# Parallel Performance Assessment of Moving Body Overset Grid Application on PC cluster

**Eugene Kim[a], Jang Hyuk Kwon[a] and Soo Hyung Park[b]**

[a]*Department of Aerospace Engineering, Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701 Republic of Korea*

[b]*NITRI & Department of Aerospace Engineering, Konkuk University, 1 Hwaynag-dong, Gwangjin-Gu, Seoul 143-701, Republic of Korea*

**Key Words**: Parallel computing, Overset Grid, PC cluster

### Introduction

The multi-block overset grid method is a powerful technique for high-fidelity computational fluid dynamics(CFD) simulations about complex aerospace configurations[1]. Furthermore, it allows the calculation of moving-body with relative motions, such as applications in the analysis of store separation and many other areas. However, a communication between overlapping grids must be re-established in every time step as the sub-grid is moved. This process of establishing communication between overlapping grids involved in hole cutting and domain connection has been referred to as “grid assembly”[2].

The parallel implementations and performance assessments of the well known chimera codes, Beggar[2] and DCF3D[3] have been accomplished on vendor’s machines, such as SGI, IBM, and Cray. In this research, a structured multi-block grid assembly is parallelized by using a static load balancing tied to flow solver based on a grid size of decomposed domain. The parallel algorithm is modified for a distributed memory system typically PC cluster, which has some advantages typically low cost & high performance and therefore is widely used in nowadays. To parallelize grid assembly, a message passing programming model based on the MPI library is implemented using the Single Program Multiple Data(SPMD) paradigm. The parallelized flow solver can be executed in each processor with the static load balancing by pre-processor and the communication load of it is fixed at initial statge. However, the communication load of grid assembly can not be determinded initially and more complex. A coarse-grained communication is optimized with the minimized memory allocation and communication load because the parallel grid assembly can access the decomposed geometry data within other processors by only message passing in the distributed memory system such as the PC cluster.

This parallel grid assembly and flow solver(KFLOW[4]) are tested on the store separation problem from the Eglin wing and pylon configuration[5] in inviscid flow field, and the parallel performance assessment is presented.

## Parallel Implementation of Grid Assembly

The overset grid method is comprised of two major steps, hole cutting and donor cell identification. A zones of interference scheme and a hole-map algorithm are used for the chimera hole cutting, the second step uses a stencil-walk and a gradient searching methods[6]. We will describe the parallel implementation of each step in the following subsection.

### Hole Cutting

Solid walls are used for the hole-cutting surfaces. The cartesian mesh is constructed in cutting boundaries and cutting surfaces are approximately modeled with a collection of cubes that touch the cutting surfaces. These algorithms are relatively simple and effective to discover whether a given point lies inside of cutting boundary or not. However, the solid body surface consists of the decomposed patches that is called facets in each processor must be closed surface to be used for the hole cutting boundary. The gathering of surface patches in each body is executed by group communication after regenerating new communicator among the processors that compose the same body.

Some processors have multi-blocks for the load balancing and block numbering in each processor has independent order like figure 1. The ‘ipblk’ is global-block number consider whole multi-blocks. The ‘idblk’ is body number that the block constitutes. Figure 2 is example of defining newrank by MPI_COMM_SPLIT library[7]. The ‘icolor’ is idblk index of figure 1 and facets are gathered in same newcomm by MPI_ALLGATHERV library[7] such as figure 3.

For the hole cutting, hole map that is cartesian meshes constructed in each body must be transferred to processors with other body. however, the direct communication of it requires heavy transfer time on the general PC cluster with poorer network system such as Ethernet than the vendor’s machines because the hole map size is dependent on initial grid size. The performance of processor in each machines is better than a network transfer capacity. Therefore, in this approach, only the facets and the computed maximum and minimum coordinates of it are transferred to other CPUs of body and the hole map is computed respectively. The size of facets are relatively smaller than total grid size, especially when the background field grid is used for moving body application. After construction of hole map and hole cutting by this, the interpolation points are given donor candidate block(CPU rank) by using facet information. The interpolation points where are beside the hole points cut by facet are inside block with this facet generally. The informations of this points are sent to candidates. The sent data is the take rank, coordinates of points and  integer tag to connect send and take ranks.

