Journal of Computational Physics 298 (2015) 237–253

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Parallel adaptive wavelet collocation method for PDEs

Alireza Nejadmalayeri [a], Alexei Vezolainen [b], Eric Brown-Dymkoski [b], Oleg V. Vasilyev [b,*]

[a] FortiVenti Inc., Suite 404, 999 Canada Place, Vancouver, BC, V6C 3E2, Canada
[b] Department of Mechanical Engineering, University of Colorado Boulder, UCB 427, Boulder, CO 80309, USA

ARTICLE INFO

Article history:
Received 22 September 2013
Received in revised form 19 May 2015
Accepted 21 May 2015
Available online 4 June 2015

Keywords:
Parallel algorithm
Parallel computing
Domain decomposition
Dynamic load balancing
Wavelets
Lifting scheme
Second generation wavelets
Adaptive grid
Multiresolution
Multilevel method
Multigrid method
Numerical method
Partial differential equations
Elliptic problem

ABSTRACT

A parallel adaptive wavelet collocation method for solving a large class of Partial Differential Equations is presented. The parallelization is achieved by developing an asynchronous parallel wavelet transform, which allows one to perform parallel wavelet transform and derivative calculations with only one data synchronization at the highest level of resolution. The data are stored using tree-like structure with tree roots starting at *a priori* defined level of resolution. Both static and dynamic domain partitioning approaches are developed. For the dynamic domain partitioning, trees are considered to be the minimum quanta of data to be migrated between the processes. This allows fully automated and efficient handling of non-simply connected partitioning of a computational domain. Dynamic load balancing is achieved via domain repartitioning during the grid adaptation step and reassigning trees to the appropriate processes to ensure approximately the same number of grid points on each process. The parallel efficiency of the approach is discussed based on parallel adaptive wavelet-based Coherent Vortex Simulations of homogeneous turbulence with linear forcing at effective non-adaptive resolutions up to 2048^3 using as many as 2048 CPU cores.

1. Introduction

The quest for highly scalable adaptive numerical methods is still ongoing despite more than three decades of extraordinary developments in supercomputing. Many attractive mathematical properties of the wavelet multi-resolution analysis such as compression, denoising, and multi-scale decomposition have made it a very promising tool in the challenging search for robust and computationally efficient multi-scale computational approach for modeling and simulation. Adaptive Wavelet Collocation Method (AWCM) is such a technique, which has been developed and thoroughly investigated for parabolic [1,2], hyperbolic [3], and elliptic [4] partial differential equations. It was successfully applied to a wide spectrum of problems including incompressible [5], compressible subsonic [6] and supersonic [3] flows, wavelet-based Adaptive Large Eddy Simulation [7–13], thermoacoustic wave propagation [14], Rayleigh–Taylor instability [15], ocean modeling [16], combustion [17], fluid–structure interactions [18,19], viscoelastic and poro-viscoelastic flows [20–22].

* Corresponding author.
E-mail addresses: Alireza.Nejadmalayeri@gmail.com (A. Nejadmalayeri), Alexei.Vezolainen@Colorado.edu (A. Vezolainen), Eric.Browndymkoski@Colorado.edu (E. Brown-Dymkoski), Oleg.Vasilyev@Colorado.edu (O.V. Vasilyev).

http://dx.doi.org/10.1016/j.jcp.2015.05.028
0021-9991/

Adaptive Wavelet Collocation Method [1–4] is based on the "second generation wavelets" [23,24]. In AWCM, the partial differential equations are solved in physical space on an adaptive nested (dyadic) computational grid. This prevents the major difficulties associated with adaptive wavelet Galerkin methods: challenging treatment of nonlinearities and general boundary conditions. The evaluation of the nonlinear terms in adaptive wavelet collocation methods is performed in a physical domain similar to pseudo-spectral methods. The grid adaptation in wavelet collocation methods is done similarly to other wavelet-based methods and is based on analyzing the wavelet coefficients. Despite enormous compression achieved by wavelets, e.g. 99%, very large-scale simulations cannot yet fit onto a single process and require highly scalable parallel algorithms.

Despite almost 30 year history of wavelets since their introduction by Grossmann and Morlet [25] and wide use in science and engineering, very little attention was paid to parallel adaptive wavelet methodologies. Until recently, most of the efforts were put into the development of parallelization strategies for non-adaptive wavelet algorithms, e.g., a communication-free parallel discrete wavelet transform [26], a communication efficient fast wavelet transform without distributed matrix transpose [27], a distributed parallel biorthogonal lifted wavelet transform [28], a parallel wavelet transform for distributed and shared memory architectures [29], and parallel GPU based discrete wavelet transforms [30–33]. The only noticeable attempts to develop parallel adaptive wavelet-based methods are multi-block adaptive wavelet method by Rossinelli et al. [34] and Adaptive Wavelet Multiresolution Representation (AWMR) method by Paolucci et al. [35], with the latter published while the manuscript was under review. Thus, the main objective of this paper is to present the extension of the AWCM [1–4] for massively parallel computers.[1]

The paper is organized as follows. The one- and multi-dimensional second generation wavelet transforms are reviewed thoroughly in Section 2. The challenges associated with the parallelization of the update-stage of the second generation wavelet transform are explained in Section 3, and a parallel asynchronous second generation wavelet transform is then introduced. The robust tree structure database utilized in this study is discussed in Section 4. The grid adaptation strategy based on wavelet-thresholding is reviewed in Section 5, where the concepts of reconstruction-check, safety/adjacent zone along with significant/adjacent masks are introduced. The finite difference based derivative algorithm for the adaptive wavelet collocation method and the use of ghost points are discussed in Section 6. After a short discussion of data migration in Section 7, the four different static and dynamic domain partitioning methods utilized in this study are explained in Section 8. The algorithm of the resulting parallel adaptive wavelet collocation method (PAWCM), based on the aforementioned components, is illustrated in Section 9. This versatile general parallel dynamically adaptive PDE solver is then used to perform a comprehensive strong-scalability study of the PAWCM for the velocity-based Coherent Vortex Simulations (CVS) of linearly forced homogeneous turbulence, Section 10. The challenges associated with the buffer zone size, the speedup slope and saturation are analyzed in detail and the numerical results are compared with an asymptotic parallel efficiency, which is derived. Finally, conclusions are given in Section 11.

2. Wavelet transform

In this section we briefly discuss the key aspects of the second generation wavelet construction, which are essential for understanding of the Parallel Adaptive Wavelet Collocation method. For more details we refer the reader to Refs. [1,2,4,23,24].

The one-dimensional second generation wavelets are constructed on an interval Ω with arbitrary distribution of grid (collocation) points. The construction is performed on an arbitrary set of interpolating points, $\{x_k^j \in \Omega\}$, which are used to form a set of nested grids

$$\mathcal{G}^j = \left\{ x_k^j \in \Omega : \; x_k^j = x_{2k}^{j+1}, \; k \in \mathcal{K}^j \right\}, \tag{1}$$

where x_k^j are the grid points of the j level of resolution.

The restriction $x_k^j = x_{2k}^{j+1}$ guarantees the nestedness of the grids, i.e. $\mathcal{G}^j \subset \mathcal{G}^{j+1}$. Following the construction of second generation wavelets described in [23,24], one-dimensional scaling functions $\phi_k^j(x)$ $(k \in \mathcal{K}^j)$ and wavelets $\psi_l^j(x)$ $(l \in \mathcal{L}^j)$ are constructed such that a function $u(x)$ can be decomposed as

$$u(x) = \sum_{k \in \mathcal{K}^0} c_k^0 \phi_k^0(x) + \sum_{j=0}^{+\infty} \sum_{l \in \mathcal{L}^j} d_l^j \psi_l^j(x), \tag{2}$$

where $\mathcal{K}^j$ and $\mathcal{L}^j$ are some index sets associated respectively with scaling functions and wavelets on level j. One may think of a wavelet decomposition as a multilevel or multiresolution representation of a function, where each level of resolution j (except the coarsest one) consists of wavelets ψ_l^j having the same scale but located at different positions. Note that scaling function coefficients represent smoothed version of the function at the current scale, while the wavelet coefficients represent the details of the function between the current scale and the next finest scale. An important property and strength

[1] PAWCM was first reported in Ref. [36].

of the wavelet transform is that wavelet coefficients can be obtained using a recursive application of a single level wavelet transform. Assuming that the scaling function coefficients c_k^{j+1} at level $j+1$ are known, the wavelet coefficients d_l^j and scaling function coefficients c_k^j at level j can be found using the following two-stage second generation wavelet transform

$$\text{Predict Stage: } d_k^j = \frac{1}{2}\left(c_{2k+1}^{j+1} - \sum_l w_{k,l}^j c_{2k+2l}^{j+1}\right), \tag{3}$$

$$\text{Update Stage: } c_k^j = c_{2k}^{j+1} + \sum_l \widetilde{w}_{k,l}^j d_{k+l}^j. \tag{4}$$

The corresponding one-dimensional inverse second generation wavelet transform is given by

$$\text{Inverse Update Stage: } c_{2k}^{j+1} = c_k^j - \sum_l \widetilde{w}_{k,l}^j d_{k+l}^j, \tag{5}$$

$$\text{Inverse Predict Stage: } c_{2k+1}^{j+1} = 2d_k^j + \sum_l w_{k,l}^j c_{2k+2l}^{j+1}, \tag{6}$$

where $w_{k,l}^j$ and $\widetilde{w}_{k,l}^j$ are coefficients associated with two stages of wavelet transform. Note, that the coloring of terms in Eqs. (4) and (5) is relevant to parallel implementation of the algorithm and is discussed in Section 3.

The first stage of forward transform is called a predict stage, since the wavelet coefficients are calculated by predicting the function value using the interpolated points on the next coarser level. The predict stage of the forward 1D second generation wavelet transform is illustrated in Fig. 1(a). The c_k^j-values that get carried down to the next lower level of resolution are then updated using the wavelet coefficients that were calculated during the predict stage. The update stage, Fig. 1(b), guarantees that the wavelet interpolating functions have zero mean. In fact, the interpolating wavelets of order N when using the update stage have N vanishing moments. For the inverse wavelet transform, the order of operations is reversed and the inverse wavelet transform is performed from low to high levels of resolution as opposed to from high to low for the forward wavelet transform.

The block diagram for one step wavelet transform is shown in Fig. 2, where S and S^{-1} denote respectively the delay and advance operators, i.e. $Sf_k = f_{k-1}$ and $S^{-1}f_k = f_{k+1}$, $(\downarrow 2)$ denotes the downsampling (decimation) operator which removes odd-numbered components from the signal, while U^j and P^j denote respectively lifting and dual lifting operators (P stands for predict and U stands for update).

