# Telescopic Approach for Extreme-scale Parallel Mesh Generation for CFD Applications

Nikos Chrisochoides, CRTCLab
Computer Science Department
Old Dominion University

## Abstract

We address *two challenges related to* Extreme-scale Mesh Generation Environments: (1) Design a multi-layered algorithmic and software framework for 3D tetrahedral parallel mesh generation using state-of-the-art functionality supported by our telescopic approach for parallel mesh generation. Our approach explores concurrency at all hardware layers using abstractions at (a) medium-grain level for many cores within a single chip and (b) coarse-grain level, i.e., sub-region and sub-domain level using proper error metric- and application-specific discrete data and continuous decomposition methods. We anticipate that the telescoping approach will be capable of sustaining a billion-way concurrency the next 15 years by (i) leveraging concurrency at different granularity levels and (ii) carefully mapping work units to take advantage of the memory and network hierarchies. (2) Design a Parallel Runtime System for extreme-scale anisotropic mesh computations to target modules like mesh generation and CFD solvers. The runtime system will provide support for: (a) One-sided explicit message passing, (b) Global name-space, (c) multi-threaded programming model for inter-layer interactions, (d) automatic and preemptive load balancing, (e) customizable data-movement and load-balancing, and (f) domain-specific energy-efficient race-to-halt, concurrency throttling, and component-level (core and memory) power scaling. In addition, we present preliminary results from our work on parallel isotropic 3D Delaunay-based guaranteed quality mesh generation on hundreds of cores and potential extensions of to 4D meshes and 3D anisotropic advancing front methods for CFD applications.

## 1. Introduction

Finite Element Mesh Generation is a critical component for CFD computations. We describe deliver a novel framework for highly scalable and energy efficient guaranteed quality mesh generation for the Finite Element (FE) analysis in three and four dimensions. We combine domain-and application-specific knowledge with run-time system support to improve energy efficiency and scalability of parallel FE mesh generation codes. Traditionally, parallel FE mesh generation methods and software are developed without considering the architectural features of the supercomputer platforms on which they are eventually used for production. The main reason is the complexity of sequential, and moreover parallel, mesh generation algorithms. As a result, it is too expensive, in terms of labor and time, to customize the performance of parallel mesh generation software for specific supercomputing architectures. For the same reason, the energy efficiency of these codes is not well understood, since it depends on the architecture. The proposed approach is to abstract and expose parallel mesh generation run-time information to the underlying run-time system which can guide the execution towards the most efficient utilization of resources on the given supercomputer. The issues of performance and energy efficiency are closely related, and we will study them in tandem.

As a result, this project is expected to deliver the first unstructured mesh generation application capable of sustaining concurrency on the order of $10^6$. At the same time, this application will allow for a range of power lowering regimes: from highest speed with opportunistic energy savings, to modest speed with high energy savings. This goal will be achieved by: (1) leveraging concurrency at different granularity levels (i.e., the work units could be sub-domains, regions within a sub-domain, and cavities), (2) carefully mapping these work units to take advantage of

the memory and network architecture of the target platforms, and (3) applying dynamic concurrency throttling in conjunction with a race-to-halt execution model, and component-level power scaling.

## 2. Telescopic Approach for Anisotropic Mesh Generation and Adaptation.

Our plan to achieve extreme-scale adaptive simulations on the complex, heterogeneous HPC architectures that will be increasingly prevalent through 2030 is to implement a telescopic approach (see Figure 1) we initially developed for 2D problems [1] and in the near future integrate it with Advancing Front/Local Reconnection (AFLR) developed at MSU [2, 3] and MOESS methods developed MIT [4]. The telescopic approach is critical to leverage the concurrency that exists at multiple levels for anisotropic and adaptive simulations. At the chip and node levels, the telescopic approach deploys a Parallel Optimistic (PO) layer and Parallel Data Refinement (PDR) layer, respectively (see Sections below). In our current effort, we will focus on the implementation of the PDR layers for AFLR and MOESS. In future efforts, the PDR layer will be implemented on top of PO layer. Based on our preliminary results [28] we expect that the PDR layer will improve PO's scalability by at least a fact of $O(10^2)$.

