A Parallel Extension for MSKT (MeSh ToolKit)

Duo Wong, Stony Brook University;
Rao Garimella, T-5

MSTK is a mesh framework that allows users to represent, manipulate, and query unstructured 3D arbitrary topology meshes in a general manner without the need to code their own data structures. The interaction of users and applications with MSTK is through a functional interface that acts as though the mesh always contains vertices, edges, faces, and regions, and maintains connectivity between all these entities.

Parallel MeSh ToolKit (MSTK) adds support for simulations with unstructured meshes on high performance computers. It manages parallel aspects of an unstructured mesh, including mesh partitioning and communication among processors. Each processor owns a subset of the global mesh and performs calculations on it, and, after each time step, synchronizes boundary data through communication with other processors. The submesh on each processor itself is a valid MSTK mesh and stores an extra layer of information to minimize communication cost. The design of parallel MSTK makes it possible to concurrently conduct communication and calculation to achieve good scaling. The interaction between users and applications of parallel MSTK is through a user-friendly function interface that hides most of the communication details so that the user can focus on the computational tasks on each processor. Parallel MSTK is still being enhanced and tested, but will soon be released for open use.

In parallel MSTK, a global mesh is partitioned into submeshes according to its highest dimension entity. The result of a partition operation is that each face or region in the global mesh gets a part number. Entities with the same part number are connected (see Fig 1). All the lower-dimensional entities may be shared among two or more parts, but only one part owns any given lower-dimensional entity. The owner partition of an entity is solely responsible for updating data on it. Each entity of a particular dimension has a unique global ID. Currently, parallel MSTK uses Metis as underlying partition algorithm, but it can use other partitioning algorithms as well.

Most numerical methods such as finite element, finite volume, and finite difference methods require adjacent entity information to perform local calculations. However, on a partition boundary, this information is not readily available unless through communication with other processors. A mesh partition in parallel MSTK stores this extra layer of information as ghost data to reduce communication cost. Ghost data gets updated after each time step through a collective communication among all the processors.

Communication in parallel MSTK is based on the message passing interface (MPI) standard. Parallel MSTK is designed such that each ghost entity knows its master, but not vice versa. Each partition keeps a record of neighboring partition IDs and sends the corresponding data to each of these partitions. On the receiving side, ghost entities find updates in the receive buffer based on their global IDs. Parallel MSTK labels each entity in a partition as interior, boundary, or ghost, from interior to exterior respectively. This separation provides the benefit of concurrent computation and communication. While waiting for the updates of ghost data, the application can continue the next time step calculations on interior entities which do not require ghost data, and finish the boundary calculations after the ghost data get updated.

There are two methods for I/O in parallel MSTK – an application can import a global mesh on the root processor, partition and scatter it to other processors, or each processor can read in the "checkpoint" MSTK file that was previously written by the same processor. The latter method can reduce the serial processing bottleneck on the root processor, and also has the benefit of quickly restarting computation for time-consuming simulations.

For more information contact Rao Garimella at rao@lanl.gov.

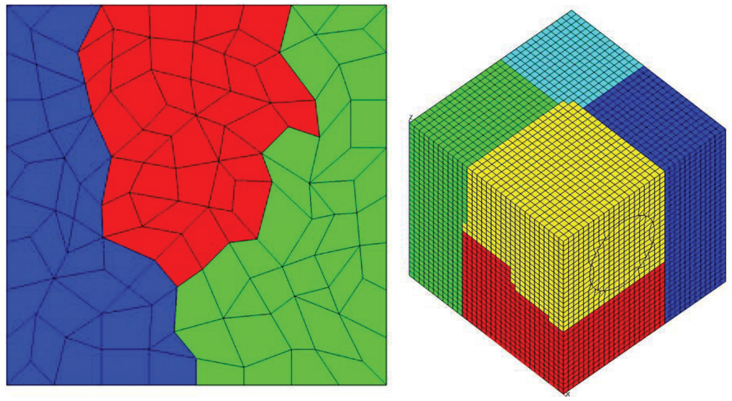

Fig. 1. Partitioning of 2D and 3D unstructured meshes.

