1) Communication optimization: We conduct experiments on tuning the performance of MPI communication based on the PETSc (Portable Extensible Toolkit for Scientific computation) library. We use a pair of PETSc APIs, `DMGlobaltoLocalBegin` and `DMGlobaltoLocalEnd`, to prepare the data for communication and to perform the neighborhood communications.

It is found that the performance of data packing and unpacking in the two APIs is more expensive than expected. To solve this problem, multi-threading is introduced in the implementations for data preparation, which greatly improves the performance of data packing and unpacking.

To investigate the performance of different MPI implementations for neighborhood communication on TH Express-2, non-blocking standard communication, non-blocking synchronous communication, non-blocking ready communication, one-side communication are implemented and evaluated as well as blocking all-to-all communication and non-blocking all-to-all communication. With careful benchmarking, we find that non-blocking standard communication is the optimal choice on TH Express-2. Moreover, we add `MPI_Testall` immediately after the non-blocking communications posted in `DMGlobaltoLocalBegin` to start the communications as early as possible. This simple method improves the communication performance by 5-10%.

There are several settings can control the communication behavior of TH Express-2. We choose two of them to improve the communication performance. One is `GLEX_BYPASS_SHM`, which should be set to 1 since there is only one MPI process per node and we do not need to use share memory for intra-node communications and pay for the overhead. The other is `GLEX_RDMA_WRITE_ONLY`, which should be set to 0. It is set to use RDMA read instead of RDMA write as the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2366754, IEEE Transactions on Computers

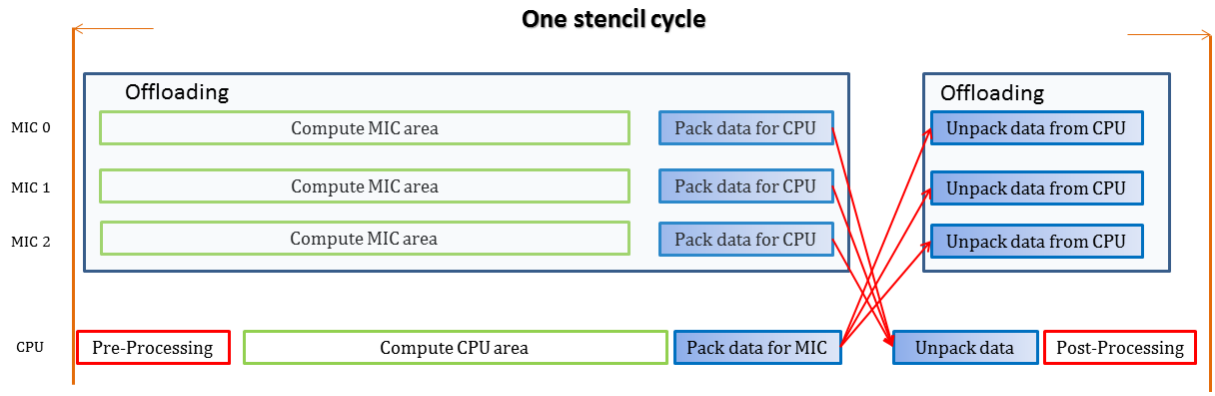

Fig. 8. An asynchronous and concurrent data transfer algorithm between CPUs and multi-accelerators

default RDMA protocol and will improve the performance by simplifying the acknowledge message in RDMA write.

As mentioned in Section. IV, each frame of Tianhe-2 has 32 tightly connected nodes. To better utilize the network bandwidth of TH Express-2, we tried to use the Hilbert space-filling curve to map the processes to physical nodes. However the process mapping did not bring us obvious performance improvement on Tianhe-2, which is still under investigation. One possible reason is that the effect of process mapping on the fat tree network is less critical than that of torus- or cube-like networks such as that of Blue Gene/Q and K computer, especially when we only assign one MPI process per node. The second reason is that the communication-computation overlapping algorithm releases the burden of network in some extents. The third is that we are not able make full use of the locality characteristic of the mapping based on the space-filling curve since it is not easy to get adequate working frames during the early-use stage of Tianhe-2.

