Parallel Adaptive Boundary Layer Meshing for CFD Analysis

Aleksandr Ovcharenko[1], Kedar Chitale[2], Onkar Sahni[1], Kenneth E. Jansen[2], and Mark S. Shephard[1]

[1] Scientific Computation Research Center (SCOREC), Rensselaer Polytechnic Institute, 110 8th St, Troy, NY 12180 shurik@scorec.rpi.edu
[2] University of Colorado at Boulder, Boulder, CO 80309 kedar.chitale@colorado.edu

Summary. This paper describes a parallel procedure for anisotropic mesh adaptation with boundary layers for use in scalable CFD simulations. The parallel mesh adaptation algorithm operates with local mesh modification operations developed for both unstructured and boundary layer parts of the mesh. The adaptive approach maintains layered elements near the viscous walls and accounts for the mesh modification operations that are carried out in parallel on a distributed mesh. In the process mesh relationships and approximations with respect to curved complex 3D geometries of interest are properly maintained. The parallel mesh adaptation procedures are applied to two problems: a heat transfer manifold and a scramjet engine.

Key words: parallel mesh adaptation, boundary layer mesh, semi-structured mesh, parallel adaptive viscous flow simulations

1 Introduction

Adaptive meshing provides a powerful tool for addressing the creation of optimal meshes for problems such as fluid flows in which highly anisotropic solution features can develop and only be located and resolved through adaptivity [1–4]. It is well known that uniformly refined meshes will not yield the desired levels of solution accuracy at an adequate computational cost and that adaptive methods provide an effective means to create meshes that will yield the requested solution quality at acceptable costs [1,5,6]. One approach to the development of these adaptive meshes is to modify a given mesh to match an anisotropic mesh size field, where such a mesh size field is derived from error correction indicators evaluated based on a computed solution [5–10].

Many physical problems of interest involve directional solution features, for example, boundary layers which form near the walls in viscous flows or shocks in high-speed flows. In such cases the application of anisotropic mesh adaptation will yield meshes that provide the same level of accuracy with

over an order of magnitude fewer elements than isotropically adapted ones. In the case of viscous flow simulations the degree of mesh anisotropy required can lead to element aspect ratios of much greater than 1000 to 1. In these regions carefully constructed boundary layers that consider the physics of the flow and the abilities of the flow models used result in the most effective means to construct the local portions of the mesh [11–14]. The best method of construction of such boundary layer meshes is to decompose the mesh in an unstructured mesh in the "plane of the surface" and a graded and structured mesh in the normal direction [11, 12, 15–17]. Layered structure of elements in boundary layer meshes near no-slip walls plays a critical role. It has been shown that maintaining a graded stack of boundary layer elements results in accurate prediction of wall and near-wall quantities (such as wall shear stress or turbulent eddy viscosity) [18]. Thus, it is crucial that the mesh adaptation maintains the structured nature of the mesh normal to the surface.

Even applying the best mesh adaptation procedures, the meshes required have millions to billions of elements. Such meshes can only be solved using massively parallel computers [19–22]. To effectively execute such simulations the mesh adaptation procedures must operate in parallel on the same computer as the flow solution using the same form of parallel decomposition which is commonly represented as a partitioned mesh [20, 21, 23].

Techniques addressing parallel anisotropic mesh adaptation with boundary layers are considered in [24–26]. Reference [24] presents a method with refinement and coarsening implemented on mixed element meshes. However, the coarsening is done through parent entity recovery of previously subdivided elements. Thus, coarsening cannot be applied to create elements larger than initial mesh. Also, derefinement constraints are limited in dealing with higher anisotropy and complex geometry curvatures. The approach in [25] uses parallel volume mesh generation to fill the holes with elements after surface parts have been remeshed in parallel. Remeshing techniques generate meshes with satisfactory resolution to capture required anisotropy and well conform to geometrical boundaries, however parallel mesh generation is a time consuming process, especially on the shared part boundaries of a mesh. It also introduces complexities in the transfer of solution fields from one mesh to another. The work discussed in [26] is similar to the approach considered in this paper.

This work outlines a procedure for parallel anisotropic mesh adaptation with boundary layers based on a set of local mesh modification operations. It is a parallelization of mesh adaptation technique previously presented and proved the robustness of the method with the analysis of simulation results [11,12]. The advantage of this method is the ability to handle curved complex 3D geometries while being able to achieve a desired degree of anisotropy with inexpensive solution transfer process. To address boundary layer specifications, the issues related to manipulation and maintenance of boundary layer stacks are considered for a partitioned mesh that is distributed across a number of parts on a parallel computer. The paper describes the importance of the mesh modification operations of swapping and node reposition, in addition to

refinement and coarsening, which allow matching the requested anisotropic size field and increasing the mesh quality. Finally, the paper studies the scalability of parallel mesh adaptation approach on different 3D geometries.

The organization of the paper is as follows. Section 2 describes the basic anisotropic mesh adaptation concepts for the meshes with boundary layers. Section 3 discusses the parallel implementation of the mesh modification operations. Section 4 shows the application of parallel anisotropic mesh adaptation procedures to two CFD problem cases.

Nomenclature

$\{M^d\}$	the set of topological mesh entities of dimension d. $d = 0$: vertex, $d = 1$: edge, $d = 2$: face, and $d = 3$: region.
M_i^d	the ith mesh entity of dimension d.
$\{M_i^d \{M^D\}\}$	the set of mesh entities of order D adjacent to M_i^d.

2 Anisotropic mesh adaptation with boundary layers

2.1 Meshes with boundary layers

A common method to construct boundary layer meshes, referred to as the advancing layers method [12, 15, 17], inflates the unstructured surface mesh on no-slip walls, where boundary layers form. This inflation is done into the volume, along the local surface normals as a structured stack of layers, up to a specified distance and fills the rest of the domain with unstructured elements. Problem cases presented in this study mostly involve attached flow and therefore, boundary layer elements fully cover no-slip surfaces with attached flow. However, the procedures described can deal with problem cases in which flow separation is encountered on a portion of a no-slip surface, where boundary layer elements will partially cover the surface. Similarly, in portions of a no-slip surface where flow anisotropy is absent or marginal, boundary layer elements can be automatically and adaptively decimated.