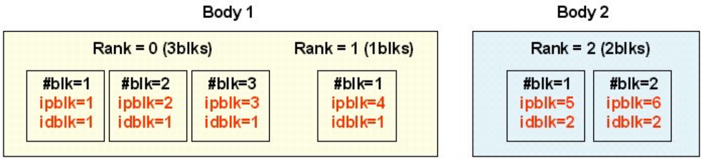


Figure 1. Block data structure of grid assembly (2 bodies and 3CPUs)

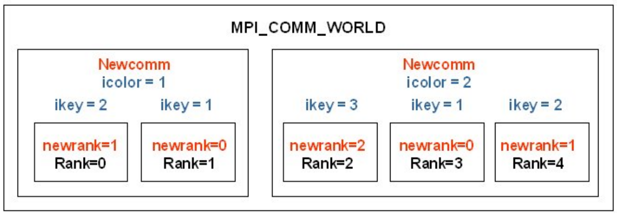


Figure 2. Example of MPI_COMM_SPLIT (2 bodies and 5 CPUs)

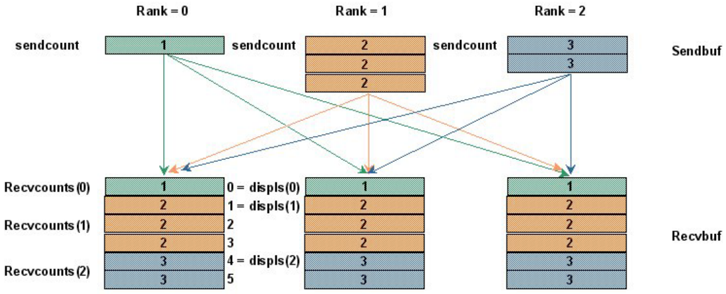


Figure 3. MPI_ALLGATHERV

## Donor Cell Identification

Every processors start searching dornor points by using the informations received from take rank. The two-step search method is used in this proccedure stated before. The stencil walk finds the nearest point to a given interpolation point by comparing the distance between the interpolation point and the donor candidate cells. However, the gradient search checks whether the interpolation point actually lies inside the cube that is made with the eight donor cells. To search through parallel boundary, we can use ghost cell coordinates like figure 4 and the received and stored facet data of this surface can be used to find nearest candidates when the searching routine meet the solid wall boundary. The informations of interpolation points that go over the next block or fail to find donor are sent to next candidate block after all processors complete the searching. These procedure is repeated until all interpolation points find donors

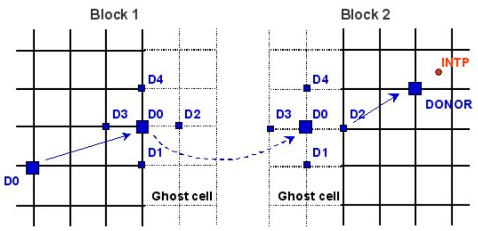


Figure 4. Jumping through parallel boundary

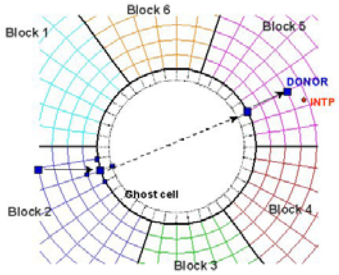
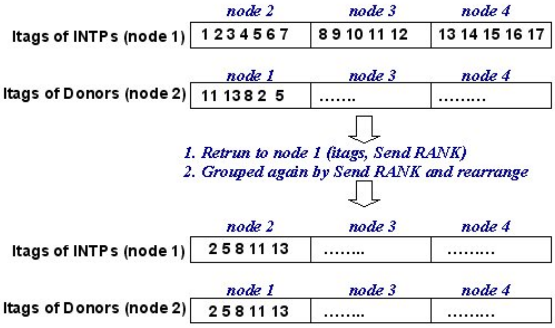