2) Detecting and resolving silent errors at large scale: There are two kinds of system errors we have taken much effort to resolve during the large scale experiments on Tianhe-2. We call these errors "silent errors" since they could not be found by the checking programs deployed on Tianhe-2. With these silent errors, we didn't catch any warning or error information at all during simulation but finally got the wrong results.

One silent error is offloading mismatch for MIC. There are three MICs in each computing node of Tianhe-2. One of the MIC cards might be offline occasionally during operation without notice, which leads to two different offloads in our program sent to the same MIC card even if we have appointed different MIC card for different offloads. In this situation, the program will not fail, but the simulation will get the wrong results since the two offload kernels on the same card overwrite the arrays and variables with the same name. Our current solution to detect the error is to check the device name of each offload at the preload stage and every several time-steps. At the same time, we have to make sure the kernels on different MIC cards will access different arrays and variables with different names. We think the best way to solve this problem is to enhance the functionality of MPSS by isolating the different offloads and dealing with the memory oversubscription, which will keep the simplicity of user programming.

The other silent error is PCI-E error. We ever found that one of the MIC card in some computing node got `NaN` (Not-a-Number) results during simulation, and the so-called problematic MIC card were mostly the card connected to the network adapter via a same PCI-E bridge. Further diagnosis shows that it is not due to the error or offline of MIC card. There was a PCI-E connection error since the point-to-point communication performance from or to this node was only around several MB/s. Our solution to detect the error is to check the results of different devices every several time steps to identify the problematic node.

The above experiences indicate that it may be necessary to investigate more efforts in fault-tolerant heterogeneous computing from both application level and system level.

VII. Validation and Performance

Several benchmark test cases are proposed and studied to validate a regional mesoscale model. Among them, we select the baroclinic instability test in a 3D channel [40]. The test is initiated by adding a confined perturbation in the zonal wind field to a geostrophically balanced background flow. The setup in the test resembles in a great way the famous spherical baroclinic wave experiment. The test is useful to examine the correct response a numerical scheme produces to interact with the unbalanced trigger. Some other important tests, have also been done to further verify the model, but we omit the details here for brevity.

A. Numerical Validation

The size of the 3D channel in the baroclinic instability test is $40,000$ km $\times$ $6,000$ km $\times$ 30 km. We use a relatively small mesh with a horizontal resolution of 100 km and a vertical resolution of 1 km to verify the model and compare with reference results. A contour plot of the 500 m level pressure distribution at day 8 is presented in Fig. 9. It is observed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2366754, IEEE Transactions on Computers

from the figure that distinct low and high pressure regions are generated with sharp fronts, which agrees well with the results in [40].

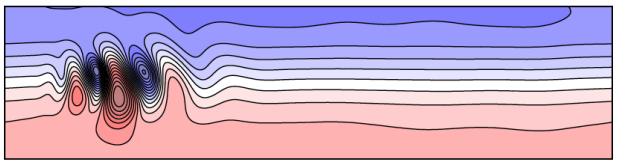

Fig. 9. The 500 m level pressure distribution of the atmosphere at day 8 for the baroclinic instability test in a 3D channel.

B. Performance results on Tianhe-2

1) The effect of model reformulation: To investigate the effectiveness of the model reformulation to avoid the `pow` operations, we measure the performance with and without using the reformulated model and show the time-to-solution of one time-step simulation as well as the speedup in Fig. 10. In the tests, we use the CPU-only code and perform the weak scaling tests using 1 to 128 computing nodes in Tianhe-2. Each node owns a same mesh size of $260 \times 240 \times 228$. In the tests, we can get average 55% performance improvement when using the proposed algorithm on CPU. It is worth noting that the performance improvement can be substantial when using hybrid code.