Boundary layers consist of stacks of prisms (or wedges) near no-slip walls with the majority of the remaining domain being unstructured tetrahedral mesh. The only additional element type introduced is a small number of pyramids that are used to cover four sided faces of boundary layer prisms that are exposed to the interior of the domain due to trimming of specific boundary layer stacks, for example, near sharp corners [17].

The boundary layer stack is defined as an ordered set of higher-order dimension entities, where the first entity in the set is instantiated from a model boundary, and each of the next entities in the set are connected with a single lower-order entity. The last entity in a boundary layer stack is one that is connected to a higher-order entity which does not belong to any boundary layer stack. Thus, the top-most entity of a boundary layer is exposed to unstructured regions and shares corresponding lower-order entities with them.

The boundary layer structure is decomposed as the product of a layer surface (2D) and thickness (1D) mesh [12]. The mesh composed of triangles located at the top of each layer is referred to as layer surface, while the lines orthogonal to the wall composed of edges are called growth curves (see Figure 1(a)). The edges that belong to layer surfaces are referred to as layer edges and ones that reside on growth curves are called growth edges. Each layer of elements is formed with the help of layer surfaces above and below and with growth edges in between. The height of each layer is referred to as layer thickness whereas the collective height of all layers is referred to as total thickness. The number of vertices (or edges) on growth curves determine its level. The vertices on walls from which growth curves originate are referred to as originating vertices. The top most layer of the stack of boundary layer elements shares an interface with the unstructured volume mesh, where the interior tetrahedral or prismatic elements are referred to as interface elements.

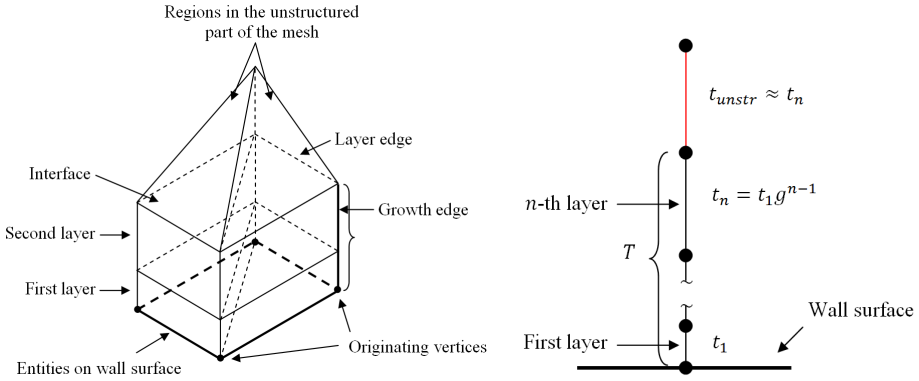

Fig. 1. Decomposition of a boundary layer stack (a) and a growth curve (b)

The vertex spacing on each growth curve determines the local normal resolution. The important parameters in defining the correct spacing of growth curve vertices for further interaction and spacing with unstructured regions are the spacing of the first vertex off the wall t_1, the gradation factor g which holds the spacing between vertices on the growth curve, the number of layers n, and the total thickness of the boundary layer T (see Figure 1(b)).

Given initial values of t_1, g and n, each spacing for the corresponding i-th vertex on the growth curve is calculated as:

$$t_i = t_1 g^{i-1} \tag{1}$$

The total thickness for the growth curve is computed as:

$$T = t_1 \sum_{i=1}^{n} g^{i-1}. \tag{2}$$

t_1 has essential information for computing the remaining parameters in Eq. (1) and Eq. (2). T points to the top-most vertex location where the

boundary layer is connected to the unstructured mesh. Note that g is a fixed constant for a particular growth curve, however it can vary from one growth curve to another as it is dictated by the problem boundary definition. Meanwhile, spacing of the last growth edge of the growth curve t_n is used to verify the ratio between the last growth edge on a boundary layer and the normal height in the unstructured mesh region immediately outside the boundary layer t_{unstr} (see Figure 1(b)). t_{unstr} is a meaningful value to compare with t_n since it is dictated by the requested mesh metric and the difference in the two should provide an acceptable gradation of the mesh at that location.

2.2 Mesh metric field

Mesh adaptivity requires size field specification that matches the desired element size in all directions [1]. At each point of the space, the computed metric field is represented by a 3D symmetric definite positive tensor $T(P)$ such that the desired directional mesh edge length distribution at this point follows an ellipsoidal surface. The transformation $XT(P)X^T = 1$, where $X = \{x_1, x_2, x_3\}$ is a coordinate vector, defines a mapping of an ellipsoid in the physical space into a unit sphere in the metric space. Since mesh metric field represents the transformation that maps an ellipsoid into a unit sphere, any tetrahedron that satisfies the mesh metric field in the unstructured part of the mesh should be a unit equilateral tetrahedron in the transformed space [1] (see left of Figure 2).

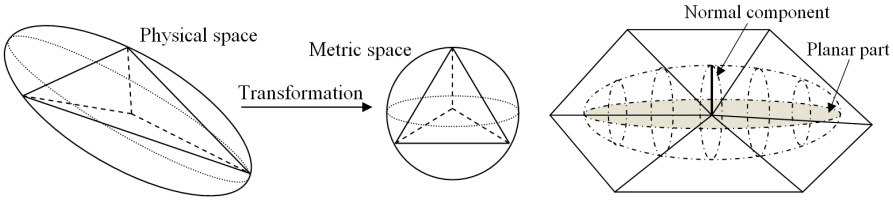

Fig. 2. The transformation associated with a mesh metric tensor (left) and its decomposition for boundary layer vertices (right)

Meanwhile, in order to align mesh metric decomposition with the boundary layer structure specification, at any vertex of a boundary layer the ellipsoid associated with the metric tensor is decomposed into a planar part, which dictates in-plane size resolution control, and a normal component which influences the the stack height adjustment [11, 12]. Figure 2 (right) illustrates the decomposition of mesh metric field for the boundary layer vertices.

2.3 Cavity concept to localize mesh modification operations

Given a mesh metric and the criteria which controls how well the size and shape of mesh entities satisfy the prescribed size field values, mesh adaptation is accomplished by applying mesh modification operations following a specific logic to create the desired mesh that remains conforming to the geometry.

