Implementation of a 3D Hilbert SFC into a Parallel Cartesian- Grid Flow Solver

Stephen M. Ruffin and Jinwook Lee*

Abstract **-** The efficient parallel computation of unstructured grid flow solver requires an adequate grid decomposition strategy because of its complex spatial data structure. The difficulties of even and block-contiguous partitioning in frequently adapting unstructured Cartesian grid are overcome by implementing the 3D Hilbert space-filling-curve (SFC). Grids constructed by SFC in a parallel environment promise shorter inter-CPU communication time while maintaining perfect load balancing between CPUs. The load imbalance due to local solution adaption is simply apportioned by re-segmenting the curve into even pieces. The detailed structure of 3D Hilbert SFC and the parallel computing efficiency results based on this grid partition method are also presented.

Keywords **–** Cartesian Grid, Parallel Computing, Space Filling Curves, SFC, Hilbert

I. INTRODUCTION

IN general, parallel algorithms are categorized into a s either task decomposition or data decomposition. Task (or event) decomposition identifies tasks that can be executed concurrently. Data (or domain) decomposition identifies local data for each task. Parallelization of a computational fluid dynamics solver (CFD) is achieved by first partitioning the computational domain into several smaller zones and distributing them to different CPUs. Next, inter-CPU boundary cell information is shared to update the state vectors of the boundary cells. In this way, parallelization saves the wall clock calculation time by sharing the total CPU time by multiple processors.

The two main conditions of parallel computing efficiency are load balancing and grid locality. Load balancing involves synchronizing the amount of CPU time per iteration between CPUs. Grid locality involves reducing the communication time between CPUs by minimizing the inter-CPU boundary surface area. Satisfying these two conditions is not a challenging task in structured grids solver where the grids are continuously indexed by constant i,j,k lines. However, in contrast to structured grids, unstructured grids need a special consideration to meet these two conditions simultaneously because cell refinement distributions of unstructured grids are random and cell numbers are scattered. Therefore, unstructured grids need a domain decomposition strategy that reorganizes or pre-conditions the spatial data before the execution of parallel computing.

*Authors are with the Georgia Institute of Technology, School of Aerospace Engineering, Atlanta, GA 30332-0150, USA.

In fact, many modern unstructured grid solvers are already parallelized with adequate domain decomposition strategies suitable for their grid topologies and solver systems. Well known NASA developed, tetrahedral based, unstructured grid solvers, FUN3D and USM3D, as well as many other academic unstructured grid solvers, utilize the MeTiS library for mesh partitioning [1]-[4]. MeTiS library is a set of serial programs for partitioning graphs, finite element meshes, and producing fill reducing orderings for sparse matrices. The algorithms implemented in MeTiS [5]-[7] are based on the multilevel recursive-bisection, multi-level k-way, and multi-constraint partitioning schemes developed by George Karypis and his colleagues.

The MeTiS library is known as the most efficient available mesh partitioning tool and is general enough to be applied to various type of unstructured graphs. However, if there are ways to construct the spatial data which do not require an iterative partitioning process, the method would be a lot more simplified. Obviously but ironically, structured grids would be the perfect example for this kind of data structure.

In order to achieve this kind of data structure within unstructured grids, researchers from a wide range of disciplines (i.e. image compression [8], vision sensing, ground mapping for GPS, motion picture [9], CAD [10], CFD [11]) recently have adopted the idea of a space-filling curve (SFC).

A SFC is a one-dimensional curve that fills every node of a multidimensional space while preserving the nodal locality. Therefore, 2D or 3D spatial data can be stored in a single dimensional array and domain decomposition is easily achieved by chopping this array into even pieces. Also the curve itself can be decomposed into higher refinement levels so it is quite suitable for adaptive grids as shown in the Fig. 2. SPC's were first described by the Italian mathematician, Giuseppe Peano [12]. A year later David Hilbert [13] published a description of another such curve, perhaps the simplest to describe among all other SFCs. Several other versions were described by other mathematicians, such as Jordan, Morton, and Moore. While the Hilbert and Morton curves are most suitable for cubic shaped spatial data, the Hilbert curve has been chosen for the present work. It is known that Hilbert outperforms Morton in preserving locality [14]-[16] because the Hilbert curves always connect the closest two nodes but Morton does not in some cases.

