Parallel Anisotropic Unstructured Grid Adaptation

Christos Tsolakis[1] and Nikos Chrisochoides[2]
Center for Real-Time Computing, Old Dominion University, Norfolk, VA 23529, USA

Michael A. Park[3]
NASA Langley Research Center, Hampton, VA 23681, USA

Adrien Loseille[4]
INRIA Paris-Saclay, Alan Turing Building, 91120 Palaiseau, France

Todd Michal[5]
The Boeing Company, St. Louis, MO, USA

Computational Fluid Dynamics (CFD) has become critical to the design and analysis of aerospace vehicles. Parallel grid adaptation that resolves multiple scales with anisotropy is identified as one of the challenges in the CFD Vision 2030 Study to increase the capacity and capability of CFD simulation. The Study also cautions that computer architectures are undergoing a radical change and dramatic increases in algorithm concurrency will be required to exploit full performance. This paper reviews four different methods to parallel anisotropic grid generation. They cover both ends of the spectrum: (i) using existing state-of-the-art software optimized for a single core and modifying it for parallel platforms and (ii) designing and implementing scalable software with incomplete, but rapidly maturing functionality. A brief overview for each grid adaptation system is presented in the context of a telescopic approach for multilevel concurrency. These methods employ different approaches to enable parallel execution, which provides a unique opportunity to illustrate the relative behavior of each approach. Qualitative and quantitative metric evaluations are used to draw lessons for future developments in this critical area for parallel CFD simulation.

I. Introduction

Parallel anisotropic grid generation and adaptation methods modify an existing mesh to conform to a specified anisotropic metric field. This metric field is constructed to specify a new grid that reduces errors estimated on the current grid and solution. Robust grid adaptation mechanics that produce and modify anisotropic elements with aspect ratios on the order of tens of thousands are required for high Reynolds number viscous flows. Grid adaptation methods have made dramatic improvements in the last decade. Alauzet and Loseille [1] showed the evolution of solution-adaptive methods that include anisotropy to resolve simulations with shocks and boundary layers. Park et al. [2] documented the current state of solution-based anisotropic grid adaptation and motivated further development for aerospace analysis and design in the broader context of the CFD Vision 2030 Study by Slotnick et al. [3]. The Vision Study provides a number of case studies to illustrate the current state of CFD capability and capacity and the potential impact of emerging High Performance Computing (HPC) environments forecasted to be available by the year 2030.

Parallel adaptive and anisotropic grid generation is at early stages of research and development compared to parallel isotropic mesh generation. In terms of concurrency, communication, and synchronization aspects, the codes for both types of grid generation share many common characteristics. Existing massively-parallel isotropic grid generation and adaptation procedures for current and emerging HPC platforms often (over-)decompose the original grid generation problem into n smaller subproblems, which are solved (i.e., meshed) concurrently using $n \gg p$ cores [4]. The subproblems can be formulated to be either tightly-coupled, partially-coupled, weakly-coupled, or decoupled. The coupling of the subproblems determines the intensity of the communication and the amount/type of synchronization

[1]Research Assistant, Center for Real-Time Computing, AIAA Member.
[2]Richard T. Cheng Chair Professor of Computer Science, AIAA Member.
[3]Research Scientist, Computational AeroSciences Branch, AIAA Associate Fellow.
[4]Researcher, GAMMA3 Team, AIAA Member.
[5]Technical Fellow, AIAA Senior Member.

required to maintain correctness and grid quality. For example, a tightly-coupled approach requires each subproblem to constantly maintain consistency with adjacent subproblems. A decoupled approach decomposes the grid generation task in a way that eliminates the need for synchronization.

Four different parallel anisotropic grid adaptation methods are presented with different communication and synchronization requirements. The methods are evaluated with a number of qualitative and quantitative criteria introduced by the Unstructured Grid Adaptation Working Group (UGAWG) in their first benchmark [5], which focused on evaluating adaptive grid mechanics for analytic metric fields on planar and simple curved domains. The UGAWG is an informal group that has been formed to mature unstructured grid adaptation technology. The first UGAWG benchmark article contains a list of future directions, among them parallel execution, the focus of this paper.

II. Parallel Strategies

The following parallel grid generation and adaptivity attributes are embodied to varying degrees by the software evaluated in this study. They range from attributes that are crucial to success in parallel execution to attributes that ensure longevity to enhance the adaptability of software for emerging computer concurrency architectures.

1) **Stability** is the requirement that the quality of the grid generated in parallel must be comparable to that of a grid generated sequentially. The quality is defined in terms of the density and shape of the elements evaluated in the metric field, and the number of the elements (fewer is better for the same level of metric conformity).
2) **Reproducibility** is separated into two forms by Chrisochoides et al. [6]. *Strong Reproducibility* requires that the grid generation code, when executed with the same input, produces identical results under the following modes of execution: (i) continuous without restarts, and (ii) with restarts and reconstructions of the internal data structures. *Weak Reproducibility* requires that the grid generation code, when executed with the same input, produces results of the same quality under the following modes of execution: (i) continuous without restarts, and (ii) with restarts and reconstructions of the internal data structures.
3) **Robustness** is the ability of the software to correctly and efficiently process any input data. Automation is critical for massively parallel computations, because operator intervention is impractical.
4) **Scalability** is the ratio of the time taken by the best sequential implementation to the time taken by the parallel implementation. Amdahl's law [7] suggests that the speedup is always limited by the inverse of the sequential fraction of the software. Therefore, all nontrivial stages of the computation must be parallelized to leverage the current and emerging architectures designed to deliver a million- to billion-way concurrency.
5) **Code Reuse** is a result of a modular design of the parallel software that builds upon previously designed sequential or parallel meshing code, such that it can be replaced and/or updated with a minimal effort. Code Reuse is feasible only if the code satisfies the Reproducibility criterion.

There are two common approaches for parallel grid generation and adaptation development, where these development approaches try to satisfy the above attributes. The first approach uses existing state-of-the-art serial software (i.e., fully functional) and modifies it for parallel execution, which will be referred as *functionality-first* approach. This paper briefly introduces and presents data from two such codes: *EPIC* and *Feflo.a*. The second approach designs and implements scalable software with an initially incomplete functionality and the intention of completing functionality as it is needed, which will be referred as *scalability-first* approach. This paper briefly introduces and presents data from two such codes: *refine* and *CDT3D*.

The grid adaptation tools used in this study leverage the parallelization methods of data decomposition, domain decomposition, or a combination. Chrisochoides [8] describes the Telescopic Approach, which applies a combination of decomposition techniques for current and emerging architectures with multiple memory/network hierarchies as shown in Fig. 1. The implementation of the Telescopic Approach is part of a long term goal for parallel grid generation and adaptation at the Center for Real-Time Computing (CRTC) to achieve and sustain a billion-way concurrency over the next 12 years. To achieve this goal, concurrency is exploited at different scales (levels) corresponding to the latency and the bandwidth of different network/memory hierarchies in order to orchestrate communication and synchronization as well as (in the future) power consumption.

The implementation of the Telescopic Approach relies on multiple abstractions used in the parallel grid generation community over the last 25 years [4]: element, cavity, data-region, and subdomain. These abstract data types vary in granularity and complexity (i.e., type and size of the data structures) and type/intensity of communication/synchronization required to implement their basic operations. The intensity/type of communication/synchronization determines their mapping to different layers of memory/network hierarchy. For example, concurrency at the element or cavity level using edge swapping is permitted only in the shared memory of the cores within a single-chip, bulk and locally synchronous