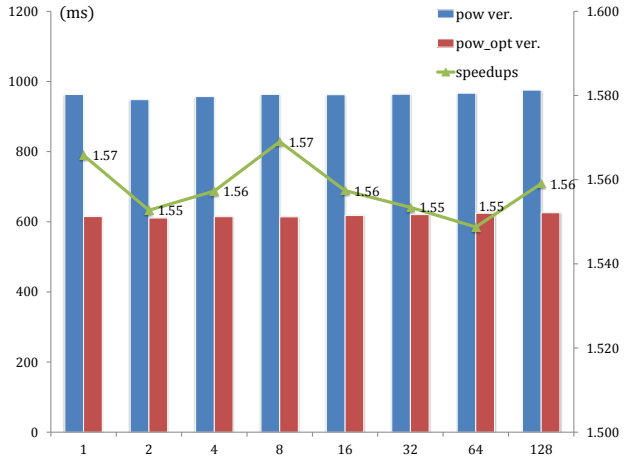

Fig. 10. Performance improvement by using the reformulated model to avoid computing `pow`. It is evaluated with the CPU-only version. Results shown is of weak scaling tests from 1 to 128 computing nodes on Tianhe-2. A same $260 \times 240 \times 228$ mesh is assigned to each node.

2) Many-core acceleration: For evaluating the many-core acceleration, we first compare the performance of the hybrid code using adaptive load balance strategy between CPUs and accelerators with that of the same code in which CPU only takes charge of the computation of boundary system. In the tests we only use one computing node and fix the mesh size to be $260 \times 240 \times 228$. It can be found in Fig. 11 that the load balance scheme can get about 41.05%, 7.90% and 3.08% performance improvement with one, two, and three MICs together with two CPUs. The red line indicates the percentage of meshpoint processed by CPUs. As shown in Fig. 11 the CPU side can compute 14.15% of the workload to make full use of the whole computing node of Tianhe-2. According to the one MIC results, we can find that the MIC code and the CPU code can get comparable performance. However, the performance improvement of adaptive load balance will decrease obviously when using more MICs. We believe that it is because the computation time of boundary system dominates the time of CPU side with the increase of MIC number, and the computation of boundary system can not get as high efficiency as the computation of internal part because of irregular memory access pattern.

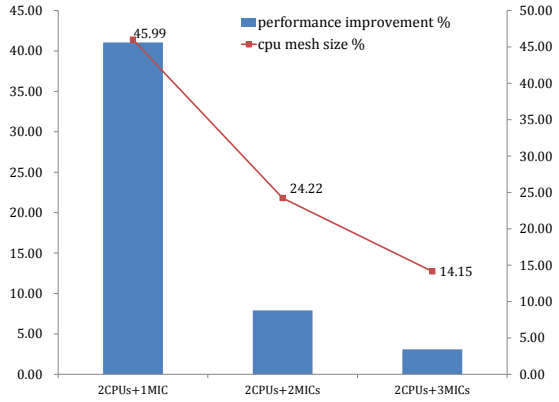

Fig. 11. The performance improvement and CPU workload in percentage with one, two, and three MICs together with 2 CPUs on one computing node of Tianhe-2. The performance results are evaluated with the hybrid code and a fixed mesh size of $260 \times 240 \times 228$.

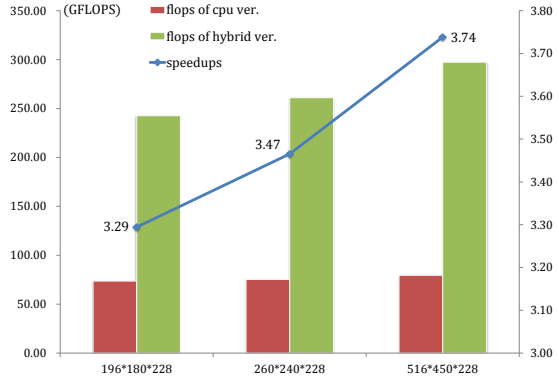

Fig. 12. Speedup and aggregated performance of the hybrid code w.r.t different mesh size in each computing node.