The second generation scaling function ϕ_m^j can be formally defined by setting $c_k^j = \delta_{k,m}\ \forall k \in \mathcal{K}^j$ and $d_l^{j'} = 0\ \forall l \in \mathcal{L}^{j'}$, $\forall j' \geq j$, and then recursively performing the inverse wavelet transform up to an arbitrary high level of resolution $J_{\max}$. This procedure results in a scaling function ϕ_m^j sampled at the locations $x_k^{J_{\max}}$. Analogously, second generation wavelet ψ_l^j can be formally defined by assuming $d_m^{j'} = \delta_{j',j}\delta_{l,m}\ \forall l \in \mathcal{L}^j,\ \forall j' \geq j$ and $c_k^j = 0\ \forall k \in \mathcal{K}^j$, and then recursively performing the inverse wavelet transform up to an arbitrary highest level of resolution $J_{\max}$. Now using the linear superposition it is easy to show that on each level of resolution J a function $u(x)$ can be approximated as

$$u^J(x) = \sum_{k\in\mathcal{K}^0} c_k^0 \phi_k^0(x) + \sum_{j=0}^{J-1}\sum_{l\in\mathcal{L}^j} d_l^j \psi_l^j(x). \tag{7}$$

The described wavelet construction can be easily extended to multiple dimensions using tensor product construction, e.g., the three-dimensional wavelets are given by

$$\psi_{i,k,l}^{\mu,j}(\mathbf{x}) = \begin{cases} \psi_i^j(x_1)\ \phi_k^j(x_2)\ \phi_l^j(x_3) & \mu = 1 \\ \psi_i^j(x_1)\ \phi_k^j(x_2)\ \psi_l^j(x_3) & \mu = 2 \\ \psi_i^j(x_1)\ \psi_k^j(x_2)\ \phi_l^j(x_3) & \mu = 3 \\ \phi_i^j(x_1)\ \phi_k^j(x_2)\ \psi_l^j(x_3) & \mu = 5 \\ \phi_i^j(x_1)\ \psi_k^j(x_2)\ \phi_l^j(x_3) & \mu = 6 \\ \phi_i^j(x_1)\ \psi_k^j(x_2)\ \psi_l^j(x_3) & \mu = 4 \\ \psi_i^j(x_1)\ \psi_k^j(x_2)\ \psi_l^j(x_3) & \mu = 7 \end{cases} \tag{8}$$

with three-dimensional scaling function $\phi_{i,k,l}^j(\mathbf{x}) = \phi_i^j(x_1)\ \phi_k^j(x_2)\ \phi_l^j(x_3)$, where $\psi_i^j(x_1)$, $\psi_k^j(x_2)$, $\psi_l^j(x_3)$, $\phi_i^j(x_1)$, $\phi_k^j(x_2)$, $\phi_l^j(x_3)$ correspond to arbitrary one-dimensional wavelets and scaling functions and $\mathbf{x} = (x_1, x_2, x_3)$.

The n-dimensional tensor product second generation wavelets [2,24] are constructed analogously on a set of nested grids

$$\mathcal{G}^j = \left\{\mathbf{x}_\mathbf{k}^j \in \Omega : \mathbf{k} \in \mathcal{K}^j\right\}, \tag{9}$$

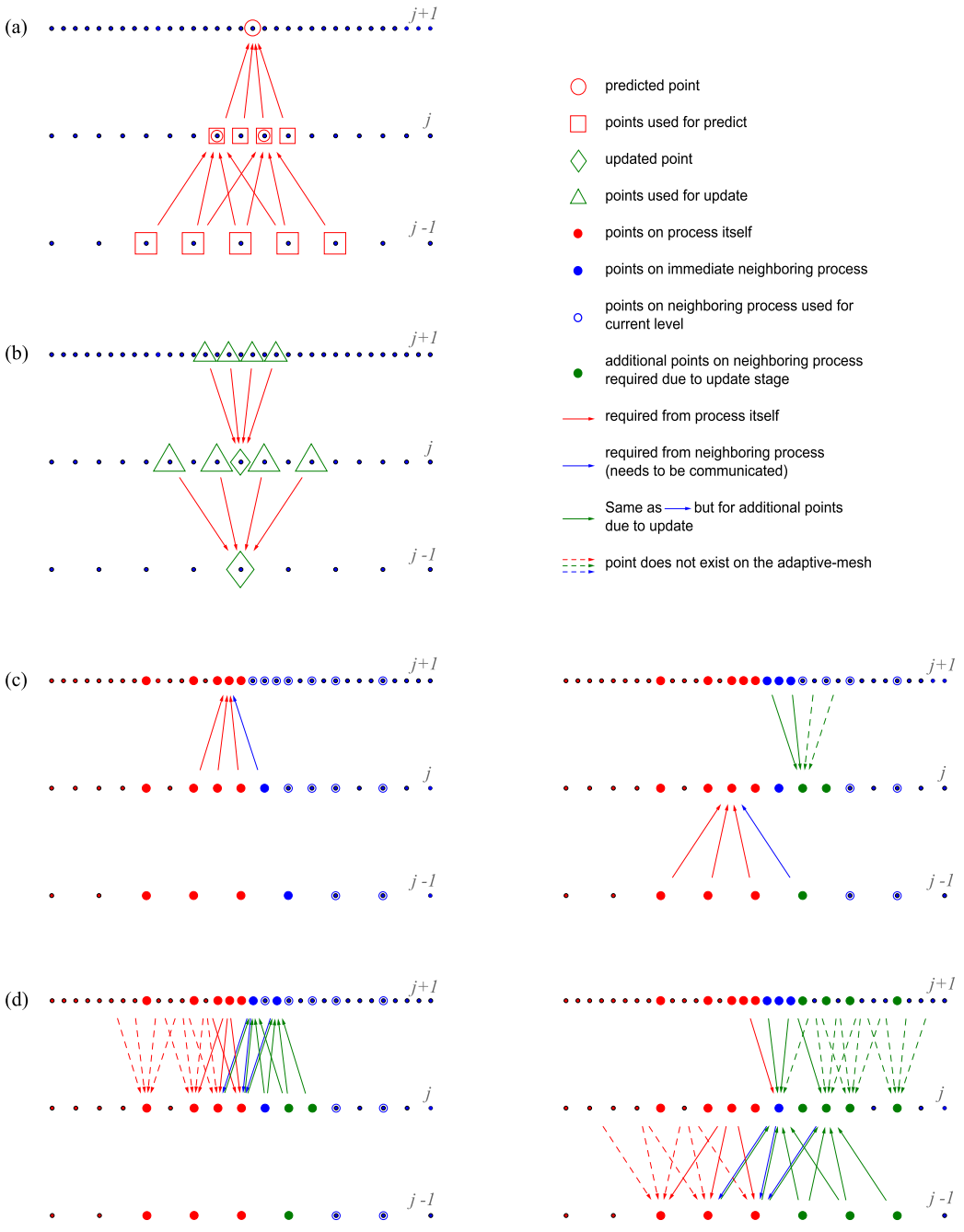

Fig. 1. Illustration of predict and update stages of the forward wavelet transform. (a) Predict stage dependency; (b) Update stage dependency; (c) Predict stages at levels $j+1$ and j; (d) Update stage at levels j and $j-1$. Figure is partially contributed by Scott Reckinger [37].

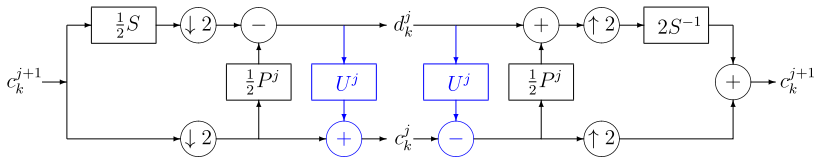

Fig. 2. Block diagram of the second generation wavelet transform. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

where $\mathcal{K}^j$ is some index set associated with scaling functions of level j, $\mathbf{k}=(k_1,\ldots,k_n)$, and the grid points $\mathbf{x}^j_{\mathbf{k}}=(x^j_{1,k_1},\ldots,x^j_{n,k_n})$ are formed by a tensor product of one-dimensional nested grids [2]. Since each individual set of one-dimensional grids is nested ($x^j_{m,k_l}=x^{j+1}_{m,2k_l}$, $m=1,\ldots,n$) the resulting set of n-dimensional grids is also nested, i.e. $\mathcal{G}^j\subset\mathcal{G}^{j+1}$. The main difference for multi-dimensional wavelet construction is that in n dimensions there are 2^n-1 distinctive wavelet families. In addition one step of forward wavelet transform consists of the sequential application of one-dimensional wavelet transform starting from x_1 direction, while the one step of inverse wavelet transform consists of the sequential application of one-dimensional inverse wavelet transform in reverse order starting from x_n direction.

Similarly to one-dimensional case, a function $u(\mathbf{x})$ can be decomposed as

$$u(\mathbf{x})=\sum_{\mathbf{k}\in\mathcal{K}^0}c^0_{\mathbf{k}}\phi^0_{\mathbf{k}}(\mathbf{x})+\sum_{j=0}^{+\infty}\sum_{\mu=1}^{2^n-1}\sum_{\mathbf{l}\in\mathcal{L}^{\mu,j}}d^{\mu,j}_{\mathbf{l}}\psi^{\mu,j}_{\mathbf{l}}(\mathbf{x}), \tag{10}$$

where $\phi^j_{\mathbf{k}}(\mathbf{x})$ ($\mathbf{k}\in\mathcal{K}^j$) and $\psi^{\mu,j}_{\mathbf{l}}(\mathbf{x})$ ($\mathbf{l}\in\mathcal{L}^{\mu,j}$) are respectively n-dimensional tensor product scaling functions and wavelets of different families, $\mathcal{L}^{\mu,j}$ is some index set associated with wavelets of family μ and level j.

3. Asynchronous parallel second-generation wavelet transform

Due to odd–even decoupling of update and predict stages of second generation wavelet transform, the non-adaptive wavelet transform is intrinsically parallel algorithm. A number of the implementations for discrete wavelet transform for both shared memory computers and GPU have been developed and discussed [30,31,33,38–40]. The parallel implementation of non-adaptive wavelet transform for parallel computers with distributed architectures have been developed as well, e.g., [26–29]. Comparing the parallel efficiency of different implementations of parallel non-adaptive discrete wavelet transform algorithms, it was identified that parallel algorithms based on Lifting Scheme (LS) are more efficient than parallel discrete wavelet transform implementations based on Filter Bank (FB) approach, while FB schemes outperform LS algorithms for GPU implementations.

The situation drastically changes for adaptive wavelet transform discussed in Section 5. The main challenge is the load balancing for highly inhomogeneous problems with spatially non-uniform distribution of grid points at each level of resolution, since the algorithms developed for non-adaptive wavelet transform fail, mainly because the parallel non-adaptive algorithms require synchronization of each stage of wavelet transform on each level of resolution, which is very impractical from either data locality or load balancing standpoint. Thus, the efficient parallel adaptive wavelet transform necessitates the development of asynchronous wavelet transform, where the data are communicated only once at the beginning of the wavelet transform.

As explained in the previous section and illustrated in Fig. 1(b), the update stage at each level of resolution for both forward and inverse wavelet-transform necessitates the inclusion of grid points at the higher level of resolution. In order to predict wavelet coefficients at these added points at the higher level of resolution more points on the lower level of resolution need to be included. Figs. 1(c)–(d) show a series of predict, update, and corresponding required extra stages on an adaptive grid to complete forward wavelet transform. Fig. 1(c) is the schematics of predict stages at levels $j+1$ and j separately, while the update stages at levels j and $j-1$ are shown by Fig. 1(d). The sequence illustrated by these four diagrams starts with predicting a point on level $j+1$ belonging to a target process (process red), Fig. 1(c). This predict stage requires four points at level j with one of them belonging to the immediate neighboring process on the right (process blue). Besides, one of the three required points from process red itself belongs to the lower level $j-1$.[2] Since $j-1$ is the lowest level of resolution in this illustration, this point (marked green) needs to be updated as shown in the right diagram of Fig. 1(c). As a result, in order to predict one point on level $j+1$ from process red, four points from process blue are required.

This procedure becomes extremely complicated for the update stage as demonstrated in Fig. 1(d), which shows that "two extra predict stages at level $j+1$ on process blue" are required for "updating four points at level j on process red". By descending the level of resolution, the number of extra stages required for update grows fast: for updating three points at level $j-1$ on process red, two extra predict as well as four extra update steps are necessary at level j. This recursive nature rapidly thickens the set of points required to perform one predict-update sequence.

For problems of arbitrary dimension (greater than one), the wavelet transform is performed by transforming each dimension independently. As the levels of resolution are descended (or ascended in the inverse wavelet transform) in one dimension, the transform is completed over the entire domain at that level of resolution. In order to completely and accurately perform the update stage of the wavelet transform (obtaining the correct c^j_k-values at each level of resolution), the points must be synchronized across subdomain boundaries. The points lying outside of the boundaries of each process are buffer zones added for proper interpolation. These points, which are required to be synchronized, are shown in Figs. 1(c)–(d) by the blue and green markers. As seen in this illustration, the entire set of blue and green points must be communicated to process red in order to perform one complete forward wavelet transform (including both predict and update stages) on an adaptive grid with only one point at the highest level of resolution.