During the execution of a mesh modification operation it is important to clearly identify the set of mesh entities that will be affected by that modification. This set of mesh entities defines a subdomain referred to as a cavity. For the effective parallel implementation of mesh modification operations the mesh modification cavity needs to be large enough that the mesh outside of the cavity boundary is not altered.

In a 3D domain, a single cavity C is comprised of regions adjacent to a mesh entity that triggers a specific mesh modification operation and has a dimension less than 3. The set of outer faces in closure of regions localized in C forms the cavity boundary which stays intact during the required altering within the cavity. The application of a local mesh modification operation then is a retriangulation of the cavity, C, into a new mesh subdomain S, which has the same cavity boundary as C, but contains a different set of regions compared to original set of regions in C.

2.4 Mesh modification operations in the semi-structured part of the mesh

There are four major operators used for mesh modification [1,12,27], namely: split, collapse, swap and vertex repositioning. In addition, there are compound operators applied in the unstructured part of the mesh, which chain multiple single step operators in the unstructured mesh in such a manner to effectively yield the desired mesh configuration. The interested reader is referred to [1] for a more detailed description of mesh modifications in the unstructured part of the mesh. This paper is focuses on mesh modification procedures associated with the boundary layer part of the mesh.

To preserve the layered nature of boundary layer elements along the normals, mesh modification for the layered part of the mesh is divided into two steps [12]: layer surface modification and thickness adjustment. Surface mesh modification operations are propagated through the stack of boundary layer entities and affect all the layer surfaces along with the interface entities within a stack in the same way. The local mesh modification operations of edge split, collapse and swap are utilized to perform the surface mesh adaptation while vertex repositioning is applied to adjust the layer thicknesses and move growth curve vertices to the geometric model boundaries. The thickness adjustment is currently in the development stage, however it is easily integrated with the set of procedures implemented to support boundary layer mesh modifications.

The layer edge split operation splits edges in the boundary layer stack and applies subdivisions in the interface to the unstructured mesh. If M_I^1 is the layer edge to be split in the stack of N boundary layer edges, then the cavity associated with it consists of a set of unique regions $\left\{\bigcup_{i=1}^{N}\left\{M_i^1\left\{M^3\right\}\right\}\right\}$, where $I \in [1..N]$. Figure 3(a) illustrates the layer edge split example.

The layer edge collapse operation is performed on stacks that contain short edges in the local mesh metric in a manner that avoids oscillation between

collapse and split operations [11,12]. The edge collapse operations can only be applied when the affected unstructured mesh entities at the top of the stack also remain valid. If $\left\{(M_i^1, M_i^0)\right\}$ are the pairs of stacks of N layer edges to be collapsed and their correspondent vertices being deleted, the cavity associated with the collapse operator is $\left\{\bigcup_{i=1}^{N}\left\{M_i^0\left\{M^3\right\}\right\}\right\}$ with a deletion of a set of regions $\left\{\bigcup_{i=1}^{N}\left\{M_i^1\left\{M^3\right\}\right\}\right\}$. Figure 3(b) shows two boundary layers and interface elements before and after the layer edge collapse operation.

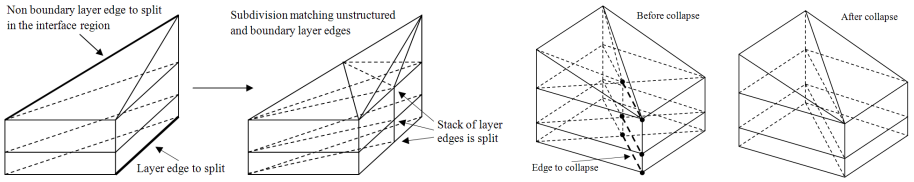

Fig. 3. Example of boundary layer split (a) and collapse (b) operations

The layer edge swap operation changes the connectivity of neighboring boundary layer stacks. In contrast to tetrahedral cavity for edge swap operation, which is reconfigured based on the equatorial plane, there is only one other possible configuration in case of layer edge swap for layer faces. If $\left\{M_i^1\right\}$ is the stack of N layer edges to be swapped, then the layer edge swap operation retriangulates the cavity $\left\{\bigcup_{i=1}^{N}\left\{M_i^1\left\{M^3\right\}\right\}\right\}$ with new layer edges being introduced inside the cavity and deletion of M_i^1. Figure 4(a) gives an example of layer edge swap operation.

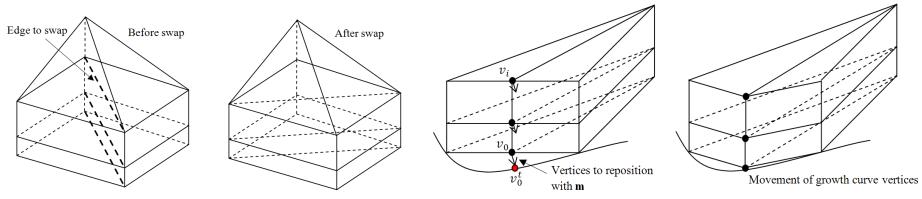

Fig. 4. Example of boundary layer swap (a) and vertex reposition (b) operations

When edge splits are applied to boundary layer edges on curved wall surfaces, the newly introduced vertices must be moved to the curved boundary to maintain the proper geometric approximation. All the vertices in the growth curve should gradually move following the correspondent originating vertices with the help of the movement vector [12] which is determined based on the originating vertex target location as: $\mathbf{m} = (\mathbf{v_0^t} - \mathbf{v_0})$, where $\mathbf{v}_0^t$ is the target location of originating vertex on the curved boundary and $\mathbf{v}_0$ is the current locations of the originating vertex. The movement vector $\mathbf{m}$ is then applied to all the vertices of the growth curve to provide layer vertex repositioning

operation. The procedure first evaluates target locations for vertices on all the growth curves, with each vertex's target location calculated as: $\mathbf{v}_i^t = \mathbf{v}_i + \mathbf{m}$, where $\mathbf{v}_i$ is the current i-th vertex location on a growth curve corresponding to its originating vertex location $\mathbf{v}_0$. It then moves vertices to their computed target locations as depicted in Figure 4(b). Similar to unstructured vertex projection to the curved boundary, layer vertex movement through simple repositioning is not always possible as it may introduce inverted elements, in which case local mesh modification operations are applied to the interior volume mesh to make the way for repositioning to be successful. After the repositioning is completed for the top most vertices, the rest of the growth curve vertices are moved to their target locations $\mathbf{v}_i^t$ resulting in originating vertex being placed on its target location on the curve boundary.