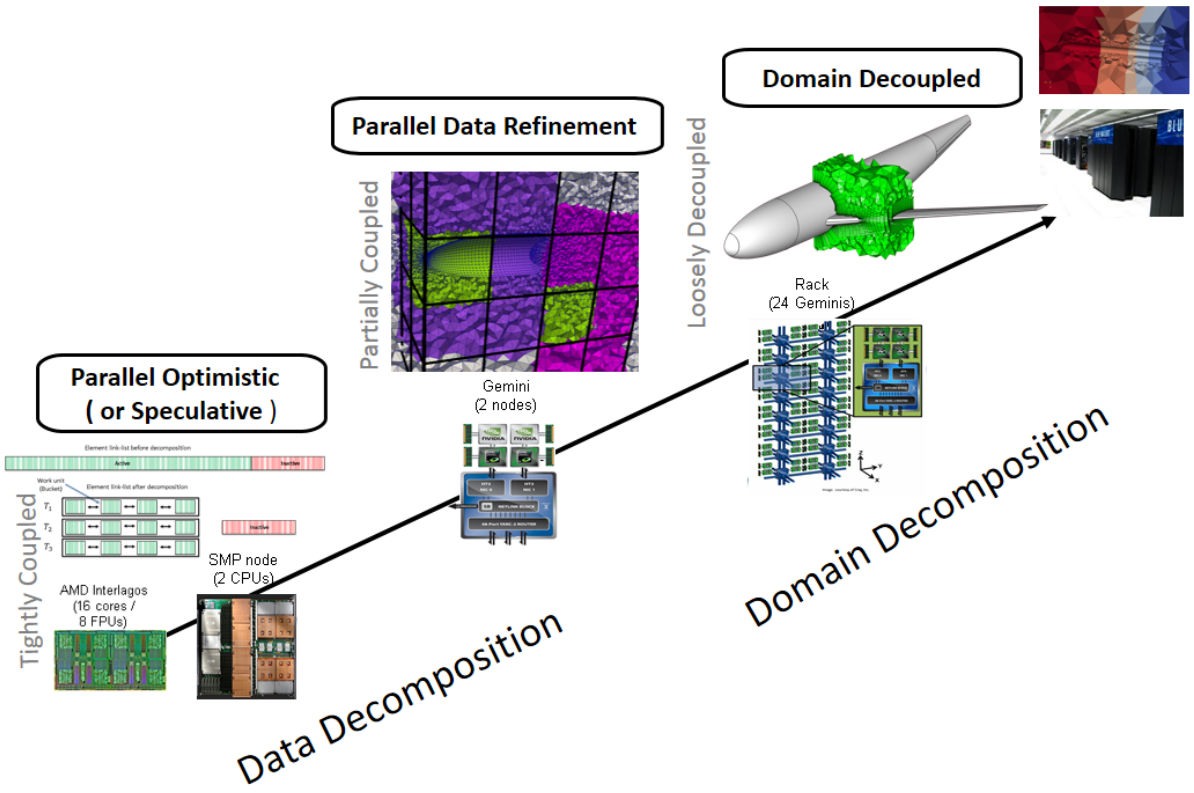


Fig. 1 The Telescopic Approach.

exchange of data among data-regions is permitted only within the distributed shared memory of a few nodes and asynchronous communication of data-buffers is permitted over distributed memory of several hundreds of nodes and/or tens of racks. Given these constraints, from the chip to the node levels, the Telescopic Approach deploys: (i) Parallel Optimistic (PO) methods similar to those presented in [9–11], Parallel Data Refinement (PDR) methods similar to those presented in [6, 12], while on supernodes and/or racks could utilize Parallel Constrained (PC) methods similar to those presented in [13], and/or loosely-coupled [14, 15] methods.

A survey of experience with isotropic mesh generation can be used to forecast the performance of future enhancements to the anisotropic algorithms. PO anisotropic grid generation codes like *CDT3D* on current and emerging Distributed Shared Memory (DSM) machines are expected to scale up to 150 to 200 cores, due to memory management issues similar to ones observed with Parallel Optimistic Delaunay Meshing (PODM) [10]. The use of sophisticated memory pools can help to sustain scaling, but do not significantly extend the practical concurrency. Locality-aware parallel implementations can help with better data affinity, but have limited impact due to the dynamic memory management aspects. For example, Locality Aware Parallel Delaunay (LAPD) [16] can improve performance for up to 200 to 250 cores, see Fig. 2 (right).

These two studies [10, 16] and the data in Fig. 2 (right) suggest that one has to explore nested parallelism at both fine- and coarse-grain levels in order to improve performance for up to 900 to 1000 cores. Namely, data from isotropic grid generation implementations indicate that the application of the PDR on PODM (PDR.PODM) on both DSM [17] and distributed memory machines using hybrid (MPI+threads) programming models [18, 19] can improve the overall performance.

However, Fig. 2 indicates that such improvements are expected to have limited impact (i.e., about 66% parallel efficiency) on higher than 3600 to 6000 cores due to local synchronization and volume of data migration. Parallel metric-based adaptive anisotropic codes expected to have the same behavior since they are very similar to isotropic ones

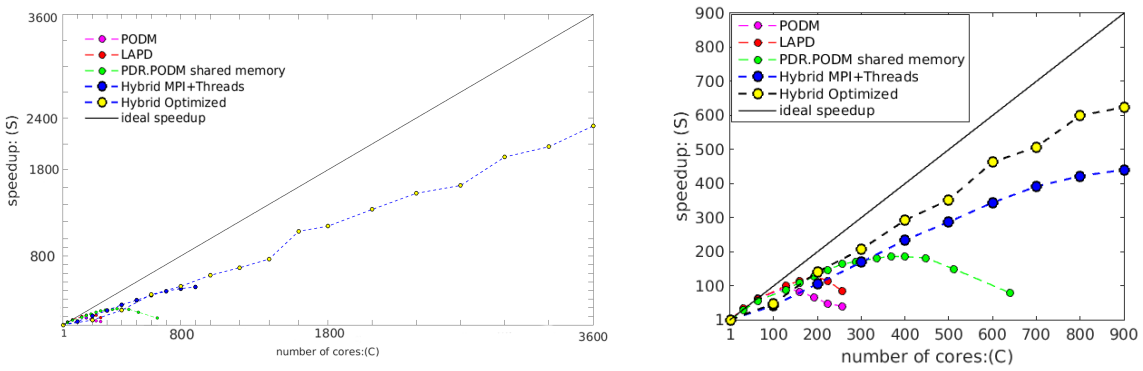


Fig. 2 Data on the first two layers of the Telescopic Approach applied on isotropic imaged-based grid generation.

when it comes to concurrency, communication and synchronization. Load balancing is a big factor in adaptive codes, but all data depicted in Fig. 2 are without load balancing.

The CRTC team plans to address the load balancing problem using a parallel runtime software system designed and implemented for load balancing [15]. However, even with load balancing for a large number of cores (i.e., $\gg$ 10,000), communication/synchronization overheads are addressed by utilizing remaining levels of Telescopic Approach (i.e., PC and loosely-coupled methods that rely on a lower volume of asynchronous communication). Anisotropic grid adaptation tools use one or more levels of this hierarchy as described in the following subsections, and the results section mirrors experience gained developing the Telescopic Approach for isotropic grid generation.

A. *CDT3D* from ODU

CDT3D implements a tightly-coupled approach and exploits fine-grain parallelism at the cavity level using data decomposition. Its current implementation targets shared memory multicore nodes using multithreaded execution at the chip level. In addition, *CDT3D* is designed for Code Reuse to simplify the implementation requirements for the communication and synchronization of both data and domain decomposition layers of the Telescopic Approach. It is designed to achieve high speed at the core level and tolerate costs due to gather/scatter operations in order to meet scalability requirements. The Stability, Robustness, and Reproducibility are constantly reinforced with the parallelization of any new operation included in *CDT3D*.

At the chip level, *CDT3D* performs concurrently multiple (but same) grid operations (e.g., edge/face swapping) on different data by using fast atomic lock instructions to guarantee correctness. The pipeline of *CDT3D* can be divided into three main steps: initial grid construction, grid refinement, and (optionally) grid quality optimization, see Fig. 3. In

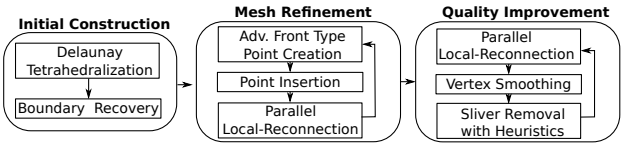


Fig. 3 The *CDT3D* grid generation pipeline.

the first stage, the input surface grid is recovered using methods based on Delaunay tetrahedralizations. Grid refinement introduces points iteratively into the grid using an advancing front point creation and direct insertion. After each point creation iteration, the grid is optimized in parallel using a fine-grained topological scheme for local reconnection [11], optimizing metric-based criteria. In the last stage, the grid quality is improved using a combination of grid smoothing, parallel local reconnection, and heuristics to target the improvement of low quality elements.

B. *EPIC* from Boeing

EPIC uses a partially-coupled approach that exploits coarse-grain parallelism at the subdomain level. Given the initial mesh, *EPIC* partitions the mesh into subdomains and performs a complete mesh operator pass consisting of refinement,

coarsening, element reconnection, and smoothing operations on the interior of each subdomain while temporarily freezing the mesh at partition boundaries. After each mesh operator pass, *EPIC* updates the decomposition by shifting elements between subdomains. Subdomain rebalancing uses an optimization technique that attempts to maintain an equal work-load balance between subdomains while ensuring that frozen mesh edges near partition boundaries are moved to the interior of a subdomain with each rebalancing step. Multithreading can be used to parallelize the mesh operators at the subdomain level, but has only been implemented for a subset of mesh operations. This incomplete multithreading implementation has seen limited use to date.