[2] On these diagrams, at each level of resolution, points belonging to the level itself and all levels below are shown.

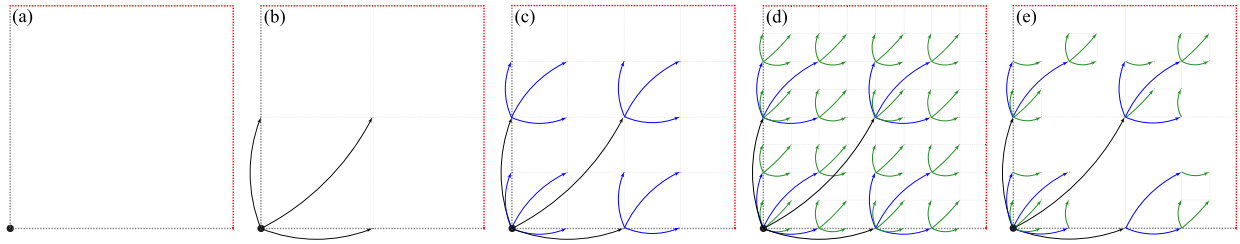

Fig. 3. Possible links from a tree root node for $n = 2$ dimensions. Each level goes to $2^n - 1$ nodes of the next level + itself: totally 2^n. Tree-root is marked by filled black circle. Black links show the first hierarchy ($j = J_{\text{root}} + 1$). Blue ($j = J_{\text{root}} + 2$) and green ($j = J_{\text{root}} + 3$) links show the second and third hierarchical links respectively. The right and top edges belong to the neighboring trees. (a) Empty tree, $j = J_{\text{root}}$; (b) All possible links to completely fill level $j = J_{\text{root}} + 1$; (c) All possible links to completely fill level $j = J_{\text{root}} + 2$; (d) All possible links to completely fill a tree with $J_{\text{root}} = J_{\max} - 3$: a full non-adaptive grid; (e) Only links required for the illustrated adaptive grid of a tree with $J_{\text{root}} = J_{\max} - 3$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Therefore, for asynchronous wavelet transform, i.e. with only one synchronization of the buffer-zone, the wavelet coefficients at the lower level of resolution in the buffer-zone need to be calculated, necessitating the inclusion of grid points at higher level of resolution required for the update stage, which results in synchronization of the entire domain on all processes to complete the update stage. This is impractical because neighboring points on different levels of resolution could end up being on different processes, which drastically complicates the logic of the parallel algorithm plus loses the efficiency. Thus, asynchronous wavelet transform with update stage is impractical for parallel implementation, alternatively, one can perform synchronization at every level of resolution after each update stage and/or predict stage depending on the algorithm as well as for every dimension. Note that the more synchronization stages are implemented, the smaller the size of the buffer zone, where wavelet coefficients need to be synchronized. Therefore, for an n-dimensional problem with $J_{\max}$ levels of resolution, $(2)nJ_{\max}$ communication stages are required to simply perform one forward wavelet transform, where 2 is put in parenthesis depending if synchronization is done for both predict and update stages or just once after update. The inverse wavelet transform requires the same amount of stages resulting in doubling synchronization stages for one time-step while using the AWCM. It is expected that the cost of so much communication could be a bottleneck, not mentioning the difficulties of load balancing at each level of resolution.

Five different parallel extensions have been investigated with the idea that the performance of these different methods would shed light on how to modify the wavelet transform so that its parallel implementation is fully optimized. These extensions include synchronization at each update and predict stages, only at predict stages, as well as modifying wavelet transform so that the wavelets close to the inter-process boundaries do not need to be updated, and finally, skipping the update stage in the entire domain.

It was found that the most efficient solution is to skip the update stage over the entire computational domain. This allows the development of an asynchronous wavelet-transform, i.e. synchronizing the data in the buffer-zone only at the beginning of the transform and performing wavelet transform inside and in the buffer-zone. The ability to perform the wavelet transform in the buffer-zone is guaranteed by the reconstruction check procedure discussed in Section 5. In addition, due to the lack of the update stage, the computational time is also cut down since the algorithm takes half as many steps. Note that the omission of the update stage does not change the convergence properties of adaptive wavelet transform (see Refs. [1,2]), since the order of polynomial interpolation and accuracy of the method is controlled by the predict stage only. The main drawback of not including update stage of wavelet transform is the loss of zero-mean properties of the interpolating wavelet.

To easily/visually address the difference between the serial and the no-update algorithm, the parts that are *not* carried out in the *parallel* algorithm are colored *blue* (gray in print version) in Equations (3)–(6). In the block diagram of the 1D one-step second generation wavelet transform (Fig. 2) also, the part that is *not* carried out in the *parallel* algorithm is colored *blue* (gray in print version).

4. Data structure

A dynamic, arbitrary dimension tree structure database is used in the algorithm presented in this paper: binary in 1D, quad-tree in 2D, and octree in 3D, Fig. 3. This tree data structure has been implemented for wavelet coefficient storage and retrieval. Trees are organized as forward link-list with a deterministic path determined by the global non-adaptive coordinates starting from the root of the tree. Roots of the tree are specified on a given resolution and trees can be empty. For parallel implementation, the trees are the smallest quanta of data (block of data) for the data migration. In other words, each individual tree cannot be broken between processes and only entire blocks of data belonging to the same trees can be moved between processes during domain repartitioning stage.

In order to decrease the number of cache-misses during tree traverse, nodes of each two levels are stored together, starting from the finest level of resolution. To summarize:

1. Nodes are arranged by levels (to simplify access during wavelet transforms) and in a cache friendly manner (to speed up the access),
2. In a tree of dimension n, a node of level j has $2^n - 1$ links to the nodes of higher level $j+1$,
3. Each node of level j is also considered as a node of level $j+1, j+2, \cdots, J_{\max}$.

For indirect data access based on coordinates, the length of the path to a data point at a level $J_{\text{root}} + j$ is j. The cost of data access to all points at a level $J_{\text{root}} + j$ is $(2^n - 1)2^{(j-1)n}\Omega_{\text{root}}(j+1)$, where Ω_{root} is the number of points on the tree root level, J_{root}, i.e. $\Omega_{\text{root}} = \mathbf{M}2^{J_{\text{root}}-1}$. The base grid size of $\mathbf{M} = [m_1 \ldots, m_n]$ and tree root level of J_{root} are user defined input parameters. In the limit when j is large, it can be seen that for non-adaptive case $\Omega \cong \Omega_{\text{root}}2^{jn}$, which implies that $j \cong \frac{1}{n}\log_2 \frac{\Omega}{\Omega_{\text{root}}}$. Therefore, the overall cost of indirect data access to the tree is $O(\Omega \log \Omega)$.

In addition, for faster access, the pointers to the data are stored in an orthogonal list based on wavelet family, location relative to the rectangular computational domain (internal, face, edge, corner), level of resolution, level of derivative calculation. This makes the cost of direct access to the data $O(\Omega)$.

Trees and the corresponding data are orthogonally distributed among processes; however, in order to facilitate the wavelet-transform and the derivative calculations, each process has an identical matrix of tree-roots that are marked by the process-rank where the actual data are stored on. Note that since the choice of the tree-root level affects the size of the tree-root matrix, the length of the tree traversing path, the number of trees and their size for data migration, the optimal value of J_{root} is problem dependent and can be chosen to optimize the computational performance.

All the trees that belong to process itself hereafter are called *internal-zone* trees and their corresponding points are called internal-zone points. All the trees that do not belong to the process itself hereafter are called *buffer-zone* trees and the points belonging to these buffer-zone trees are called buffer-zone points. To distinguish these points, it is convenient to define, for each process $p \in \{0, \cdots, \mathsf{n_p} - 1\}$, a mask $\mathcal{M}_{p,r}$, $r \in \{0, \cdots, \mathsf{n_p} - 1\}$, that consists of points on process p that belong to process r, i.e. the set $\mathcal{M}_{p,p}$ consists of internal-zone points of the process p and the sets $\mathcal{M}_{p,r\neq p}$ are the masks of buffer zone points on the process p. It should be noted that the buffer-zone points are always a subset of the points on the corresponding process, i.e. $\mathcal{M}_{p,r} \subset \mathcal{M}_{r,r}$, and consist of the points that are necessary for the asynchronous wavelet-transform that is discussed in Section 3. Each process should have at least one tree with the possibility of some of them being empty. Each tree-root also has descriptor indicating to which process it belongs.

5. Grid adaptation

The major strength of wavelet decomposition (10) is the ability to compress functions. For functions that contain isolated small scales on a large-scale background (i.e. intermittent functions), most wavelet coefficients are small. Thus, a good approximation can be retained even after discarding a large number of wavelets with small coefficients. Intuitively, the coefficient $d_{\mathbf{l}}^{\mu,j}$ is small unless the $u(\mathbf{x})$ has variation on the scale of j in the immediate vicinity of wavelet $\psi_{\mathbf{l}}^{\mu,j}(\mathbf{x})$. More precisely, if we rewrite (10) as the sum of two terms composed of wavelets whose amplitudes are respectively above and below some prescribed non-dimensional (relative) threshold parameter, ϵ,

$$u(\mathbf{x}) = u_{\geq}(\mathbf{x}) + u_{<}(\mathbf{x}), \tag{11}$$

where

$$u_{\geq}(\mathbf{x}) = \sum_{\mathbf{k}\in\mathcal{K}^0} c_{\mathbf{k}}^0 \phi_{\mathbf{k}}^0(\mathbf{x}) + \sum_{j=0}^{+\infty}\sum_{\mu=1}^{2^n-1}\sum_{\substack{\mathbf{l}\in\mathcal{L}^{\mu,j}\\ |d_{\mathbf{l}}^{\mu,j}|\geq\epsilon\|u\|}} d_{\mathbf{l}}^{\mu,j}\psi_{\mathbf{l}}^{\mu,j}(\mathbf{x}), \tag{12}$$

$$u_{<}(\mathbf{x}) = \sum_{j=0}^{+\infty}\sum_{\mu=1}^{2^n-1}\sum_{\substack{\mathbf{l}\in\mathcal{L}^{\mu,j}\\ |d_{\mathbf{l}}^{\mu,j}|<\epsilon\|u\|}} d_{\mathbf{l}}^{\mu,j}\psi_{\mathbf{l}}^{\mu,j}(\mathbf{x}), \tag{13}$$

$\|\cdot\|$ is the norm that provides the (absolute) dimensional scaling for the filtered variable u, then, following Donoho [41], it can be shown that for a sufficiently smooth function $u(\mathbf{x})$

$$\left\|u(\mathbf{x}) - u_{\geq}(\mathbf{x})\right\|_2 \leq C\epsilon\|u(\mathbf{x})\|, \tag{14}$$

with C of order unity, which can be utilized to actively control the accuracy of the approximation.