Figure 5. Searching near the solid wall

**Connection between Initerpolation and Donor points**

During the parallel donor searching, the minimized informations of interoplation points are communicated among the take and send rank described before and the take rank need not acknoledge the donor informations. Therefore separated rank data is connected by sorting with tag index. Figure 6 shows this procedure between two processors, first step is grouping and second step is sorting by tag index in same groups. A Heap-sort algorithm is used that is known as one of the fastest sorting algorighm with *O(nlogn)* operations and that it has good performance in the worst case.



| | node 2 | node 3 | node 4 |
|---|---|---|---|
| Itags of INTPs (node 1) | 1 2 3 4 5 6 7 | 8 9 10 11 12 | 13 14 15 16 17 |

| | node 1 | node 3 | node 4 |
|---|---|---|---|
| Itags of Donors (node 2) | 11 13 8 2 5 | ……. | ……… |

*1. Retrun to node 1 (itags, Send RANK)*
*2. Grouped again by Send RANK and rearrange*

| | node 2 | node 3 | node 4 |
|---|---|---|---|
| Itags of INTPs (node 1) | 2 5 8 11 13 | …….. | ……… |

| | node 1 | node 3 | node 4 |
|---|---|---|---|
| Itags of Donors (node 2) | 2 5 8 11 13 | …….. | ……… |

Figure 6. Coupling of interpolation and donor points

Figure 7 is the flow chart of parallel grid assembly. The coarse-grained communication between each processor are inserted in construction of hole-map, hole cutting and donor searching routine.

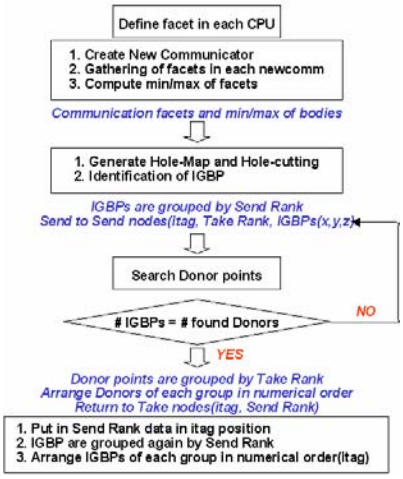

Define facet in each CPU
1. Create New Communicator
2. Gathering of facets in each newcomm
3. Compute min/max of facets
Communication facets and min/max of bodies
1. Generate Hole-Map and Hole-cutting
2. Identification of IGBP
IGBPs are grouped by Send Rank
Send to Send nodes(itag, Take Rank, IGBPs(x,y,z))
Search Donor points
# IGBPs = # found Donors
NO
YES
Donor points are grouped by Take Rank
Arrange Donors of each group in numerical order
Return to Take nodes(itag, Send Rank)
1. Put in Send Rank data in itag position
2. IGBP are grouped again by Send Rank
3. Arrange IGBPs of each group in numerical order(itag)

Figure 7. Process of parallel grid assembly

## Test results

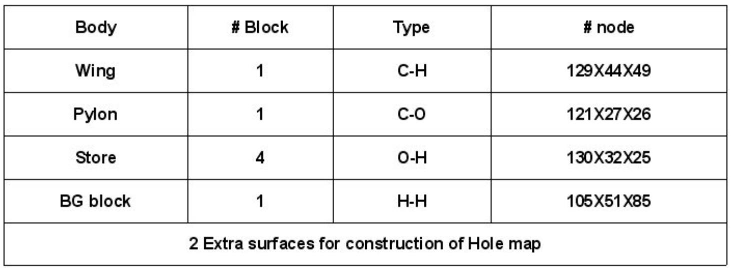
The flow solver is cell centered finite volume code. The numerical fluxes at cell interface are constructed Roe’s flux difference splitting method and MUSCL(Monotone Upwind Scheme for Conservation Law) scheme. For the time integration, diagonalized ADI method is used with 2$^{nd}$-order dual time stepping algorithm.

