Efficient Distribution of a Parallel Job Across Different Grid Sites

E. Yilmaz, R.U. Payli, H.U. Akay*, and A. Ecer
Computational Fluid Dynamics Laboratory
Department of Mechanical Engineering
Indiana University-Purdue University Indianapolis (IUPUI)
Indianapolis, Indiana 46202 USA
*E:mail: hakay@iupui.edu; Web Page: http://www.engr.iupui.edu/cfdlab

Key Words: TeraGrid, Distributed Computing, Parallel Computing, Load Balancing, Graph Partitioning

ABSTRACT

Introduction

With the significant infrastructure investments in recent years, grid computing continues to challenge the way scientific computing has been done up until a decade ago. TeraGrid [1], an NSF-supported tera-scale grid computing environment established by combining computational resources of eight geographically different centers in USA as a single resource pool, is now readily available to researchers for solving grand-challenge problems. As demands towards scientific discoveries through high-performance computing are increasing rapidly, the use of these resources are expected to become more routine. With the recent initiative of the NSF Office of Cyberinfrastructure [2] to fund peta-scale high-performance grid computing capabilities, formation of a PetaGrid may soon become a reality, paving the way to scientific discoveries.

Along with the tera or peta-scale investments, computational researchers need more advanced tools which can facilitate using full capacity of these environments. One such tool for the researchers in the parallel computational fluid dynamics field is MPICH-G2, which is the Grid enabled implementation of the Message Passing Interface (MPI) [3]. This utility is capable of starting a parallel job across Grid sites with some predetermined CFD data block distribution. The fact that CFD data blocks are highly interconnected requires efficient distribution of the blocks such that communication between the Grid sites for the particular parallel job would be at minimum. While the blocks communicate at high speed within the same Grid site, they communicate with whatever bandwidth is provided between the Grid sites. This makes the inter-connection between the Grid sites a bottleneck. Therefore, CFD code developers and researchers need to optimize this communication by efficiently allocating the blocks among the Grid sites.

In the past, we have developed Dynamic Load Balancing (DLB) tools [4, 5] which can facilitate the CFD block distribution among available computing resources, mostly clusters of processors. DLB is very useful in environments where processors loads are changing dynamically. However, if the computing resource is dedicated for a specific job, then a static distribution would be sufficient for this job. In TeraGrid sites, resource schedulers handle internal distributions of parallel tasks or chunks of dispersed sub-tasks across Grid sites. Therefore, a static distribution of parallel CFD blocks between Grid sites would be sufficient provided that this distribution is optimum or efficient.

We had experience in TeraGrid with running jobs independently at different sites [6]. In this paper, we will demonstrate an efficient CFD block distribution algorithm across Grid sites such as TeraGrid. The blocks will be grouped together such that inter-group communication volume is at minimum. This will ensure optimum distribution of the blocks, hence minimum communication between the Grid sites. We apply MeTis graph partitioning libraries [7] for the distribution of the CFD blocks across the clusters. MeTis ensures optimum partitioning of the graph, same as in partitioning CFD mesh blocks for parallel flow solver. We will also consider improvements to this block distribution regarding different computational power of the Grid sites.

Block Distribution Across Grid Sites

A parallel CFD job executes on partitioned data block, called mesh blocks. As the number of the partitioned mesh blocks increase, dependencies of one block to other increases, hence the communication overhead. When using multiple sites, a user can balance the load and communication between the clusters so that overall time for the job would be lowest possible. If users are unaware of which block communicates to which, then the opportunity of using different clusters will be lost.

In our studies, we used MeTis graph partitioner for the mesh block distribution across more than one Grid site. In partitioning the graph, communication volume between a pair of mesh block is the size of actual interface mesh points, since those mesh points will be used to communicate with neighbor blocks. MeTis graph partitioner can use communication volume between the blocks as weights between interconnected edges. Based on these weight, graph is partitioned for optimum cut into as many as number of Grid sites to be used. Figure 1 shows a schematic of four mesh blocks and their inter-connect distributed between two Grid sites.

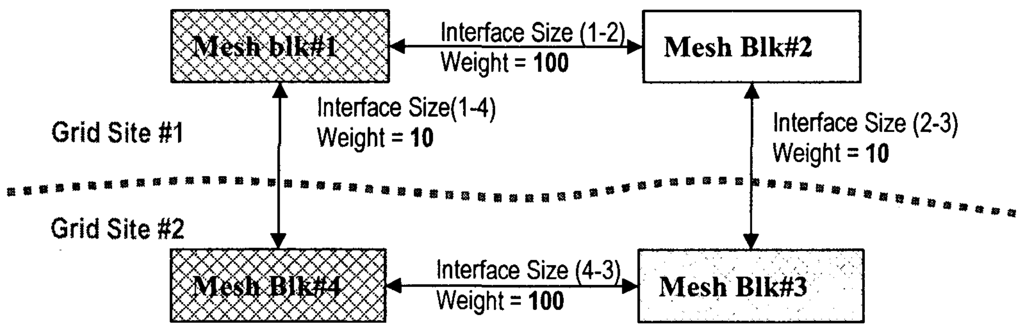

Figure 1: Four mesh blocks distributed between two Grid sites based on communication volume