Grid adaptation occurs naturally in wavelet methods, e.g. [42,43]. To illustrate the algorithm, let us consider a function $u(\mathbf{x})$, defined on a closed n-dimensional rectangular domain Ω. Relation (14) provides the framework for representing a function with significantly fewer degrees of freedom, while still retaining a good approximation. However, in order to realize all the benefits of the wavelet compression, a function $u_{\geq}(\mathbf{x})$ needs to be reconstructable from the subset of Ω_{S} *significant* grid points, hereafter denoted by mask $\mathcal{M}^S$. We recall that every scaling function $\phi_{\mathbf{k}}^j(\mathbf{x})$, $\mathbf{k} \in \mathcal{K}^j$, is uniquely associated with a grid point $\mathbf{x}_{\mathbf{k}}^j$, while each wavelet $\psi_{\mathbf{l}}^{\mu,j}(\mathbf{x})$, $\mathbf{l} \in \mathcal{L}^{\mu,j}$ is uniquely associated with a corresponding collocation point. Once the

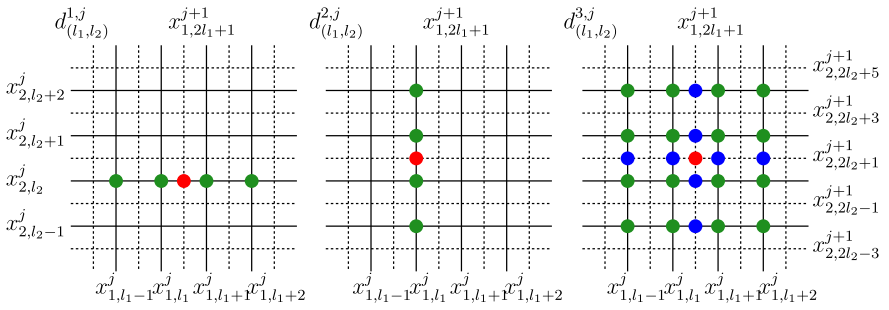

Fig. 4. Ancestry points at the coarser level j (marked ●) and finer level $j+1$ (marked ●) where $c^{j+1}_{\mathbf{k}}$ are needed for calculation of the wavelet coefficient $d^{\mu,j}_{\mathbf{l}}$, $\mu = 1, 3$ (marked ●) for two-dimensional fourth-order ($N = 4$) wavelet transform.

wavelet decomposition is performed, each grid point is uniquely associated either with the wavelet or the scaling function at the coarsest level of resolution. Consequently, the collocation point should be omitted from the computational grid if the associated wavelet is omitted from the approximation. Note that for the stability of the reconstruction algorithm all the grid points associated with the scaling function at the coarsest level of resolution need to be kept, and, thus, are included in mask $\mathcal{M}^S$.

For the asynchronous parallel wavelet transform discussed in Sections 3, following the notation for internal-zone and buffer-zone points, $\mathcal{M}_{p,r}$, introduced in Section 4, the set $\mathcal{M}^S_{p,p}$ will denote all the significant grid points on the process $p \in \{0, \cdots, \mathsf{n_p} - 1\}$. Note that all the sets of significant buffer-zone points are empty, i.e., $\mathcal{M}^S_{p,r\neq p} \equiv \varnothing$. Also note that $\mathcal{M}^S \equiv \bigcup_{p\in\{0,\cdots,\mathsf{n_p}-1\}} \mathcal{M}^S_{p,p}$.

When solving evolution or elliptic problems one should add an additional criterion for grid adaptation, which ensures that the wavelet basis or computational mesh is sufficient to approximate the solution throughout the time integration step for an evolution problem or at the next iteration in the elliptic case. In particular, as suggested by Liandrat and Tchamitchian [42], the computational grid should consist of grid points associated with wavelets whose coefficients are or can possibly become significant during the period of time or iteration when the grid remains unchanged. In actual implementation the adjacent zone includes neighboring wavelets at the same, one above (children), and one below (ancestors) levels of resolution. In other words, at any instant in time or iteration, the computational grid should include points associated with wavelets belonging to an *adjacent zone* of significant wavelets $\mathcal{M}^S$. For convenience of the discussion the combined set of significant and adjacent points is denoted by mask $\mathcal{M}^{S+A}$. Note that the set $\mathcal{M}^S \subset \mathcal{M}^{S+A}$.

For parallel wavelet transform one needs to introduce the combined set of significant and adjacent points for each process p, $\mathcal{M}^{S+A}_{p,r}$. Note that in contrast to the significant buffer-zone set $\mathcal{M}^S_{p,r\neq p}$, which is empty, the set $\mathcal{M}^{S+A}_{p,r\neq p}$ is not empty and consists of buffer-zone points of process p that include wavelets on process r belonging to an adjacent zone of significant internal-zone wavelets $\mathcal{M}^{S+A}_{p,p}$. Also note that $\mathcal{M}^{S+A} \equiv \bigcup_{p\in\{0,\cdots,\mathsf{n_p}-1\}} \mathcal{M}^{S+A}_{p,p}$ and $\mathcal{M}^{S+A}_{p,r} \subset \mathcal{M}^{S+A}_{r,r}$ for any $p, r \in \{0, \cdots, \mathsf{n_p} - 1\}$.

In order to be able to perform wavelet transforms on an adaptive grid, an additional step, hereafter called *reconstruction step procedure*, needs to be performed to ensure that all ancestry grid points required for the recursive computation of the wavelet coefficients $d^{\mu,j}_{\mathbf{l}}$ belonging to the set $\mathcal{M}^{S+A}$ are also available. Due to the recursive nature of the wavelet transform, the points added as a result of reconstruction step procedure are also included into the combined significant and adjacent set $\mathcal{M}^{S+A}$. To illustrate the reconstruction step procedure, let us consider one step of n-dimensional wavelet transform. As discussed earlier, the n-dimensional wavelet transform consists of the sequential application of n one-dimensional wavelet transforms in x_i, $i = 1, \ldots, n$, directions. Hence, in order to find the ancestry grid points necessary for the calculation of the wavelet coefficient $d^{\mu,j}_{\mathbf{l}}$, we start with the collocation point associated with $d^{\mu,j}_{\mathbf{l}}$ and recursively, $i = n, \ldots, 1$, add points that are needed to perform one step of the one-dimensional wavelet transform in the x_i direction at the locations that are added to perform the one-dimensional wavelet transforms in x_l, $l = i+1, \ldots, n$, directions. At the end of this recursive procedure, we will have a minimal set of grid points that are necessary for calculation of wavelet coefficient $d^{\mu,j}_{\mathbf{l}}$ provided that wavelet coefficients at other locations are either zero or negligible (below an a priori prescribed threshold). Fig. 4 illustrates the minimal set of grid points in two dimensions that are necessary for calculation of wavelet coefficient $d^{\mu,j}_{\mathbf{l}}$ belonging to three different families of wavelets, i.e. $\mu = 1, 3$. The perfect reconstruction check procedure guarantees that all wavelet coefficients obtained by performing the wavelet transform on the adapted grid are the same as those found by performing the wavelet transform of $u_{\geq}(\mathbf{x})$ on the complete grid.

In order to be able to perform asynchronous parallel wavelet transform, all the data at the ancestry points for the wavelet-transform at the internal-zone points $\mathcal{M}^{S+A}_{p,p}$ of the process $p \in \{0, \cdots, \mathsf{n_p} - 1\}$ are required, including points that are stored on different processes. The reconstruction check procedure on $\mathcal{M}^{S+A}_{p,p}$ is performed on each process, as it is a serial algorithm, to construct the set of buffer-zone $\mathcal{M}^{S+A}_{p,r\neq p}$. After the reconstruction procedure, the data at the buffer-zone points $\mathcal{M}^{S+A}_{p,r\neq p}$ need to be synchronized among processes to ensure that all ancestry and adjacent buffer-zone points are

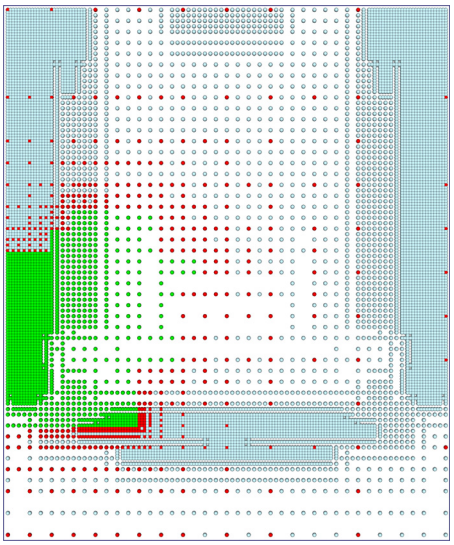

Fig. 5. Parallel Wavelet Transform. Green: one process ($\mathcal{M}^{S+A}_{p,p}$), Red: nodes used in communication (to be synchronized, $\mathcal{M}^{S+A}_{p,r\neq p}$), Blue: rest of processes, i.e., $\bigcup_{q\neq p} \mathcal{M}^{S+A}_{q,q} - \mathcal{M}^{S+A}_{p,r\neq p}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Algorithm 1 Reconstruction Check Procedure (RCP) for the parallel wavelet transform: $\mathcal{M}^{S+A}_{p,p} \xrightarrow{\text{RCP}} \mathcal{M}^{S+A}_{p,r}$.

```
for ∀p = 0 : 1 : n_p − 1
    set M^{S+A}_{p,r≠p} = ∅ at j = J_max
    for all levels j = J_max − 1 : −1 : 1
        extend mask M^{S+A}_{p,r} to include the ancestry points at level j
    end
end
```

communicated to the internal-zone points $\mathcal{M}^{S+A}_{p,p}$, which ensures that $\mathcal{M}^{S+A}_{p,r} \subset \mathcal{M}^{S+A}_{r,r}$. Once sets $\mathcal{M}^{S+A}_{p,r}$ are synchronized, the wavelet-transform on each process can be performed as a regular serial algorithm. These masks, $\mathcal{M}^{S+A}_{p,p}$ and $\mathcal{M}^{S+A}_{p,r\neq p}$, are illustrated in Fig. 5, where they are shown as a subset of an entire mesh $\bigcup_{p\in\{0,\cdots,\mathsf{n_p}-1\}} \mathcal{M}^{S+A}_{p,p}$. Note that for some process r, $\mathcal{M}^{S+A}_{p,r\neq p}$ could be an empty set.

The pseudocode for the parallel perfect reconstruction check procedure is shown in Algorithm 1. At the end of this procedure the masks $\mathcal{M}^{S+A}_{p,r}$ are constructed. These masks can be used to construct for each process $p \in \{0,\cdots,\mathsf{n_p}-1\}$ a set of nested adaptive computational grids $\mathcal{G}^j_{p\geq} = \left\{ \mathbf{x}^j_{\mathbf{k}} \in \Omega : \mathbf{k} \in \mathcal{K}^j, \mathbf{x}^j_{\mathbf{k}} \in \bigcup_{r\in\{0,\cdots,\mathsf{n_p}-1\}} \mathcal{M}^{S+A}_{p,r} \right\}$ such that $\mathcal{G}^j_{p\geq} \subset \mathcal{G}^{j+1}_{p\geq}$ for any $j < J_{\max} - 1$, where $J_{\max}$ is the finest level of resolution present in the approximation (12). Note that grids $\mathcal{G}^j_{p\geq}$ are local on each process.