We further compare the performance of the hybrid code mentioned above with the performance of tuned multi-core code. The performance results by using three MICs and two CPUs in each computing node are described as the speedups (compared to the tuned CPU code) and the aggregated flops, shown in Fig. 12. In the tests we only use one computing node and fix the mesh size to be $196 \times 180 \times 228$, $260 \times 240 \times 228$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2014.2366754, IEEE Transactions on Computers

and $516 \times 450 \times 228$, respectively. When using more MICs, the aggregated performance increases proportionally as well as the speedups. The aggregated flops of the three meshes using the entire node are 242.60 Gflops, 260.98 Gflops, and 297.33 Gflops respectively. The speedups with the use of three MIC cards are 3.29x, 3.47x, and 3.74x for different mesh size.

The performance results shown above prove the effectiveness and feasibility of our hybrid algorithm and implementation.

3) Weak scaling results: Based on the observations made above, we fix the mesh size of each computing node to be $196 \times 180 \times 228$, $260 \times 240 \times 228$ and $516 \times 450 \times 228$ and gradually increase the number of computing nodes from 64 to $6,144$. In the largest run of the tests, we solve about 53.8 billion mesh elements and 269 billion unknowns using about 1.2 million cores. The results are summarized in Fig. 13, where the averaged compute time per time step is plotted with respect to the number of computing nodes as well as the aggregated flops. We observe from the figure that the compute time increases from 458.71 ms to 476.86 ms, leading to a parallel efficiency of about 96% for the largest runs. Moreover, we can get 1.45 Pflops, 1.63 Pflops, and 1.74 Pflops with different mesh size scaling to 6,144 computing nodes.

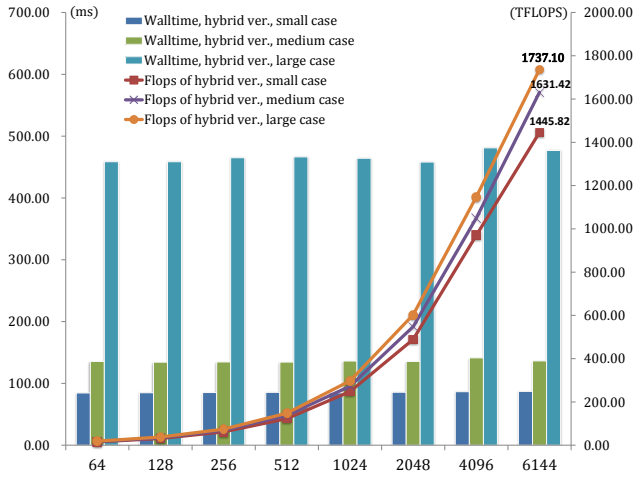

Fig. 13. Weak scaling results on Tianhe-2.

These suggest that our computation-communication overlapping strategy is successful. Most overhead caused by the increased amount of communication is hidden and a nearly ideal weak scaling curve is obtained.

4) Strong scaling results: In the strong scaling tests, we fix the total problem size to be largest run used previously and increase the number of computing nodes from 1,024 nodes to 6,144 nodes. The test results are provided in Fig. 14. From the figure, we observe that the parallel efficiency decreases to 88.0% when using 2,048 nodes and further to 73% when using 6,144 nodes. The observed degradation of the strong scaling efficiency is acceptable because the working load is too small to scale at a higher node count. Further performance improvement is still under investigation.

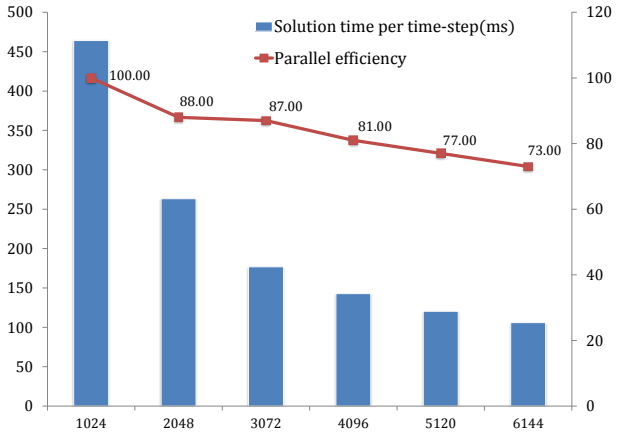

Fig. 14. Strong scaling results on Tianhe-2.