The test problem is the released store from Eglin wing/ pylon configuration in transonic free stream condition, the Mach number is 0.95. A C-H type grid generated over the wing and an O-O type grid over the pylon are overlapped and used as a major grid system, together with an H-H grid to improve the interpolation quality. The total number of grids is approximately one million. Table 1 is list of grid system.

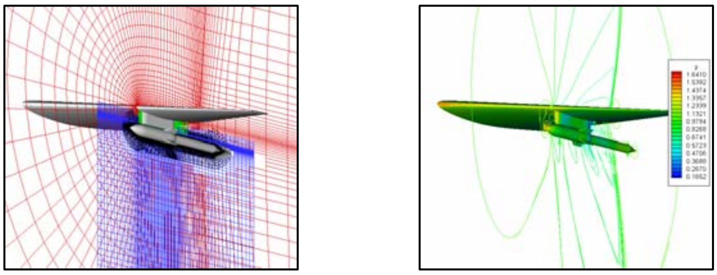
Linux PC cluster with P4-2.6GHz CPU in each node and linked by 100Mbps Ethernet network is used. Serial and parallel computing with 15 and 30 nodes are performed.

Figure 8 shows the constructed overlap grids and the pressure contour at initial position and the number of IGBPs(Inter-Grid Boundary Points) at this condition are 126,534.

Table 1. Grid system for Eglin wing/pylon/store case



| Body | # Block | Type | # node |
|---|---|---|---|
| Wing | 1 | C-H | 129X44X49 |
| Pylon | 1 | C-O | 121X27X26 |
| Store | 4 | O-H | 130X32X25 |
| BG block | 1 | H-H | 105X51X85 |
| 2 Extra surfaces for construction of Hole map | | | |



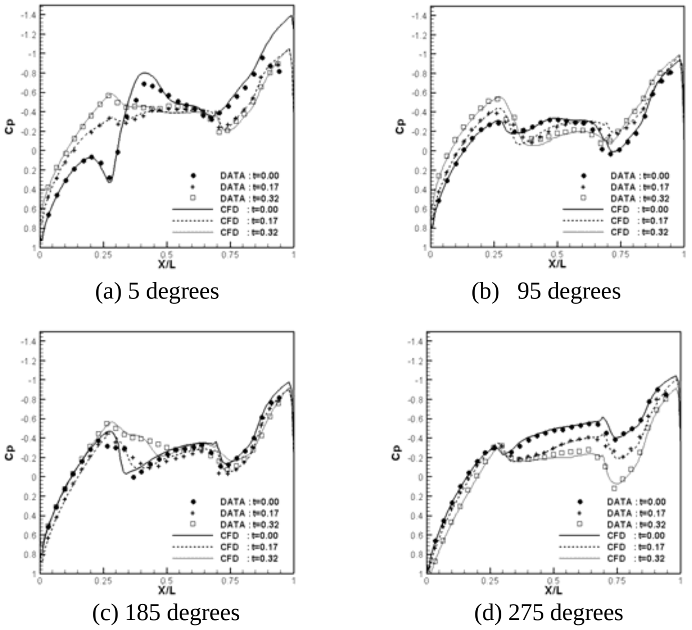
(a) Overlapped grids (b) Pressure distributions

Figure 8. Grids and flow solution in store symmetric section at the initial condition

To simulate dynamics of store, the unsteady time step is set to 0.00195 second and we simulate to 0.4 second.