6. Calculation of derivatives on the adapted grid

The differentiation procedure for obtaining derivatives of a function from its values at collocation points is based on the interpolating properties of second generation wavelets. It is recalled that wavelet coefficients $d^{\mu,j}_{\mathbf{l}}$ measure the difference between the approximation of the function at the $j+1$ level of resolution and its representation at the j level of resolution. Thus, if there are no points in the immediate vicinity of a grid point $\mathbf{x}^j_{\mathbf{k}}$, i.e. $|d^{\mu,j}_{\mathbf{m}}| < \epsilon$ for all the neighboring points, and points $\mathbf{x}^{j+1}_{(2k_1\pm1,\ldots,2k_d\pm1)}$ are not present in $\mathcal{G}^{j+1}_{\geq}$, then there exists some neighborhood of $\mathbf{x}^j_{\mathbf{k}}$, $\Omega^j_{\mathbf{k}}$, where the actual function is well approximated by a wavelet interpolant based on $c^j_{\mathbf{m}}$ ($\mathbf{m} \in \mathcal{K}^j$), i.e.

$$\left| u(\mathbf{x}) - \sum_{\mathbf{m}\in\mathcal{K}^j} c^j_{\mathbf{m}} \phi^j_{\mathbf{m}}(\mathbf{x}) \right| \leq \tilde{C}\epsilon\|u\|, \quad \mathbf{x} \in \Omega^j_{\mathbf{k}}. \tag{15}$$

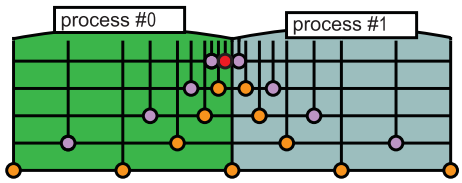

Fig. 6. Illustration of Zones and Masks (includes nodes required for wavelet transform and derivatives) in parallel. Red: wavelet above the threshold ($\mathcal{M}_{0,0}^{S+A}$), Orange: nearest neighbors to capture evolving solution (reconstruction masks $\mathcal{M}_{0,0}^{S+A}$ and $\mathcal{M}_{0,1}^{S+A}$), Violet: nodes required for derivatives (Ghost masks $\mathcal{M}_{0,0}^{G}$ and $\mathcal{M}_{0,1}^{G}$). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

Let us denote by $\mathcal{D}_p^j$ a collection of such points at each level of resolution for each process $p \in \{0, \cdots, \mathsf{n_p} - 1\}$. In other words $\mathcal{D}_p^j$ is the subset of the mask $\mathcal{M}_{p,p}^{S+A}$ where the differentiation is taken at level j, i.e.

$$\mathcal{D}_p^j = \left\{ \mathbf{x}_{\mathbf{k}}^j \in \Omega : \mathbf{x}_{\mathbf{k}}^j \in \mathcal{M}_{p,p}^{S+A},\ \text{differentiation taken at level } j \right\}. \tag{16}$$

Note that $\mathcal{D}_p^j$ is an orthogonal set, $\bigoplus_{j=1}^{J_{\max}} \mathcal{D}_p^j = \mathcal{M}_{p,p}^{S+A}$. The differentiation of the interpolant (15) results in the value of the derivative of the function at the $\mathbf{x}_{\mathbf{k}}^j$ location. Rewriting this interpolant as local Lagrange polynomial of order N, i.e. the same order as the wavelet, differentiating the polynomial, and evaluating it at $\mathbf{x}_{\mathbf{k}}^j$ location would result in local finite difference operator that uses the neighboring points of the interpolant on level j with some of these points, hereafter called *ghost points*, not being present in the union mask $\bigcup_{r\in\{0,\cdots,\mathsf{n_p}-1\}} \mathcal{M}_{p,r}^{S+A}$. If $\mathcal{M}_{p,r}^{S+A+G}$, $r \in \{0, \cdots, \mathsf{n_p} - 1\}$ are sets of points used for calculation of the derivatives at internal-zone points of process p, then in order to evaluate finite difference operators defined by local interpolants one would need to interpolate the function to the ghost points defined by the combined mask $\bigcup_{r\in\{0,\cdots,\mathsf{n_p}-1\}} \mathcal{M}_{p,r}^{S+A+G}$. The pseudocode for the evaluation of derivatives at all grid points is given in Algorithm 2. At the end of this procedure, the derivatives of the function at all grid points are found. The computational cost of calculating spatial derivatives is roughly the same as the cost of forward and inverse wavelet transforms. For the details on the accuracy of this differentiation procedure, the readers are referred to [1,2].

Algorithm 2 Calculation of derivatives on the adapted grid.

```
for ∀p = 0 : 1 : n_p − 1
    perform forward wavelet transform for each component of u_k^m on M_{p,r}^{S+A}
    for all levels j = 1 : 1 : J_max − 1
        perform one step of inverse wavelet transform for level j on M_{p,r}^{S+A+G}
        find derivatives at grid points that belong to D_p^j
    end
end
```

The internal- and buffer-zone masks for significant, adjacent and ghost points are illustrated in Fig. 6, where red-points are $\mathcal{M}_{0,0}^{S+A}$; all the orange-points on the green-background belong to mask $\mathcal{M}_{0,0}^{S+A}$, while all the orange-points on the blue-background belong to the set $\mathcal{M}_{0,1}^{S+A}$; all the violet-points on the green-background are $\mathcal{M}_{0,0}^{G}$, while all the violet-points on the blue-background represent $\mathcal{M}_{0,1}^{G}$.

7. Data migration

The mask of points $\mathcal{M}_{p,r\neq p}^{S+A}$ for either transform or derivative-calculation are constructed separately and then communicated by means of either all-to-all or one-to-one communication techniques.

8. Domain partitioning and dynamic load balancing

Several partitioning approaches with different user controls are implemented, Fig. 7. More advanced Zoltan [44–49] library based partitions provide nearly optimal load balancing, thanks to the asynchronous wavelet transform resulting in linear dependence of computational cost on the number of grid points per process. A summary of the approaches is as follows: For the geometric prime-number partitioning, all spatial directions of the domain are dissected based on prime-number factorization of tree-roots assuming nearly equal distribution of grid points among trees. The major deficiency of this approach is its static nature and poor load balancing for a non-uniform wavelet distribution. For the geometric sequential partitioning, the domain is subdivided by planes normal to an axis on rounded to the nearest integer $\sqrt[n]{\mathsf{n_p}}$ sub-domains, where n is the problem dimension and $\mathsf{n_p}$ is the total number of processes. The available $\mathsf{n_p}$ processes are distributed

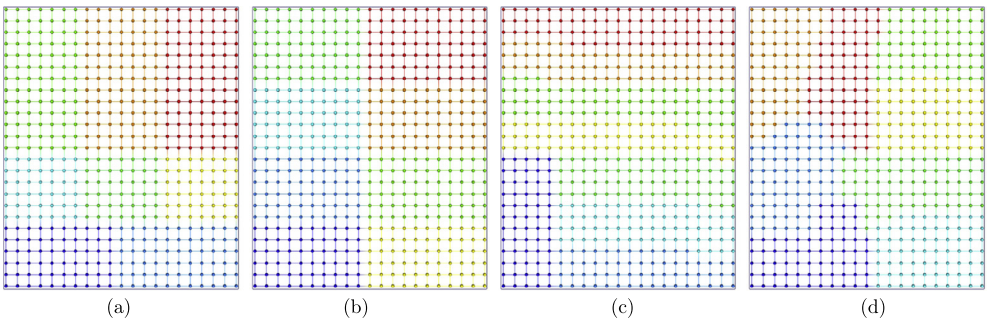

Fig. 7. Domain partitionings: (a) geometric prime-number, (b) geometric sequential, (c) Zoltan geometric, (d) Zoltan hypergraph.

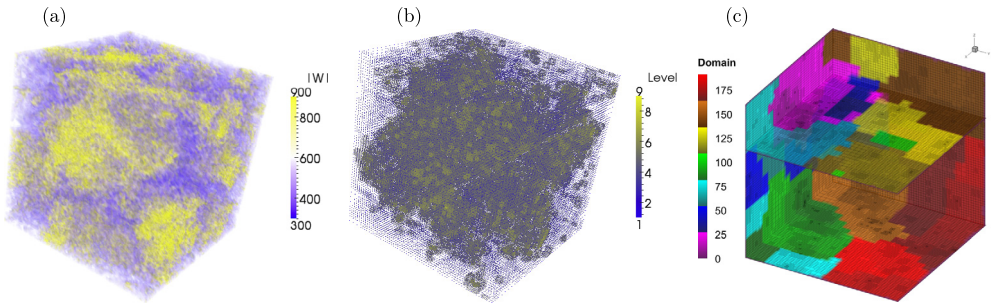

Fig. 8. Coherent Vortex Simulation of linearly forced incompressible homogeneous turbulence at $Re_\lambda = 320$ using PAWCM at effective non-adaptive resolution of $\mathcal{N}_{\max} = 2048^3$: (a) volume rendered vorticity magnitude, (b) the adaptive computational mesh colored by the level of resolution with compression ratio $\mathcal{N}/\mathcal{N}_{\max} = 2.1 \times 10^{-4}$, (c) domain partitioning for Dynamic Load Balancing using Zoltan hypergraph repartitioning for 192 processes.

among these sub-domain according to the number of active wavelets inside each of the sub-domains. This recursion step is repeated n times to get the final partitioning. The load balancing might not quite be optimal, though it may be more usable for uni-directional wavelet distributions across the domain. For significantly non-uniform wavelet distribution, the domain is partitioned using Zoltan partitioning library [44–49] by Sandia National Laboratories. Zoltan geometric (Recursive Coordinate Bisection) (Fig. 7(c)) and Zoltan hypergraph (Fig. 7(d)) parallel partitioning algorithms were used. In both cases, the Zoltan library is supplied with the matrix of computational weights based on the number of grid points in each tree and the communication weights based on the number of grid points on the interface between neighboring trees.

Dynamic load balancing (DLB) is implemented via domain repartitioning during the grid adaptation step and reassigning tree data structure nodes to the appropriate processes. The user provides an imbalance tolerance vector to trigger the repartitioning if necessary. Depending on the imbalance of wavelet distribution, a different kind of repartitioning is performed. Highly imbalanced data are partitioned without considering initial decomposition, moderately imbalanced data are repartitioned while trying to stay close to the current decomposition, and nearly balanced data are refined by small changes only. Figs. 8 and 9 present three dimensional examples of domain partitioning for PAWCM and corresponding flow fields with adaptive computational meshes for Coherent Vortex Simulation [50,51] of linearly forced homogeneous turbulence at effective non-adaptive resolution of $\mathcal{N}_{\max} = 2048^3$ as well as adaptive wavelet-based Direct Numerical Simulation of compressible flow past sphere at $Re = 1000$ and $Ma = 0.7$ at effective non-adaptive resolution of $3713 \times 2305 \times 2305$.

9. Parallel AWCM

Algorithm 3 illustrates major components of both serial and parallel AWCM. In Algorithm 3, the operations that are *only* carried out in the *parallel* algorithm are colored in *blue* (gray in print version). The operator $\xrightarrow{\text{A}}$ is the adjacent (safety) zone inclusion that extends the significant mask on each rank $\mathcal{M}^{S}_{p,p}$ to its corresponding significant + adjacent mask $\mathcal{M}^{S+A}_{p,r}$, which may belong to any rank $r = 0, \cdots, \mathsf{n_p} - 1$. Similarly, the operator $\xrightarrow{\text{RCP}}$ is the reconstruction check procedure on $\mathcal{M}^{S+A}_{p,p}$ to extend it to $\mathcal{M}^{S+A}_{p,r}$.

10. Scalability studies

All parallel scalability studies were performed on the U.S. Air Force Research Laboratory's Spirit SGI ICE X system. Each node consists of a 16-core Intel Xeon E5 Sandy Bridge processor with 32 GB RAM, and the cluster utilizes an FDR 14x Infiniband interconnect. For all results, a single process is run on each processor core. All results presented in this section have been obtained using a linearized Crank–Nicolson time integration method (for details see Ref. [19]). The sustained parallel performance of the code was assessed for 40 time integration/grid adaptation cycles.