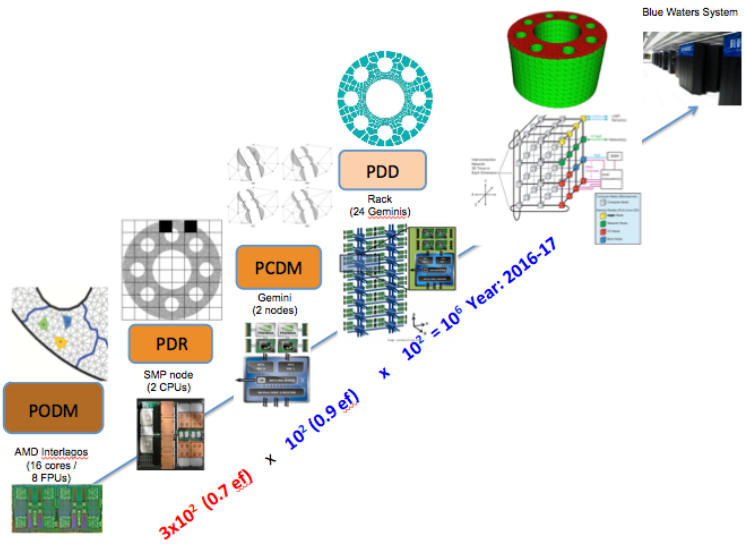


**Figure 1. Telescopic Approach to parallel mesh generation and adaptation.**

Together after optimizations we expect to get about $10^4$ -way concurrency i.e., close to linear weak speedup: in other words for $10^4$ cores achieve about 0.8 parallel efficiency. Then at the super-node and/or rack level, the telescopic approach deploys a Parallel Constrained layer [29], with its asynchronous communication is expected to perform well at higher latency and lower bandwidth network switches utilized at the super-node and/or rack levels. In combination with PDR, it is expected to deliver about $10^5$-way concurrency. Finally, for inter-rack level, we plan to use a Parallel Domain Decoupled (PDD) layer [5] that requires very little to no communication. Our experience indicates that PDD can easily scale up to $10^3$, so in combination all four layers-methods together are expected to deliver concurrency in the order of $10^6$ to $10^7$. The "work-horse" of the telescopic approach is the PO layer i.e., all cores in the system run PO at the lowest level. The rest of the layers like PDR etc. are used to explore additional concurrency at additional levels of granularity (eg. mesh data-block, sub-region and subdomain). Both mesh generation and adaptation challenges will be addressed at the lowest level, with one exception the PDD. The domain decomposition should take advantage of existing metric information available in our adaptive algorithm. In the past we developed application-specific decompositions [6]. In this project we will extend them to be error metric-specific, too. Both the data and domain decomposition methods will over-decompose the geometry so that issues like load-balancing and out-of-core (in the future) in parallel mesh generation and adaptation will be managed by a multi-layered runtime system (see below). This separation of concerns helps portability, maintainability, and performance and thus enables labor-effective and performance-efficient extreme-scale parallel mesh generation and adaptation software.

The design of the telescopic approach is based on a combination of data and domain decomposition methods in order to explore locality at the chip, single node, and multiple nodes levels. In this section, we discuss the layers of the telescopic approach for the AFLR-based meshing. In the next section, we describe the telescopic implementation of the MOESS algorithm.

