Simulating the Wenchuan Earthquake with Accurate Surface Topography on Sunway TaihuLight

Bingwei Chen*‡, Haohuan Fu†‡‖, Yanwen Wei†‡, Conghui He*‡, Wenqiang Zhang¶, Yuxuan Li*‡,
Wubin Wan‡, Wei Zhang‡, Lin Gan*‡‖, Wei Zhang§, Zhenguo Zhang§, Guangwen Yang*†‡, Xiaofei Chen§
* Department of Computer Science and Technology, Tsinghua University
† Ministry of Education Key Lab. for Earth System Modeling, and
Department of Earth System Science, Tsinghua University
‡ National Supercomputing Center in Wuxi
§ Department of Earth and Space Sciences, Southern University of Science and Technology
¶ School of Earth and Space Sciences, University of Science and Technology of China
‖ Laboratory for Regional Oceanography and Numerical Modeling,
Qingdao National Laboratory for Marine Science and Technology
{chenbwei2012,weiyw17,heconghui,zhangwq.zhang,wanwuko,lin.gan27}@gmail.com,jflfy2255@163.com
{zhangwei,zhangzg,chenxf}@sustc.edu.cn,haohuan@tsinghua.edu.cn,ygw@mail.tsinghua.edu.cn,zhangwei@mail.nsccwx.cn

***Abstract*—This paper reports our efforts on performing a 50-m resolution earthquake simulation of the Wenchuan Earthquake (Ms 8.0, China) on Sunway TaihuLight. To accurately capture the surface topography, we adopt a curvilinear grid finite-difference method with a traction image free surface implementation and redesign the algorithm to reduce memory access costs for heterogeneous many-core architectures. We then derive a performance model of our algorithm to guide and drive the further optimization and tuning of various parameters using a genetic algorithm. A data layout transformation is also proposed to improve the direct memory access (DMA) efficiency further. Our efforts improve the simulation efficiency from 0.05% to 7.6%, with a sustained performance of 9.07 Pflops using the entire machine of the Sunway TaihuLight (over 10 million cores), and a large-scale simulation of the Wenchuan earthquake with accurate surface topography and improved coda wave effects.**

***Index Terms*—Sunway TaihuLight, computational seismology, earthquake ground motions, parallel scalability, accurate surface topography**

I. Introduction

Li, Bai, the well-known 'Immortal Poet' in China's history (701-762 AD), wrote in one of his poems, 'the steep road in Shu, even more perilous than the road to the Green Heavens'. Shu is the traditional name of Sichuan province, which sits inside a large basin surrounded by mountains that gradually grow from the basin to the Tibetan Plateau. While such a unique geological environment provides some of the marvelous scenery frequently depicted in masterpiece poets and paintings in China, it also brings one of the most complex domains for seismologists to investigate and to simulate. Figure 1 shows the region of the Wenchuan earthquake. The Wenchuan earthquake (Ms 8.0, 2008) occurred in one of the critical points along the fault zone [1]. The damage caused by the ground motion, the hill slide, as well as the barrier lake formed after the earthquake, led to the losses of over 69,000 lives, and over 100 billion USD.

For seismologists who work on earthquake problems, numerical simulation is an essential tool to investigate underground geological structures and earthquake mechanisms.

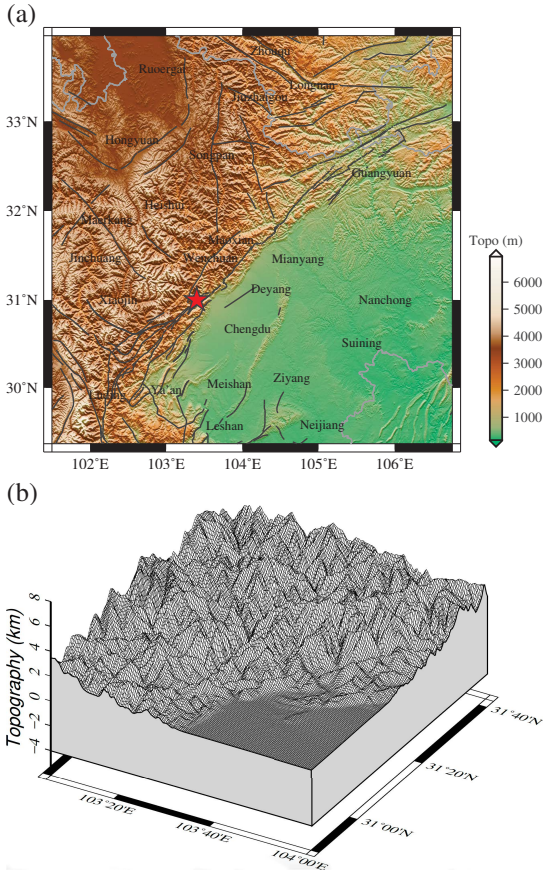

Fig. 1: (a) The simulation region of the Wenchuan earthquake, the red star indicates epicenter, and the black lines indicate the faults in this area. (b) The grid discretization of a small chunk of the 3D simulation region. The vertical transformation is used to generate the grid.

However, the complicated scenario of the Wenchuan earthquake brings a number of tough challenges when compared to that of the Tangshan earthquake that was already simulated

SC18, November 11-16, 2018, Dallas, Texas, USA
978-1-5386-8384-2/18/$31.00

with high fidelity on Sunway TaihuLight [2]. To achieve an accurate simulation of both the basin and the mountainous regions, we need to cover a 3D domain $\times$ 640 km $\times$ 640 km $\times$ 100 km, which is roughly eight times the size of the domain simulated in the Tangshan earthquake. Moreover, the complex geology features of this region (an elevation range of at least 7,000 meters) make it extremely important to accurately describe the surface topography in order to capture the local amplification effects and margin effects of basins.

To tackle these challenges, we design and develop a Sunway-based software platform to perform high-resolution simulations of the Wenchuan earthquake (Ms 8.0, China) with accurate surface topography. Major innovations include:

- We build our simulation program − Seismic Wave Simulation with surface Topography (SWST), which uses a curvilinear grid finite difference method (FDM) to achieve accurate and stable free-surface boundary condition, and uses the traction image method to describe accurate surface topography. We also make algorithmic adjustments to alleviate the memory bandwidth constraint of this application.
- To derive the optimal configuration of the Wenchuan Earthquake simulation with any given settings of resource and problem size, we build a performance model to identify the major factors that determine the performance of the applications, as well as the tools to facilitate the tuning process guided by the model.
- To further improve the efficiency of direct memory access (DMA) operations, we propose a data layout transformation scheme to maximize the continuity of the data items partitioned to each computing processing element (CPE) thread.

While the simulation of the Wenchuan earthquake already brings the challenge of the significantly large problem space (640 km $\times$ 640 km $\times$ 100 km), our numerical method for capturing the complex surface topography also increases the number of unknowns per grid point. The SWST software requires 53 unknowns for each grid point, which is roughly twice the number required by the non-topography algorithm, such as the AWP-ODC [3], [4], [5]. Therefore, when compared to the Tangshan earthquake simulation on Sunway TaihuLight in 2017, our target scenario requires roughly 16 times more memory space and twice the bandwidth to achieve the same performance for the same simulation resolution.

Combining both the algorithmic and architectural optimizations based on the memory and bandwidth features, we are able to improve the computing efficiency of SWST from 0.05% to 7.6%. When using the entire machine of the Sunway TaihuLight (over 10 million cores), SWST provides a sustained performance of 9.07 Pflops, and is capable of resolving a 10000 $\times$ 10000 $\times$ 2000 mesh, with over 110.5 trillion unknowns in the seismic wave equation. Using SWST, we can perform the Wenchuan Earthquake (Ms 8.0, 2008) simulation with complex surface topography at 50-meter resolution for a region of 640 km $\times$ 640 km $\times$ 100 km.

II. Background

A. Existing Efforts

In recent decades, many different methods have been proposed and applied in large scale earthquake simulation:

1) Spectral element method (SEM): SEM is a high order finite element method (FEM) [6], [7]. In 2003, American and Japanese scientists used 5.5 billion grid points on 1944 cores of the Earth Simulator, achieving a performance of 5 Teraflops. This work was later evolved into the SPECFEM3D package that further integrated more useful features [8], [9], [10]. In 2008, by using 28,000 processors on Jaguar, SPECFEM3D obtained a sustained performance of 35.7 Tflops. In 2012, another effort [11] that ported SPECFEM3D to a large GPU cluster with 896 GPUs of Cray XK6 employed 8 billion grid points and reached 135 Tflops.