VIII. Concluding Remarks

Recent scientific demands are pushing the global atmospheric model towards an ultra-high resolution within 1km and the capability to resolve the clouds. Currently, most existing global climate models are still struggling with the poor scalability and the inability to use many-core accelerators, such as GPUs or MICs in many dominant heterogeneous supercomputers. To solve these challenges and to enable cloud-resolving atmospheric simulation, we design and implement an ultra-scalable algorithm to accelerate a 3D atmospheric model on the CPU-MIC hybrid system of Tianhe-2. We extend our previous work on a highly-scalable 2D shallow water model to the mesoscale fully compressible 3D Euler model. To achieve the high performance, we reformulate the model to avoid long-latency operations and reorganize both the computation pattern and data layout to enable all possible vectorizations and locality. We also carefully choose domain decomposition algorithms on both inter-node and intra-node levels. Performance is tuned among the CPU cores and the multi-MICs as well as the communication across nodes, so as to ensure perfect overlapping of computation and communication, and to achieve best utilization of various computation resources within hybrid system. Our final design can scale up to 6,144 nodes with a nearly ideal weak scaling efficiency, and achieve over 8% of the peak performance. This ultra-scalable hybrid algorithm may provide a guidance for our further development of a large-scale cloud-resolving atmospheric model on increasingly dominated heterogeneous supercomputers.

References

[1] National Research Council of the National Academies, *A National Strategy for Advancing Climate Modeling*. The National Academies Press, 2012.

[2] W. M. Putman, "Development of the Finite-Volume Dynamical Core On Cubed-Sphere", Ph.D. dissertation, The Florida State University, May 2007.

[3] J. M. Dennis, M. Vertenstein, P. H. Worley, A. A, Mirin, A. P, Craig, R. Jacob, and S. Mickelson, "Computational performance of ultra-high-resolution capability in the Community Earth System model", *Int'l J. High Perf. Comput. Appl.*, vol. 26, pp. 5–16, 2012.

[4] 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 *Proc. PPoPP'13*, ACM, New York, NY, USA, 2013, pp. 1–12.

[5] Wei Xue, Chao Yang, Haohuan Fu, Xinliang Wang, Yangtong Xu, Lin Gan, Yuotng Lu, Xiaoqian Zhu, "Enabling and scaling a global shallow-water atmospheric model on Tianhe-2", In *Proc. IPDPS'14*, IEEE Computer Society, Phoenix, Arizona, USA, 2014, to appear.

[6] J. Michalakes, J. Hacker, R. Loft, M. O. McCracken, A. Snavely, N. J. Wright, T. Spelce, B. Gorda, and R. Walkup, "WRF nature run", In *Proc. SC'07*. ACM, New York, NY, USA, 2007, pp. 59:1-59:6.

[7] P. Johnsen, M. Straka, M. Shapiro, A. Norton, and T. Galarneau, "Petascale WRF simulation of hurricane Sandy deployment of NCSA's cray XE6 Blue Waters", In *Proc. SC'13*, ACM, New York, NY, USA, 2013, pp. 63:1–63:7.

[8] J. Michalakes and M. Vachharajani, "GPU acceleration of numerical weather prediction", in *Proc. IPDPS'08*, IEEE, 2008, pp. 1–7.

[9] M. W. Govett, J. Middlecoff, and T. Henderson, "Running the NIM next-generation weather model on GPUs", in *Proc. CCGrid'10*, IEEE, 2010, pp. 792–796.

[10] W. Putman, "Graphics Processing Unit (GPU) Acceleration of the Goddard Earth Observing System Atmospheric Model", *NASA Technical Report*, Goddard Space Flight Center, 2011.