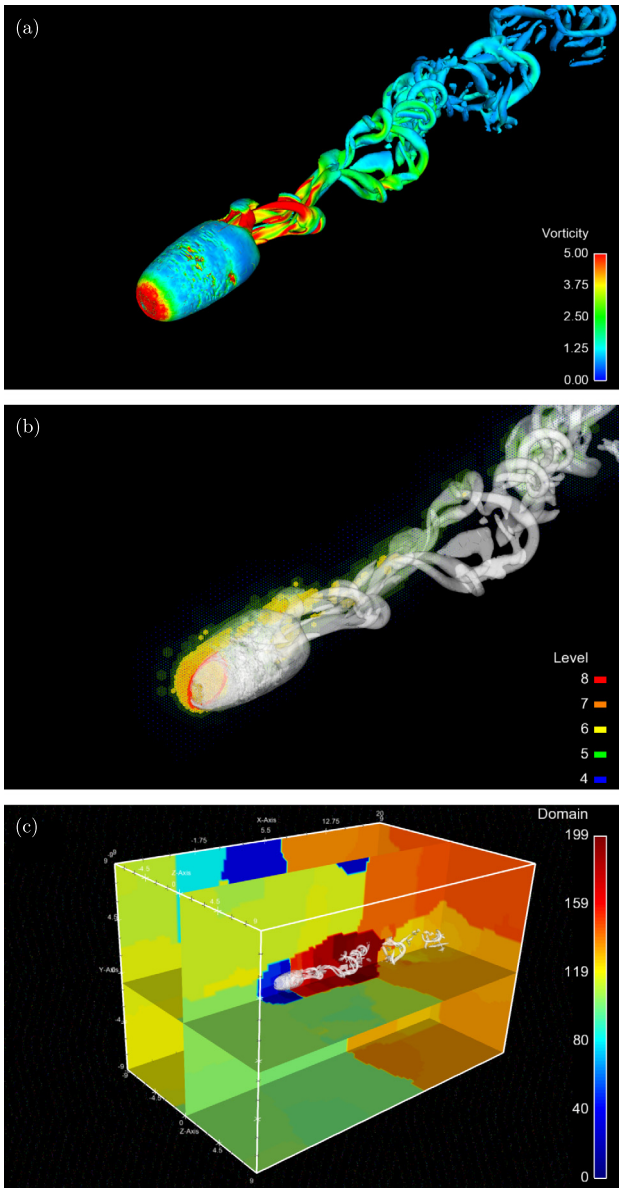

Fig. 9. Adaptive Wavelet-based Direct Numerical Simulation of compressible flow past sphere at $Re = 1000$ and $Ma = 0.7$ using PAWCM with characteristic-based volume penalization [52] at effective non-adaptive resolution of $3713 \times 2305 \times 2305$: (a) main vortical structures in the near wake identified by the iso-surfaces of $Q = 0.25$ colored by the magnitude of vorticity, (b) a slice of wavelet collocation points at higher levels of resolution ($4 \leq j \leq 8$) superimposed by iso-surfaces of main vortical structures, (c) domain partitioning for Dynamic Load Balancing using Zoltan hypergraph repartitioning for 200 processes superimposed by iso-surfaces of main vortical structures. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Strong scalability studies for PAWCM have been performed for velocity-based Coherent Vortex Simulations (CVS) [50,51] of linearly forced incompressible homogeneous turbulence [53,54] at Taylor micro-scale Reynolds number of $Re_\lambda = 320$. The simulations with constant wavelet threshold $\epsilon = 0.1$ were performed on dynamically adaptive computational grid corresponding to effective non-adaptive resolution of 2048^3. The coefficient $C_f = 6.\bar{6}$ for turbulent forcing was used. The time interval is chosen to be smaller than eddy turnover time to ensure that the number of localized vortical structures remain unchanged and the variation of the total number of grid points is small. For the details of the problem formulation and the description of the model parameters, the reader is referred to Ref. [55]. An example of the turbulent flow field, the corresponding adaptive computational mesh and domain partitioning are given in Fig. 8. Strong scalability studies were performed for both geometric prime-number and dynamic Zoltan hypergraph partitionings. Due to high spatial intermittency of turbulent flows, the geometric prime-number partitioning was very imbalanced and is not reported in this paper. The parallel speedup results are shown in Fig. 10 for dynamic Zoltan hypergraph partitioning for both all-to-all and one-to-one

Algorithm 3 Parallel Adaptive Wavelet Collocation Method (PAWCM).

for $\forall p = 0:1:\mathsf{n_p}-1$
  initial guess $(m=0)$**:** $\mathbf{u}_{\mathbf{k}}^{m}$ and $\mathcal{G}_{\geq}^{m}$
end

while $m=0$ **or** $\left\{ m \geqslant 1 \text{ and } \left\{ \underbrace{\left[\mathcal{G}_{\geq}^{m} \neq \mathcal{G}_{\geq}^{m-1} \text{ or } \|\mathbf{u}_{\mathbf{k}}^{m}-\mathbf{u}_{\mathbf{k}}^{m-1}\|_{\infty} > \delta_{\epsilon}\right]}_{\text{only for Elliptic problem}} \text{ or } \underbrace{[t_m < t_{\text{end}}]}_{\substack{\text{only for}\\ \text{time evolution}\\ \text{problem}}} \right\} \right\}$

  request $\mathcal{M}_{p,r\neq p}^{S+A}$ for migration
  perform forward wavelet transform for each component of $\mathbf{u}_{\mathbf{k}}^{m}$
  for $\forall p = 0:1:\mathsf{n_p}-1$
    for all levels $j = J_{\max}:-1:1$
      create a mask $\mathcal{M}_{p,p}^{S}$ for $|d_{\mathbf{l}}^{\mu,j}| \geq \epsilon$
    end
    $\mathcal{M}_{p,p}^{S} \xrightarrow{\text{A}} \mathcal{M}_{p,r}^{S+A}$, $r = 0,\cdots,\mathsf{n_p}-1$
    synchronize mask: $\mathcal{M}_{p,r\neq p}^{S+A}$ send to r process: $\mathcal{M}_{r,r}^{S+A} = \bigcup_{p=0}^{\mathsf{n_p}-1} \mathcal{M}_{p,r}^{S+A}$
    perform the reconstruction check procedure: $\mathcal{M}_{p,p}^{S+A} \xrightarrow{\text{RCP}} \mathcal{M}_{p,r}^{S+A}$
    add ghost mask: $\mathcal{M}_{p,p}^{S+A} \xrightarrow{\bigoplus_{j=1}^{J_{\max}} \mathcal{D}_{p}^{j}} \mathcal{M}_{p,r}^{S+A+G}$. $\mathcal{M}_{p,r}^{S} \subset \mathcal{M}_{p,r}^{S+A} \subset \mathcal{M}_{p,r}^{S+A+G}$
  end
  if imbalanced
    domain-**repartitioning** and **migration** of trees
    clean $\mathcal{M}_{p,r\neq p}^{S+A} = \varnothing$ and $\mathcal{M}_{p,r\neq p}^{S} = \varnothing$
    perform the reconstruction check procedure: $\mathcal{M}_{p,p}^{S+A} \xrightarrow{\text{RCP}} \mathcal{M}_{p,r}^{S+A}$
  end if
  for $\forall p = 0:1:\mathsf{n_p}-1$
    construct $\mathcal{G}_{\geq}^{m+1}$
    if $\mathcal{G}_{\geq}^{m+1} \neq \mathcal{G}_{\geq}^{m}$
      interpolate $\mathbf{u}_{\mathbf{k}}^{m}$ to $\mathcal{G}_{\geq}^{m+1}$
    end if
  end
  Either **Solve** the Elliptic problem (using Local Multilevel Elliptic Solver)
  or **Advance** in Time (using Krylov/RK Time-Integration)
  $m = m+1$
end

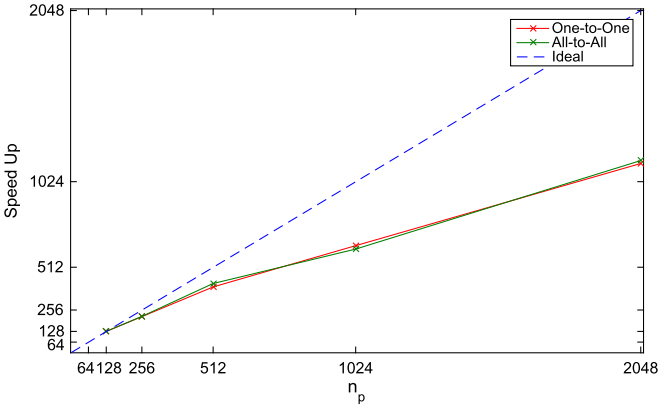

Fig. 10. Parallel Speedup for PAWCM of CVS at $Re_{\lambda} = 320$ on dynamically adaptive computational grid at effective resolution of 2048^3 using dynamic Zoltan hypergraph partitioning with both all-to-all and one-to-one communication methods. Green marker: All-to-all communication. Red marker: One-to-one communication. The base simulation with 128 CPU cores is assumed to perform with the ideal linear speedup. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

communication methods. The results demonstrate the linear scalability up to 512 processes with subsequent noticeable slow down for higher number of processes. Multiple mechanisms could contribute to such slow down, with the most significant being large load imbalance among processes, interprocess communication, and increase of computational cost due to use of buffer-zones in the asynchronous wavelet transform.

In order to assess the effect of interprocess communication on the parallel efficiency of the method, two different mechanisms are implemented: all-to-all and one-to-one MPI communications. It is well known that for all-to-all communication

cost increases significantly with the increase of the buffer size and total number of ranks. The fact that the parallel speedup curves for both all-to-all and one-to-one communication mechanisms are almost identical indicates that the deterioration in the performance of the solver compared to the ideal linear speedup cannot be solely attributed to the communication problems, at least for the number of processes studied.

The next possible mechanism responsible for the deterioration of parallel scalability is the load imbalance, which can be understood by studying the distribution of the grid points among processes, namely the mean, the minimum and maximum bounds, the standard deviation, and the standard deviation bound of the number of active wavelets, $\Omega_{\mathrm{SA_I}}$, $\Omega_{\mathrm{SA_B}}$, and $\Omega_{\mathrm{SA_T}}$, where the subscript SA stands for significant and adjacent points, while subscripts I, B, T respectively denote internal, buffer-zone, and total (internal and buffer zone) points. The statistical data for $\Omega_{\mathrm{SA_I}}$ and $\Omega_{\mathrm{SA_T}}$ grid points, averaged between processes and over 40 time steps, are given in Fig. 11.

All four measures for internal-zone points, $\Omega_{\mathrm{SA_I}}$, are decreasing with the increase in number of processes, while the relative load imbalance measured by the ratio $\overline{\max(\Omega_{\mathrm{SA_I}})}/\overline{\langle\Omega_{\mathrm{SA_I}}\rangle}$ does not change much with the increase of the number of processes. It should be noted, that the relatively constant load imbalance demonstrates that load balancing procedure works and the relative load imbalance actively controlled by the imbalance tolerance discussed in Section 8, which was kept at for all scalability studies. Thus, there is an additional mechanism of deterioration of parallel scalability. As mentioned in Section 5, the requirement of having an asynchronous wavelet transform results in additional operations for significant and adjacent buffer-zone points, effectively increasing the cost of wavelet transform. When the asynchronous wavelet transform is performed, the cumulative computational cost scales as the total number of SA points, $\mathsf{n_p}\langle\Omega_{\mathrm{SA_T}}\rangle = \mathsf{n_p}\left(\langle\Omega_{\mathrm{SA_I}}\rangle + \langle\Omega_{\mathrm{SA_B}}\rangle\right)$, where operator $\langle\cdot\rangle$ denotes averaging across all processes. In the serial case, there is no buffer-zone wavelets, thus, the cost of wavelet transform scales as $\Omega_{\mathrm{SA_{I_{serial}}}} \cong \mathsf{n_p}\langle\Omega_{\mathrm{SA_I}}\rangle$. For well balanced partitioning, the parallel efficiency of the wavelet transform can be defined as the ratio of total number of serial and parallel points, $\Omega_{\mathrm{SA_{I_{serial}}}} / \left(\mathsf{n_p}\langle\Omega_{\mathrm{SA_T}}\rangle\right) = \langle\Omega_{\mathrm{SA_I}}\rangle/\langle\Omega_{\mathrm{SA_T}}\rangle$. In actual calculation, the process with the maximum number $\Omega_{\mathrm{SA_T}}$ points determines the speed of the calculation. Thus, the average efficiency of the parallel wavelet transform over a time interval can be measured as

$$\eta_{\mathsf{p}}^{WT} \cong \frac{\overline{\langle\Omega_{\mathrm{SA_I}}\rangle}}{\overline{\max\left(\Omega_{\mathrm{SA_T}}\right)}}, \tag{17}$$

where operator $\overline{(\cdot)}$ denotes time-averaged value.