**2.1 Parallel Optimistic (PO) layer at the chip level.** This is a tightly coupled method [7,8]. Our preliminary data on 3D Delaunay methods indicate that speculative (optimistic) methods perform well on hardware shared memory [8] i.e., single chip. We expect the number of cores to emerging chips and for HPC nodes will be in the order of hundreds. Figure 4, depicts the concurrent expansion of three cavities, however because of the speculative nature of this layer, roll-backs are possible, due to a possible intersection of cavities. Such intersections lead to non-conforming or non-Delaunay meshes. In [8] we present a way to efficiently implement such codes for 3D Delaunay based methods. We will investigate a new Parallel Optimistic for AFLR method based on our earlier Parallel Optimistic Delaunay Mesh (PODM) work [8,9] on speculative execution model for parallel Delaunay-based methods. AFLR similar to Delaunay refinement algorithms works by inserting additional points (so-called Steiner for Delaunay methods) into an existing mesh to improve the quality of the elements.

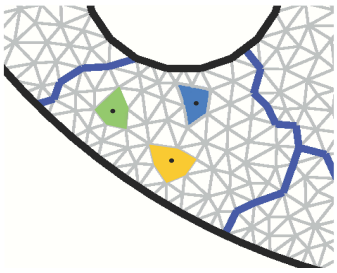

**Figure 2. Concurrent cavity expansion according to PODM.**

**2.2 Parallel Data Refinement (PDR) layer at the node level.** This is partially coupled method [10,11] with locally synchronous communication. In our previously published results we developed a distributed memory (MPI) implementation of two-dimensional [10,11] and 3-dimensional [12] Parallel Delaunay Refinement methods. The partitioning and the decoupling of the subdomains was achieved by creating a special buffer zone around every subregion, so that we can mathematically guarantee that the insertion of a point in one subregion can modify the triangles only inside this subregion and its buffer zone, but the changes do not propagate to other subregions and their buffer zones. After refining the subregions, the buffer zones are refined in a similar way. An adjustment to the sequential Delaunay refinement code allowed for the insertion of only those circumcenters that satisfy some user defined condition (i.e., being inside a given box). The drawback is that the amount of work assigned to different processors can vary significantly.

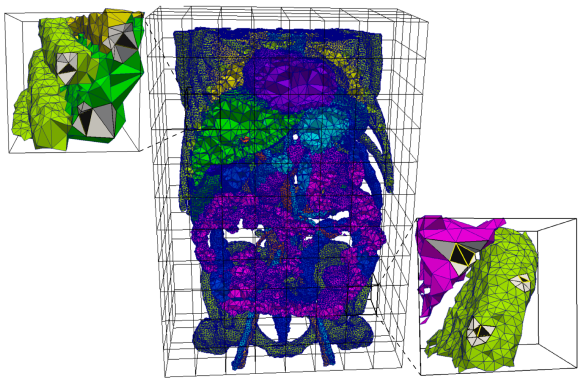

**Figure 3. The first two layers of the telescopic approach for a 3D abdominal multi-material atlas. The PDR blocks left and right are meshed in parallel (eg. on different nodes) and within each block there are cavities that are computed concurrently (eg. on different cores of each node) using speculative execution PODM layer.**

However, by using over-decomposition in a combination with runtime software system we developed [13] for dynamic load balancing, we can redistribute the work among the nodes. Figure 5 depicts the results of the first two layers of telescopic approach for Delaunay-based methods using a 3D abdominal multi-material atlas; the mesh is generate on 256 cores using Parallel Data

Refinement (PDR) by exploring concurrency at data block level [10-12] and Parallel Optimistic Delaunay Mesh (PODM) by exploring concurrency at the cavity level [7, 8, 9]. The telescopic approach works both for parallel mesh generation and adaptation. The adaptation step is more efficient since all data structures are in place for either refinement or de-refinement operations that rely on an abstraction similar to the cavity.

Like Delaunay-based methods AFLR is a volume meshing method that will be used (after proper modifications) for the individual sub-regions (i.e., leaves of the octree). Figure 6 depicts the weak speedup on a 256 core Distributed Shared Memory (DSM) machine [14]. These results show the impact of PDR on 3D Parallel Optimistic Delaunay Meshing (PODM). PDR improves scalability due to improved data locality and the absence of rollbacks. We expect the implementation of the PDR layer on distributed memory machines to further improve the performance we are getting on DSM (cc-NUMA) machines like the Blacklight at Pittsburgh Supercomputing center.