2) DG-FEM: In 2014, German scientists created a simulation software called SeisSol [12], [13]. With 191 million tetrahedrons running on 1,597,440 cores of Tianhe-2, the 1992 Landers (M7.2) Earthquake was simulated successfully with a performance of 8.6 Pflops. In 2017, EDGE was created [14], with 341 million tetrahedrons, a performance of 10.4 Pflops was achieved. In the same year, the 2004 Sumatra Megathrust Earthquake was successfully simulated [15] using 221 million tetrahedrons, providing a sustained performance of 1.59 Pflops on Haswell processors. DG-FEM converts the numerical problem into compute bound dense matrix operations, but the high order method also increases the memory demand and the computational complexity, limiting the largest size of solvable problem and time-to-solution.

3) Implicit FEM: In 2014, scientists in Japan built the GAMERA system [16], [17], [18], which provides a sustained performance of 1.97 Pflops when processing 27 billion DOFs with 663,552 cores on the K computer. They built the GOJIRA system [19] in the following year, processing 1.08 trillion DOFs with the entire K computer, achieving a performance of 1.97 Pflops. Implicit FEM can gain larger time increments by adding some extra computation in each time step. However, the implicit FEM also increase memory requirements and computational costs, making it difficult to tackle a larger simulation area (hundreds of kilometers).

4) Staggered FDM: Staggered FDM is popular due to its straightforward and parallel-friendly implementation. The Southern California Earthquake Center (SCEC) developed AWP-ODC, which tackled 859 billion grid points in 2013, with a performance of 2.3 Pflops [16]. In 2017, researchers from Tsinghua University ported and redesigned AWP-ODC for Sunway TaihuLight, achieved a performance of 15.2 Pflops when simulating the Tangshan earthquake using 3.99 trillion grid points [2]. A balance between the computational cost and the simulation capability is maintained for large-scale simulation using AWP-ODC. However, for the numerical simulation of the Wenchuan Earthquake, which involves an altitude variation in the range of 7 kilometers, it is too costly, if not completely impossible, to describe the complex surface using a staggered grid discretization.

5) Summary: After considering all these different factors, we decided to adopt the curvilinear grid FDM, to achieve both an accurate description of the complex topography, which is a crucial factor for simulating the Wenchuan earthquake accurately, and efficient utilization of the heterogeneous many-core platforms.

B. Main Challenge of the architecture

Sunway TaihuLight supercomputer was the first system in the world to have a peak performance of over 100 Pflops. The computing power of TaihuLight comes from China's custom SW26010 CPU [20], [21]. Fig 2 shows the architecture of SW26010. One CPU chip contains four core groups (CG), each of which includes one management processing unit (MPE), one computing processing element cluster with 8 by 8 computing processing elements (CPE), and one memory controller. In general, the MPE is suitable for controlling while the CPE is suitable for computing. These four CGs are connected via the network on chip (NoC). With 260 processing elements in total, one SW26010 can provide a peak performance of over 3 Tflops. With 40,960 CPUs (10,140,000 cores), the TaihuLight supercomputer can provide a peak performance of 125 Pflops.

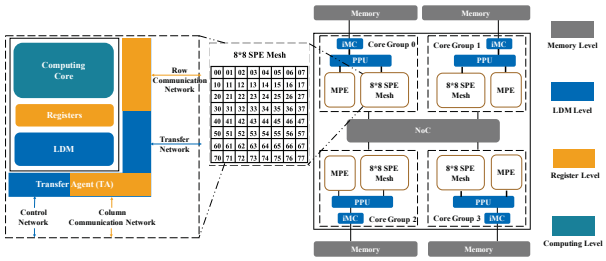

Fig. 2: The Architecture of SW26010 [21]

As for the memory hierarchy, each CG includes three levels. On the first level, all CPEs share the 8-GB DDR3 memory through the DDR3 interface. On the second level, each CPE has an independent 64-KB Scratch Pad Memory (SPM), which can be used as a user-controlled cache. The peak bandwidth of DMA operations to and from the DDR3 memory is roughly 30GB/s per CG. The effective bandwidth depends on the block size of each DMA transaction. The third level consists of 32 256-bit registers.

In summary, the significant challenges of the architecture are mostly because of the memory and bandwidth features. With the increasing requirements in memory capacity and bandwidth of modern scientific computing applications, each of the TaihuLight nodes only provides 32 GB memory and around 120 GB/s bandwidth, which is limited compared with other many-core architectures that usually contain 300GB/s to 500 GB/s of bandwidth. Furthermore, each CPE only provides a 64-KB scratchpad buffer, so the design of a user-controlled cache scheme has to be done.

III. A Memory-Oriented Redesign of the Curvilinear FDM for the Sunway Architecture

A. Collocated FDM with Curvilinear Grid

In 3-D inhomogeneous isotropic elastic media, the propagation of elastic waves is governed by the elastodynamic equations (including the momentum equation and the generalized stress-strain relationship), which can be written as a first order velocity-stress equations, i.e.,

$$\rho v_{i,t} = \sigma_{ij,j} + f_i, \tag{1}$$

$$\sigma_{ij,t} = \lambda v_{k,k}\delta_{ij} + \mu(v_{i,j} + v_{j,i}), \tag{2}$$

where λ and μ are the Lamé parameters in isotropic media.

To write the equations as a compact matrix form (ignoring the source term), we have:

$$\frac{\partial \mathbf{W}}{\partial t} = \mathbf{A}\frac{\partial \mathbf{W}}{\partial x} + \mathbf{B}\frac{\partial \mathbf{W}}{\partial y} + \mathbf{C}\frac{\partial \mathbf{W}}{\partial z}. \tag{3}$$

The unknowns that need to be solved are the wavefield, which includes both the velocity field, and the stress field:

$$\mathbf{W} = (v_x, v_y, v_z, \sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{xz}, \sigma_{yz})^T. \tag{4}$$

In this paper, we use (x, y, z) to denote the coordinates in physical space, and (ξ, η, ζ) to denote coordinates in computational space (as shown in Fig 3(1)). Using a curvilinear coordinate transformation, the irregular physical space can be transformed into a regular computational space:

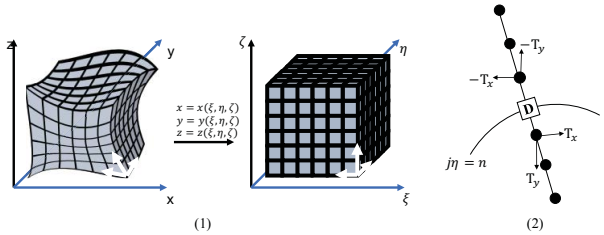

Fig. 3: (1) Curvilinear grid transformation; (2) Traction image method.

$$x = x(\xi, \eta, \zeta), y = y(\xi, \eta, \zeta), z = z(\xi, \eta, \zeta). \tag{5}$$

Using the finite difference method to calculate the coordinate transformation coefficients, we can then obtain covariant and contravariant basis vectors in the curvilinear coordinate system. Converting the derivatives of the physical space into the derivatives of the computational space through the chain rule [22], the compact matrix form of elastodynamic equations can be written as:

$$\frac{\partial \mathbf{W}}{\partial t} = \tilde{\mathbf{A}}\frac{\partial \mathbf{W}}{\partial \xi} + \tilde{\mathbf{B}}\frac{\partial \mathbf{W}}{\partial \eta} + \tilde{\mathbf{C}}\frac{\partial \mathbf{W}}{\partial \zeta}. \tag{6}$$

We compute the derivatives in the computational domain (ξ, η, ζ), but solve the wavefield in the physical domain (x, y, z). We adopt a collocated-grid scheme instead of the staggered-grid scheme for the convenience of coordinate transformation and boundary conditions. The spatial difference operator is split into the forward and backward one-sided difference operators, which are alternately used in Runge-Kutta time marching schemes. The dispersion relation preserving (DRP) method is used to optimize the dispersion and dissipation error of the one-sided differences in the MacCormack-type schemes. This dispersion and dissipation optimization scheme is called the DRP/opt MacCormack scheme and has been used by Zhang & Chen [23], [24] to model 2-D and 3-D seismic waves in the presence of surface topography. If the three directional derivatives of DRP/opt MacCormack scheme are calculated

TABLE I: A summary of existing work of large-scale earthquake simulations on supercomputers. The numbers are obtained from the published papers. Unreported values are labeled as '–'. For the numercial method, FD refers to finite difference method, SEM refers to the spectral element method, and DG-FEM refers to the discontinuous Galerkin finite element method.