2.5 Procedure of anisotropic mesh adaptation with boundary layers

The mesh adaptation is executed in three stages: mesh coarsening, iterative mesh refinement, and shape correction [10, 12]. The first two stages are controlled by mesh edge length analysis in the transformed space, whereas the third stage is dictated by both element quality and mesh edge length control.

The coarsening stage eliminates the majority of edges which are shorter than requested by the local mesh size field. An advantage to applying the coarsening first is that it makes the traversals required during mesh adaptation faster and limits the peak memory usage.

The second stage refines mesh regions based on splitting mesh edges if they are longer when measured in the transformed space. It also places newly created boundary vertices on the model boundary and coarsens any new short mesh edges introduced.

Shape correction routines improve the shape of poorly shaped entities in the transformed space. Those entities are modified using swap / collapse / vertex reposition and compound operators [10] to obtain the best possible element quality while preserving the correct edge length in the metric space.

Since each pyramid is easily decomposed into two tetrahedra, the mean ratio [28] is used to measure the quality of unstructured regions in the transformed space, namely:

$$\eta' = 15552 \frac{V'^2}{\left(\sum_{i=1}^{6} l'^2_i\right)^3}, \tag{3}$$

with V' and l'^2_i being the volume and edge length square of a tetrahedron in the transformed space. For the boundary layer part of the mesh, layer face quality control is performed, where each layer face is measured as follows:

$$S' = 48 \frac{A'^2}{\left(\sum_{i=1}^{3} l'^2_i\right)^2}, \tag{4}$$

where A' is the layer edge face area in the transformed space.

3 Parallel implementation

3.1 Distributed mesh representation

The execution of parallel mesh adaptation is based on the fact that a single mesh is distributed [29–31] to a number of parts that consist of a set of mesh entities assigned to the corresponding processor. Each part is treated as a serial mesh with the addition of mesh part boundaries to describe groups of mesh entities that are on inter-part boundaries.

The implementation of parallel boundary layer mesh adaptation is aided by requiring all mesh entities in a stack to be on the same part. This process is supported by employing the entity set concept [32] in which the mesh regions in a stack are put in a single set that must be properly maintained during mesh modification and migration [33]. To provide proper partitioning the set is represented as one weighted node during graph partitioning.

The direct consideration of cavities on the part boundary for such mesh modification operations as collapse and swap is a complex and expensive procedure since it leads to a number of communication steps to properly update the parts with the mesh modification decisions carried out. Thus, regions from such cavities are localized on one processor. To support cavity localization, a mesh migration is used, where all regions and stacks of regions involved in the mesh modification operation are migrated onto a single part [20, 29].

The Flexible Distributed Mesh Data Base [29] is used to support the needed mesh-based parallel operations on distributed meshes including part boundary management and mesh migration. The Zoltan library [34] is used for dynamic load balancing. The Inter-Processor Communication Manager is used [35] for efficient parallel communications between processors.

3.2 Refinement and vertex reposition to geometrical boundaries

Mesh edges and their adjacent mesh faces on part boundaries are subdivided the same way as it is done in serial since replicated faces across part boundaries have their bounding edges and vertices in the same order such that the entities properly match [20, 36]. Note that triangular faces can be split using any combination of edges tagged for refinement, whereas quadrilateral faces as part of the boundary layer stack, are limited to be bisected in a layer surface direction only, without subdivision of growth curve edges.

Each subdivision of an entity on the part boundary is matched with the same subdivision on the part the entity is shared with. The inter-part links between newly created mesh entities are appropriately updated across the part boundary [36]. The algorithm involves the same logic of updating inter-part links for both boundary layer and unstructured parts of the mesh. The only difference for the boundary layer procedure is that once the stack of quadrilateral faces exposed to the part boundary is split, each has to be completely updated with the pair or copy on another part.

After refinement is completed, each part holds a list of mesh vertices thatneed to be projected onto the solid model boundaries [20]. For the boundary layer part of the mesh, the newly created originating vertices are projected onto the solid model surfaces with the help of the movement vector as described in Section 2.4. In parallel, the migration might be involved, to complete the local mesh modification procedures in order to move the vertices in the corresponding stacks for the correct vertex adjustment.

3.3 Coarsening and surface optimization

Layer edge collapse operation is performed on the on-part localized cavity. The vicinity of boundary layer stacks sharing vertices of the same growth curve from a specific originating vertex M_d^0 to its associated top most growth curve vertex $M_{d_{top}}^0$ and their adjacent layer edges shorter than the desired size in a metric space are checked for the layer edge collapse. If neither of the stack of layer edges adjacent to $M_d^0..M_{d_{top}}^0$ can be collapsed locally, the boundary layer coarsening procedure migrates all the boundary layers and interface regions adjacent to growth curve vertices from M_d^0 to $M_{d_{top}}^0$ to one part and checks for the possibility of layer edge collapse operation with the full local cavity.

Figure 5 shows the example of layer edge collapse operation requesting mesh migration. It can be seen from the picture that the top-most vertex $M_{d_{top}}^0$ for surrounding boundary layers is located on the part boundary and layer edge collapse for its adjacent edges cannot be evaluated. Thus, $M_{d_{top}}^0$ requests all the adjacent boundary layer stacks and interface regions to be migrated onto the part P_2 such that the layer edge collapse operation can be executed. Different line colors indicate shared entities between specific parts.

At the end of mesh adaptation, the boundary layer surface optimization step is provided [12]. The logic for the optimization operation on the part boundary is similar to the layer edge collapse one as it acquires the cavity associated with the operation locally on the part. The difference for the surface optimization compared to coarsening is that the procedure calculates shapes of layer faces (see Eq. (4)) and determines which operation should be applied to the adjacent layer edges in order to improve the boundary layer faces quality.