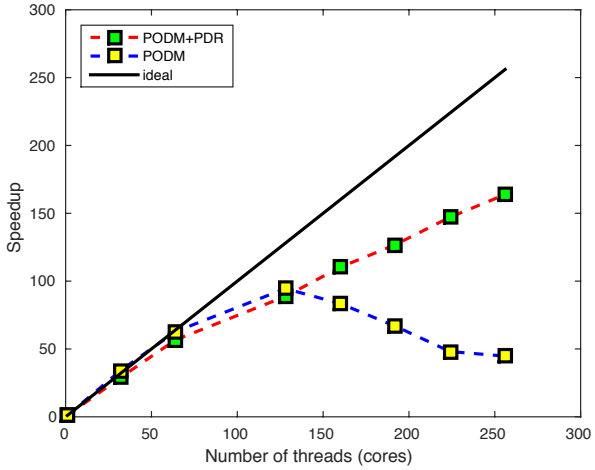


**Figure 4. Weak Speedup comparison of PODM and its combination with PDR (PDR.PODM) for meshes that vary from 3M tets on a single core to 745M tets on 256 cores [28].**

**2.3 Parallel Constrained (PC) layer.** This is partially coupled method [15] with asynchronous communication. With PC we will combine first the PDR and PO into a single highly efficient parallel and scalable engine: the PDR.PO.AFLR which will be used for the implementation of the PC. Each subdomain contains the collections of the constrained faces, edges, and points. In the case of Delaunay-based methods we used use the Bowyer-Watson kernel for mesh generation. The constrained (boundary) segments are protected by diametral lenses and each time a segment is encroached, it is split in the middle; as a result, a split message is sent to the neighboring subdomain [15]. In this project, we will explore the use of similar tools in the case of AFLR-based methods. PC.AFLR will be designed to run on multi-processor nodes and clusters of nodes, i.e., it uses the message-passing paradigm. Each process lies in its own address space and uses its own copy of a custom memory allocator. In addition to reducing substantially the communication PC.AFLR can be further optimized by using message aggregation of the split-messages, which are sent between neighboring subdomains as a result of inserting points on the common boundaries [29].

**2.4 Parallel Domain Decoupled (PD²) layer** is similar to PC layer, however before the subdomains become available for further processing by the PC.PDR.PO layer they are discretized using the pre-processing step similar to one we presented in [5, 6]. This guarantees that any Delaunay-based algorithm can generate independently a mesh on each of the subdomains in a way that does not introduce any new points on the boundary of the subdomains (i.e., the algorithm terminates and can guarantee conformity and Delaunay properties without the need to communicate with any of the neighbor subdomains).

**2.5 Domain Decomposition**. There are numerous methodologies that could be employed for obtaining a domain decomposition. Different approaches will be used for the over-decomposition of the parallel mesh generation and adaptation phases. In addition to the data decomposition, a domain decomposition for the parallel mesh generation phase will be considered based on a