Work	Year	Machine	Arch	Scale	# grid points	Flops	Mem	Method	Surface Topography
SPECFEM3D	2012	Cray XK6	Fermi GPU	896 GPUs	8 billion	135 Tflops	3.5 T	SEM	Yes
EDGE	2017	Cori-II	Xeon Phi	612,000 cores	341 million tetrahedrons	10.4 Pflops	32 TB	DG-FEM	Yes
GOJIRA	2015	K Computer	SPARC64	663,552 cores	270 billion	1.97 Pflops	–	implicit FEM	Yes
AWP-ODC	2017	Sunway	SW26010	10,014,000 cores	3.99 trillion	15.2 Pflops	892 TB	FD	No
Our work	2018	Sunway	SW26010	10,014,000 cores	4.3 trillion	9.07 Pflops	902 TB	Collocated CG FD	Yes

by a forward operator in the ξ-axis, a backward operator in the η-axis, and a forward operator in the ζ-axis, respectively, then the 3D operator will be written as (take L^{FBF} as an example):

$$\hat{L}^{FBF}(\mathbf{W}) = \tilde{\mathbf{A}}L_{\xi}^{F}(\mathbf{W}) + \tilde{\mathbf{B}}L_{\eta}^{B}(\mathbf{W}) + \tilde{\mathbf{C}}L_{\zeta}^{F}(\mathbf{W}). \quad (7)$$

To avoid numerical bias, we mix the forward and backward difference operators for the ξ-,η-, ζ-derivatives, resulting in 8 possible pairs of the 3-D biased difference operators in the spatial domain, and create a cycle of 8 time steps to update the wavefield [22]. In the time domain of step n to $n+1$, the fourth-order Runge-Kutta scheme evaluates the wavefield at different intermediate times between nt and $(n+1)t$:

$$\mathbf{W}^{(1)} = \mathbf{W}^{n}, \quad (8)$$

$$\mathbf{W}^{(2)} = \mathbf{W}^{n} + \alpha_2 \Delta t \hat{L}^{FFF}(\mathbf{W}^{(1)}), \quad (9)$$

$$\mathbf{W}^{(3)} = \mathbf{W}^{n} + \alpha_3 \Delta t \hat{L}^{BBB}(\mathbf{W}^{(2)}), \quad (10)$$

$$\mathbf{W}^{(4)} = \mathbf{W}^{n} + \alpha_4 \Delta t \hat{L}^{FFF}(\mathbf{W}^{(3)}), \quad (11)$$

$$\mathbf{W}^{n+1} = \mathbf{W}^{n} + \Delta t[\beta_1 \hat{L}^{FFF}(\mathbf{W}^{(1)}) + \beta_2 \hat{L}^{BBB}(\mathbf{W}^{(2)}) + \beta_3 \hat{L}^{FFF}(\mathbf{W}^{(3)}) + \beta_4 \hat{L}^{BBB}(\mathbf{W}^{(4)})], \quad (12)$$

these forward and backward one-sided difference operators will be interchanged at the next step of the Runge-Kutta update.

The implementation of a free surface boundary condition is an essential issue in seismic wave simulations. For finite difference schemes with regular grid discretizations, the implementation of a free boundary condition can be directly manipulated in the wavefield components. For example, the stress image method directly mirrors the stress component within the finite difference stencil with respect to the free surface. However, in curvilinear grids, the traction image method is used to update the velocity field. As a result, we cannot directly apply the boundary condition on each wavefield component since traction is the combination of the stress components (as shown in Fig 3(2)):

$$T_x = (\zeta_{,x}\sigma_{xx} + \zeta_{,y}\sigma_{xy} + \zeta_{,z}\sigma_{xz})\frac{1}{|\Delta\zeta|} = 0, \quad (13)$$

$$T_y = (\zeta_{,x}\sigma_{xy} + \zeta_{,y}\sigma_{yy} + \zeta_{,z}\sigma_{yz})\frac{1}{|\Delta\zeta|} = 0, \quad (14)$$

$$T_z = (\zeta_{,x}\sigma_{xz} + \zeta_{,y}\sigma_{yz} + \zeta_{,z}\sigma_{zz})\frac{1}{|\Delta\zeta|} = 0, \quad (15)$$

$$T_x|_{k_0+n} = -T_x|_{k_0-n}, T_y|_{k_0+n} = -T_y|_{k_0-n}, \quad T_z|_{k_0+n} = -T_z|_{k_0-n}. \quad (16)$$

As each traction force component is a combination of three stress components, when updating the velocity field at the boundary, the momentum equation must be written in a conservative form to facilitate the application of the traction image method. In other regions with no boundary points, the normal form of the elastodynamic equations is used to update the wavefield.

B. Memory-Oriented Algorithm Redesign

To make the algorithm more suitable for the on-chip heterogeneity of the SW26010 processors, the algorithm is redesigned to make the best of the memory bandwidth.

Based on equations 8-12, we use mW to represent the wavefield of the last timestep, W to represent the wavefield of the current timestep, hW to represent the derivatives of the wavefield, and tW to represent the wavefield of the next timestep. As for the direction of the difference operator, we mark it as a boolean variable $Flag$. The algorithm in each time step is shown in Algorithm 1.

The first naive implementation is to design seven kernels including four for updating variables in space domain ($Calc\_DerivV$, $Calc\_DerivS$, $Tract\_Img$, Vel_Low), and three for updating variables in time domain (RK_begin, RK_inner, RK_end), as shown in Algorithm 1. In each CG, the MPE manages the boundary exchange and the CPEs process the computation for the seven kernels. However, such implementation has some drawbacks. First, different computational kernels in spatial domain usually result in different calculation areas that require different partition schemes to keep acceptable efficiencies. Second, each kernel needs to load W (variables stand for wave field) from memory to SPM due to the limited memory size, which leads to the repetitive consumption of the bandwidth. Third, the surface layer of only three points in the z-axis results in a small block size of DMA operations and low bandwidth utilization in two of the space domain calculating ($Tract\_Img$ and Vel_Low) kernels.

To resolve these issues above, by analyzing the dependencies of all variables among these kernels, we merge four spatial domain kernels into one named as $DRP\_MacCormack$. Afterward, we only load W one time at the beginning, and store

Algorithm 1 Runge Kunta Synthesis for Collocated FDM with Curvilinear Grid

RK_Init: $mW \leftarrow W$
DRP_MacCormack: $hW \leftarrow$
    Calc_DerivV($Flag, W.S$)
    Calc_DerivS($Flag, W.V$)
    Tract_Img($Flag, W.S$)
    Vel_Low($Flag, W.V$)
RK_begin: $W \leftarrow mW + \alpha * hW$
    $tW \leftarrow mW + \beta * hW$
Exchange: Exchange_Boundaries(W)
DRP_MacCormack: $hW \leftarrow$
    Calc_DerivV($Flag, W.S$)
    Calc_DerivS($Flag, W.V$)
    Tract_Img($Flag, W.S$)
    Vel_Low($Flag, W.V$)
RK_inner: $W \leftarrow mW + \alpha * hW$
    $tW += \beta * hW$
Exchange: Exchange_Boundaries(W)
DRP_MacCormack: $hW \leftarrow$
    Calc_DerivV($Flag, W.S$)
    Calc_DerivS($Flag, W.V$)
    Tract_Img($Flag, W.S$)
    Vel_Low($Flag, W.V$)
RK_inner: $W \leftarrow mW + \alpha * hW$
    $tW += \beta * hW$
Exchange: Exchange_Boundaries(W)
DRP_MacCormack: $hW \leftarrow$
    Calc_DerivV($Flag, W.S$)
    Calc_DerivS($Flag, W.V$)
    Tract_Img($Flag, W.S$)
    Vel_Low($Flag, W.V$)
RK_end: $W \leftarrow tW + \beta * hW$
Exchange: Exchange_Boundaries(W)

Algorithm 2 Runge Kunta Synthesis for Collocated FDM with Curvilinear Grid

$mW \leftarrow W$
$tW \leftarrow hW$
for $istep \in [0, 3]$ **do**
    $Flag$ = generateFlag($istep$)
    $hW \leftarrow$ MainCalculateKernel($Flag, istep, W$)
    swap_addr(hW, W)
    Exchange_Boundaries(W)
end for

hW one time at the end. However, after $DRP\_MacCormack$ finished, the following temporal domain kernel needs to load W, hW, and a number of other variables for the second time, i.e., consuming the memory bandwidth for the same data items for second time.