4 Application results

The capabilities of the parallel anisotropic mesh adaptation with boundary layers developed in this work are demonstrated with two CFD applications. In the first case, simulations were performed on a heat transfer manifold and mesh convergence was studied in terms of pressure drops between inlet and multiple outlets. The other case involves a scramjet simulation of the NASA CIAM [37] scramjet geometry. In the heat transfer manifold problem domain the CFD analysis is performed by PHASTA [38]. The scramjet geometry was simulated with FUN3D [39] solver.

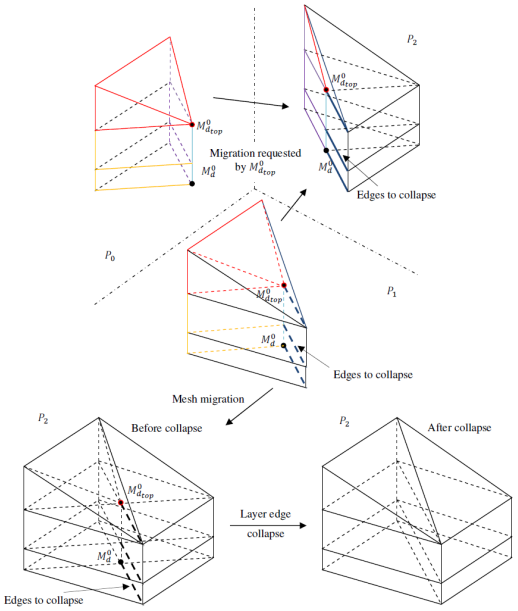

Fig. 5. Example of parallel layer edge collapse operation involving mesh migration

The studies have been executed on Hopper Cray XE6 [40] at National Energy Research Scientific Computing Center. It is configured with 2 twelve-core AMD 2.1 GHz processors per node, with separate L3 caches and memory controllers, 32 GB or 64 GB DDR3 SDRAM per node. Hopper has Gemini interconnect with 3D torus topology. The 3D torus topology provides powerful bisection and global bandwidth characteristics as well as support for dynamic routing of messages.

4.1 Parallel anisotropic boundary layer adaptivity on a heat transfer manifold case

The heat transfer manifold consists of a large diameter cylindrical pipe for the inflow, a plate and twenty small outflow pipes (see Figure 6). Flow computations are done using steady, incompressible RANS equations along with Spalart-Allmaras turbulence model. No-slip boundary conditions are prescribed on viscous surfaces and a natural pressure is assumed at outflows. Hessian matrix field based on computed solution [11] is used to form the mesh metric size field in order to drive mesh adaptation procedures.

The adaptive simulation consists of a flow solve and adaptation, and is carried out twice with the flow solve started from previous solution. Each cycle was divided into 1000 time steps with a constant time step size of $0.1s$. The initial mesh consists of 3M regions, the first adapted boundary layer mesh has 16M regions, and the second adapted mesh results in 81M regions. The initial, first and second adapted inflow pipe mesh cuts with the pressure distribution are shown on Figure 7.

The initial mesh is too coarse and the figures demonstrate its inability to capture the flow phenomena accurately. The stagnation point and the fillets of the inflow pipe are refined which reflects in smoother and more accurate adaptive solution results. The walls of the manifold, especially the wall closest to the inflow pipe, are refined to a higher degree. With adaptivity, accurate predictions are obtained.

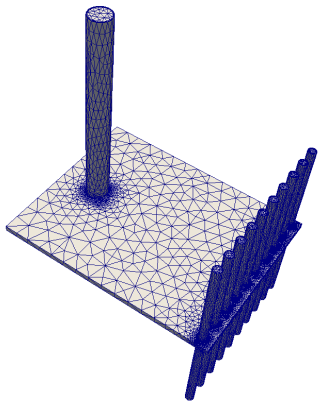

Fig. 6. Initial mesh for the heat transfer manifold test case

To measure the parallel performance results with the strong scaling studies, a second cycle mesh adaptation tests were executed on 256 to 4,096 processors. The scaling is based on the execution time on 256 processors and defined as $(n_{proc-base} * time_{base})/(n_{proc-test} * time_{test})$. All available cores per node were requested during the adaptation runs. Table 1 gives the scaling of second cycle mesh adaptation run times with the initial mesh of 16M regions, and the final one consisting of 81M regions.

The analysis part of a simulation defines the number of cores the particular problem is being executed on. It makes sense to run mesh adaptation routines on the same number of cores since bringing the mesh and size field to a smaller number of cores and repartition it again to a bigger number after mesh adaptation is done is an expensive procedure. Meanwhile, in order to guarantee faster simulation time, mesh modification routines should perform a reasonable scaling.

Table 1. Mesh adaptation run times and scaling for the manifold simulation

N/proc	256	512	1024	2048	4096
Time	1194.34	785.44	514.45	421.09	339.38
Scaling	1	0.76	0.58	0.36	0.22

As indicated in Table 1 the mesh adaptation times decrease with the increased number of cores. Since there is little computation performed during mesh adaptation relative to the substantial increase in communications required as the given mesh is distributed to more processors, the scaling decreases on high core counts (note that a strong scaling study is performed and therefore, the problem size is the same). However, the analysis have been shown to scale strongly with the similar amount of work load provided [21,41]. The fact that mesh modification routines are able to scale on bigger core count with more entities involved into communication supports the statement that it is likely to at a minimum provide the equivalent scaling with more work load on the same number of parts.

Note that the work required for mesh modifications in both unstructured and boundary layer parts of the mesh is proportional, and the overall mesh adaptation took 0.7% of the adaptation cycle time on 256 processors, which is relatively small compared to time taken by the analysis step. Therefore, even with the loss of strong scaling in mesh adaptation step, it is reasonable to run it on the same number of processors together with the flow solver, delivering the massively parallel adaptive boundary layer mesh simulation capabilities to billions of elements.