Medial Axis Domain Decomposition (MADD) software, similar to one we presented [5, 6], for 2D geometries. MADD can produce domain decompositions which satisfy the following three basic criteria: (1) the boundary of the subdomains create good angles, i.e., angles no smaller than a given tolerance, where the value of the tolerance is user-defined, (2) the size (area) of the separator should be relatively small compared to the volume of the subdomains and (3) the subdomains should have approximately equal mesh-size (even if the mesh is non-uniform) given application-specific size function. The domain decomposition uses an approximation of a Medial Axis as an auxiliary structure for constructing the boundary of the subdomains (separators). Figure 7, depicts the result of a 2D MADD. Notice that the decomposition satisfies constrains 1-3, in contrast to data decompositions (see Figure 4) where those conditions are not guaranteed. However, in the PODM and PO.AFLR the interfaces are not preserved and thus, there is no need to satisfy conditions 1-3. The domain decomposition approach is based on "Divide and conquer" algorithmic paradigm and a smoothing procedure for eliminating the creation of any new artificial features in the subdomains. However, in the adaptation phase we will only consider those approaches that impose no restrictions on the mesh generation, e.g. a fixed artificial boundary like we plan to use in the PDR layer. We also have available, after initial mesh generation, the existing mesh (from the previous solution adaptation step) along with the new metric values. A suitable decomposition can be derived from that mesh. In all cases we will utilize the metric values to determine the estimated mesh size in each subdomain. This estimate progressively will become more accurate as the solution evolves and the mesh is further resolved via the adaptation. In addition, exploratory work with AFLR indicates it could be used to generate a very coarse mesh that mimics the true mesh and can be used as a domain decomposition that satisfies conditions 1-3 naturally.

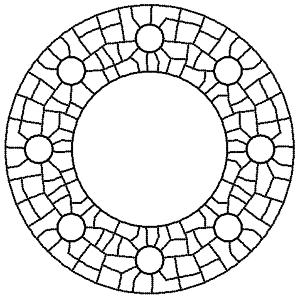

**Figure 5. Medial Axis Domain Decomposition.**

## 3. Parallel Runtime Software System

The upcoming Aurora supercomputer, and likely many others in the future, will have multiple levels of memory with slower but larger non-volatile memory supplementing faster RAM. Burst SSD storage can be seen as yet another slower intermediate layer of memory. Also, due to power constraints, the amount of RAM per core will likely shrink considerably as more cores are integrated into a single CPU. This makes our out-of-core system [16] even more relevant since exploiting memory locality and controlling placement of data in fast or slow memory will be critical to achieve adequate performance. In addition, having "traditional" out-of-core support by temporarily retiring data to a disk would be beneficial since it would facilitate such tasks as checkpointing (out of the scope of this project; it will be addressed in follow up phases). Indeed, resilience is important to current supercomputers (e.g., Blue Waters requires some remedial repair actions every few hours ) and is expected to be a

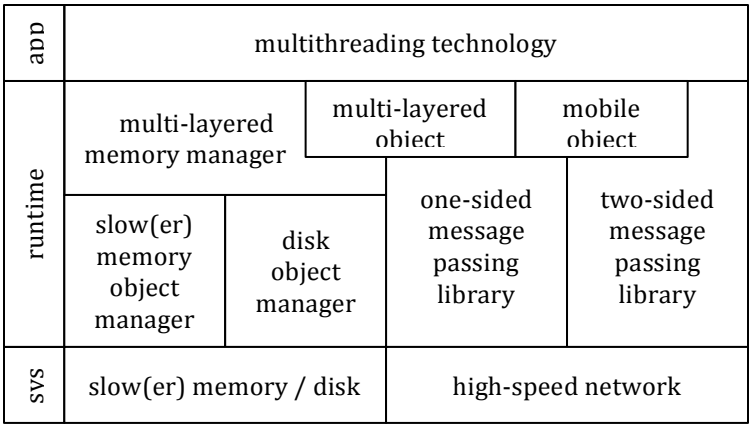


**Figure 6. Potential proposed MRTS design based on current and emerging hardware and software technologies for HPC.**

major challenge to exascale computing. Having built-in capabilities that enable checkpointing or other forms of algorithmic fault tolerance would be highly beneficial.

Given these current and emerging trends in parallel hardware, our long-term goal is to redesign a Multi-Layer Runtime System (MRTS) [13,16] based on the evaluation of current and emerging software technologies we reviewed earlier. We also plan to extensively evaluate new technologies in the next three years. This study will help us to identify and adopt suitable state-of-the-art communication substrates of interest to NASA, so that we can implement a global address space with one-sided communication and load balancing libraries. These libraries will be used to implement the Telescopic approach to parallel anisotropic mesh generation and adaptation.