To further reduce the redundant memory access, we merge $DRP\_MacCormack$ and three temporal domain kernels into a new kernel by adjusting the calculation order. The new algorithm is shown in Algorithm 2 and Algorithm 3. The major modifications include: (1) exchange the order of calculating tW and W; (2) storing the final result in hW to resolve the dependencies of different grid points in a loop (we need to swap the address of W and hW afterward). As shown in Algorithm 3, the final algorithm only needs to load and store the arrays once.

For the matrices that need to reside in the SPM during the computation of specific kernels (such as the $matVx2Vz, matVy2Vz, matF2Vz$ matrices required for the Vel_Low kernel, which derive the vertical derivative indirectly by other-orient variables), we adopt an on-the-fly compute strategy, to further reduce the memory access counts by recomputing these matrices at the initialization stage.

Algorithm 3 MainCalculateKernel

for $(i, j, k) \in Whole\_Region$ **do**
    if $(i, j, k) \in Underground\_Region$ **then**
        $hW \leftarrow$ Calc_Deriv($Flag, W$)
    end if
    if $(i, j, k) \in Surface\_Region$ **then**
        $hW \leftarrow$ Tract_Img($Flag, W$)
    end if
    $tW += \beta * hW$
    if istep != 3 **then**
        $hW \leftarrow mW + \alpha * hW$
    else
        $hW \leftarrow tW$
    end if
end for

IV. Auto-Tuned Process/Thread Parallelization Schemes Driven by an Analytic Performance Model

A. Our General Parallelization Scheme

In order to scale our large-scale simulation to over 10 million cores, we propose a hierarchical decomposition scheme with multiple levels, as shown in Fig 4. First, the complex surface topography is transformed into a regular 3D mesh using our proposed Curvilinear FDM, which enables the regular decomposition hereafter. Second, the entire problem domain (e.g., $x_1 \times y_1 \times z_1$)is decomposed via a 3D MPI decomposition scheme (e.g. $p_{x1} \times p_{y1} \times p_{z1}$), with each sub-domain (e.g., $x_2 \times y_2 \times z_2$) being assigned to one CG. Sub-domains communicate with each other via asynchronous MPI routines.

In each CG, the sub-domain requires further partitioning (e.g., $p_{x2} \times p_{y2} \times p_{z2}$) as the SPM within each CPE is limited and generally cannot accommodate the entire sub-domain. The sub-domain is further decomposed into a sub-volume of size $(x_3 \times y_3 \times z_3)$. The 8×8 CPE cluster can further be rearranged (e.g. $p_{x3} \times p_{y3} \times p_{z3}$) to process the computation. Note that the CPEs located in the same row or same column can share data via the register communication (RC) feature, which enables efficient data exchange among CPEs. Each CPE is responsible for processing a small volume of size $(x_4 \times y_4 \times z_4)$. In our case, we assume that the memory continues along the z-axis. Therefore, z_4 is the maximal number of elements that can be transferred between SPM and main memory via DMA. The DMA bandwidth to and from SPM is denoted as $v_{Bi}(z)$, and $v_{Bo}(z)$, respectively.

For a specific setting of the problem size and the available resources of the machine, it is usually difficult to identify the optimal configuration parameters. To derive suitable configurations to simulate the Wenchuan earthquake efficiently when given an arbitrary problem size and resource quantity, we build a performance model driven auto-tuning framework.

The performance model is the basis of the automatic tuning tool. The performance model at the very top level can be described as follows:

$$T_{total} = T_0 + NT * T_{iter} + N_{IO} * T_{IO}. \quad (17)$$

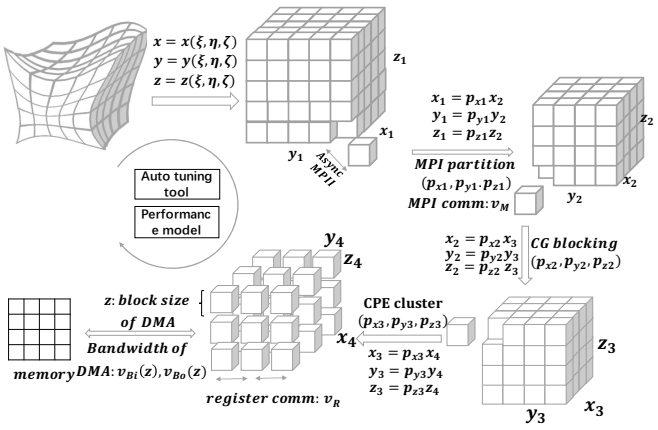

Fig. 4: The general parallelization scheme to map the earthquake simulation to millions of cores.

The cost of a task includes an initial process, loops of iterations, and IO. T_0 refers to the initialization, which is invoked once. IO in this work is mainly used for storing checkpoints, which need to dump almost all the data from the memory into the filesystem. N_{IO} means the number of checkpoints, while T_{IO} means the cost of each checkpoint. We adopt balanced IO, group IO and lossless data compression techniques to guarantee the performance of IO transmission. The large timestep NT would make the iteration time $NT * T_{iter}$ the major factor to optimize, in which T_{iter} means the cost for each iteration. The detailed model is decomposed into the process level and thread level, discussed in the following sections.

B. Process-Level Performance Model

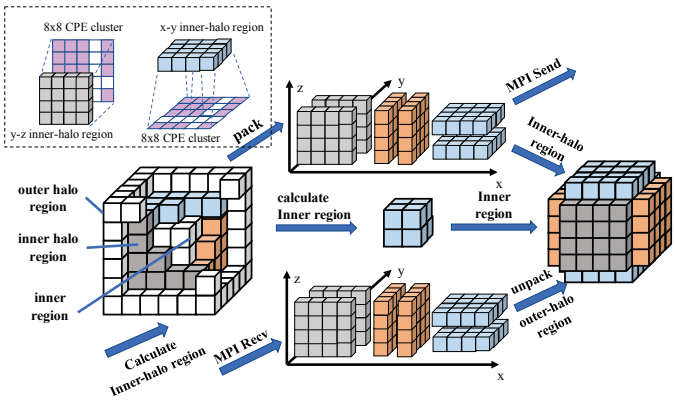

Fig. 5: Asychronize MPI

On the MPI process level (first level), according to Fig 5, we decompose the 3D region into three layers (outer halo region, inner halo region and inner region) and decompose MPI into several parts. The cost of each iteration contains the inner-halo region computation ($T_{inner-halo}$), data packing (T_{pack}), MPI (T_{MPI}), data unpacking (T_{unpack}) and inner region calculation (T_{inner}). In Asynchronous MPI, the inner region computation shall be performed by CPEs, and can be overlapped by MPI performed by MPE, so the cost of the kernel is:

$$T_{iter} = T_{inner-halo} + T_{pack} + max(T_{inner}, T_{MPI}) + T_{unpack}. \quad (18)$$

The cost of computing 6 inner-halo regions is described as follows:

$$T_{inner-halo} = 2T_c(x_2, y_2, N_{var}H), + 2T_c(x_2, H, N_{var}z_2) + 2T_c(H, y_2, N_{var}z_2) \quad (19)$$

where $T_c(x, y, z)$ refers to the computation time for a domain of $(x \times y \times z)$, N_{var} refers to the number of unknowns per grid point (the data is arranged in array of structures), and H refers to the thickness of the halo region.