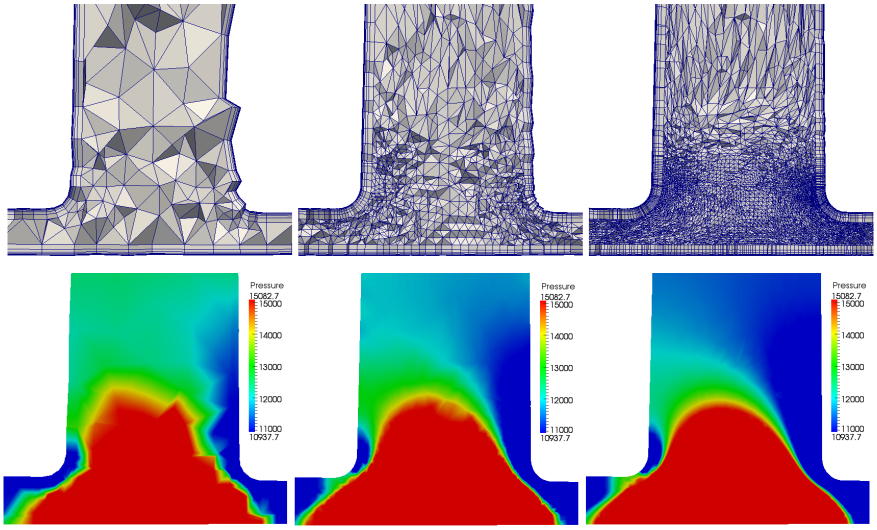

Fig. 7. Initial (left), first (middle) and second (right) adapted meshes and pressure distribution for the cut of the inflow pipe and the manifold

4.2 Parallel anisotropic boundary layer adaptivity on a scramjet geometry

The NASA CIAM [37] scramjet case was run with a freestream Mach number of 6.2, and a freestream reference temperature of 203.5 Kelvin. The initial mesh has 2.86M regions and its slice is depicted in Figure 8. Hessian reconstruction based on Mach number was used to get the mesh size field.

Three adaptation cycles were required. The first adapted mesh had 7.2M regions, the second adapted mesh consisted of 16M regions, and the third adapted mesh had 43M regions. Figure 9 presents first and second adapted mesh zoomed at the tip of the scramjet geometry, whereas Figure 10 shows the mesh and Mach number contour plots at the cowl lip on the inlet of the combustor area for the initial and two adapted meshes.

The flow solution resolution is greatly improved through the use of anisotropic mesh adaptation. The second adapted mesh captures the shock far better as compared to the initial mesh. In the far field upstream of the

shock, where flow is uniform and parallel, the mesh was appropriately coarsened. Expected mesh refinement was obtained at the tips, at the cowl lip, within the combustor area, at the sharp edges of the combustor area liner, and behind the engine. Mesh anisotropy followed the shock emanating from the nose cone tip, with coarsening in the direction tangential to the shock.

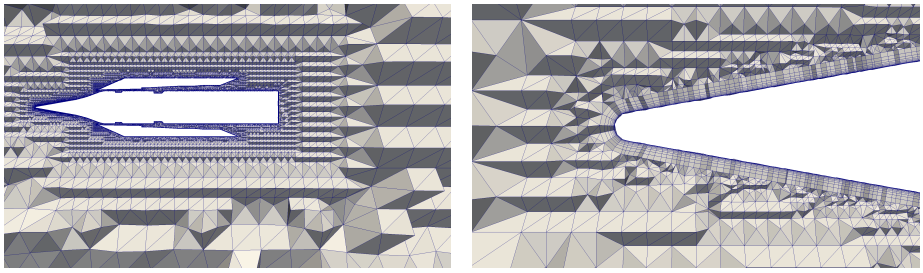

Fig. 8. Slice of the whole (left) initial mesh and the zoom to its tip (right) for the scramjet test case

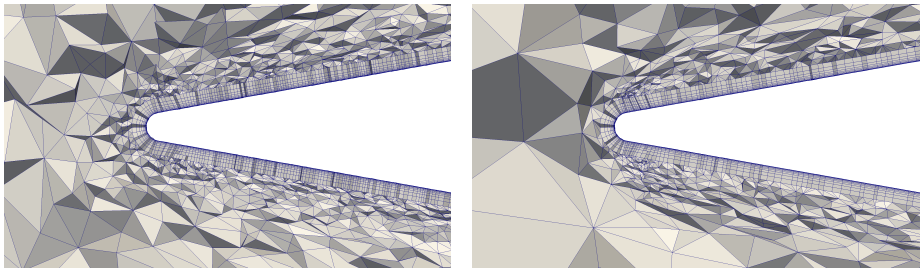

Fig. 9. Scramjet tip during the first (left) and second (right) adaptation cycles

Table 2 provides timings and strong scaling values for the mesh adaptation runs in the third adaptive cycle with input mesh of 16M regions, and the final one consisting of 43M regions. The strong scaling studies were performed on 128 to 4,096 processors. The scaling is based on the execution time on 128 processors and is calculated the same way it is defined in the heat transfer manifold test case.

Table 2. Mesh adaptation run times and scaling for the scramjet test case

N/proc	128	256	512	1024	2048	4096
Time	957.80	532.29	339.39	202.17	136.83	90.85
Scaling	1	0.9	0.7	0.59	0.44	0.33

Table 2 supports the observation that the mesh adaptation is able to decrease run times with the growing number of cores. Although the simulation study experiences the fixed size problem phenomena on a bigger processor count observed in the previous test case, the scalability factors show better

parallel performance of the scramjet test case compared to the heat transfer manifold one. Note that mesh adaptation took 1.2% of the adaptation cycle time on 128 processors with relatively equal amount of computation needed for mesh modifications in both unstructured and boundary layer parts of the mesh, and therefore, is significantly small compared to time taken by analysis step or flow solver. Again, a massively parallel boundary mesh adaptation capability is able to perform efficient large-scale simulations.