The *EPIC* anisotropic grid adaptation process [20] provides a modular framework for anisotropic unstructured grid adaptation that can be linked with external flow solvers. *EPIC* relies on repeated use of mesh operator passes to modify a grid such that element edge lengths match a given anisotropic metric tensor field. The metric field on the adapted grid is continuously interpolated from the initial metric field. Several methods are available to preprocess the metric to limit minimum and maximum local metric sizes, control metric stretching rates and/or anisotropy, and ensure smoothness of the resulting distribution. In addition, the metric distribution can be limited relative to the initial grid and/or to the local geometry surface curvature. The surface grid is maintained on an IGES geometry definition with geometric projections and a local regridding. *EPIC* is routinely used on production applications at the Boeing Company and has been applied on several workshop cases where the parallel implementation makes it practical for large scale problems [21, 22].

C. *refine* from NASA

refine relies on the implementation of a partially-coupled approach that exploits coarse-grain parallelism at the subdomain level using domain decomposition and a homogeneous programming model. The parallel execution strategy is described in Park and Darmofal [23]. The interior portion of each subdomain is modified in parallel while the border regions between subdomains are fixed. Elements that span boundaries and need to be modified to improve metric conformity are marked for future refinement. A combined load-balancing and migration is performed to equalize the number of nodes on each partition while penalizing elements marked for modification that span subdomains after migration. The repartitioning step provides edge weights to either ParMETIS [24] or Zoltan [25] graph-based partitioning libraries. The current load-balancing and migration approach has improved parallel scaling properties over the transcript approach described in [23].

refine is available at `https://github.com/NASA/refine` under the Apache 2.0 open source license. *refine* is designed to output a unit grid [26] for a given metric field. A combination of edge split and collapse operations proposed by Michal and Krakos [20] is used to modify long and short edges toward unity length in the metric. Node relocation is performed to improve adjacent element shape. A new ideal node location of the node is created for each adjacent element. A convex combination of these ideal node locations is chosen to yield a new node location update that improves the element shape measure in the anisotropic metric [27]. Geometry is accessed through the EGADS [28] and EGADSlite [29] application program interface.

D. *Feflo.a* from INRIA

Feflo.a employs a partially coupled, coarse-grained approach that exploits parallelism at the subdomain level. The initial grid is decomposed in multiple levels (i.e., domain decomposition). The initial volume is split and adapted in parallel while treating the interface between subdomains as a constrained surface. Once the initial subdomains are complete, a new set of subdomains are constructed entirely of the constrained interface elements of the previous subdomains. This process recurses until all the constrained elements are adapted [30].

Feflo.a is an adaptation code developed at INRIA that can process manifold or nonmanifold surface and/or volume grids composed of simplicial elements. It creates a unit grid [31, 32] in two steps. The first step improves the edge length distribution with respect to the input metric field. Only classical edge-based operators (insertion and collapse) are used during this step. The second step is the optimization of grid element shape measures with node smoothing and tetrahedra edge and face swaps. For the surface grid adaptation, a dedicated surface metric is used to control the deviation of the metric and surface curvature. This surface metric is then combined with the input metric. New points created on the surface are evaluated on a (fine) background surface grid and optionally on a geometry model via the EGADS API.

The classical edge-based operators are implemented by a unique cavity-based operator [30, 33]. This cavity-based operator simplifies code maintenance, increases the success rate of grid modifications, has a constant execution time for many different local operations, and robustly inserts boundary layer grids [34]. When the cavity operator is combined with advancing-point techniques, it outputs metric-aligned and metric-orthogonal grids [35].

III. Experimental Evaluation

A series of experiments are performed to evaluate the parallel strategies defined in the previous section. In each experiment, a given mesh is adapted to conform to an anisotropic metric field M. Loseille and Alauzet [26] provide a thorough introduction of the definition and properties of the metric tensor field. The complexity C of a continuous metric field $\mathcal{M}$ is defined as the integral,

$$C(\mathcal{M}) = \int_{\Omega} \sqrt{\det(\mathcal{M}(x))}\, dx. \tag{1}$$

Complexity is computed on the discrete grid by sampling $\mathcal{M}$ at each vertex i as the discrete metric field M,

$$C(M) \equiv \sum_{i=1}^{N} \sqrt{\det(M_i)} V_i, \tag{2}$$

where V_i is the volume of the Voronoi dual surrounding each node. The relationship between C and the number of vertices and elements in the adapted grid is shown theoretically by [26] and experimentally by [36, 37]. The complexity has a linear dependency with respect to the number of vertices and tetrahedra, where the vertices are approximately $2C$ and tetrahedra are approximately $12C$.

The complexity of a metric can be scaled to create a uniformly refined (or coarsened) mesh with the same relative distribution of element density and shape. The metric tensor M_{C_T} that corresponds to the target complexity C_T is evaluated by [26]:

$$M_{C_T} = \left(\frac{C_T}{C(M)}\right)^{\frac{2}{3}} M, \tag{3}$$

where M is the metric tensor before the scaling and $C(M)$ is the complexity of the discrete metric before scaling.

The objective of the evaluation is to support parallel anisotropic grid adaptation method development. Two evaluation methods are used: (i) quantitative with respect to parallel performance of the codes and (ii) qualitative with respect to metric conformity of the adapted mesh. The goal of metric conformity is to create a unit grid [26], where the edges are unit-length and the elements are unit-volume with respect to the given metric. Computing an edge length in a continuous metric field requires an integral. If assumptions are made about the interpolation of the metric between vertex a and vertex b of the mesh, an analytical expression for the edge length in the metric L_e is available as [38],

$$\begin{aligned} L_e &= \begin{cases} \frac{L_a - L_b}{\log(L_a/L_b)} & |L_a - L_b| > 0.001 \\ \frac{L_a + L_b}{2} & otherwise \end{cases} . \\ L_a &= (v_e^T M_a v_e)^{\frac{1}{2}},\ L_b = (v_e^T M_b v_e)^{\frac{1}{2}} \end{aligned} \tag{4}$$

The Mean Ratio shape measure is also approximated in the discrete metric,

$$Q_k = \frac{36}{3^{1/3}} \frac{\left(|k| \sqrt{\det(M_{\text{mean}})}\right)^{\frac{2}{3}}}{\sum_{e \in L} v_e^T M_{\text{mean}} v_e}, \tag{5}$$

where v is a vertex of element k and M_{mean} the interpolated metric tensor evaluated at the centroid of element k. The parallel performance is evaluated in terms of traditional metrics like strong speedup and speed for generating elements (S_e) and grid points (S_p) defined as follows:

$$S_e = \frac{N_e}{T_{Prep} + T_{E2E}}$$
$$S_p = \frac{N_p}{T_{Prep} + T_{E2E}}$$

where

T_{Prep} file access time, boundary recovery time, partitioning time, metric interpolation time, etc.

T_{E2E} time spent on grid adaptation

N_e number of elements in the final grid minus elements in the initial grid

N_p number of vertices in the final grid minus vertices in the initial grid

A cube with an analytically-defined metric field and a delta wing with a solution-based metric field in laminar flow are examined. These two simple geometries focus on the details of parallel execution without the difficulty of evaluating curved geometries. Materials for these two cases are available at `https://github.com/UGAWG`. The rest of this section will be organized with respect to those two geometries.

The execution times and hardware specifications are omitted for the *EPIC* results to protect proprietary data. This might seem contrary to open discussions in forums like this, but the authors feel that this real-life constraint does not affect the lessons learned. In fact, the value of including the normalized scaling and metric conformity evaluations of an industrial code exceeds the minor limitation of an incomplete comparison.

A. Cube

The first geometry is a cube with an analytically defined metric field M referred to as polar-2 in the first UGAWG benchmark [5], where a cube-cylinder geometry was specified. Here a unit cube is used, see Fig. 4. The metric is defined as,

$$M = \begin{bmatrix} \cos(t) & -\sin(t) & 0 \\ \sin(t) & \cos(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} h_r^{-2} & 0 & 0 \\ 0 & h_t^{-2} & 0 \\ 0 & 0 & h_z^{-2} \end{bmatrix} \begin{bmatrix} \cos(t) & \sin(t) & 0 \\ -\sin(t) & \cos(t) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{6}$$

where $r = \sqrt{x^2 + y^2}$, $t = \text{atan2}(y, x)$, $h_z = 0.1$, $h_0 = 0.001$ and $h_r = h_0 + 2(0.1 - h_0)\,|r - 0.5|$. The subscript t is in the θ direction and subscript r is the radial direction. The spacing in the tangential direction is defined by

$$d = 10\,(0.6 - r) \quad \text{and} \quad h_t = \begin{cases} 0.1 \;\text{ if }\; d < 0 \\ d/40 + 0.1(1 - d) \;\text{ if }\; d \le 0 \end{cases} \tag{7}$$

This metric field represents a curved shear layer. This polar distribution has low gradation and is possible to satisfy with high-quality elements by resolving curvature in the tangential direction near the layer.