The CPEs packing six inner-halo regions via DMA is described on the left part in Fig 5. We map each region to the CPE cluster, for example, we map the y-z plane of fore-and-aft inner-halo regions to 8×8 CPE cluster and map the x-y plane of upward and downward inner-halo regions to 8×8 CPE cluster. Data packing is a discontinuous memory copy process. The key is the DMA memory bandwidth from memory on CG to SPM on CPE, which is according to the block size of one continuous access. The cost of data packing is shown as follows:

$$T_{pack} = 2N_{var}H(\frac{x_2y_2}{v_{Bi}(N_{var}H)} + \frac{x_2z_2}{v_{Bi}(N_{var}z_4)} + \frac{y_2z_2}{v_{Bi}(N_{var}z_4)}), \quad (20)$$

where $v_{Bi}(z)$ is the DMA memory bandwidth (from memory on CG to SPM on CPE) function with respect to z.

The cost of MPI communication is:

$$T_{MPI} = 2N_{var}H\frac{x_2y_2 + x_2z_2 + y_2z_2}{v_M}, \quad (21)$$

where v_M means the average MPI bandwidth. Similar to the process of inner-halo region computation, the cost of computing inner region is shown as follows:

$$T_{inner} = 2T_c(x_2 - 2H, y_2 - 2H, N_{var}(z_2 - 2H)). \quad (22)$$

The cost of data unpacking is opposite to data packing:

$$T_{unpack} = 2N_{var}H(\frac{x_2y_2}{v_{Bo}(N_{var}H)} + \frac{x_2z_2}{v_{Bo}(N_{var}z_4)} + \frac{y_2z_2}{v_{Bo}(N_{var}z_4)}), \quad (23)$$

where $v_{Bo}(z)$ is the DMA memory bandwidth (from SPM on CPE to memory on CG) function with respect to z.

C. Thread-level Performance Model

On the CPE thread level, to achieve the best utilization of the 64 CPEs within one CG, we design three different kinds of schemes: a pure DMA scheme (T_{PD}), a 2D register communication (RC) scheme (T_{2DR}), and a 1D register communication scheme (T_{1DR}). Different kinds of data domain sizes and partitions will lead to different decisions among these three schemes. So the computational cost of the optimal scheme is:

$$T_c(x_2, y_2, z_2) = p_{x3}p_{y3}p_{z3}min(T_{PD}(x_3, y_3, z_3) T_{2DR}(x_3, y_3, z_3), T_{1DR}(x_3, y_3, z_3)). \quad (24)$$

A detailed discussion of these schemes is shown below.

(1) Pure DMA scheme: In thread-level optimization, the CPE needs to fetch the region $(x_4 + 2H)$ by $(y_4 + 2H)$ by $N_{var}(z_4 + 2H)$, compute and then write back the region x_4

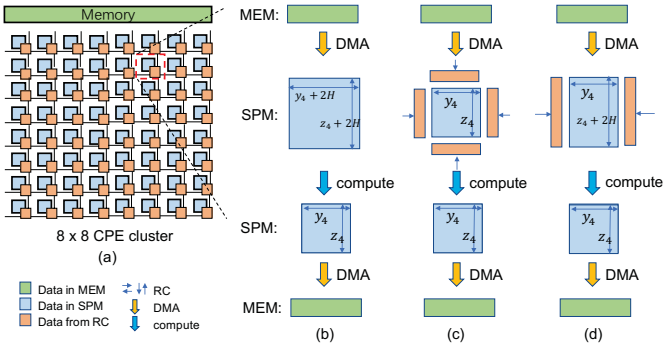

Fig. 6: Three thread-level schemes.

by y_4 by $N_{var}z_4$ to memory, which is shown in part (b) of Fig 6. Due to the asynchronous DMA, the computational time will be partly overlapped by DMA. The cost of this process is:

$$T_{PD}(x_3, y_3, N_{var}z_3) = p_{x3}p_{y3}p_{z3}max(\frac{N_{var}(x_4+2H)(y_4+2H)(z_4+2H)}{v_{Bi}(N_{var}z_4+2H)} + \frac{N_{var}x_4y_4z_4}{v_{Bo}(N_{var}z_4)}, \frac{N_{flop}}{v_{flop}}), \quad (25)$$

where N_{flop} is the number of floating point operations within the kernel and v_{flop} is the number of real floating point operations per second on CPE.

(2) 2D register communication (RC) scheme: using pure DMA, it is necessary for each CPE to read four halo boundaries to compute the inner region, which will lead to low data utilization. To solve this problem, we design a 2D register communication (RC) scheme. Each CPE reads only the inner region, and then exchange the halo boundaries by register communication with its neighbor CPEs, which is shown in part (c) of Fig 6. This scheme can effectively improve the data utilization. However, CPEs located at four boundaries still need to read halo boundaries by DMA, and the bandwidth of DMA when reading upper and lower boundaries is $v_{Bi}(N_{var}H)$, which is small and largely constrained by the thickness of halo region and would lead to a large latency for the whole kernel. So it is unclear that this scheme is better than the pure DMA scheme. The cost of this scheme includes RC and DMA for the inner region and extra boundaries, which is shown below:

$$T_{2DR}(x_3, y_3, N_{var}z_3) = max(\frac{N_{var}x_3(y_3z_3+2Hz_3)}{v_{Bi}(N_{var}z_4)} + \frac{2N_{var}Hx_3y_3}{v_{Bi}(N_{var}H)} + \frac{2N_{var}Hx_3(y_3+z_3)}{v_R} + \frac{N_{var}x_3y_3z_3}{v_{Bo}(N_{var}z_4)}, \frac{N_{flop}}{v_{flop}}), \quad (26)$$

where v_R is the speed of RC.

(3) 1D register communication scheme: although 2D register communication communicates in the 2D pattern, it still involves complicated issues. As mentioned above, CPEs located at the four boundaries need to read additional halo boundaries through DMA, which will lead to a large latency. To alleviate this issue, as shown in part (d) of Fig 6, we use each CPE to read an area of $x_4y_4(z_4+2H)$, and then exchanges the left and right halo regions by RC with its left and right neighbor CPEs. In addition, the leftmost and rightmost CPEs needs to load halo boundaries by DMA. This scheme is quite reasonable, but it reads more data than the 2D register communication scheme and so it is still not clear that this is the optimal choice.

The cost for this scheme is shown below:

$$T_{1DR}(x_3, y_3, N_{var}z_3) = max(\frac{N_{var}x_3(y_3z_3+2H(z_3+2H)}{v_{Bi}(z_4+2H)} + \frac{2N_{var}Hx_3(z_3+2H)}{v_R} + \frac{N_{var}x_3y_3z_3}{v_{Bo}(N_{var}z_4)}, \frac{N_{flop}}{v_{flop}}). \quad (27)$$

D. Auto-Tuning Tool Using a Genetic Algorithm

Based on our performance model, we can describe the performance and bandwidth of a simulation task. But with so many tuning parameters it is hard for us to find the optimal configuration easily. Therefore, we create an automatic tuning tool based on our performance model and a genetic algorithm([25], [26]).

The major constraints for optimization are as follows: (1) the available resources of the computing nodes, memory size, and SPM space; (2) the size in a specific axis of the upper level is larger than the lower level; (3) all sizes must be positive integers.

The tuning tool is designed to find the optimal configuration parameters when given specific set of resources and problem size, The key unknowns are partition scheme $p_{x1}, p_{y1}, p_{z1}, p_{x2}, p_{y2}, p_{z2}, p_{x3}, p_{y3}, p_{z3}$. We use genetic algorithm (GA) to solve this optimal problem ([25], [26], [27], [28]), which is shown in Fig 7. The genetic representation is the set of unknowns in binary represenation with a search range within the bounds, and the fitness function is the cost of the whole program. The process is shown as follows:

Fig. 7: Auto tuning based on a genetic algorithm.

For example, we use the auto-tuning tool before our 50-meter resolution simulation. First, given the resources of

nodes, memory size, problem size, we produce the first random population of 500 individuals within the bounds, each of which is a set of 9 unknowns. Then we apply the crossover operators with a probability of 0.1 to every two individuals, the mutation with a probability of 0.01 to every gene, and the copy selection operations with a probability of 0.01 to every individual randomly, respectively. Now we are able to produce a new generation with better characteristics than previous generations. After 400,000 iterations, the optimal configuration parameters can be found in the last generation. The auto-tuning tool costs about 6 hours, which is near the cost of one simulation, but saves weeks over exhaustively searching all possible parameters for an optimal parameter combination.