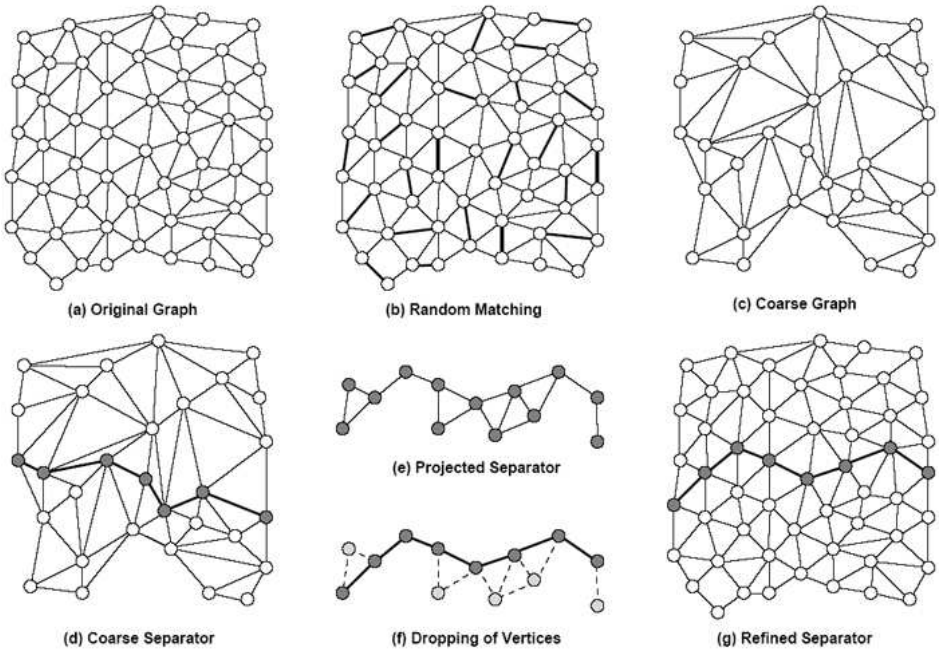


Fig. 1. Example of MeTiS Algorithm: The sequence of one level coarsening, finding a separator for the coarse graph, projecting the separator to the original graph, and refining the separator by dropping vertices [5]

Therefore, grids constructed by the SFC for parallel environment promise short inter-CPU communication time by maintaining cells in contiguous blocks and also it is extremely simple to achieve good load balancing between CPUs because the SFC arranges the three dimensional data into a linear data format. This method is typically applicable to unstructured Cartesian grids, and in this work, it is implemented in NASCART-GT.

NASCART-GT is a solution adaptive unstructured Cartesian grid based flow solver [17], [18]. NASCART-GT's governing equations are Euler, Euler + Integral boundary layer method, Navier-Stokes with two-equation (K-epsilon) turbulence model, and species conservation equations + vibrational energy equation for hypersonic reacting flows [19].

II. DOMAIN DECOMPOSITION WITH 3D HILBERT SFC

The Hilbert SFC has a very unique and organized internal structure. The SFC visiting millions of points in 3D space with non-uniform refinement levels might look somewhat complicated but the 3D Hilbert SFC actually has only twenty four base elements (only four for 2D), regardless of how complex the curve is. Each of these twenty four elements has eight nodal points at every turning location of the curve and any of these nodal points can decompose into higher refinement levels. Each nodal point corresponds to a cell center of the Cartesian grid. An important character of this curve is that each of these elements is always refined into the same combinations of sub-elements in the same order. In Fig. 4, the basic twenty four elements are shown. The numbers labeled for each element is arbitrarily assigned by the author and they can be differently represented in other sources. All elements have the exact same shape and are oriented in four directions for all six faces. The curve within each element has a direction and this direction has to be consistent because the cell numbers are assigned in that order. Table 1 shows its refinement combination and the number corresponding to each element number in Fig. 4. Using these elements and the refinement order, the 3D SFC can be constructed at multiple levels (Fig. 3).

Once the SFC fills the space, domain decomposition can be performed by simply chopping the curve into even array pieces. The implementing code, NASCART-GT, contains some inactive cells during its array so only the active cells are counted in the array cutting process.

As a 2D analog to the 3D curve, Fig. 5 shows how a 2D domain in decomposed and is evenly distributed to 4 CPUs by cell numbering based on the SFC. Each color represents a CPU index and the numbers inside cells represent the actual cell numbers in the computational array of the unstructured grid. The cell numbers written on the cell intersections represent the parent cells in a tree data structure. The parent cells do not participate in computation but are often used for unrefining the grids or for finding neighbor's information. 3D domain decomposition follows the exact same procedure as the 2D example, and the 2D example is shown simply because it is easy to be understood and illustrated.

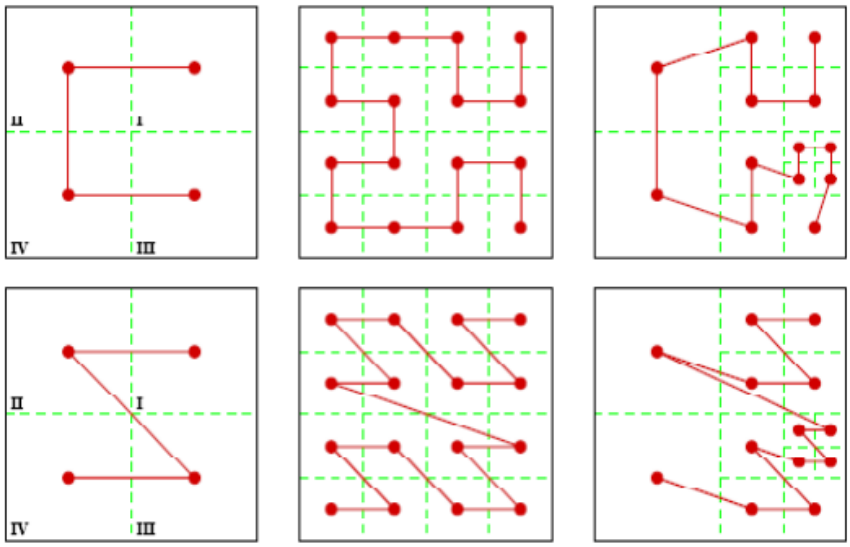


Fig. 2. 2D Hilbert(Top) and Morton(Bottom) Space Filling Curve [15]