Demonstration Case

In our application, the mesh data contains 512 mesh blocks. Mesh has about 18 millions of tetrahedral elements. Computational mesh of this problem at airplane surface is given in Figure 2. The airplane geometry is known as DLR-F6 wing-body configuration. The flow solver used is PACER3D [8], a parallel unstructured mesh flow solver.

To demonstrate the mesh block distribution, we assumed four Grid sites. Initially, user is unaware of block-to-block interconnection structure. User does not know that there are 3090 connections between blocks; this is actually edges between the graph nodes. Maximum number of neighbor of a block is 18. Therefore, mesh blocks are randomly assigned to four grid sites. One can imagine that such an unstructured communication can cause serious overhead problem.

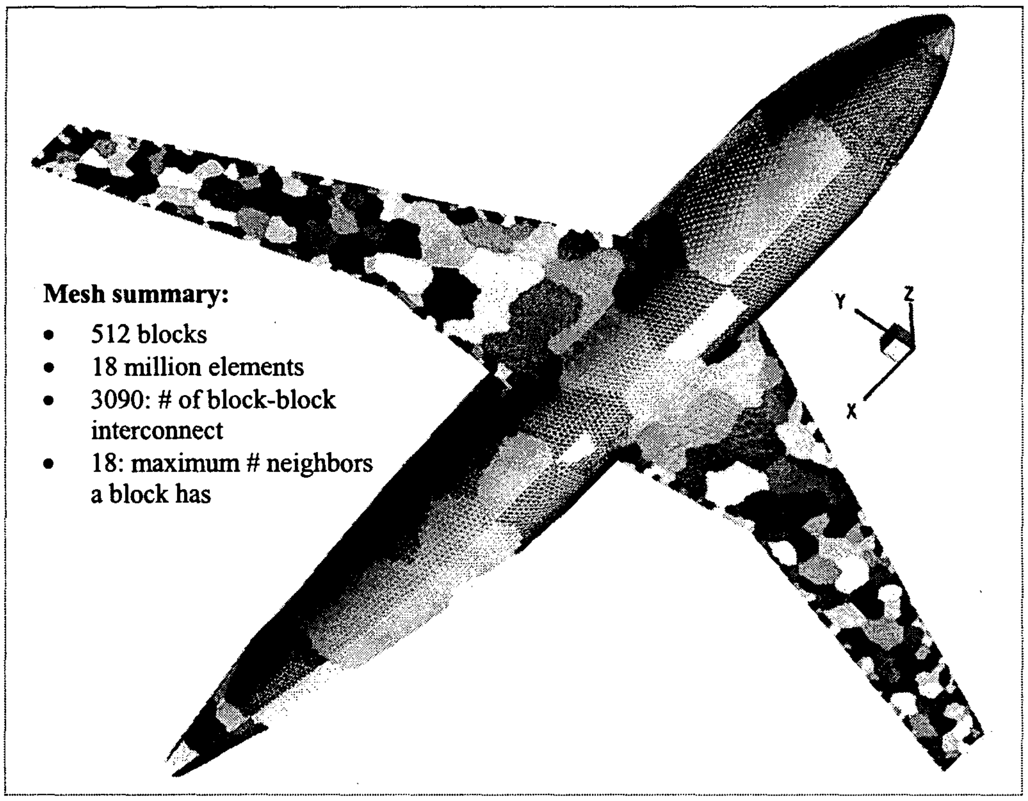
Mesh summary:
• 512 blocks
• 18 million elements
• 3090: # of block-block interconnect
• 18: maximum # neighbors a block has
Y
Z
X

Figure 2: DLR F6 wing body with 512 CFD mesh blocks

Tables 1 and 2 show number of block-to-block interconnect between the Grid sites and regarding communication volume using random block allocation and using graph partitioning, respectively. Considering total number of blocks, 512, random block distribution has five times more block-to-block interconnect between Grid sites than that of using graph partitioning. Even communication volume is about eight times much higher than graph partitioning. Actual communication time for each case depends on the synchronization of send-receive orders of all these blocks and actual bandwidth between Grid sites. Therefore, time measurement for actual job is essential to conclude real benefit of the graph partitioning. Figure 3 shows block distribution on actual airplane geometry. While blocks are scattered around without considering neighbor blocks in random distribution, the graph partitioning grouped neighbors together so that there would be optimum communication volume between Grid sites as well as between the blocks within each Grid site.