V. Data Layout Transformation

One tough challenge for earthquake simulations is the large number of unknowns that needs to be processed for each grid point. For the AWP-ODC linear and nonlinear versions, each grid point involves 30 to 40 unknowns. In our curvilinear grid FDM, the number of variables per grid point is further increased to over 50. Considering the DMA behavior for TaihuLight, the co-located array fusion strategy proposed in [2] is a common approach to effectively improve the DMA bandwidth utilization. However, such a strategy requires that the set of fused arrays share exactly the same pattern of reads and writes. Some variables such as the $density(\rho)$ are independent of other variables, resulting in a terrible DMA bandwidth and computing efficiency as the co-located array fusion trick does not work for them. For these different sets of variables, we propose a layout transformation scheme for those variables to improve DMA bandwidth effectively. The basic idea for the layout transformation is shown in Fig 8. In the original parallelization scheme, each CPE thread would read the corresponding 2D data block. Such a straightforward parallel decomposition, as shown in part (b) of Fig 8, would break the continuity of the memory space, and thus significantly reduce the efficiency of DMA instructions.

To keep the continuity of the data items in each CPE thread, and to improve the DMA operation efficiency, at the initialization stage, we use 64 CPEs to perform the layout transformation of the corresponding memory region in a parallel way. The main transformation mechanism is shown in part (c) of Fig 8. The data layout transformation is performed for each CG independently.

To ensure the continuity of memory access during the computation, the memory region for each CPE thread is enlarged to include halo areas. The first step of the transformation is to read data from memory to SPM. When performing the transformation, if the CPE thread was sitting at the boundary of the CPE cluster, it needs to read the extra halo points from the memory. Otherwise, the CPE thread acquires the halo points from the neighboring threads using the unique register communication feature. Now the SPM has a continuous data layout, and the MPE allocates the extra space to store the transformed data.

After the layout transformation, the CPE can read in the data to compute with a block size of the entire $y - z$ plane while the original block size of DMA is just one stripe. If the array only needs to be read and does not need to be updated, we only need to transform once at the initial stage. As shown in part (d) of Fig 8, if data needs to be updated in each iteration, we would need to apply the reverse process and then write the whole region (including yellow region and blue region) back to memory to keep the new structure. The data size for transmission is larger than the original scheme (the yellow region), which will eliminate the advantage of this scheme. So the data transformation scheme is only suitable for the $density(\rho)$ read-only array, enlarging the DMA bandwidth.

VI. Performance and Scalability

A. How Performance was Measured

The performance is measured using the average time for one time step, by running the benchmark test for 100 time steps. In formula 17, for example, when running a 50-meter resolution simulation on the whole machine of Sunway TaihuLight, T_{total} is about 5 hours, the initialization time T_0 is about 10 minutes. We usually have one checkpoint during the whole simulation, where T_{IO} is about 10 minutes, and the other IO time can be negligible when hundreds of thousands of time steps are involved. The initialization time and IO time account for only 6.7% of the total time. So we just focus on the iteration time in the following sections. The number of floating point operations is measured using two different methods, namely by counting all floating point arithmetic instruction in the assembly code as well as using the hardware performance monitor, PERF, which is provided along with the Sunway TaihuLight compiler. The two different methods generate similar operation counts, and we use the PERF tool to measure the average floating point operations in this study.

B. Kernel Optimization Results

The Runge-Kunta synthesis step is the most time-consuming part (the redesigned function that integrates seven different kernels) of the entire program. Fig. 9 demonstrates the performance and bandwidth improvements for different kinds of optimization methods on different resolutions. With the resolution becoming higher (generally more data times to process per thread), the performance benefits of different optimization methods also increase. We set the 'MPE' version as the starting point of the performance curve, which only uses single MPE in each CG to perform the computation. The 'naive' version performs a straightforward mapping from a single MPE to 64 CPEs. Other bars indicate the performance improvements after applying different optimization strategies accordingly. After performing the memory-oriented algorithm redesign, the performance has improved by roughly four times based on different resolutions. With the tuned parameters for both process and thread configurations, we can further improve the performance by another 1.5 times. After applying the data layout transformation, the performance was finally improved by another 1.26 times. The final design is 152.9 times faster than the 'MPE' version and demonstrates a memory bandwidth

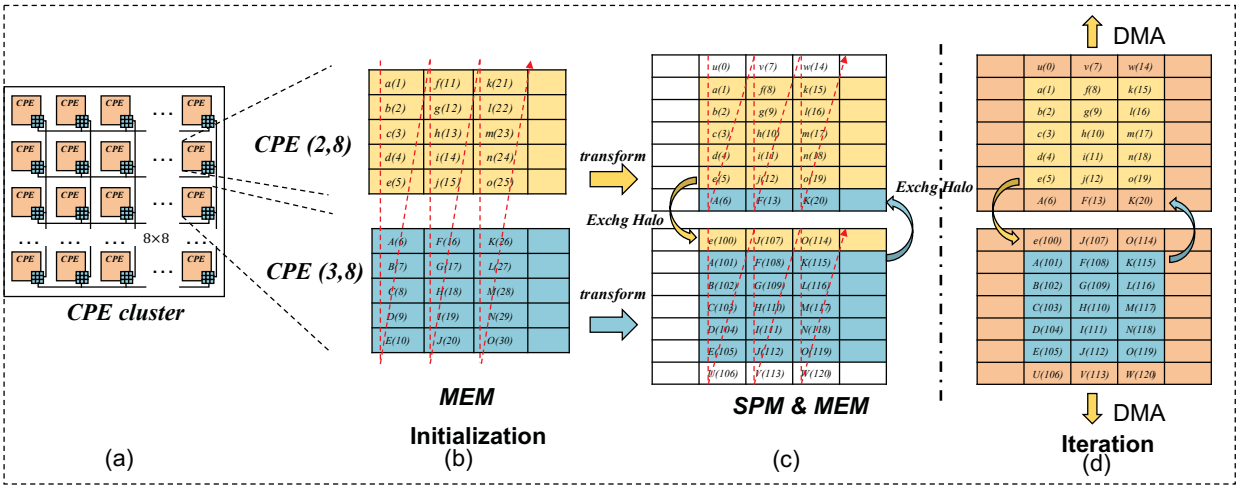

Fig. 8: Data layout transformations

utilization of 22 GB/s bandwidth, which is about 73.3% of the theoretical peak.

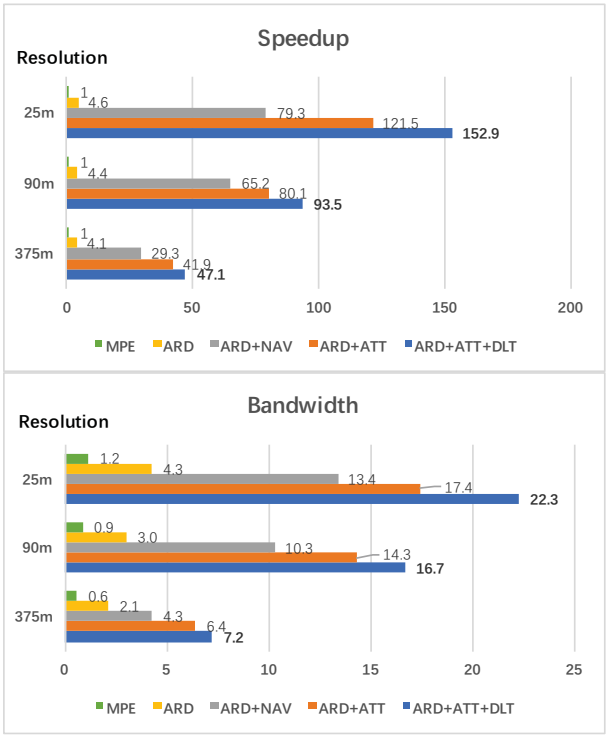

Fig. 9: The speedup and DMA bandwidth of the Runge-Kunta synthesis step, when applying different optimization methods under the guidance of the performance model. 'MPE' stands for the original version that uses MPI only; 'ARD' means algorithm redesign, 'NAV' refers to naive partition scheme, 'ATT' means auto tuning tool optimization, 'DLT' refers to data layout transformation optimization.