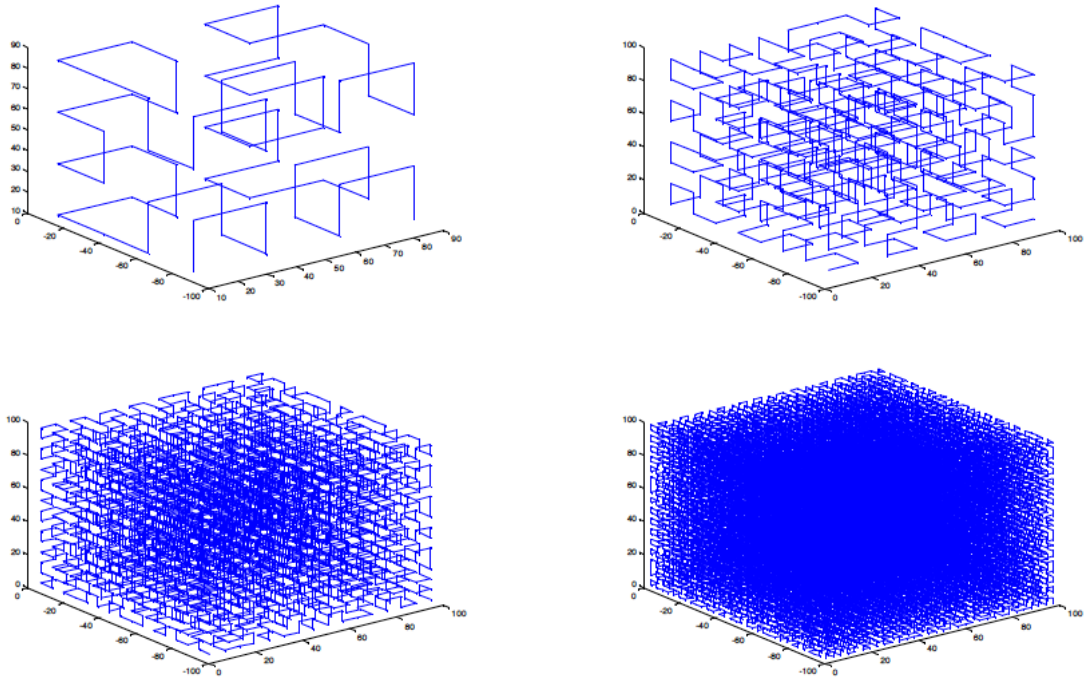


Fig. 3. 3D Space-Filling-Curve in Level 2 to 5

For the current work, a 3D Hilbert SFC is implemented to the initial grid generation routine, and the grid refinement and unrefinement routines of NASCART-GT. It requires different methods for finding the neighbor pointer and for shifting cell numbers after the refinement/unrefinement process than the previous algorithm. Actually, three neighbor pointers out of six pointers can be trivially found by knowing the element number within which the point is located in.

Once the domain is evenly decomposed to individual the CPUs, state vectors of CPU boundary cells should be communicated among the CPUs that share the CPU boundary surface. This communication is performed by using usual MPI routines, and detailed algorithms for the communication will not be discussed in this paper.

III. DESTINATION CELL FLAGGING TECHNIQUE

Due to the inherent nature of unstructured grids, where the demarcation for splitting up the domain by the SFC for different CPUs need not involve smooth boundaries, it takes a searching procedure to identify these cells, based on their destination CPUs that they are to communicate with. This process can be somewhat intricate in terms of efficiently managing the computer memory when a communicating cell has multiple destinations such 'A', 'B', 'C', and 'D' cell in Fig. 6, which frequently happens in 3D.

A simple way of flagging these cells would be by creating a memory slot of the size of the number of communicating cells times the number of communicating CPUs. However, it would waste a lot of available memory. Therefore, the present work suggests a simple technique to avoid this memory wasting problem as its idea is illustrated in Fig. 6. In the Fig., each color of the domains represents a CPU number, and the number inside of the cells represents the flagging integer to be stored in the memory. If a cell has multiple destinations, it uses three digits for each destination of the FORTRAN long integer data type, which has sixteen digits, so that it can store up to five destinations. As an example, if the state vector information of cell number 1937483 has to be sent to CPU2, CPU13, CPU14, CPU17, and CPU 31, then the representation of this flagging integer would be "2,013,014,017,031". This technique can be flexibly expanded or contracted based on the total number of CPUs available and the size of the problem.

IV. PARALELLIZATION RESULTS

Using the presented domain decomposition techniques, the partitioned domains are distributed to multiple CPUs. Fig. 7 show how domains are distributed to 8 CPUs while performing the solution adaption. Distinct colors in the figure represent the CPU number. The example shown in the figures are M_{∞}=1.2 flow over a 3D sphere with 3 degree angle of attack.