[11] I. Carpenter, R. K. Archibald, K. J. Evans, J. Larkin, P. Micikevicius, M. Norman, J. Rosinski, J. Schwarzmeier, and M. A. Taylor, "Progress towards accelerating HOMME on hybrid multi-core systems", *Int'l J. High Perf. Comput. Appl.*, vol. 27, pp. 335–347.

[12] H. Xiao, J. Sun, X. Bian, and Z. Dai, "GPU acceleration of the WSM6 cloud microphysics scheme in GRAPES model", *Computers & Geosciences*, vol. 59, 2013, pp. 156–162.

[13] T. Shimokawabe, T. Aoki, C. Muroi, J. Ishida, K. Kawano, T. Endo, A. Nukada, N. Maruyama, and S. Matsuoka, "An 80-fold speedup, 15.0 TFlops full GPU acceleration of non-hydrostatic weather model ASUCA production code", in *Proc. SC'10*. Washington, DC, USA: IEEE Computer Society, 2010, pp. 1–11.

[14] 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, Proc. ICCS'11*, vol. 4, 2011, pp. 1535–1544.

[15] I. Demeshko, N. Maruyama, H. Tomita, and S. Matsuoka, "Multi-GPU Implementation of the NICAM Atmospheric Model", In *Proc. HeteroPar'12*. Rhodes Island, Greece, 2012, pp. 175–184.

[16] K. Datta, M. Murphy, V. Volkov, S. Williams, J. Carter, L. Oliker, D. Patterson, J. Shalf, and K. Yelick, "Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures", in *Proc. SC'08)*. Piscataway, NJ, USA: IEEE Press, 2008, pp. 4:1–4:12.

[17] S. Kamil, K. Datta, S. Williams, L. Oliker, J. Shalf, and K. Yelick, "Implicit and explicit optimizations for stencil computations", In *Proc. MSPC'06*, 2006, pp. 51–60.

[18] M. Frigo and V. Strumpen, "The memory behavior of cache oblivious stencil computations", *J. Supercomput.*, vol. 39, 2007, pp. 93–112.

[19] A. Nguyen, N. Satish, J. Chhugani, C. Kim, and P. Dubey, "3.5-D blocking optimization for stencil computations on modern CPUs and GPUs", In *Proc. SC'10*, IEEE Press, 2010, pp.1–13.

[20] M. Christen, O. Schenk, and Y. Cui, "Patus for convenient high-performance stencils: evaluation in earthquake simulations", In *Proc. SC'12*, IEEE Computer Society Press, Los Alamitos, CA, USA, 2012, pp. 11:1–11:10.

[21] Szustak L, Rojek K, Wyrzykowski R, et al. "Toward efficient distribution of MPDATA stencil computation on Intel MIC architecture", In *Proce. HiStencils'14*, 2014, pp. 51–56.

[22] J. Park, P. Tang, M. Smelyanskiy, D. Kim, and T. Benson, "Efficient backprojection-based synthetic aperture radar computation with many-core processors", In *Proc. SC'12*, IEEE Computer Society Press, Los Alamitos, CA, USA, 2012, pp. 28:1–11.

[23] A. Heinecke, K. Vaidyanathan, M. Smelyanskiy, A. Kobotov, R. Dubtsov, G. Henry, A. G. Shet, G. Chrysos, and P. Dubey, "Design and Implementation of the Linpack Benchmark for Single and Multi-node Systems Based on Intel Xeon Phi Coprocessor", In *Proce. IPDPS'13*, IEEE Computer Society, Washington, DC, USA, 2013.

[24] X. Liu, M. Smelyanskiy, E. Chow, and P. Dubey, "Efficient sparse matrix-vector multiplication on x86-based many-core processors," In *Proc. ICS '13*, ACM, New York, NY, USA, 2013, pp. 273–282.

[25] D. Schmidl, T. Cramer, S. Wienke, C. Terboven, and M. S. Müller, "Assessing the Performance of OpenMP Programs on the Intel Xeon Phi", In *Proc. Euro-Par'13*, Aachen, Germany, 2013.