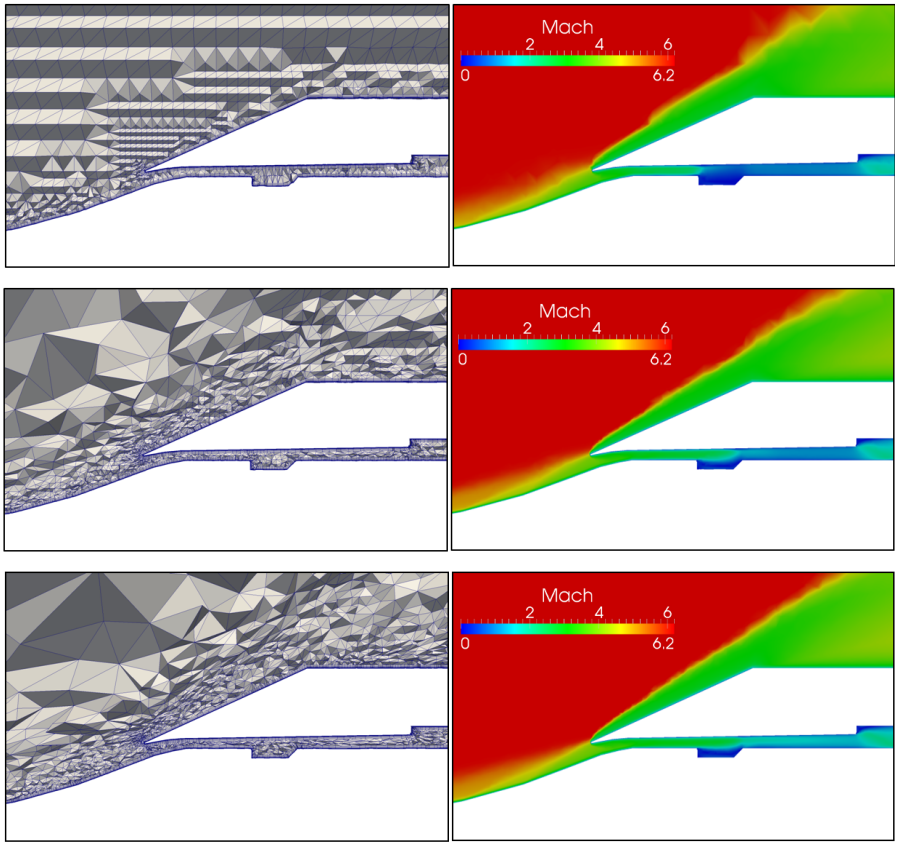

Fig. 10. Initial (top), first (middle) and second (bottom) adapted meshes and Mach number contour plots near the cowl lip and at the entry to the combustor region

5 Closing remarks

An adaptive parallel boundary layer meshing procedure was presented. The approach described works with the distributed meshes and effectively supports layered structure of the mesh. Mesh modification procedures were applied in parallel using the anisotropic size field provided by the application. The parallelization of the anisotropic mesh adaptation routines with boundary layers

based on local mesh modifications allows the procedure to be applied to large and complex 3D problem cases which are usually attacked with remeshing methods. At the same time, the approach provides the ability to perform a very inexpensive solution transfer process which also allows solution not to deteriorate over the mesh adaptation cycles.

It has been demonstrated that boundary layer mesh adaptation resulted in an increase of the accuracy of key quantities of interest and helped resolve critical areas of the flow to supply the analysis with better solution. During the adaptation process, the approach was able to preserve layered elements near the walls and effectively account for the mesh modification operations carried out in the distributed environment for both boundary layer and unstructured parts of the mesh, at the same time being capable of handling curved geometries of interest with the required mesh resolution and element quality control.

The method described scales while utilizing different mesh modification operation types and not only simple refinement. The parallel performance results carried out on problem domains indicate that mesh adaptation is capable of decreasing the simulation run times with higher number of cores.

6 Acknowledgements

This work was supported by the National Science Foundation under Grant No. 0749152, by the U.S. Department of Energy under DOE Grant No. DE-FC02-06ER25769, and by the NASA STTR Part II Grant No. BEE103/NNX11CC69C. Computing support is provided by National Energy Research Scientific Computing Center for granting access to the Cray XE6 supercomputers. The authors would like to acknowledge F. Nihan Cayan and Oren Breslouer for the efforts in helping with the scramjet test case.

References

1. X. Li, M.S. Shephard, and M.W. Beall. 3D anisotropic mesh adaptation by mesh modifications. *Comput. Method. Appl. M.*, 194(48-49):4915–4950, 2005.
2. G. Compere, J.-F. Remacle1, J. Jansson, and J. Hoffman. A mesh adaptation framework for dealing with large deforming meshes. *Int. J. Numer. Meth. Eng.*, 82(7):843–867, 2010.
3. K.-J. Bathe and H. Zhang. A mesh adaptivity procedure for CFD and fluid-structure interactions. *Computers and Structures*, 87(11-12):604–617, 2009.
4. M. D. Piggott, G. J. Gorman, C. C. Pain, P. A. Allison, A. S. Candy, B. T. Martin, and M. R. Wells. A new computational framework for multi-scale ocean modelling based on adapting unstructured meshes. *Int. J. Numer. Meth. Fl.*, 56(8):1003–1015, 2008.
5. P.J. Frey and F. Alauzet. Anisotropic mesh adaptation for CFD computations. *Comput. Method. Appl. M.*, 194(48-49):5068–5082, 2005.