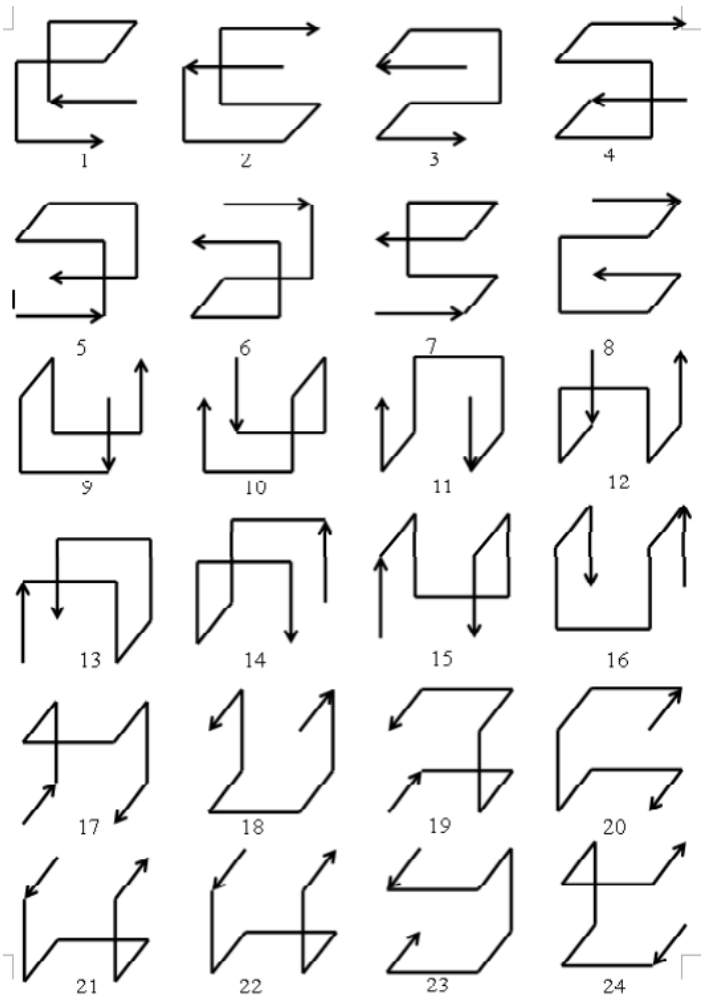


Fig. 4. 24 Base Elements of 3D Hilbert Curve

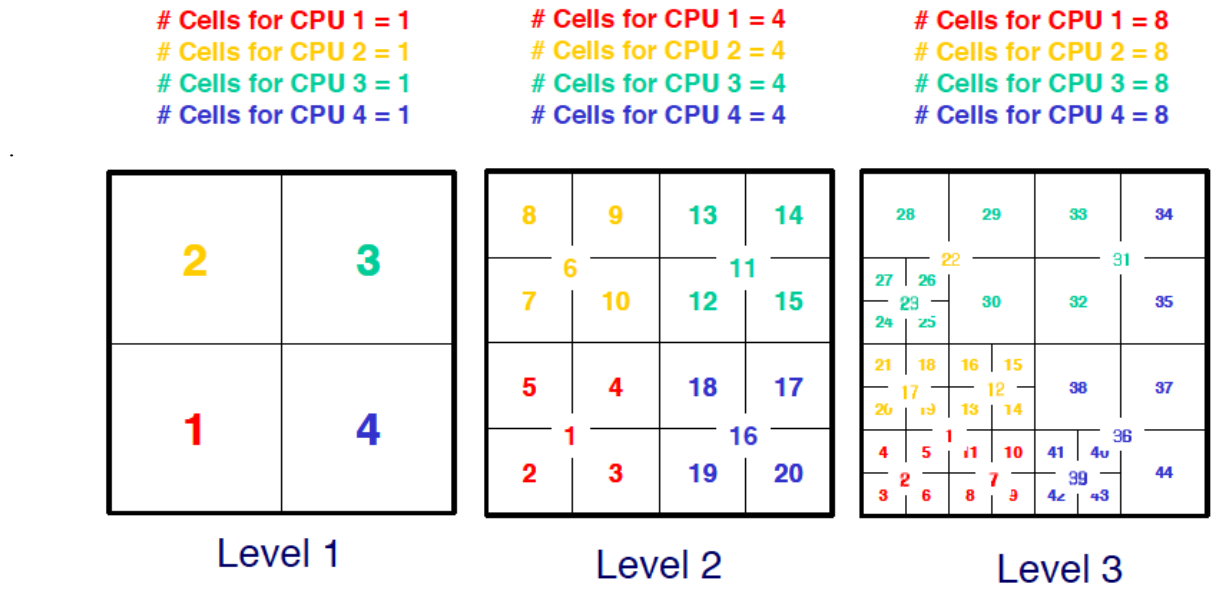
As it is seen in the Fig. 7, CPU boundaries dynamically change as local region is finely refined due to solution adaption.