[26] S. Ramos, and T. Hoefler, "Modeling communication in cache-coherent SMP systems: a case-study with Xeon Phi", In *Proc. HPDC'13*. ACM, New York, NY, USA, 2013, pp. 97–108.

[27] S. J. Pennycook, C. J. Hughes, M. Smelyanskiy, and S. A. Jarvis, "Exploring SIMD for molecular dynamics, using Intel Xeon processors and Intel Xeon Phi coprocessors", In *Proc. IPDPS'13*, IEEE Computer Society, Washington, DC, USA, 2013.

[28] S. Cadambi, G. Coviello, C.-H. Li, R. Phull, K. Rao, M. Sankaradass, and S. Chakradhar. "COSMIC: middleware for high performance and reliable multiprocessing on Xeon Phi coprocessors", In *Proce. HPDC'13*. ACM, New York, NY, USA, 2013, pp. 215–226.

[29] P. H. Lauritzen, C. Jablonowski, M. A. Taylor, and R. D. Nair, Eds., *Numerical Techniques for Global Atmospheric Models*. Springer, 2011.

[30] Y. Ogura and A. Phillips, "Scale analysis of deep and shallow convection in the atmosphere", *Mon. Wea. Rev.*, vol. 19, 1962, pp. 173–179.

[31] C. Yang and X.-C. Cai, "A scalable fully implicit compressible euler solver for mesoscale nonhydrostatic simulation of atmospheric flows", *SIAM J. Sci. Comput.*, 2014, to appear.

[32] B. Hejazialhosseini, D. Rossinelli, C. Conti, and P. Koumoutsakos, "High throughput software for direct numerical simulations of compressible two-phase flows", In *Proc. SC'12*. IEEE Computer Society Press, Los Alamitos, CA, USA, 2012, pp. 16:1–16:12.

[33] D. Rossinelli, B. Hejazialhosseini, P. Hadjidoukas, C. Bekas, A. Curioni, A. Bertsch, S. Futral, S. J. Schmidt, N. A. Adams, and P. Koumoutsakos, "11 PFLOP/s simulations of cloud cavitation collapse", In *Proc. SC'13*, ACM, New York, NY, USA, 2013, pp. 3:1–3:13.

[34] T. Shimokawabe, T. Aoki, T. Takaki, T. Endo, A. Yamanaka, N. Maruyama, A. Nukada, and S. Matsuoka, "Peta-scale phase-field simulation for dendritic solidification on the TSUBAME 2.0 supercomputer", in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11)*. New York, NY, USA: ACM, 2011, pp. 3:1–3:11.

[35] M. Crovella, R. Bianchini, T. LeBlanc, E. Markatos, R. Wisniewski, "Using communication-to-computation ratio in parallel program design and performance prediction", In *Proc. IPDPS'92*, IEEE Computer Society, Arlington, TX, USA, 1992.

[36] V. Tarokh, H. Jafarkhani, "On the computation and reduction of the peak-to-average power ratio in multicarrier communications", In IEEE Transactions on Communications, IEEE Computer Society, vol. 48, 2000, pp. 37–44.

[37] F. Zhang, M. Sakr, "Cluster-size Scaling and MapReduce Execution Times", In *Proceedings of The International Conference on Cloud Computing and Science*, IEEE Computer Society, Bristol, UK, 2013.

[38] S.U.Khan, A.Y.Zomaya, L. Wang, "Scalable Computing and Communications: Theory and Practice", Wiley-IEEE Computer Society Press, New Jersey, USA, 2013.

[39] M. Satoh, "Conservative scheme for the compressible nonhydrostatic models with the horizontally explicit and vertically implicit time integration scheme", *Mon. Wea. Rev.*, vol. 130, 2002, pp. 1227–1244.

[40] P. Ullrich and C. Jablonowski, "Operator-split RungeKuttaRosenbrock methods for nonhydrostatic atmospheric models", *Mon. Wea. Rev.*, vol. 140, 2012, pp. 1257–1284.