It should be noted, that while the number of active wavelets represents the cost of the wavelet-transform, the number of SAG (significant + adjacent + ghost) points, which are used for derivatives calculations, is a better measure of the overall computational time, since, as mentioned in Algorithm 3, the number of SAG points is generally greater than the number of active wavelets: $\mathcal{M}_{p,r}^{S} \subset \mathcal{M}_{p,r}^{S+A} \subset \mathcal{M}_{p,r}^{S+A+G}$. Thus, a more accurate measure of parallel efficiency of the PAWCM can be written as

$$\eta_{\mathsf{p}} \cong \frac{\overline{\langle\Omega_{\mathrm{SAG_I}}\rangle}}{\overline{\max\left(\Omega_{\mathrm{SAG_T}}\right)}}. \tag{18}$$

The statistical data for $\Omega_{\mathrm{SAG_I}}$ and $\Omega_{\mathrm{SAG_T}}$ grid points, averaged between processors and time, are given in Fig. 12. Using these actual data, the theoretical parallel speedup is given by $S = \mathsf{n_p}\eta_{\mathsf{p}}$. The theoretical speedup curve is shown in Fig. 13. Assuming that the base simulation with 128 CPU cores performs at theoretical efficiency, the parallel speedup curves for all-to-all and one-to-one communication showing in Fig. 10 are rescaled and plotted in Fig. 13. This rescaled plot clearly indicates that PAWCM performs at maximum theoretical efficiency up to 512 processes, with slight deterioration of performance for larger number of processes. It is worth pointing out that in the reported theoretical efficiency, the cost associated with parallel communication is not taken into account, i.e. η_{p} assumes zero wall-time for parallel communication. The increase of the number of the buffer-zone points relative to the number of internal-zone points also has an effect on the total size of the send-receive buffers that need to be communicated between processes. As it was discussed in Sections 5 and 6, in order to perform parallel wavelet transform in asynchronous manner, the data at the SA points in the buffer-zone of each process need to be synchronized. The results presented in Fig. 11 demonstrate that with the increase of the number of processes the ratio $\overline{\langle\Omega_{\mathrm{SA_B}}\rangle}/\overline{\langle\Omega_{\mathrm{SA_I}}\rangle}$ increases as number of processes increase, which indicates an increase in the total size of all send-receive buffers, which in turn decreases the communication efficiency.

In the current implementation of the asynchronous wavelet-transform algorithm, all ranks have to wait until the communication is complete among all processes. This is true regardless of all-to-all or one-to-one communication algorithm, i.e. even in one-to-one case, there is barrier at the end of communication. Since the size of the buffer-zone and resulting very large send-receive buffers cannot be reduced, even more efficient one-to-one communication mechanisms will not be able to drastically reduce the communication wall-time to have the algorithm performed without deterioration of parallel efficiency. One possible way to further improve the scalability of the PAWCM and to make it work at the theoretical parallel efficiency is to remove the communication barrier by changing the way wavelet transform is performed. In the current asynchronous algorithm, all ranks should wait for the communication until each rank can start the wavelet transform on its corresponding mask $\mathcal{M}_{p,r}^{S+A}$; however, while the communication is executing each rank indeed can perform the wavelet transform on a subset of points $\mathcal{M}_{p,p}^{S+A}$ that does not require points form $\mathcal{M}_{p,r\neq p}^{S+A}$. Furthermore, the data communication can be done in multiple stages starting from buffer-zone data on the highest level of resolution for the process required for the one stage

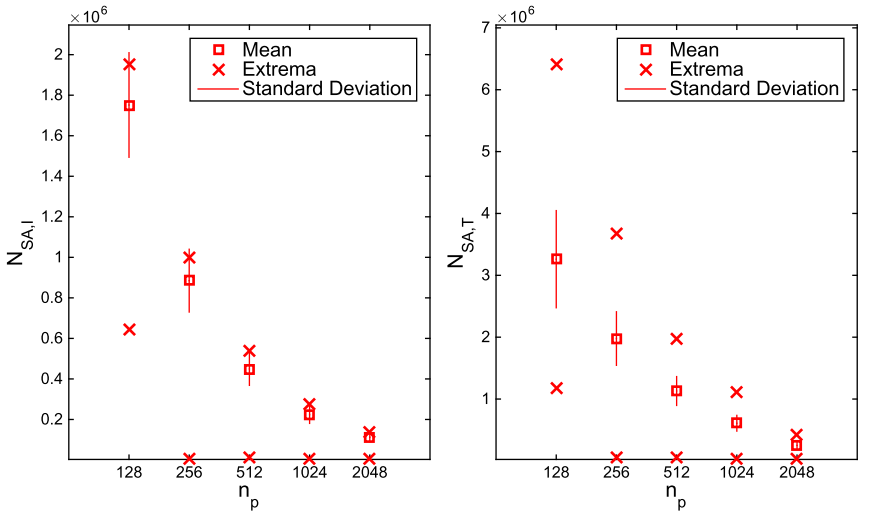

Fig. 11. Statistics of $\overline{\min(\Omega_{SA_I})}$, $\overline{\max(\Omega_{SA_I})}$, $\overline{\langle\Omega_{SA_I}\rangle}$ and $\overline{\min(\Omega_{SA_T})}$, $\overline{\max(\Omega_{SA_T})}$, $\overline{\langle\Omega_{SA_T}\rangle}$ for PAWCM of CVS at $Re_\lambda = 320$ and 2048^3 effective resolution using dynamic Zoltan hypergraph partitioning. Vertical lines indicate the range encompassing ± 1 standard deviations.

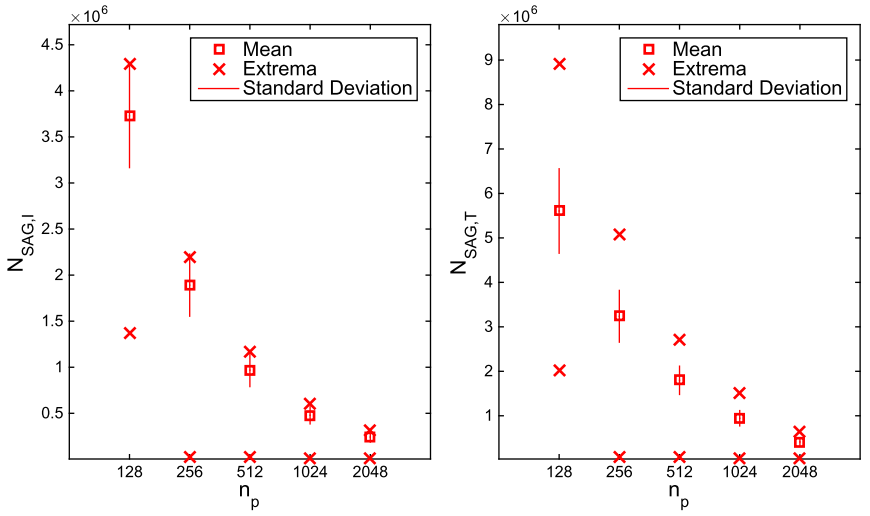

Fig. 12. Statistics of $\overline{\min(\Omega_{SAG_I})}$, $\overline{\max(\Omega_{SAG_I})}$, $\overline{\langle\Omega_{SAG_I}\rangle}$ and $\overline{\min(\Omega_{SA_T})}$, $\overline{\max(\Omega_{SAG_T})}$, $\overline{\langle\Omega_{SAG_T}\rangle}$ for PAWCM of CVS at $Re_\lambda = 320$ and 2048^3 effective resolution using dynamic Zoltan hypergraph partitioning. Vertical lines indicate the range encompassing ± 1 standard deviations.

of wavelet transform in one dimension, recursively send and receive data for each direction and level of resolution, while performing wavelet transform on the subset of internal-zone and/or buffer-zone points that either do not require the use of buffer-zone points, which are not received yet. The use of such multi-stage scheduling algorithms removes or postpones the effect of the communication barrier on the wavelet transform, while migration is occurring and would ensure that the wavelet transform engine is not idle during the migration. Different scheduling strategies are currently under investigation and will be discussed elsewhere.

11. Conclusions

A parallel adaptive wavelet-based method for simulation of partial differential equations on arbitrary physical space bounded within a rectangular computational domain without limitation on simply-connected partitioning is presented. The method is a parallel extension of adaptive wavelet collocation method with no update stage. The data are stored in tree like structure, which provides a robust data management. An efficient parallel implementation is achieved by developing an asynchronous parallel wavelet transform, which allows one to perform wavelet transform and derivative calculations on each process with only one data synchronization at the highest level of resolution. Both static and dynamic domain partitioning approaches are developed. For the dynamic domain partitioning, trees are considered to be the minimum quanta of data to be migrated between the processes. This allows fully automated and efficient handling of non-simply

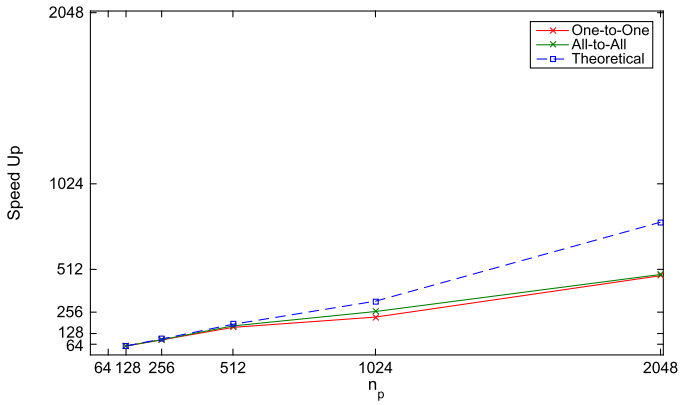

Fig. 13. Comparison of theoretical and actual parallel speedup for PAWCM of CVS at $Re_\lambda = 320$ on dynamically adaptive computational grid at effective resolution of 2048^3 using dynamic Zoltan hypergraph partitioning with both all-to-all and one-to-one communication methods. The base simulation with 128 CPU cores is assumed to perform at theoretical efficiency. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

connected partitioning of a computational domain. Dynamic load balancing is achieved via domain repartitioning during grid adaptation step and reassigning trees to the appropriate processes to ensure approximately the same number of nodes on each process. The detailed strong scalability study is reported for PAWCM of Coherent Vortex Simulations of homogeneous turbulence with linear forcing at Taylor micro-scale Reynolds number of $Re_\lambda = 320$ on dynamically adaptive computational grid at effective resolution of 2048^3 using as many as 2048 CPU cores. Two main mechanisms for saturation of strong scalability are identified: the increase of the total number of buffer-zone points and parallel communication. The theoretical limit of parallel efficiency due to additional operations performed in buffer-zone required for asynchronous parallel wavelet transform is identified. Possible future improvements of the algorithm that will minimize the communication cost and eliminate or postpone saturation are discussed.

The successful outcome of a longtime effort to develop a novel parallel adaptive wavelet collocation method for numerical solution of PDEs has been completed during the course of the current work. Even though the currently developed parallel algorithm is capable of performing very large scale simulations, we envision that further improvements in the parallel wavelet-transform algorithm as well as the buffer-zone communication and scheduling strategies can further improve the efficiency of the method.

Acknowledgements

This work was supported by NSF under grants Nos. CBET-0756046 and CBET-1236505. This support is gratefully acknowledged. EBD and OVV also acknowledge partial support from ONR under grant No. N00014-11-1-069. Authors thankfully acknowledge the use of TeraGrid, XSEDE, sharcnet, and AFRL DSRC computational resources. This work also utilized extensively the Janus supercomputer, which is supported by the National Science Foundation (award number CNS-0821794) and the University of Colorado Boulder. The Janus supercomputer is a joint effort of the University of Colorado Boulder, the University of Colorado Denver and the National Center for Atmospheric Research. Authors acknowledge the initial efforts of Scott Reckinger in studying different possibilities for the efficient asynchronous parallel extension of the second generation wavelet transform.