C. Weak Scaling Results

Fig. 10 demonstrates the weak scaling results on 25-m and 20-m resolution cases. For both cases, each MPI process (corresponding to one CG) computes a sub-domain with the size of $120 \times 120 \times 2000$. The decomposed mesh size is also used for the scientific simulations discussed later in Section VII. The total number of the grid points in this domain becomes $120 \times 120 \times 2000 \times N$, where N is the number of CGs. In the benchmark tests, we see a close-to-linear speedup from 900 processes to 160,000 processes for the weak scaling. So the proposed process-level optimization for communication model is proved to be highly efficient. Since the application performs only nearest-neighbor communications, we would expect a continued linear scaling when further applying the entire machine.

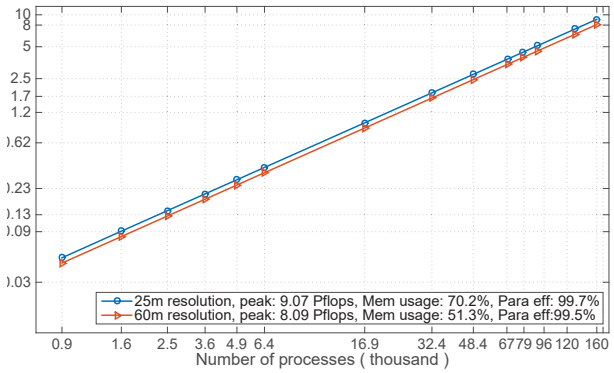

Fig. 10: The weak scaling results of the simulation, scaling from 900 to 160,000 MPI processes. Each CG corresponds to one MPI process.

D. Strong Scaling Results

Fig. 11 shows the results of the strong scaling tests for 125-m to 50-m resolutions based on three different mesh sizes. Our software achieves the similar speedup for the scenarios of 125-m and 80-m resolutions. With the increase of the number of CGs, performance degradation is expected, due to two possible reasons. The ratio of computation over communication decreases, and so does the ratio of the outer halo region over the sub-domain size in proportion. Our software is thereby less effective in overlapping computations and communication.

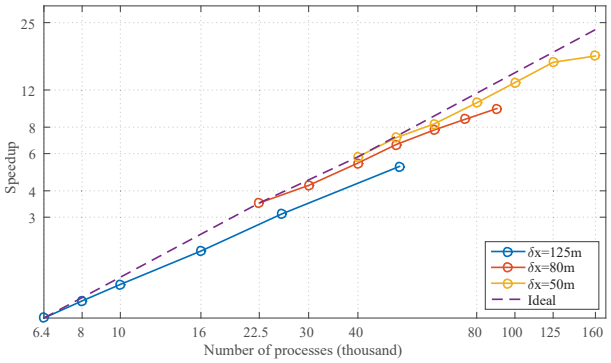

Fig. 11: The strong scaling results scaling from 6,400 to 160,000 MPI processes for three different problem sizes. Each CG corresponds to one MPI process.

VII. The Wenchuan Earthquake Simulation on Sunway TaihuLight

Seismic wave propagation for large-scale earthquakes is affected by the process of source rupture, the velocity medium structure in the source area, and the free surface boundary. The source process can be obtained by waveform inversion, geodetic survey, and GPS data inversion of predecessors. Limited by the difficulty of seismic inversion, the source model obtained after source inversion is relatively smooth, with a lack of high-frequency signals. The medium velocity model is difficult to observe directly and is often obtained through model inversion by other work. Therefore, the inversion is done with more smooth constraints, with not too much of the high-frequency information. However, the terrain data can be directly observed.

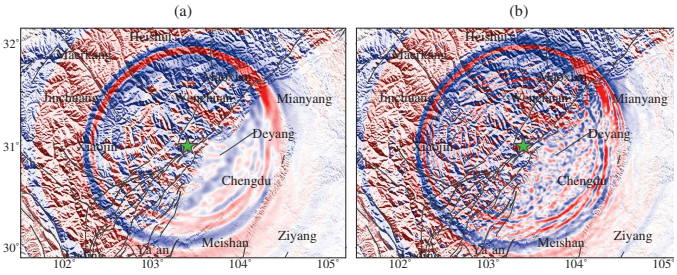

Fig. 12: The up-down velocity field snapshot at 32 sec with different seismic source: (a) rise time = 4s; (b) rise time = 2s

In this simulation, we selected a 512km × 512km × 80km and 3D area including the Wenchuan earthquake fault for calculation. We set the spatial resolution to 64m. Therefore, the number of grid points is as high as 8000x8000x1250. Such a large scale is a difficult task to accomplish for ordinary multi-core computing clusters but can be simulated by the Sunway TaihuLight within a few hours. We chose the inverted seismic source [29] with a resolution of 10km x 7.5km (32x8 points in total). We interpolate the seismic source to 1280x320 points for simulation. The source time function uses a Gaussian function to control the primary frequency of the source by the rise time. A homogeneous structural model is used, excluding the effects of complex media. In Fig 12, the rise time in (a) is 4s and in (b) is 2s. It can be found that the seismic wavefield is affected by the reflection and scattering of the topography, and the coda wave effect is obvious especially for high frequency (corresponding to small rise time) seismic source. In order to verify the influence of the topography effect during the wavefield simulation, we made a comparative verification case, which is shown in section A.

From this comparison, it can be seen that the topography contributes greatly to the coda wave effect of seismic waves. Therefore, for the simulation of seismic waves with complex topography, large-scale simulation is necessary.

VIII. Conclusion

In this work, we proposed a curvilinear grid FDM with the traction image method to handle the complex geology features of the Wenchuan earthquake. To develop an efficient implementation on the Sunway TaihuLight system, we first redesigned the algorithm to reduce the memory access cost. A performance model guide and further optimizations and tunings of the parameters are then derived using a genetic algorithm. Our efforts improve the simulation performance for complex scenarios, such as the Wenchuan earthquake, from 0.05% to 7.6% of the hardware peak. When using the entire machine with over 10 million cores, SWST provides a sustained performance of 9.07 Pflops. SWST succeeded in performing the Wenchuan Earthquake (Ms 8.0, 2008) simulation with complex surface topography in 50-meter resolution for a region of 640 km by 640 km by 100 km, demonstrating significantly improved depiction of the coda wave effects, and a high potential to achieve a more accurate description of the seismic events.

Acknowledgment

The authors are grateful to the reviewers for valuable comments that have greatly improved the paper. H. Fu, B. Chen, and Y. Wei are supported by the National Key Research & Development Plan of China (grant# 2017YFA0604500), the National Natural Science Foundation of China (grant no. 91530323, 41661134014, 41504040 and 61361120098); and the Tsinghua University Initiative Scientific Research Program (no. 20131089356). L. Gan, Y. Li are supported by the National Natural Science Foundation of China (grant no. 61702297); and the China Postdoctoral Science Foundation (grant no. 2016M601031). G. Yang, C. He, W. Wan, and W. Zhang[3] are supported by the National Key Research & Development Plan of China (grant no 2016YFA0602200). The corresponding author is Haohuan Fu (email: haohuan@tsinghua.edu.cn). Last but not least, a special girl Bingwei Chen would like to thank sincerely is Dr. Chuling Fang from Peking Union Medical College for her patience, understanding and encouragement during our time together. And darling, will you marry me?

References

[1] B. der Hilst, "A geological and geophysical context for the Wenchuan earthquake of 12 May 2008, Sichuan, People's Republic of China," *GSA today*, vol. 18, no. 7, p. 5, 2008.

[2] H. Fu, C. He, B. Chen, Z. Yin, Z. Zhang, W. Zhang, T. Zhang, W. Xue, W. Liu, W. Yin *et al.*, "18.9-Pflops nonlinear earthquake simulation on Sunway TaihuLight: enabling depiction of 18-Hz and 8-meter scenarios," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2017, p. 2.

[3] Y. Cui, K. B. Olsen, T. H. Jordan, K. Lee, J. Zhou, P. Small, D. Roten, G. Ely, D. K. Panda, A. Chourasia *et al.*, "Scalable earthquake simulation on petascale supercomputers," in *High Performance Computing, Networking, Storage and Analysis (SC), 2010 International Conference for*. IEEE, 2010, pp. 1–20.