Fig. 8 shows the parallelization efficiency results based on the presented domain de- composition method for the case of Fig. 7. The cluster used for the sample parallel speed up results has 16 AMD opteron248 processors with 1MB of L2 cache each. The speed up results without solution adaption show nearly 90% parallel speed up with 16 CPUS. The major sources of the 10% sublinear performance are the increase in communication data and also the fact that surface boundary cells require a little extra computational time than regular flow cells in Cartesian grids. The speed up results with solution adaption show only 57% parallel efficiency. This is expected because the current solution adaption and load balancing processes are not parallelized at this stage of the solver development.

Table 1. Refinement Combination and Order

Parent Element	Child1	Child2	Child3	Child4	Child5	Child6	Child7	Child8
1	16	24	24	10	10	3	3	15
2	11	20	20	13	13	4	4	12
3	18	2	2	23	23	14	14	17
4	21	1	1	19	19	9	9	22
5	15	19	19	9	9	8	8	16
6	12	23	23	14	14	7	7	11
7	17	5	5	24	24	10	10	18
8	22	6	6	20	20	13	13	21
9	3	11	11	5	5	22	22	4
10	8	12	12	1	1	18	18	7
11	20	2	2	21	21	10	10	19
12	23	6	6	17	17	9	9	24
13	7	15	15	2	2	21	21	8
14	4	16	16	6	6	17	17	3
15	19	5	5	22	22	14	14	20
16	24	1	1	18	18	13	13	23
17	13	19	19	12	12	3	3	14
18	9	20	20	16	16	7	7	10
19	5	15	15	4	4	18	18	6
20	2	11	11	8	8	17	17	1
21	14	24	24	11	11	8	8	13
22	10	23	23	15	15	4	4	9
23	6	12	12	3	3	21	21	5
24	1	16	16	7	7	22	22	2

Fig. 5. Domain Decomposition Process in 2D



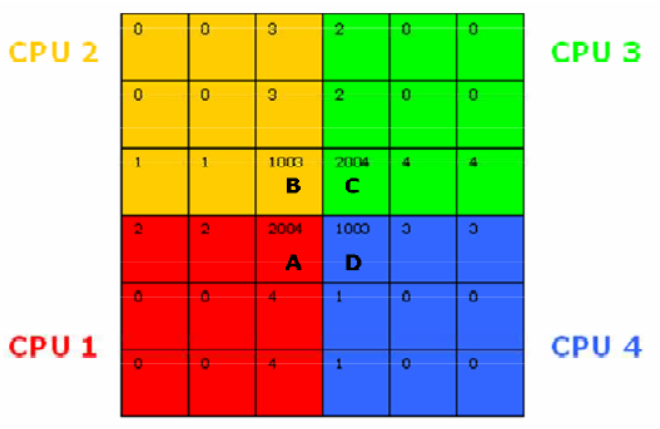
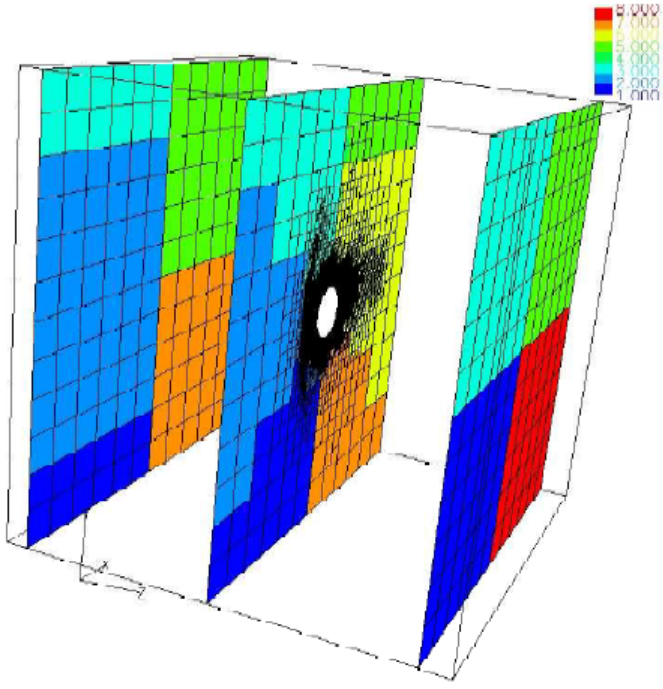


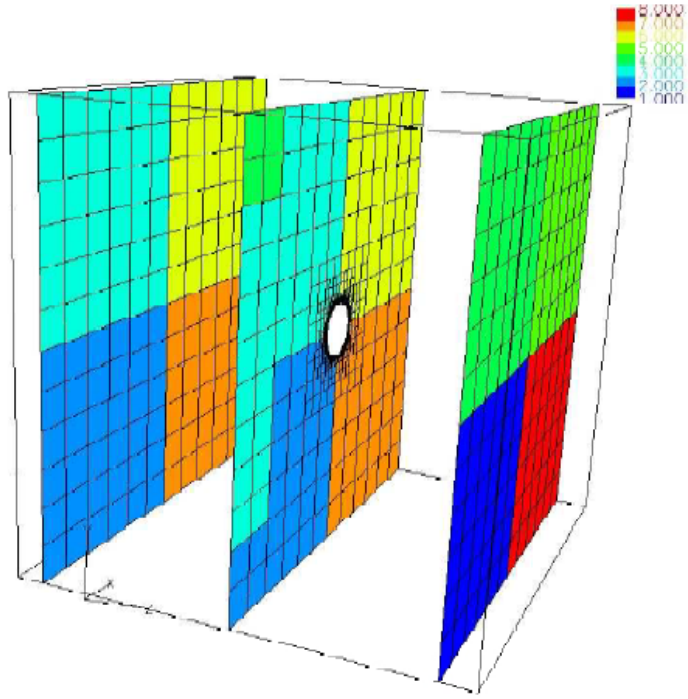
Fig. 6. Illustration of Communicating Cell Flagging Technique

In order to verify the validity of parallel schemes, the result from parallel run is compared with the serial run at a free stream Mach number of 0.85 over a sphere. Fig. 9 shows that the pressure coefficients obtained from the both cases are almost right on top of each other.