Figure 6 depicts the high level design of the runtime system. At the top, the application will directly interact with a multithreading technology to exploit fine grain parallelism. The data of the application will be organized into distributed data-structures consisting of mobile objects [17] linked via mobile pointers. The mobile object layer will provide a global addressing scheme and an efficient mechanism for communicating between objects and forwarding to enable seamless migration of the mobile objects. While the mobile object layer is responsible for communication between nodes, the multi-layered object manager controls the placement of local objects within a local nodes as well as scheduling the delivery of messages to/from the local objects. It is integrated with a multi-layered memory manager that is responsible for managing message and object pools in all available layers of memory. Low-level one-sided message passing technology will serve as the communication substrate to the runtime system. Interoperability with MPI will be preserved to leverage collective communication and legacy parallel codes as well as codes that will be developed independently of our system and this project. The runtime system will manage the movements and storage of objects, as well as placement of objects into fast or slow memory. There is a trade-off between reusing as much of an existing application code as possible and taking full advantage of the runtime system. The less the application will rely on using MPI the more control will be given to the runtime system and ultimately better performance can be achieved and more features (e.g., “free” checkpointing) can be utilized. Moreover, moving to a different platform should require virtually no porting and considerably less tuning since that should be handled by the runtime system transparently to the user. The benefits of the MRTS system will become apparent in the adaptation phase, where only some parts of the mesh need to be refined and de-refined. In this case, the Implicit Load Balancing (ILB) library [13] will be used to develop error- and application-specific policies for load balancing. The ILB library will be built on top of MRTS system.

## 4. Putting It All Together

First we will focus on the implementation of PDR layer of the telescopic approach. Each block of the PDR phase will be prepared so that it can eventually become a Mobile Work Unit (MWU), which can migrate to any node of the system without the programmer being concerned about hardware-dependent communication and load balancing issues. As we showed in [38], this abstraction is extremely convenient for the development of parallel mesh generation codes, and is indispensable for two of the most challenging problems in parallel unstructured mesh generation: dynamic data movement & control and load balancing. Given the ability to create and migrate MWUs, the MRTS system will implement high-level logic [16] by monitoring the status of the system and the available objects (i.e., MWUs), and rearranging them accordingly across the distributed memory of the hardware platform. In the case of adaptation, we will use either existing or we can develop new policies using the ILB library that will take into account the error-based metrics in order to tolerate latencies i.e., migrate MWU based on look ahead mechanism for work-load imbalances or power savings using information by analyzing error-based metrics.

**Acknowledgements** This work in part is funded by NSF grant no. CCF-1439079, NASA grant no. NNX15AU39A and DoD's PETTT Special Project PP-CFD-KY07-007. In addition, it utilized resources from the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1053575. Finally, it is partially supported by the Richard T. Cheng Endowment. The content is solely the responsibility of the authors and does not necessarily represent the official views of the government agencies that support this work. The author developed the parallel mesh generation methods with his PhD students Dr. A. Chernikov, Dr. L. Linardakis and Dr. P. Foteinos and runtime systems with his PhD students Dr. K. Barker and A. Kot. On going work in parallel mesh generation involves his current graduate students T. Kennedy, D. Feng and C. Tsolakis. In addition he recently discussed his telescopic approach to parallel mesh generation with Dr. D. Marcum (MSU) and D. Darmofal (MIT) for its application to AFLR and Error-based metrics for large scale CFD, in the context of the NRA and DoD grant applications. The author thanks Dr. D. Darmofal for his help in improving the clarity of the ideas presented in this paper.

**References**

[1] N. Chrisochoides, A. Chernikov, A. Fedorov, A. Kot, L. Linardakis, and P. Foteinos, "Towards Exascale Parallel Delaunay Mesh Generation," B. W. Clark, Ed., ed: Springer Berlin Heidelberg, 2009, pp. 319-336.