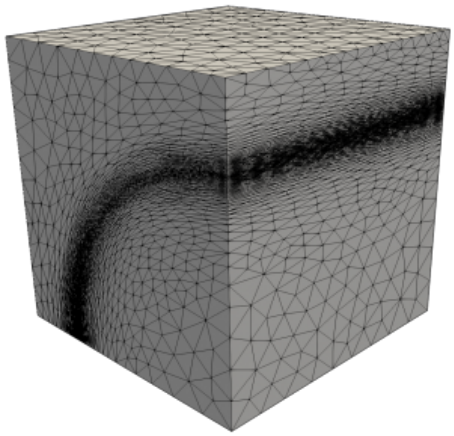


Fig. 4 Cube with polar-2 analytic metric, complexity of 8,000.

The initial grid conforms to the polar-2 metric with a complexity of 8,000. The polar-2 metric field is scaled to 500,000 complexity for this test. Adapted grids with approximately 1,000,000 vertices and 6,000,000 tetrahedra are expected, which is a relatively small example based on the size of a typical fluid simulation. This small size makes the strong scaling tests a challenge for a large number of cores. The scaling results obtained using *refine*, *CDT3D* and *EPIC* to adapt the initial 8,000 complexity grid to conform to the 500,000 complexity as a function of number of cores is shown in Fig. 5. All three methods exhibit linear scaling in a low number of cores. At higher core numbers, the speedup becomes constant for both distributed memory methods. The main reason for this is the small size of the mesh, which does not offer enough concurrency. In other words, the computation time per core becomes very small and the communication overhead dominates the running time.

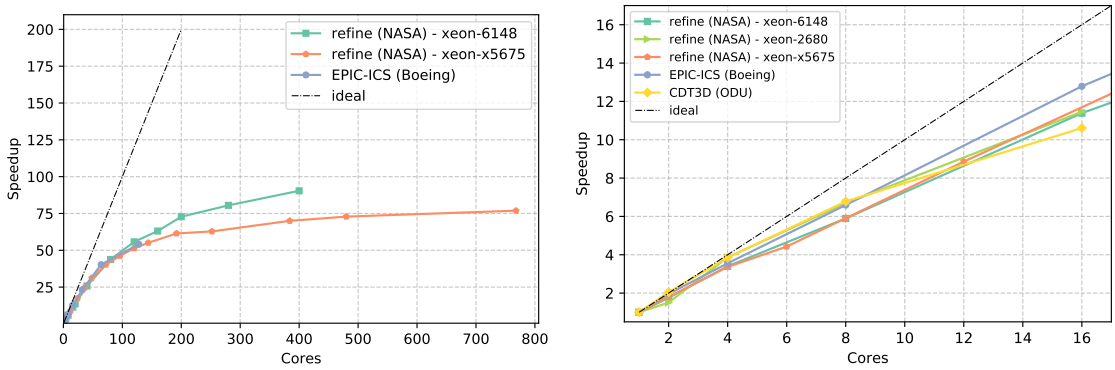


Fig. 5 Left: Speedup for the cube case adapted from 8,000 complexity to 500,000 complexity. Right: Zoom-in view of the data for up to 16 cores.

Metric conformity, characterized by element shape measure and edge length histograms of the generated grids, is shown for *refine*, *EPIC*, *CDT3D* and *Feflo.a* in Fig. 6 and 7. The mean ratio is bounded between one and zero, where a mean ratio near one indicates better metric conformity than a mean ratio near zero. In linear scale, all methods appear to exhibit good overall quality. The log scale makes the presence of low quality elements in the grid generated with *CDT3D* more visible. One of the reasons for these elements is that currently *CDT3D* does not adapt on the boundary of the grid, resulting thus in low quality elements near and on the boundary. *refine* produces elements with the highest minimum mean ratio of 0.1, while the lowest mean ratio is around 0.01 for *EPIC* and *Feflo.a*. The ideal edge length distribution is clustered tightly around unity. Figure 7 (left) reveals that *refine* and *EPIC* generated edges with less variance, while *Feflo.a* and *CDT3D* produce both the shortest edges and the largest edges.

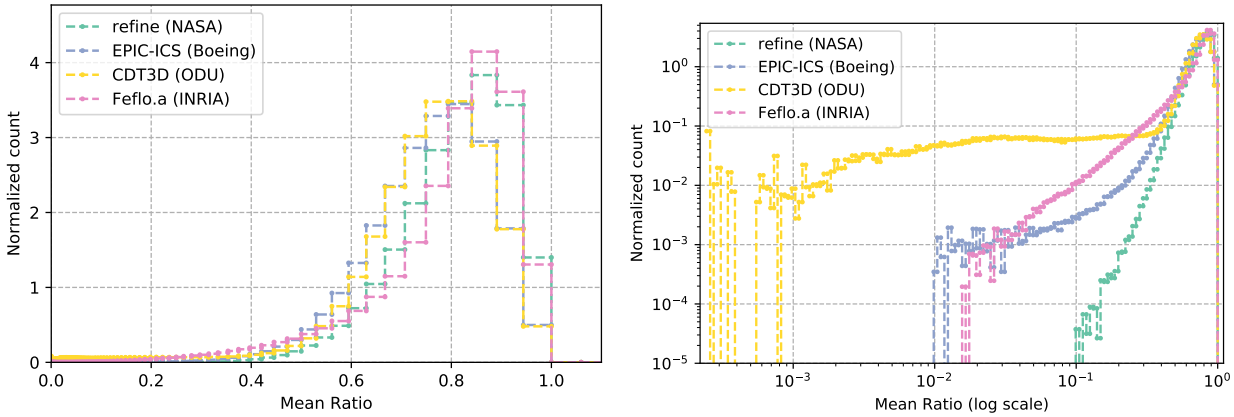


Fig. 6 Comparison of the mean ratio of the generated meshes for the Cube 500k case with *refine*, *EPIC* and *CDT3D* using 16 cores, in linear and logarithmic scales.

B. Delta Wing

The second geometry, Fig. 8(a), is a delta wing constructed of planar facets. A multiscale metric [39] is constructed based on the Mach field of this subsonic laminar flow. The initial grid is adapted to a specified complexity of 50,000 and details of the verification of the delta wing grid adaptation process is provided by Park et al. [40]. The multiscale metric is scaled to have a complexity of 500,000 for the input to the adaptation evaluation. Adapted grids with approximately 1,000,000 vertices and 6,000,000 tetrahedra are expected, which is a relatively small example based on the maximum number of 23M vertices in [40].

The initial grid conforms to the metric with a complexity of 50,000. The execution time required by *refine*, *CDT3D*, and *EPIC* to adapt the initial 50,000 complexity grid to conform to a 500,000 complexity metric field as a function of