(b) After Solution Adation

Fig. 7. NASCART-GT 3D Decomposition with 8 CPUs at Solution Adapted Grids over 3D Sphere



(a) Initial Grids

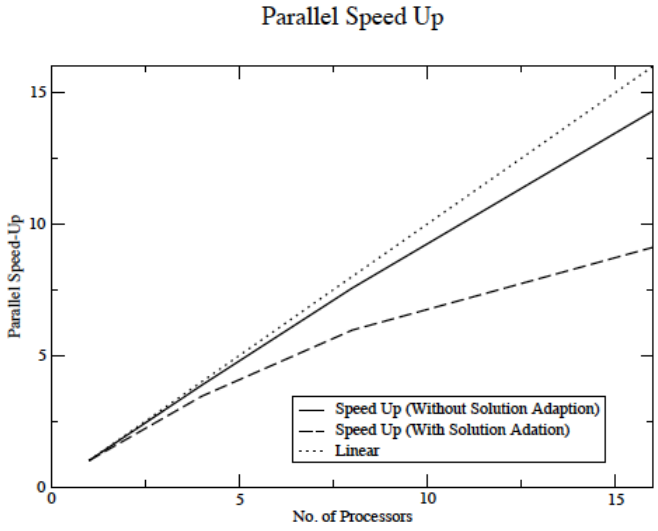


Fig. 8. Current Parallel Speed Up vs. No. of CPUs

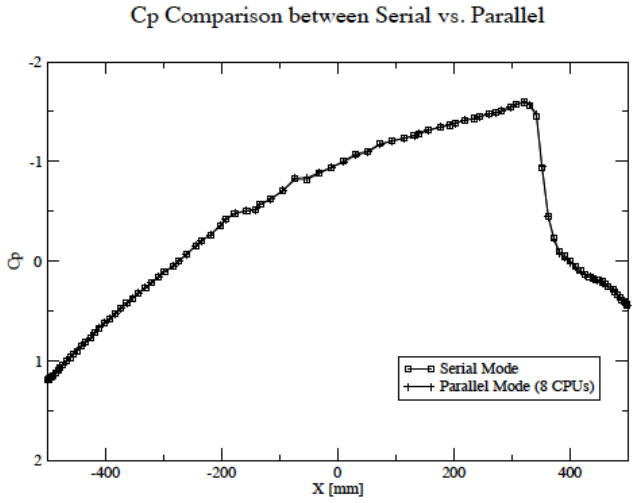


Fig. 9. Cp Distributions on sphere surface at M=0.85

V. ONERA M6 VALIDATION

An inviscid simulation of ONERA M6 wing case was performed as a test case for the current parallelization work. The case was run at a freestream Mach number of 0.84 and an angle-of-attack of $\alpha = 3.06^{o}$. Fig. 10 shows the solution adapted grids clustered around the transonic shock and the pressure contours. A total of 14 CPUs were used and the domain decomposition is shown in Fig. 11. Each color on the figure represents the CPU number from 1 to 14. In this figure, pressure contour lines are superimposed on the domain color map. The pressure contour results show effective and seamless communication across the CPU interfaces.

Since this case was run in the inviscid mode, the location of transonic lambda shock is slightly different from the experimental data[16] as shown in Fig. 12. Despite the use of solution adaption to well resolve the shock waves, perfect load balancing is achieved through use of the SFC.

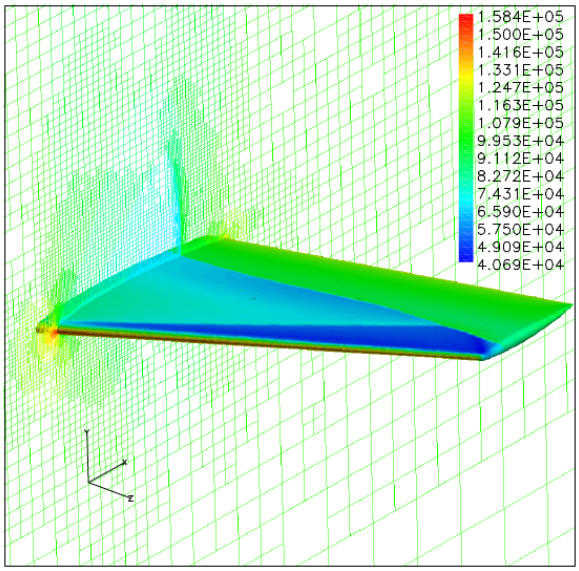


Fig. 10. Unstructured Cartesian grid and pressure contour over ONERA M6 wing.