References

[1] O.V. Vasilyev, C. Bowman, Second generation wavelet collocation method for the solution of partial differential equations, J. Comput. Phys. 165 (2000) 660–693.
[2] O.V. Vasilyev, Solving multi-dimensional evolution problems with localized structures using second generation wavelets, Int. J. Comput. Fluid Dyn.: Special issue on High-Resolution Methods in Computational Fluid Dynamics 17 (2003) 151–168.
[3] J.D. Regele, O.V. Vasilyev, An adaptive wavelet-collocation method for shock computations, Int. J. Comput. Fluid Dyn. 23 (2009) 503–518.
[4] O.V. Vasilyev, N.K.-R. Kevlahan, An adaptive multilevel wavelet collocation method for elliptic problems, J. Comput. Phys. 206 (2005) 412–431.
[5] N.K.R. Kevlahan, J.M. Alam, O.V. Vasilyev, Scaling of space–time modes with Reynolds number in two-dimensional turbulence, J. Fluid Mech. 570 (2007) 217–226.
[6] Q. Liu, O.V. Vasilyev, A Brinkman penalization method for compressible flows in complex geometries, J. Comput. Phys. 227 (2007) 946–966.
[7] D.E. Goldstein, O.V. Vasilyev, N.K.R. Kevlahan, CVS and SCALES simulation of 3D isotropic turbulences, J. Turbul. 6 (2005) 1–20.
[8] O.V. Vasilyev, G. De Stefano, D.E. Goldstein, N.K.R. Kevlahan, Lagrangian dynamic SGS model for stochastic coherent adaptive large eddy simulation, J. Turbul. 9 (2008) 1–14.
[9] G. De Stefano, O.V. Vasilyev, D.E. Goldstein, Localized dynamic kinetic energy-based models for stochastic coherent adaptive large eddy simulation, Phys. Fluids 20 (2008) 045102.1–045102.14.
[10] G. De Stefano, O.V. Vasilyev, Stochastic coherent adaptive large eddy simulation of forced isotropic turbulence, J. Fluid Mech. 646 (2010) 453–470.
[11] G. De Stefano, O.V. Vasilyev, A fully adaptive wavelet-based approach to homogeneous turbulence simulation, J. Fluid Mech. 695 (2012) 149–172.

[12] A. Nejadmalayeri, A. Vezolainen, O. Vasilyev, Reynolds number scaling of coherent vortex simulation and stochastic coherent adaptive large eddy simulation, Phys. Fluids 25 (2013) 110823.
[13] A. Nejadmalayeri, A. Vezolainen, G. De Stefano, O.V. Vasilyev, Fully adaptive turbulence simulations based on lagrangian spatio-temporally varying wavelet thresholding, J. Fluid Mech. 749 (2014) 794–817.
[14] O.V. Vasilyev, S. Paolucci, A fast adaptive wavelet collocation algorithm for multi-dimensional PDEs, J. Comput. Phys. 125 (1997) 16–56.
[15] S.J. Reckinger, D. Livescu, O.V. Vasilyev, Adaptive wavelet collocation method simulations of Rayleigh–Taylor instability, Phys. Scr. T142 (2010) 1–6.
[16] S. Reckinger, O. Vasilyev, B. Fox-Kemper, Adaptive volume penalization for ocean modeling, Ocean Dyn. 62 (2012) 1201–1215.
[17] J. Regele, D.R. Kassoy, O.V. Vasilyev, Effects of high activation energies on acoustic timescale detonation initiation, Combust. Theory Model. 16 (2012) 650–678.
[18] O.V. Vasilyev, N.K.R. Kevlahan, Hybrid wavelet collocation – Brinkman penalization method for complex geometry flows, Int. J. Numer. Methods Fluids 40 (2002) 531–538.
[19] N.K.R. Kevlahan, O.V. Vasilyev, An adaptive wavelet collocation method for fluid–structure interaction at high Reynolds numbers, SIAM J. Sci. Comput. 26 (2005) 1894–1915.
[20] O.V. Vasilyev, D.A. Yuen, S. Paolucci, The solution of PDEs using wavelets, Comput. Phys. 11 (1997) 429–435.
[21] O.V. Vasilyev, Y.Y. Podladchikov, D.A. Yuen, Modeling of viscoelastic plume–lithosphere interaction using adaptive multilevel wavelet collocation method, Geophys. J. Int. 147 (2001) 579–589.
[22] O.V. Vasilyev, Y.Y. Podladchikov, D.A. Yuen, Modeling of compaction driven flow in poro-viscoelastic medium using adaptive wavelet collocation method, Geophys. Res. Lett. 25 (1998) 3239–3242.
[23] W. Sweldens, The lifting scheme: a custom-design construction of biorthogonal wavelets, Appl. Comput. Harmon. Anal. 3 (1996) 186–200.
[24] W. Sweldens, The lifting scheme: a construction of second generation wavelets, SIAM J. Math. Anal. 29 (1998) 511–546.
[25] A. Grossmann, J. Morlet, Decomposition of hardy functions into square integrable wavelets of constant shape, SIAM J. Math. Anal. 15 (1984) 723–736.
[26] F. Marino, V. Piuri, E. Swartzlander, A parallel implementation of the 2-d discrete wavelet transform without interprocessor communications, IEEE Trans. Signal Process. 47 (1999) 3179–3184.
[27] O. Nielsen, M. Hegland, Parallel performance of fast wavelet transforms, Int. J. High Speed Comput. 11 (2000) 55–74.
[28] P. Gonzalez, J. Cabaleiro, T. Pena, Parallel computation of wavelet transforms using the lifting scheme, J. Supercomput. 18 (2001) 141–152.
[29] R. Kutil, A. Uhl, Parallel adaptive wavelet analysis, Future Gener. Comput. Syst. 18 (2001) 97–106.
[30] C. Tenllado, J. Setoain, M. Prieto, L. Pinuel, F. Tirado, Parallel implementation of the 2d discrete wavelet transform on graphics processing units: filter bank versus lifting, IEEE Trans. Parallel Distrib. Syst. 19 (2008) 299–310.
[31] A. Garcia, H. Shen, GPU-based 3d wavelet reconstruction with tileboarding, in: 13th Pacific Conference on Computer Graphics and Applications, Macao, Peoples Republic of China, Oct. 12–14, 2005, Vis. Comput. 21 (2005) 755–763.
[32] M. Strengert, M. Magallon, D. Weiskopf, S. Guthe, T. Ertl, Large volume visualization of compressed time-dependent datasets on GPU clusters, in: Symposium on Parallel Graphics and Visualization, Grenoble, France, Jun. 2004, Parallel Comput. 31 (2005) 205–219.
[33] V. Galiano, O. Lopez, M.P. Malumbres, H. Migallon, Parallel strategies for 2d discrete wavelet transform in shared memory systems and GPUs, J. Supercomput. 64 (2013) 4–16.
[34] D. Rossinelli, B. Hejazialhosseini, D.G. Spampinato, P. Koumoutsakos, Multicore/multi-GPU accelerated simulations of multiphase compressible flows using wavelet adapted grids, SIAM J. Sci. Comput. 33 (2011) 512–540.
[35] S. Paolucci, Z.J. Zikoski, T. Grenga, WAMR: an adaptive wavelet method for the simulation of compressible reacting flow. Part II. The parallel algorithm, J. Comput. Phys. 272 (2014) 842–864.
[36] A. Nejadmalayeri, Hierarchical multiscale adaptive variable fidelity wavelet-based turbulence modeling with Lagrangian spatially variable thresholding, Ph.D. thesis, University of Colorado Boulder, Boulder, CO, 2012.
[37] S. Reckinger, Personal communication, 2012.
[38] W.J. van der Laan, A.C. Jalba, J.B.T.M. Roerdink, Accelerating wavelet lifting on graphics hardware using cuda, IEEE Trans. Parallel Distrib. Syst. 22 (2011) 132–146.
[39] V. Galiano, O. Lopez-Granado, M.P. Malumbres, L.A. Drummond, H. Migallon, GPU-based 3d lower tree wavelet video encoder, EURASIP J. Adv. Signal Process. (2013).
[40] J. Franco, G. Bernabe, J. Fernandez, M. Ujaldon, The 2d wavelet transform on emerging architectures: GPUs and multicores, J. Real-Time Image Process. 7 (2012) 145–152.
[41] D.L. Donoho, Interpolating wavelet transforms, Technical Report 408, Department of Statistics, Stanford University, 1992.
[42] J. Liandrat, P. Tchamitchian, Resolution of the 1D regularized Burgers equation using a spatial wavelet approximation, Technical report, NASA Contractor Report 187480, ICASE Report 90-83, NASA Langley Research Center, Hampton, VA 23665-5225, 1990.
[43] A. Harten, Adaptive multiresolution schemes for shock computations, J. Comput. Phys. 115 (1994) 319–338.
[44] E. Boman, K. Devine, L.A. Fisk, R. Heaphy, B. Hendrickson, V. Leung, C. Vaughan, U. Catalyurek, D. Bozdag, W. Mitchell, Zoltan home page, http://www.cs.sandia.gov/Zoltan, 1999.
[45] E. Boman, K. Devine, L.A. Fisk, R. Heaphy, B. Hendrickson, C. Vaughan, U. Catalyurek, D. Bozdag, W. Mitchell, J. Teresco, Zoltan 3.0: parallel partitioning, load-balancing, and data management services; Developer's guide, Tech. Report SAND2007-4749W, Sandia National Laboratories, Albuquerque, NM, 2007, http://www.cs.sandia.gov/Zoltan/dev_html/dev.html.
[46] E. Boman, K. Devine, L.A. Fisk, R. Heaphy, B. Hendrickson, C. Vaughan, U. Catalyurek, D. Bozdag, W. Mitchell, J. Teresco, Zoltan 3.0: parallel partitioning, load-balancing, and data management services; User's guide, Tech. Report SAND2007-4748W, Sandia National Laboratories, Albuquerque, NM, 2007, http://www.cs.sandia.gov/Zoltan/ug_html/ug.html.
[47] U. Catalyurek, E. Boman, K. Devine, D. Bozdag, R. Heaphy, L. Riesen, Hypergraph-based dynamic load balancing for adaptive scientific computations, in: Proc. of 21st International Parallel and Distributed Processing Symposium, IPDPS'07, IEEE, 2007, Best Algorithms Paper Award.
[48] K. Devine, E. Boman, R. Heaphy, B. Hendrickson, C. Vaughan, Zoltan data management services for parallel dynamic applications, Comput. Sci. Eng. 4 (2002) 90–97.
[49] K.D. Devine, E.G. Boman, R.T. Heaphy, R.H. Bisseling, U.V. Catalyurek, Parallel Hypergraph Partitioning for Scientific Computing, IEEE, 2006.
[50] M. Farge, K. Schneider, N.K.-R. Kevlahan, Non-Gaussianity and coherent vortex simulation for two-dimensional turbulence using an adaptive orthogonal wavelet basis, Phys. Fluids 11 (1999) 2187–2201.
[51] D.E. Goldstein, O.V. Vasilyev, Stochastic coherent adaptive large eddy simulation method, Phys. Fluids 16 (2004) 2497–2513.
[52] E. Brown-Dymkoski, N. Kasimov, O.V. Vasilyev, A characteristic based volume penalization method for general evolution problems applied to compressible viscous flows, J. Comput. Phys. 262 (2014) 344–357.
[53] T. Lundgren, Linearly forced isotropic turbulence, in: Annual Research Briefs, 2003, pp. 461–473.
[54] C. Rosales, C. Meneveau, Linear forcing in numerical simulations of isotropic turbulence: physical space implementations and convergence properties, Phys. Fluids 17 (2005) 1–8.
[55] G. De Stefano, O.V. Vasilyev, Stochastic coherent adaptive large eddy simulation of forced isotropic turbulence, J. Fluid Mech. 646 (2010) 453–470.