6. C.L. Botasso. Anisotropic mesh adaption by metric-driven optimization. *Int. J. Numer. Meth. Eng.*, 60(3):597–639, 2004.
7. X. Li, M.S. Shephard, and M.W. Beall. Accounting for curved domains in mesh adaptation. *Int. J. Numer. Meth. Eng.*, 58(2):247–276, 2003.
8. C.C. Pain, A.P. Umpleby, C.R.E. de Oliveira, and A.J.H. Goddard. Tetrahedral mesh optimization and adaptivity for steady-state and transient finite element calculations. *Comput. Method. Appl. M.*, 190(29-30):3771–3796, 2001.
9. J.F. Remacle, X. Li, M.S. Shephard, and J.E. Flaherty. Anisotropic adaptive simulation of transient flows using discontinuous Galerkin methods. *Int. J. Numer. Meth. Eng.*, 62(7):899–923, 2005.
10. X. Li. *Mesh Modification Procedures for General 3D Non-Manifold Domains*. PhD thesis, Rensselaer Polytechnic Institute, 2003.
11. O. Sahni, J. Muller, K.E. Jansen, M.S. Shephard, and C.A. Taylor. Efficient anisotropic adaptive discretization of the cardiovascular system. *Comput. Method. Appl. M.*, 195(41-43):5634–5655, 2006.
12. O. Sahni, K.E. Jansen, M.S. Shephard, C.A. Taylor, and M.W. Beall. Adaptive boundary layer meshing for viscous flow simulations. *Eng. Comput.*, 24(3):267–285, 2008.
13. Y. Kallinderis and C. Kavouklis. A dynamic adaptation scheme for general 3-D hybrid meshes. *Comput. Method. Appl. M.*, 194(48-49):5019–5050, 2005.
14. A. Khawaja, T. Minyard, and Y. Kallinderis. Adaptive hybrid grid methods. *Comput. Method. Appl. M.*, 189(4):1231–1245, 2005.
15. C.L. Botasso and D. Detomi. A procedure for tetrahedral boundary layer mesh generation. *Eng. Comput.*, 18(1):66–79, 2002.
16. A. Loseille and R. Lohner. Boundary layer mesh generation and adaptivity, 2011. 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition.
17. R.V. Garimella and M.S. Shephard. Boundary layer mesh generation for viscous flow simulations. *Int. J. Numer. Meth. Eng.*, 49:193–218, 2000.
18. O. Sahni, K.E. Jansen, X.J. Luo, and M.S. Shephard. Curved boundary layer meshing for adaptive viscous flow simulations. *Finite Elem. Anal. Des.*, 46(1-2):132–139, 2010.
19. M.S. Shephard, J.E. Flaherty, C.L. Bottasso, H.L. de Cougny, C. Ozturan, and M.L. Simone. Parallel automated adaptive analysis. *Parallel Computing*, 23(9):1327–1347, 1997.
20. F. Alauzet, X. Li, E.S. Seol, and M.S. Shephard. Parallel anisotropic 3D mesh adaptation by mesh modification. *Eng. Comput.*, 21(3):247–258, 2006.
21. M. Zhou, O. Sahni, H.J. Kim, C.A. Figueroa, C.A. Taylor, M.S. Shephard, and K.E. Jansen. Cardiovascular flow simulation at extreme scale. *Computational Mechanics*, 46(1):71–82, 2010.
22. M.S. Shephard, C. Smith, and J.E. Kolb. Bringing hpc to engineering innovation. *Computers in Science and Engineering*, 2013. Accepted for publication. `http://www.scorec.rpi.edu/REPORTS/2012-1.pdf`.
23. S. Chandra, X. Li, T. Saif, and M. Parashar. Enabling scalable parallel implementations of structured adaptive mesh refinement applications. *The Journal of Supercomputing*, 39(2):177–203, 2007.
24. C. Kavouklis and Y. Kallinderis. Parallel adaptation of general three-dimensional hybrid meshes. *J. Comput. Phys.*, 229(9):3454–3473, 2010.

25. O. Hassan, K. Sorensen, K. Morgan, and N.P. Weatherill. A method for time accurate turbulent compressible fluid flow simulation with moving boundary components employing local remeshing. *Int. J. Numer. Meth. Fl.*, 53(8):1243–1266, 2007.
26. S. Tendulkar, M. Beall, M.S. Shephard, and K.E. Jansen. Parallel mesh generation and adaptation for CAD geometries. In *Proc. of the NAFEMS World Congress*, 2011. `https://www.scorec.rpi.edu/REPORTS/2011-2.pdf`.
27. C.C. Pain, A.P. Umpleby, C.R.E. de Oliveira, and A.J.H. Goddard. Tetrahedral mesh optimization and adaptivity for steady-state and transient finite element calculations. *Int. J. Numer. Meth. Eng.*, 190(29-30):3771–3796, 2001.
28. A. Liu and B. Joe. On the shape of tetrahedra from bisection. *Mathematics of Computation*, 63(207):141–154, 1994.
29. E.S. Seol and M.S. Shephard. Efficient distributed mesh data structure for parallel automated adaptive analysis. *Eng. Comput.*, 22(3-4):197–213, 2006.
30. C. Ollivier-Gooch, L.F. Diachin, M.S. Shephard, T. Tautges, J. Kraftcheck, V. Leung, X. Luo, and M. Miller. An interoperable, data-structure-neutral component for mesh query and manipulation. *ACM Transactions on Mathematical Software*, 37:1–28, 2010.
31. M. Zhou, O. Sahni, K.D. Devine, M.S. Shephard, and K.E. Jansen. Controlling unstructured mesh partitions for massively parallel simulations. *SIAM Journal on Scientific Computing*, 32(6):3201 – 3227, 2010.
32. K.K. Chand, L.F. Diachin, X. Li, C. Ollivier-Gooch, E.S. Seol, M.S. Shephard, T. Tautges, and H. Trease. Toward interoperable mesh, geometry and field components for pde simulation development. *Eng. Comput.*, 24(2):165–182, 2008.
33. T. Xie, S. Seol, and M.S. Shephard. Generic components for petascale automated adaptive simulations. *Eng. Comput.*, 2012. Accepted for publication. `http://www.scorec.rpi.edu/FMDB/doc/xie_seol_shephard_EngComp.pdf`.
34. E. Boman, K. Devine, L.A. Fisk, R. Heaphy, B. Hendrickson, V. Leung, C. Vaughan, U. Catalyurek, D. Bozdag, and W. Mitchell. Zoltan home page, 2012. `http://www.cs.sandia.gov/Zoltan`.
35. A. Ovcharenko, D. Ibanez, F. Delalondre, O. Sahni, K.E. Jansen, C.D. Carothers, and M.S. Shephard. Neighborhood communication paradigm to increase scalability in large-scale dynamic scientific applications. *Parallel Computing*, 38(3):140–156, 2012.
36. H.L. de Cougny and M.S. Shephard. Parallel refinement and coarsening of tetrahedral meshes. *Comput. Method. Appl. M.*, 46:1101–1125, 1999.
37. NASA. Ciam axisymmetric scramjet, 2012. `http://hapb-www.larc.nasa.gov/Public/Engines/Ciam/Ciam.html`.
38. C.H. Whiting and K.E. Jansen. A stabilized finite element method for the incompressible Navier–Stokes equations using a hierarchical basis. *Int. J. Numer. Meth. Fl.*, 35(1):93–116, 2001.
39. NASA. FUN3D online manual, 2012. `http://fun3d.larc.nasa.gov/`.
40. Cray XE6, 2012. `http://www.cray.com/Products/XE/CrayXE6System.aspx`.
41. O. Sahni, M. Zhou, M.S. Shephard, and K.E. Jansen. Scalable implicit finite element solver for massively parallel processing with demonstration to 160K cores. In *Proc. of the 2009 ACM/IEEE Conference on Supercomputing*, 2009.