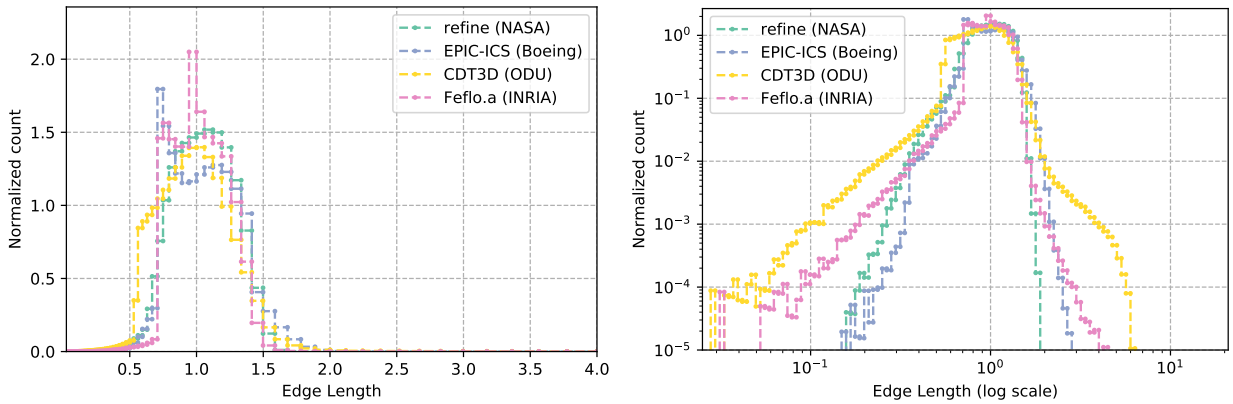
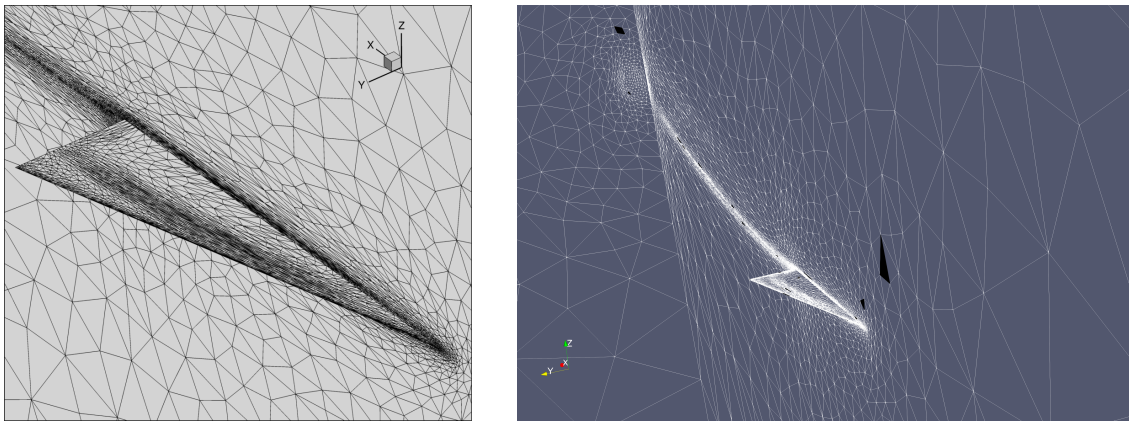


Fig. 7 Comparison of the edge lengths of the generated meshes for the Cube 500,000 case with *refine*, *EPIC* and *CDT3D* using 16 cores, in linear and logarithmic scales.



(a) Delta wing with multiscale metric in laminar flow, 50,000 complexity.

(b) *CDT3D* adapting the delta wing: the current missing functionality of boundary adaptation creates low quality elements (black tetrahedra)

Fig. 8 Delta wing geometry.

the number of cores is shown in Fig. 9. At high core numbers, both distributed memory codes exhibit improved scaling over the performance of the cube case due to the larger size of the initial grid for the delta wing. The speedup becomes almost constant above 400 cores for *refine*. For *EPIC*, the maximum speedup is about 75 on 256 cores. When the complexity of the target mesh is linearly scaled to 10,00,000, *EPIC* offers more than 100 times the speedup on 500 cores, as shown in Fig. 10. At lower core counts, *EPIC* exhibits the best scaling while *CDT3D* falls between *EPIC* and *refine*.

Returning to the 500,000 complexity target metric, metric conformity (characterized by element shape measure and edge length histograms of the generated grids) is shown in Figs. 11 and 12, respectively. On a linear scale, all methods appear to exhibit good overall quality. The log scale makes the differences more prevalent. *refine*'s grid quality exhibits the best lower bound among these metric conformity measures. *CDT3D* exhibits a slightly inferior quality in the mean ratio because of the current lack of boundary adaptation. Figure 8(b) depicts the locations of all elements with mean ratio quality less that 0.1 near the wing, it is easily seen that they are attached on the boundary. These low quality features are expected to be eliminated once boundary adaptation is implemented. The edge-length distribution is similar for all methods with *CDT3D* producing the longest edge and *Feflo.a* the shortest.

The concepts of Stability and Reproducibility were introduced in Section II. Adherence to these attributes is measured by evaluating the metric conformity of the same case with different numbers of cores. Histograms of edge length in the metric are evaluated for three codes for execution with 1, 8, and 16 cores in Fig. 13. *refine*, *CDT3D*, and *EPIC* show an almost perfect overlap of the histograms, but they do not produce the same mesh (i.e., they offer a weak form of the Reproducibility attribute). Producing metric conformity that is independent of the number of cores

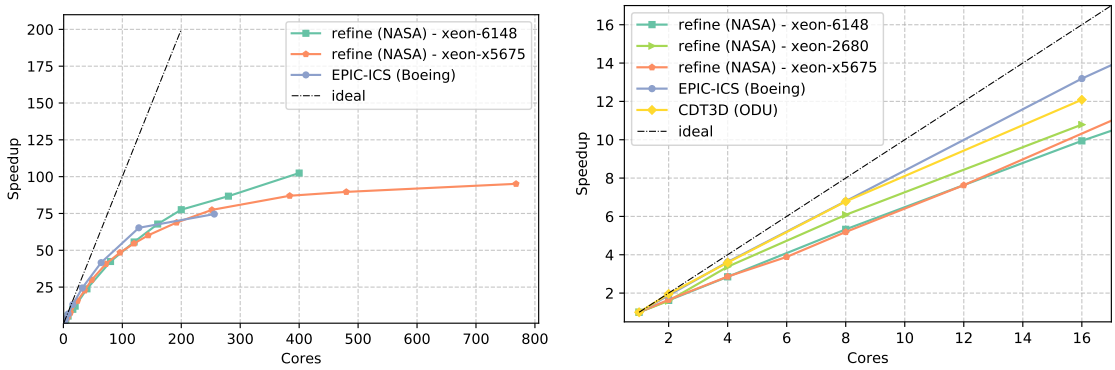


Fig. 9 Left: Speedup data for the delta wing adapted from 50,000 complexity to 500,000 complexity. Right: Zoom-in view of the data for up to 16 cores.

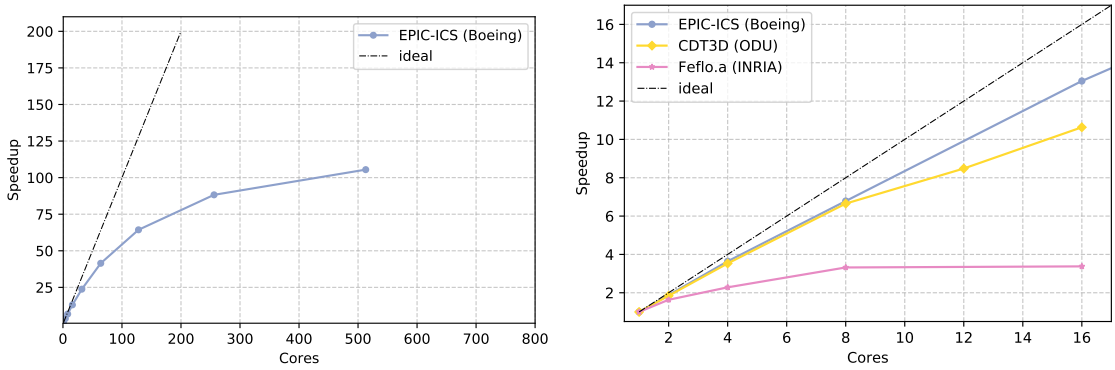


Fig. 10 Left: Speedup data for the delta wing adapted from 50,000 complexity to 10,000,000 complexity. Right: Zoom-in view of the data for up to 16 cores.

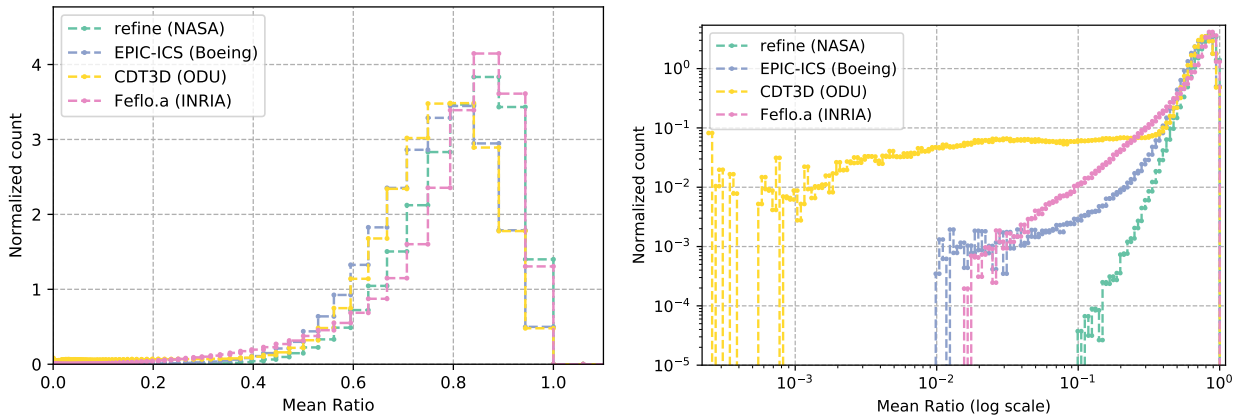


Fig. 11 Comparison of the mean ratio of the generated grids for the delta wing 500,000 complexity case with *refine*, *EPIC*, *CDT3D* using 16 cores and *Feflo.a* using 1 core, in linear and logarithmic scales.

satisfies the requirement of Stability. The mean ratio histograms result in the same conclusion that metric conformity is independent of the number of cores for these tools and the mean ratio plot is omitted for brevity.