VI. CONCLUSIONS

This paper presented parallelization of the solution adaptive unstructured Cartesian grids flow solver using 3D Hilbert space filling curve (SFC). As expected, the domain decomposition strategy using the 3D Hilbert Space Filling Curve (SFC) provided evenly distributed and block contiguous grid partitioning. The master CPU redistributed the workload to each CPU after solution adaption so that load balancing is still maintained. Effective and seamless communication across CPU interfaces is demonstrated even when shock waves present. However, the grid generation part of NASCART-GT including the grid refinement routine was not parallelized in the present work. Therefore, it showed sub-linear speed up performance with solution adaption turned on, but the one without solution adaption showed almost a linear speed up. The recommendation for future work would be to parallelize the grid generation process to achieve linear speed up even with the solution adaption turned on.

REFERENCES

[1] Bhat, M. K. and Parikh, P., "Parallel Implementation of an Unstructured Grid-Based Navier-Stokes Solver," *AIAA Paper 99–16526*, Jan 1999.

[2] Nompelis, I., Drayna, T. W., and Candler, G. V., "A Parallel Unstructured Implicit Solver for Hypersonic Reacting Flow Simulation," *AIAA Paper 2005–4867,* Jun 2005.

[3] Parikh, P., "Application of a Scalable, Parallel, Unstructured Grid-Based Navier-Stokes Solver," *AIAA Paper 2001–2584*, Jun 2001.

[4] Park, Y. M. and Kown, O. J., "A Parallel Unstructured Dynamic Mesh Adaptation Algorithm for 3-D Unsteady Flows," *International Journal for Numerical Methods in Fluids*, Vol. 48, No. 6, June 2005, pp. 671–690.

[5] Karypis, G. and Kumar, V., "Analysis of multilevel graph partitioning," Technical Report TR 95-037, Department of Computer Science, University of Minnesota, 1995, Also available on WWW at URL http://www.cs.umn.edu/users/kumar/papers/mlevel analysis.ps.

[6] Karypis, G. and Kumar, V., "A fast and high quality multilevel scheme for partitioning irregular graphs," Technical Report TR 95-035, Department of Computer Science, University of Minnesota, 1995, Also available on WWW at URL http://www.cs.umn.edu/users/kumar/papers/mlevel serial.ps.

[7] Karypis, G. and Kumar, V., "Unstructured graph partitioning and sparse matrix ordering system," Tech. rep., Department of Computer Science, University of Minnesota, 1995, Available on WWW at URL http://www.cs.umn.edu/users/kumar/metis/metis.html.

[8] Pajarola, R. and Widmayer, P., "An Image Compression Method for Spatial Search," *IEEE TRANS- ACTIONS ON IMAGE PROCESSING*, Vol. 9, No. 3, 2003.

[9] Mitchell, L. K., "'Techniques for Artistically Rendering Space-Filling Curves," Associate Professor of University of Advancing Technology.

[10] Hill, D. C., "Cartesian Mesh Generation and Highly-Compressed Storage Using Hilbert Codes," *AIAA Paper 2004–240*, Jan 2004.

[11] Aftosmis, M. J., Berger, M. J., and Murman, S. M., "Applications of Space-Filling-Curves to Cartesian Methods for CFD," *AIAA Paper 2004–1232*, Jan 2004.

Peano, G., "Sur une Courbe qui Remplit Toute une Aire Plane,"

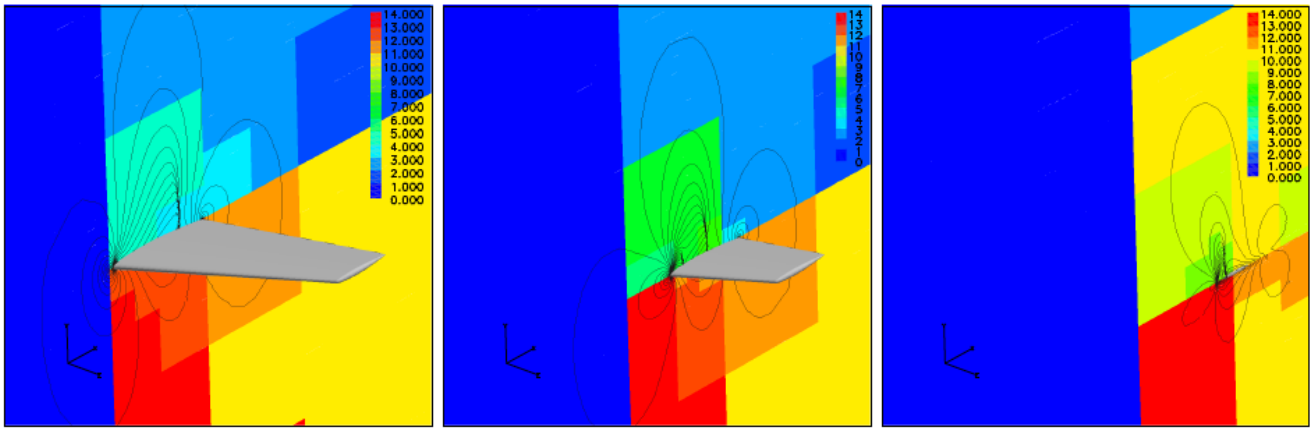


Fig. 11. Domain Decomposition over OneraM6 wing. Left: Hub, Center: Mid-Span, Right: Tip