[4] S. Potluri, K. Hamidouche, A. Venkatesh, D. Bureddy, and D. K. Panda, "Efficient inter-node MPI communication using GPUDirect RDMA for InfiniBand clusters with NVIDIA GPUs," in *Parallel Processing (ICPP), 2013 42nd International Conference on*. IEEE, 2013, pp. 80–89.

[5] M. Christen, O. Schenk, and Y. Cui, "Patus for convenient high-performance stencils: evaluation in earthquake simulations," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 11.

[6] D. Komatitsch, S. Tsuboi, C. Ji, and J. Tromp, "A 14.6 billion degrees of freedom, 5 teraflops, 2.5 terabyte earthquake simulation on the Earth Simulator," in *Supercomputing, 2003 ACM/IEEE Conference*. IEEE, 2003, pp. 4–4.

[7] R. Jankowski, "Non-linear FEM analysis of earthquake-induced pounding between the main building and the stairway tower of the Olive View Hospital," *Engineering Structures*, vol. 31, no. 8, pp. 1851–1864, 2009.

[8] L. Carrington, D. Komatitsch, M. Laurenzano, M. M. Tikir, D. Michéa, N. Le Goff, A. Snavely, and J. Tromp, "High-frequency simulations of global seismic wave propagation using SPECFEM3D_GLOBE on 62K processors," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press, 2008, p. 60.

[9] L. Bie, T. Garth, A. Rietbrock, J. Collier, S. Goes, C. Rychert, T. Henstock, N. Harmon, and A. Anglade, "Towards full waveform simulation using SPECFEM3D for earthquakes in the Lesser Antilles subduction zone," in *AGU Fall Meeting Abstracts*, 2016.

[10] F. Rodríguez Cardozo, V. Hjörleifsdóttir, and M. Caló, "Homogenization and implementation of a 3D regional velocity model in Mexico for its application in moment tensor inversion of intermediate-magnitude earthquakes," in *EGU General Assembly Conference Abstracts*, vol. 19, 2017, p. 10428.

[11] M. Rietmann, P. Messmer, T. Nissen-Meyer, D. Peter, P. Basini, D. Komatitsch, O. Schenk, J. Tromp, L. Boschi, and D. Giardini, "Forward and adjoint simulations of seismic wave propagation on emerging large-scale GPU architectures," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 38.

[12] A. Heinecke, A. Breuer, S. Rettenberger, M. Bader, A.-A. Gabriel, C. Pelties, A. Bode, W. Barth, X.-K. Liao, K. Vaidyanathan *et al.*, "Petascale high order dynamic rupture earthquake simulations on heterogeneous supercomputers," in *High Performance Computing, Networking, Storage and Analysis, SC14: International Conference for*. IEEE, 2014, pp. 3–14.

[13] V. Ginting, G. Lin, and J. Liu, "On application of the weak Galerkin finite element method to a two-phase model for subsurface flow," *Journal of Scientific Computing*, vol. 66, no. 1, pp. 225–239, 2016.

[14] A. Breuer, A. Heinecke, and Y. Cui, "EDGE: Extreme Scale Fused Seismic Simulations with the Discontinuous Galerkin Method," in *International Supercomputing Conference*. Springer, 2017, pp. 41–60.

[15] C. Uphoff, S. Rettenberger, M. Bader, E. H. Madden, T. Ulrich, S. Wollherr, and A.-A. Gabriel, "Extreme scale multi-physics simulations of the tsunamigenic 2004 sumatra megathrust earthquake," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2017, p. 21.

[16] Y. Cui, E. Poyraz, K. B. Olsen, J. Zhou, K. Withers, S. Callaghan, J. Larkin, C. Guest, D. Choi, A. Chourasia *et al.*, "Physics-based seismic hazard analysis on petascale heterogeneous supercomputers," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. ACM, 2013, p. 70.

[17] X. Xu, X. Wen, G. Yu, G. Chen, Y. Klinger, J. Hubbard, and J. Shaw, "Coseismic reverse-and oblique-slip surface faulting generated by the 2008 Mw 7.9 Wenchuan earthquake, China," *Geology*, vol. 37, no. 6, pp. 515–518, 2009.

[18] T. Parsons, C. Ji, and E. Kirby, "Stress changes from the 2008 Wenchuan earthquake and increased hazard in the Sichuan basin," *Nature*, vol. 454, no. 7203, p. 509, 2008.

[19] T. Ichimura, K. Fujita, P. E. B. Quinay, L. Maddegedara, M. Hori, S. Tanaka, Y. Shizawa, H. Kobayashi, and K. Minami, "Implicit nonlinear wave simulation with 1.08 T DOF and 0.270 T unstructured finite elements to enhance comprehensive earthquake simulation," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2015, p. 4.

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

[21] J. Fang, H. Fu, W. Zhao, B. Chen, W. Zheng, and G. Yang, "swdnn: A library for accelerating deep learning applications on sunway taihulight," in *Parallel and Distributed Processing Symposium (IPDPS), 2017 IEEE International*. IEEE, 2017, pp. 615–624.

[22] Z. Zhang, W. Zhang, and X. Chen, "Three-dimensional curved grid finite-difference modelling for non-planar rupture dynamics," *Geophysical Journal International*, vol. 199, no. 2, pp. 860–879, 2014.

[23] W. Zhang and X. Chen, "Traction image method for irregular free surface boundaries in finite difference seismic wave simulation," *Geophysical Journal International*, vol. 167, no. 1, pp. 337–353, 2006.

[24] W. Zhang, Z. Zhang, and X. Chen, "Three-dimensional elastic wave numerical modelling in the presence of surface topography by a collocated-grid finite-difference method on curvilinear grids," *Geophysical Journal International*, vol. 190, no. 1, pp. 358–378, 2012.

[25] K. Deb, "An introduction to genetic algorithms," *Sadhana*, vol. 24, no. 4-5, pp. 293–315, 1999.

[26] C. R. Houck, J. Joines, and M. G. Kay, "A genetic algorithm for function optimization: a Matlab implementation," *Ncsu-ie tr*, vol. 95, no. 09, pp. 1–10, 1995.

[27] L. Fan and E. M. Joo, "Design for auto-tuning PID controller based on genetic algorithms," in *Industrial Electronics and Applications, 2009. ICIEA 2009. 4th IEEE Conference on*. IEEE, 2009, pp. 1924–1928.

[28] A. H. Wright, "Genetic algorithms for real parameter optimization," in *Foundations of genetic algorithms*. Elsevier, 1991, vol. 1, pp. 205–218.

[29] Y. Yagi, N. Nishimura, and A. Kasahara, "Source process of the 12 May 2008 Wenchuan, China, earthquake determined by waveform inversion of teleseismic body waves with a data covariance matrix," *Earth Planets & Space*, vol. 64, no. 7, pp. e13–e16, 2012.

APPENDIX A
VERIFY THE INFLUENCE OF THE TOPOGRAPHY EFFECT

In order to verify the influence of the topography effect during the wavefield simulation, we made a comparative verification case. We eliminated the influence of the media and the source by setting uniform media and point Ricker sources (1Hz). Hence we can only focus on the topographic effect. As shown in Fig 13 (a) and (b), the complex topography causes more high-frequency scattered waves and has a great influence on the simulation of the seismic wave field. It can be concluded that the role of topography in the Wenchuan earthquake simulation cannot be ignored. The effect of the terrain on the simulation can be observed through high-resolution simulations. For large-scale earthquake simulations

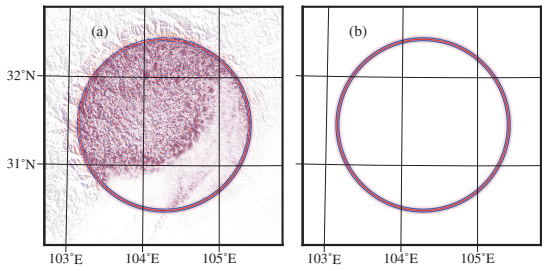

Fig. 13: A simple case to verify topography effect during earthquake simulations. (a) with topography (b) without topography

(especially for the Wenchuan earthquake, which is a complex terrain earthquake), the effect of the terrain on the simulation can be observed through high-resolution simulations.