The speeds S_e and S_p, depicted in Table 1 for a variety of hardware types, provide an insight on the behavior of parallel anisotropic adaptive grid methods for HPC CFD capability needs. This table helps to anchor the relative strong scaling

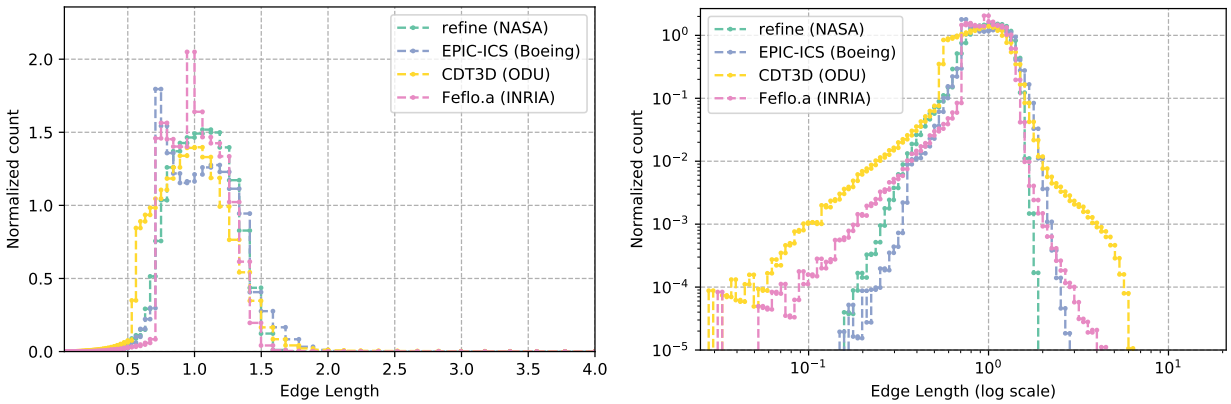


Fig. 12 Comparison of the edge lengths of the generated grids for the delta wing 500,000 complexity case with *refine*, *EPIC*, *CDT3D* using 16 cores and *Feflo.a* using 1 core, in linear and logarithmic scales.

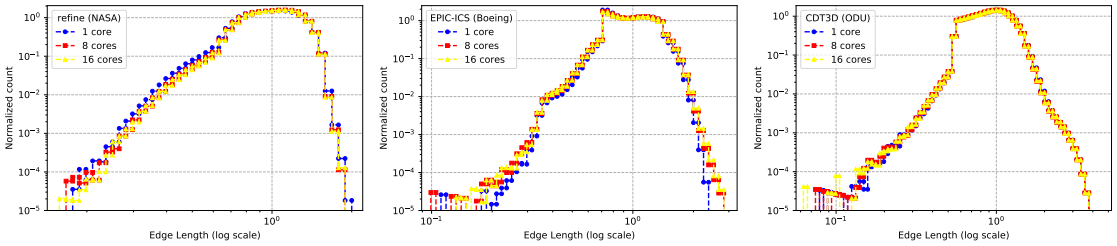


Fig. 13 Stability data for the delta wing 50,000 to 500,000 complexity case using *EPIC*, *refine* and *CDT3D*.

performance of the tools as previously shown in Fig. 9. Table 2 indicates the versions of the Intel®Xeon®processors, which have launch dates ranging from 2011 to 2017. This wide range of processor capabilities introduced in this time period makes direct comparison of the speeds between different codes problematic. Even for the same code, the speed may be as much as 3 times slower (see *CDT3D* in Table 1) depending on the hardware used for the evaluation.

The current timing information provides limited insight on the potential behavior of the parallel methods for extreme-scale current and emerging architectures. For example, both scalability-first *CDT3D* and functionality-first *EPIC* include parallel and sequential parts. Amdahl's law predicts that the serial fraction of the code reduces the potential for parallel speedup as the number of cores grows.

Table 1 Performance results for adapting the delta wing grid from an initial complexity of 50,000 to a 500,000 complexity using *CDT3D*, *refine* and *Feflo.a* on different hardware.

Software	Cores	T_{Prep}	T_{E2E}	N_e	N_p	S_e (elms/sec)	S_p (points/sec)	Machine
Feflo.a	1	1.9	52.0	5,305,631	835,632	87,651.86	13,568.22	xeon2697
CDT3D	24	2.43	40.88	5,558,489	933,876	114,922.49	19,154.26	crtc.lab
CDT3D	24	2.62	50.24	5,563,424	934,712	94,253.27	15,709.55	turing.odu
CDT3D	24	2.07	118.84	5,566,176	935,010	41,228.85	6,870.44	bridges.psc
refine	16	1.18	1084.76	5,273,053	929,649	4,861.03	857.01	xeon2680
refine	24	1.45	866.34	4,413,879	772,160	5,094.90	891.29	xeonX5675
refine	16	1.39	902.84	4,434,490	776,647	4,911.70	860.23	xeon6148

Weak scaling of *CDT3D* (Table 3) and *Feflo.a* (Table 4) is shown to complement the strong scaling studies. The initial grid is the 50,000 complexity delta wing. The output mesh from an adaptation to a target complexity is used

Table 2 Hardware specifications of the different machine names in Table 1.

Machine	Memory (GB)	L3 cache (MB)	Model Name	Turbo	Sockets	Cores per Socket	Total Cores
crtc.lab	757	30	Intel®Xeon®E5-2697 v2 @ 2.70GHz	3.50GHz	2	12	24
turing.odu	757	16	Intel®Xeon®E5-4610 v2 @ 2.30GHz	2.70GHz	4	8	32
bridges.psc	3000	40	Intel®Xeon®E7-8860 v3 @ 2.20GHz	3.20GHz	4	16	64
xeon2680	264	35	Intel®Xeon®E5-2680 v4 @ 2.40GHz	3.30GHz	2	14	28
xeonX5675	24	24	Intel®Xeon®X5675 @ 3.06GHz	3.46GHz	2	6	12
xeon6148	96	55	Intel®Xeon®Gold 6148 @ 2.40GHz	3.70GHz	2	20	40
xeon2697	-	30	Intel®Xeon®E5-2697 v2 @ 2.70GHz	3.50GHz	-	12	12

as the input to the next adaptation step after scaling the metric to the new target complexity. The target complexity is scaled linearly with the number of cores at a ratio of 500,000 complexity per core. This assumes that the work required to adapt the grid scales linearly with the complexity. Methods with sublinear or superlinear work scaling should be evaluated with a problem size that accounts for the work scaling. Element creation speed is evaluated by $(elms_{final} - elms_{initial})/(t_{prep} + t_{e2e})$ and similarly for the points. The *CDT3D* grid is kept in memory throughout the simulation and is passed from each case to the next as is expected in the context of a parallel CFD solver. The *CDT3D* T_{Prep} is a small fraction of T_{E2E}, independent of the number of cores. There is a slight increase in *CDT3D* T_{E2E} at 24 cores, which may be due to memory contention. The *Feflo.a* T_{Prep} is a significant and growing fraction of T_{E2E}. If T_{Prep} is removed from T_{E2E}, good weak scaling is observed, which indicates that T_{Prep} is a good target of optimization or reformulation. Even with a growing T_{Prep}, *Feflo.a* shows an increasing S_e and S_p for increasing cores.