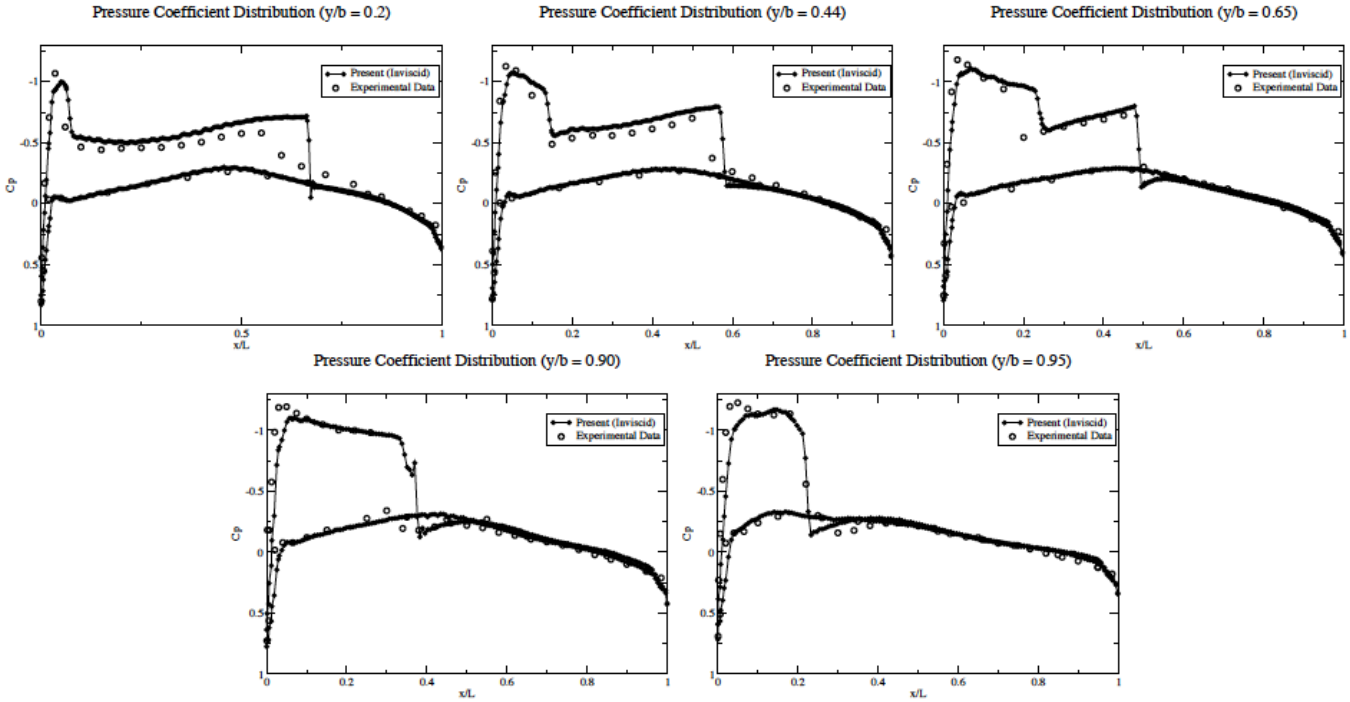


Fig. 12. Pressure coefficient on surface of ONERA M6 wing.

[12] *Math. Ann.*, Vol. 36, 1890, pp. 157–160.
[13] Hilbert, G., "U ber die stetige Abbildung einer Linie auf Flachenstu ck," *Math. Ann.*, Vol. 38, 1891, pp. 459–460.
[14] Jagadish, H. V., "Linear Clustering of Objects with Multiple Attributes," Proc. ACM SIGMOD Conf., May 1990, pp. 332–342.
[15] Moon, B, Jagadish, H. V., Faloutsos, C., and Saltz, J. H., "Analysis of the Clustering Properties of the Hilbert Space-Filling Curve," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 1, Jan 2001, pp. 459–460.
[16] Schmitt, V. and Charpin, F., "Pressure Distributions on the ONERA-M6-Wing at Transonic Mach Numbers," AGARD Advisory Report 138, North Atlantic Treaty Organization, May 1979.
[17] Ruffin, S.M., Lee, J.D., "Adaptation of a k-Epsilon Model to a Cartesian Grid Based Methodology," *International Journal of Mathematical Models and Methods in Applied Sciences*, Vol. 3, No. 1, 2009.
[18] Lee J.D., "Development of an Efficient Viscous Approach in a Cartesian Grid Framework And Application To Rotor-Fuselage Interaction," PhD Dissertation, Georgia Tech, 2006.
[19] Lee, Jinwook, Orsini, A., and Ruffin, S.M., "Unstructured Cartesian-Grid Methodology for Non-equilibrium Hypersonic Flows," *Journal of Thermophysics and Heat Transfer*, Vol. 24, No. 1, Jan-Mar 2010.