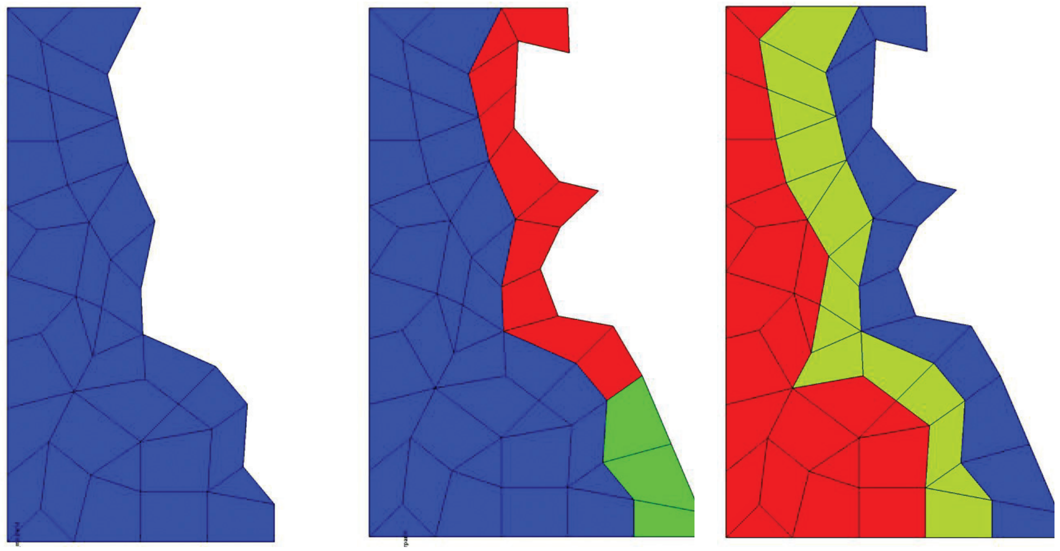

Fig. 2. From left to right: partition 0 of unstructured 2D mesh shown in Fig. 1, partition with ghost entities (color indicated master partition ID), and the color map of interior, boundary, and ghost faces.

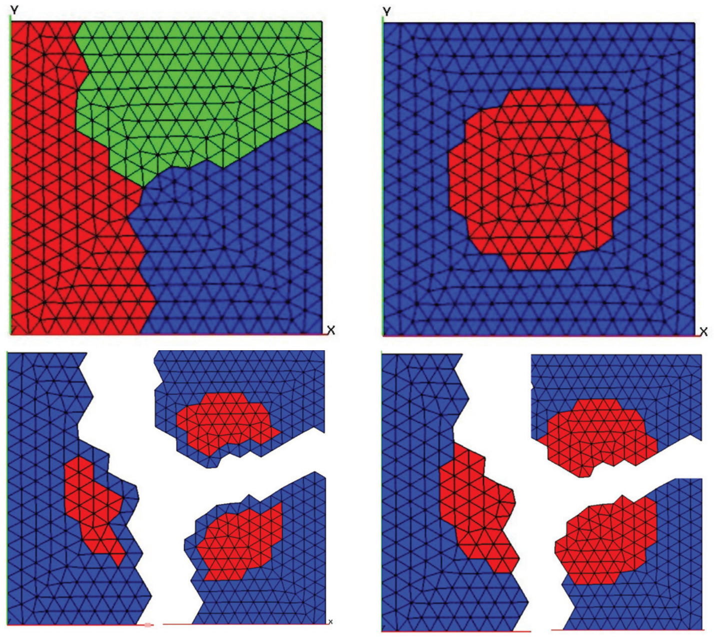

Fig. 3. Global mesh partitioned into three parts. Top right: faces that have at least one vertex inside a circle are marked red. Bottom left: each partition deals with faces belonging to itself; no ghost faces are updated. Bottom right: after communication, ghost faces are updated.

Funding Acknowledgments
Advanced Simulation Capability program