Table 3 Weak scaling performance of *CDT3D* on crtc.lab.

cores	complexity	# tetrahedra	# vertices	t_{prep}	t_{e2e}	elms/sec	pts/sec
1	50k → 500k	5,516,567	926,766	2.26	719.59	6,858.59	1,146.56
3	500k → 1.5m	15,908,630	2,669,073	0.18	666.35	15,595.50	2,615.41
6	1.5m → 3m	34,169,056	5,712,669	0.48	586.14	31,153.69	5,196.87
12	3m → 6m	69,272,450	11,571,320	0.51	648.17	54,157.70	9,045.87
24	6m → 12m	138,892,161	23,186,740	0.97	859.06	81,041.73	13,536.37

Table 4 Weak scaling performance of *Feflo.a*.

cores	complexity	# tetrahedra	# vertices	t_{prep}	t_{e2e}	elms/sec	pts/sec
1	50k → 500k	5,305,631	835,632	1.9	52.0	87,651.86	13,568.22
3	500k → 1.5m	17,084,296	2,653,433	6.2	82.5	132,777.20	20,491.50
6	1.5m → 3m	35,701,908	5,533,547	23.3	111.2	138,420.91	21,413.49
12	3m → 6m	71,634,403	11,072,009	68.3	155.0	160,915.79	24,802.79
24	6m → 12m	144,592,969	22,332,731	175.0	223.4	183,128.93	28,264.86

IV. Conclusions and Future Work

This paper presents four parallel anisotropic grid generation and adaptation methods from both ends of the spectrum for parallel mesh generation: functionality-first (i.e., *EPIC* and *Feflo.a*) and scalability-first (i.e., *refine* and *CDT3D*). In a follow-up study, we expect to increase the pool of the methods in this study by including: *Pragmatic* (`https://meshadaptation.github.io`), a 2D and 3D anisotropic adaptation code that targets distributed memory

machines developed at Imperial College London and *Omega_h* (`https://github.com/ibaned/omega_h`) an open-source grid adaptation library that exploits coarse-grain concurrency on subdomains with both CPU and GPU processors. *Omega_h* is unique in its support for Strong Reproducibility [41, 42].

In the rest of this section, we summarize the lessons learned with respect to five parallel mesh generation criteria defined in Section II. The experimental data from *EPIC*, *refine*, *Feflo.a* and *CDT3D* suggest:

- **Stability** For the target geometries, all four codes exhibit stability as depicted in Fig. 13. These codes are tested in a large set of geometries independently and experience the same behavior in terms of their stability.
- **Reproducibility** There is high cost for delivering strong reproducibility, but weak reproducibility can be attained at a lower cost. Weak reproducibility is sufficient for most flow solvers and adaptive mesh processes.
- **Robustness** No special effort is made to test robustness. However, independent of this study, there is evidence [21, 22, 43] that these codes are robust, which is not a trivial task especially for the methods that rely on discrete domain decomposition. Unexpected artifacts on the surfaces of discrete domain decomposition can disrupt boundary recovery.
- **Scalability** The scalability results on shared-memory nodes with a lower number of cores are encouraging. Strong speedup data from *EPIC* and *refine* suggest high-end user-productivity. Weak scaling speedup data in Fig. 2 from two layers (Parallel Optimistic and Parallel Data refinement [17]) of the Telescopic Approach applied on isotropic grid generation suggest similar end-user productivity and promising scalability (i.e., 100 billion elements on 16K cores in about 1000 seconds [19]). However, the data from tables 1, 3, and 4 suggest that for scalability-first methods like *CDT3D*, there are opportunities for runtime reduction. Namely, tables 3 and 4 suggest that, it takes 24 cores for *CDT3D* to deliver the same performance as a single-core, functionality-first, and highly-optimized code like *Feflo.a*. Functionality-first parallel anisotropic methods like *Feflo.a* can pay a high price for preprocessing subdomain data (see Table 4) that impact the scalability of the method (see Fig. 10). On the other hand, the scalability-first methods tend to have lower subdomain preprocessing time, see *refine* in Table 1 and *CDT3D* in Table 3. As it is evident from *refine* in Fig. 9, lower preprocessing cost helps with the scalability of the method. Scalability is impacted by the speed per single core and preprocessing cost. This paper presents two different viable approaches to achieve the same objective, where the difference is in the implementation priorities.
- **Code Reuse** By design, all four codes leverage code reuse at different levels. For example, *EPIC* and *Feflo.a* rely on existing sequential fine-tuned highly-optimized fully-functional code. The current version of *refine* is structured to reuse low-level data structures based on experience and code from an earlier version with lower scalability potential. *CDT3D* is designed from the ground-up to meet all the requirements for each of the layers of the Telescopic Approach and is expected to accomplish this with more than 95% code reuse, which is a lower bound from CRTC's experience with TetGen [6] and PODM [17].

Designing and implementing scalable software from ground-up leads to short-term incomplete, but rapidly maturing functionality. Evidence from this group's experience suggest that scalability-first methods like *CDT3D* with proper design decisions can accelerate efforts to extend functionality [44] and improve element quality. Work remains for both approaches, but sharing experiences from very targeted efforts like this paper will aid all parties. For example, scalability-first methods like *CDT3D* can improve conformity of the metric by targeting and prioritizing areas of interest suggested by functionality-first software like *EPIC*, which has been optimized to meet industrial needs. Functionality-first methods like *EPIC* could benefit by using a tightly-coupled and Telescopic Approach adopted by *CDT3D* to improve scalability on current and emerging hardware.

The pluralism in the different methods and their implementation (even when they belong to the classification presented in [4]) is a mutual beneficial to this community. The main contribution of the lessons learned in this paper is to identify very specific improvements for both functionality-first and scalability-first methods in a labor-efficient way. Given that mesh generation and specifically parallel mesh generation is a labor intensive task, our hope is that this study (and future studies) will provide insight to meet the challenges stated in the CFD Vision 2030 Study.

The effort started by the UGAWG has already returned value to the participants and wider grid adaptation community. The general consensus of the UGAWG is that parallel anisotropic grid adaptation codes could improve their scalability by exploring concurrency at several nested levels of abstractions like the Telescopic Approach depicted in Fig. 1 for isotropic methods.

Acknowledgments

Cameron Druyor and Kyle Anderson provided feedback that improved this manuscript. This research was sponsored in part by the NASA Transformational Tools and Technologies Project (NNX15AU39A) of the Transformative

Aeronautics Concepts Program under the Aeronautics Research Mission Directorate, NSF grant no. CCF-1439079, the Richard T. Cheng Endowment, and the Modeling and Simulation fellowship of Old Dominion University. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Government.

References

[1] Alauzet, F., and Loseille, A., "A Decade of Progress on Anisotropic Mesh Adaptation for Computational Fluid Dynamics," *Computer-Aided Design*, Vol. 72, 2016, pp. 13–39. doi:10.1016/j.cad.2015.09.005, 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation.

[2] Park, M. A., Krakos, J. A., Michal, T., Loseille, A., and Alonso, J. J., "Unstructured Grid Adaptation: Status, Potential Impacts, and Recommended Investments Toward CFD Vision 2030," AIAA Paper 2016–3323, 2016.

[3] Slotnick, J., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E., and Mavriplis, D., "CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences," NASA CR-2014-218178, Langley Research Center, Mar. 2014. doi:2060/20140003093.

[4] Chrisochoides, N., *Numerical Solution of Partial Differential Equations on Parallel Computers*, Springer-Verlag, 2006, Lecture Notes in Computational Science and Engineering, Vol. 51, Chap. Parallel Mesh Generation, pp. 237–264. doi:10.1007/3-540-31619-1_7.

[5] Ibanez, D., Barral, N., Krakos, J., Loseille, A., Michal, T., and Park, M., "First Benchmark of the Unstructured Grid Adaptation Working Group," *Procedia Engineering*, Vol. 203, 2017, pp. 154–166. doi:10.1016/j.proeng.2017.09.800, 26th International Meshing Roundtable, IMR26, 18-21 Sept. 2017, Barcelona, Spain.

[6] Chrisochoides, N., Chernikov, A., Kennedy, T., Tsolakis, C., and Garner, K., "Parallel Data Refinement Layer of a Telescopic Approach for Extreme-Scale Parallel Mesh Generation for CFD Applications," AIAA Paper 2018–2887, 2018.

[7] Amdahl, G. M., "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities," *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, ACM, New York, NY, USA, 1967, pp. 483–485. doi:10.1145/1465482.1465560.

[8] Chrisochoides, N., "Telescopic Approach for Extreme-Scale Parallel Mesh Generation for CFD Applications," AIAA Paper 2016–3181, 2016.

[9] Chrisochoides, N., and Nave, D., "Parallel Delaunay Mesh Generation Kernel," *International Journal for Numerical Methods in Engineering*, Vol. 58, No. 2, 2003, pp. 161–176. doi:10.1002/nme.765.

[10] Foteinos, P., and Chrisochoides, N., "High Quality Real-Time Image-to-Mesh Conversion for Finite Element Simulations," *Journal on Parallel and Distributed Computing*, Vol. 74, No. 2, 2014, pp. 2123–2140. doi:10.1016/j.jpdc.2013.11.002.

[11] Drakopoulos, F., Tsolakis, C., and Chrisochoides, N., "Fine-Grained Speculative Topological Transformation Scheme for Local Reconnection Method," AIAA Paper 2018–2889, 2018.

[12] Chernikov, A., and Chrisochoides, N., "Parallel Guaranteed Quality Delaunay Uniform Mesh Refinement," *SIAM Journal on Scientific Computing*, Vol. 28, 2006, pp. 1907–1926. doi:10.1137/050625886.