[2] D. L. Marcum, "Unstructured Grid Generation Using Automatic Point Insertion and Local Reconnection," in Handbook of Grid Generation, ed: CRC Press, 1998.

[3] D. L. Marcum and N. P. Weatherill, "Unstructured grid generation using iterative point insertion and local reconnection," AIAA Journal, vol. 33, pp. 1619-1625, 1995.

[4] M. Yano and D. L. Darmofal, "An optimization-based framework for anisotropic simplex mesh adaptation," Journal of Computational Physics, vol. 231, pp. 7626-7649, Sep. 2012.

[5] L. Linardakis and N. Chrisochoides, "Graded Delaunay Decoupling Method for Parallel Guaranteed Quality Planar Mesh Generation," SIAM Journal on Scientific Computing, vol. 30, pp. 1875-1891, Jan. 2008.

[6] L. Linardakis and N. Chrisochoides, "Algorithm 870: A Static Geometric Medial Axis Domain Decomposition in 2D Euclidean Space," ACM Trans. Math. Softw., vol. 34, pp. 4:1–4:28, Jan. 2008.

[7] N. Chrisochoides, "Parallel Mesh Generation," in Numerical Solution of Partial Differential Equations on Parallel Computers, A. M. Bruaset and A. Tveito, Eds., ed: Springer Berlin Heidelberg, 2006, pp. 237-264.

[8] P. Foteinos and N. Chrisochoides, "High quality real-time Image-to-Mesh conversion for finite element simulations," Journal of Parallel and Distributed Computing, vol. 74, pp. 2123-2140, Feb. 2014.

[9] D. Nave, N. Chrisochoides, and L. P. Chew, "Guaranteed-quality parallel Delaunay refinement for restricted polyhedral domains," Computational Geometry, vol. 28, pp. 191-215, Jun. 2004.

[10] A. Chernikov and N. Chrisochoides, "Parallel Guaranteed Quality Delaunay Uniform Mesh Refinement," SIAM Journal on Scientific Computing, vol. 28, pp. 1907-1926, Jan. 2006.

[11] A. N. Chernikov and N. Chrisochoides, "Practical and Efficient Point Insertion Scheduling Method for Parallel Guaranteed Quality Delaunay Refinement," in Proceedings of the 18th Annual International Conference on Supercomputing, 2004, pp. 48–57.

[12] A. Chernikov and N. Chrisochoides, "Three-dimensional Delaunay Refinement for Multi-core Processors," in Proceedings of the 22nd Annual International Conference on Supercomputing 2008, pp. 214–224.

[13] K. Barker, A. Chernikov, N. Chrisochoides, and K. Pingali, "A load balancing framework for adaptive and asynchronous applications," IEEE Transactions on Parallel and Distributed Systems, vol. 15, pp. 183-192, Feb. 2004.
[14] D. Feng, C. Tsolakis, A. Chernikov, and N. Chrisochoides, "Scalable 3D Hybrid Parallel Delaunay Image-to-Mesh Conversion Algorithm for Distributed Shared Memory Architectures," Proceedings of 24th International Meshing Roundtable, 2015.
[15] A. Chernikov and N. P. Chrisochoides, "Algorithm 872: Parallel 2D Constrained Delaunay Mesh Generation," ACM Trans. Math. Softw., vol. 34, pp. 6:1-6:20, Jan. 2008.
[16] A. Kot, A. N. Chernikov, and N. P. Chrisochoides, "The Evaluation of an Effective Out-of-Core Run-Time System in the Context of Parallel Mesh Generation," in Parallel Distributed Processing Symposium (IPDPS), 2011 IEEE International 2011, pp. 164-175.
[17] N. Chrisochoides, Barker, K., Nave, D., Hawblitzel, C., "Mobile object layer: a runtime substrate for parallel adaptive and irregular computations," Advances in Engineering Software, vol. 31, pp. 621-637, 2000.