Timing of different combinations run on the TeraGrid will be presented in the full paper along with the parallel efficiency studies.

Table 1a: number of connected blocks across Grid sites with random block distribution

	Site #1	Site #2	Site #3	Site #4	Total
Site #1		427	389	396	1,212
Site #2	427		409	412	1,248
Site #3	389	409		428	1,226
Site #4	396	412	428		1,236

Table 1b: Communication volume between Grid sites with random block distribution

	Site #1	Site #2	Site #3	Site #4	Total
Site #1		156,021	108,597	123,634	388,252
Site #2	156,021		115,114	122,325	393,460
Site #3	108,597	115,114		165,192	388,903
Site #4	123,634	122,325	165,192		411,151

Table 2a: Number of connected blocks across Grid sites with graph partitioning

	Site #1	Site #2	Site #3	Site #4	Total
Site #1		186	16	109	311
Site #2	186		29	27	242
Site #3	16	29		115	160
Site #4	109	27	115		251

Table 2b: Communication volume between Grid sites with graph partitioning

	Site #1	Site #2	Site #3	Site #4	Total
Site #1		44,338	2,970	26,431	73,739
Site #2	44,338		6,105	6,434	56,877
Site #3	2,970	6,105		27,034	36,109
Site #4	26,431	6,434	27,034		59,899

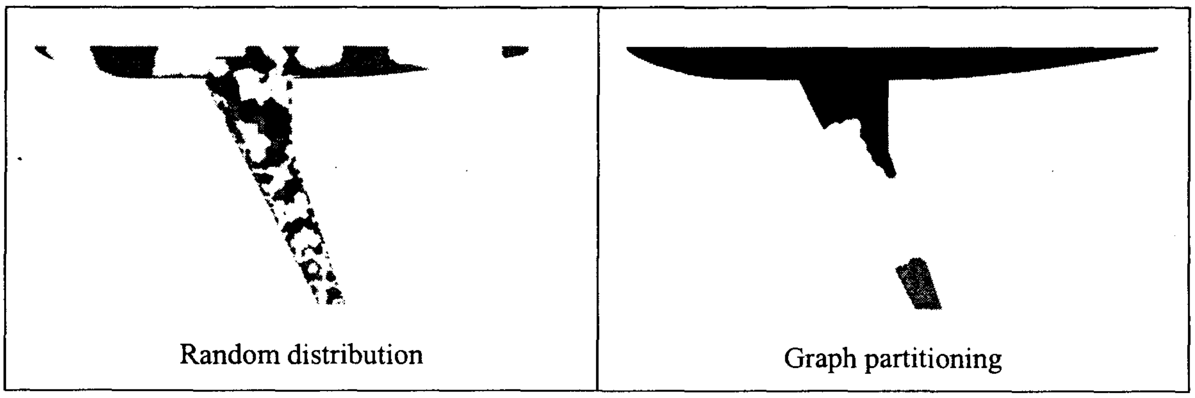

Figure 3: Block distribution across four Grid sites (colors show different Grid sites)

REFERENCES

1. TeraGrid, *http://www.teragrid.org*.
2. NSF Office of Cyberinfrastructure, *http://www.nsf.gov/dir/index.jsp?org=OCI*.
3. N.T. Karonis, B. Toomen, and I. Foster, 밝 PICH-G2: A Grid-Enabled Implementation of the Message Passing Interface,? *J. Parallel and Distributed Computing*, vol. 63, no. 5, 2003, pp. 551-563.
4. A. Ecer, H.U. Akay, W.B. Kemle, H. Wang, D. Ercoskun, and E.J. Hall, 밣arallel Computation of Fluid Dynamics Problems,? *Computer Methods in Applied Mechanics and Engineering*, 112, 1994, pp. 91-108.
5. S. Chien, Y. Wang, A. Ecer, and H.U. Akay, 및 rid Scheduler with Dynamic Load Balancing for Parallel CFD,? *Parallel CFD 2003,* Edited by B. Chetverushkin, et al., Elsevier Science, pp. 259-266, 2004.
6. R.U. Payli, H.U. Akay, A. Baddi, E. Yilmaz, A. Ecer, and E. Oktay, 뮗 FD Applications on TeraGrid, *Proceedings of Parallel CFD 2005*, Edited by A. Deane, et al., Elsevier Science, 2006 (in print).
7. G. Karypis and V. Kumar, 밝 ultilevel Algorithms for Multi-constraint Graph Partitioning,? *Technical Report TR 98-019*, Department of Computer Science, University of Minnesota, 1998.
8. E. Yilmaz, M.S. Kavsaoglu, H.U. Akay, and I.S. Akmandor 뮐 ell-Vertex Based Parallel and Adaptive Explicit 3D Flow Solution on Unstructured Grids,? *The International Journal of CFD*, Vol. 14, pp. 271-286, 2001.