[13] Chernikov, A., and Chrisochoides, N., "Algorithm 872: Parallel 2D Constrained Delaunay Mesh Generation," *ACM Transactions on Mathematical Software*, Vol. 34, 2008, pp. 6–25. doi:10.1145/1322436.1322442.

[14] Linardakis, L., and Chrisochoides, N., "Graded Delaunay Decoupling Method for Parallel Guaranteed Quality Planar Mesh Generation," *SIAM Journal on Scientific Computing*, Vol. 30, 2008, pp. 1875–1891. doi:10.1137/060677276.

[15] Thomadakis, P., Tsolakis, C., Vogiatzis, K., Kot, A., and Chrisochoides, N., "Parallel Software Framework for Large-Scale Parallel Mesh Generation and Adaptation for CFD Solvers," AIAA Paper 2018–2888, 2018.

[16] Feng, D., Chernikov, A. N., and Chrisochoides, N. P., "Two-Level Locality-Aware Parallel Delaunay Image-to-Mesh Conversion," *Parallel Computing*, Vol. 59, 2016, pp. 60–70. doi:10.1016/j.parco.2016.01.007.

[17] Feng, D., Tsolakis, C., Chernikov, A. N., and Chrisochoides, N. P., "Scalable 3D Hybrid Parallel Delaunay Image-to-Mesh Conversion Algorithm for Distributed Shared Memory Architectures," *Computer-Aided Design*, Vol. 85, 2017, pp. 10–19. doi:10.1016/j.cad.2016.07.010.

[18] Feng, D., Chernikov, A. N., and Chrisochoides, N. P., "A Hybrid Parallel Delaunay Image-to-Mesh Conversion Algorithm Scalable on Distributed-Memory Clusters," *Computer-Aided Design*, 2017. doi:10.1016/j.cad.2017.11.006, (in press).

[19] Chrisochoides, N. P., Chernikov, A. N., Foteinos, P., Feng, D., and Tsolaki, C., "On the Scalability of Delaunay-based Isotropic Grid Generation, Codes: A Case Study Using I2M Conversion," *Proceedings of the National Academy of Sciences of the United States of America*, 2019. (to be submitted).

[20] Michal, T., and Krakos, J., "Anisotropic Mesh Adaptation Through Edge Primitive Operations," AIAA Paper 2012–159, 2012.

[21] Michal, T., Kamenetskiy, D., and Krakos, J., "Anisotropic Adaptive Mesh Results for the Third High Lift Prediction Workshop (HiLiftPW-3)," AIAA Paper 2018–1257, 2018.

[22] Michal, T. R., Kamenetskiy, D. S., Krakos, J., Mani, M., Glasby, R. S., Erwin, T., and Stefanski, D., "Comparison of Fixed and Adaptive Unstructured Grid Results for Drag Prediction Workshop 6," *AIAA Journal of Aircraft*, Vol. 55, No. 4, 2018, pp. 1420–1432. doi:10.2514/1.C034491.

[23] Park, M. A., and Darmofal, D. L., "Parallel Anisotropic Tetrahedral Adaptation," AIAA Paper 2008–917, 2008.

[24] Schloegel, K., Karypis, G., and Kumar, V., "Parallel Static and Dynamic Multi-Constraint Graph Partitioning," *Concurrency and Computation: Practice and Experience*, Vol. 14, No. 3, 2002, pp. 219–240. doi:10.1002/cpe.605.

[25] Devine, K. D., Boman, E. G., Heaphy, R. T., Hendrickson, B. A., Teresco, J. D., Faik, J., Flaherty, J. E., and Gervasio, L. G., "New Challenges in Dynamic Load Balancing," *Applied Numerical Mathematics*, Vol. 52, No. 2–3, 2005, pp. 133–152. doi:10.1016/j.apnum.2004.08.028.

[26] Loseille, A., and Alauzet, F., "Continuous Mesh Framework Part I: Well-Posed Continuous Interpolation Error," *SIAM Journal on Numerical Analysis*, Vol. 49, No. 1, 2011, pp. 38–60. doi:10.1137/090754078.

[27] Alauzet, F., "A Changing-Topology Moving Mesh Technique for Large Displacements," *Engineering with Computers*, Vol. 30, No. 2, 2014, pp. 175–200. doi:10.1007/s00366-013-0340-z.

[28] Haimes, R., and Drela, M., "On The Construction of Aircraft Conceptual Geometry for High-Fidelity Analysis and Design," AIAA Paper 2012–683, 2012.

[29] Haimes, R., and Dannenhoffer, J. F., III, "EGADSlite: A Lightweight Geometry Kernel for HPC," AIAA Paper 2018–1401, 2018.

[30] Loseille, A., Menier, V., and Alauzet, F., "Parallel Generation of Large-size Adapted Meshes," *Procedia Engineering*, Vol. 124, 2015, pp. 57–69. doi:10.1016/j.proeng.2015.10.122, 24th International Meshing Roundtable.

[31] Loseille, A., and Löhner, R., "Anisotropic Adaptive Simulations in Aerodynamics," AIAA Paper 2010–169, 2011.

[32] Loseille, A., "Unstructured Mesh Generation and Adaptation," *Handbook of Numerical Methods for Hyperbolic Problems: Applied and Modern Issues*, Handbook of Numerical Analysis, Vol. 18, edited by R. Abgrall and C.-W. Shu, Elsevier, 2017, pp. 263–302. doi:10.1016/bs.hna.2016.10.004.

[33] Loseille, A., and Menier, V., "Serial and Parallel Mesh Modification Through a Unique Cavity-Based Primitive," Sandia National Laboratories, Springer International Publishing, 2014, pp. 541–558. doi:10.1007/978-3-319-02335-9_30.

[34] Loseille, A., and Löhner, R., "Robust Boundary Layer Mesh Generation," Sandia National Laboratories, Springer Berlin Heidelberg, 2013, pp. 493–511. doi:10.1007/978-3-642-33573-0_29.

[35] Loseille, A., "Metric-Orthogonal Anisotropic Mesh Generation," *Procedia Engineering*, Sandia National Laboratories, 2014, pp. 403–415. doi:10.1016/j.proeng.2014.10.400.

[36] Loseille, A., and Alauzet, F., "Continuous Mesh Framework Part II: Validations and Applications," *SIAM Journal on Numerical Analysis*, Vol. 49, No. 1, 2011, pp. 61–86. doi:10.1137/10078654X.

[37] Park, M. A., Loseille, A., Krakos, J. A., and Michal, T., "Comparing Anisotropic Output-Based Grid Adaptation Methods by Decomposition," AIAA Paper 2015–2292, 2015.

[38] Alauzet, F., "Size Gradation Control of Anisotropic Meshes," *Finite Elements in Analysis and Design*, Vol. 46, No. 1–2, 2010, pp. 181–202. doi:10.1016/j.finel.2009.06.028.

[39] Alauzet, F., and Loseille, A., "High-Order Sonic Boom Modeling Based on Adaptive Methods," *Journal of Computational Physics*, Vol. 229, No. 3, 2010, pp. 561–593. doi:10.1016/j.jcp.2009.09.020.

[40] Park, M. A., Balan, A., Anderson, W. K., Galbraith, M. C., Caplan, P. C., Carson, H. A., Michal, T., Krakos, J. A., Kamenetskiy, D. S., Loseille, A., Alauzet, F., Frazza, L., and Barral, N., "Verification of Unstructured Grid Adaptation Components," AIAA SciTech Forum 2019, American Institute of Aeronautics and Astronautics, Reston, VA (submitted for publication).

[41] Ibanez, D. A., "Conformal Mesh Adaptation on Heterogeneous Supercomputers," Ph.D. thesis, Rensselaer Polytechnic Institute, Nov. 2016.

[42] Ibanez, D., and Shephard, M., "Mesh Adaptation for Moving Objects on Shared Memory Hardware," *25th International Meshing Roundtable Research Notes*, Sandia National Laboratories, 2016, pp. 1–5.

[43] Park, M. A., Barral, N., Ibanez, D., Kamenetskiy, D. S., Krakos, J., Michal, T., and Loseille, A., "Unstructured Grid Adaptation and Solver Technology for Turbulent Flows," AIAA Paper 2018–1103, 2018.

[44] Zhou, B. Y., Diskin, B., Gauger, N. R., Pardue, J., Chernikov, A., Tsolakis, C., Drakopoulos, F., and Chrisochoides, N., "Hybrid RANS/LES Simulation of Vortex Breakdown Over a Delta Wing," AVIATION Forum 2019, American Institute of Aeronautics and Astronautics, Reston, VA (submitted for publication).