Figure 9 is the Cp distributions on the store during the separations and the experimental data[5]. The flow solver include the developed grid assembly agree well with the trend and the magnitude of the CTS data. Figure 10 is the simulated CG location and attitude of store. The results of serial and parallel computing are almost same in this figure.

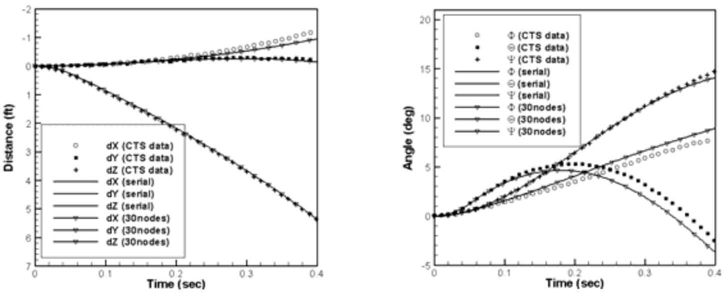
Table 2 and 3 show parallel performance of each routine for the grid assembly, the domain connection(total interpolation work time in every time step) and the flow solving. Total speed up ratio in case of using 30 processors is 17.58. This is about 59% of a ideal parallel performance. Most time consuming parts of this procedures is domain connection routine that is just data communication through network.



(a) 5 degrees (b) 95 degrees

(c) 185 degrees (d) 275 degrees

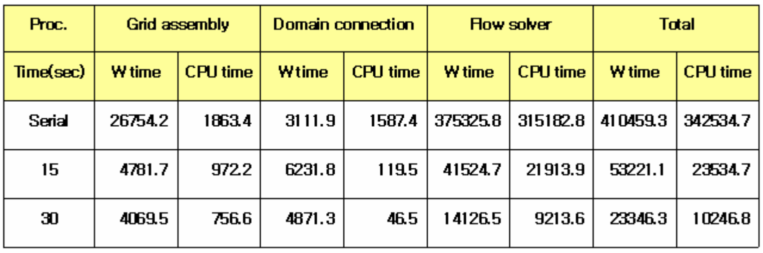
Figure 9. Sectional Cp distributions on the store surface (various Φ angles)



(a) Displacements (b) Attitudes

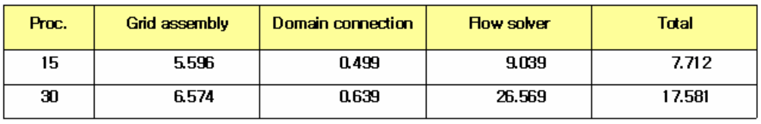
Figure 10. Trajectory of released store (CTS vs CFD : serial and parallel)

Table 2. Parallel performance (wall clock time and CPU time)



| Proc. | Grid assembly | | Domain connection | | Flow solver | | Total | |
|---|---|---|---|---|---|---|---|---|
| Time(sec) | W time | CPU time | W time | CPU time | W time | CPU time | W time | CPU time |
| Serial | 26754.2 | 1863.4 | 3111.9 | 1587.4 | 375325.8 | 315182.8 | 410459.3 | 342534.7 |
| 15 | 4781.7 | 972.2 | 6231.8 | 119.5 | 41524.7 | 21913.9 | 53221.1 | 23534.7 |
| 30 | 4069.5 | 756.6 | 4871.3 | 46.5 | 14126.5 | 9213.6 | 23346.3 | 10246.8 |

Table 3. Speed up of each module (fraction of wall clock time)



| Proc. | Grid assembly | Domain connection | Flow solver | Total |
|---|---|---|---|---|
| 15 | 5.596 | 0.499 | 9.039 | 7.712 |
| 30 | 6.574 | 0.639 | 26.569 | 17.581 |

## Conclusion

The multi-block parallel grid assembly has been developed. The coarse-grained communication is optimized with the minimized memory allocation and communication load without modifications of main algorithms to cut hole and search donor. The flow solver include this parallel grid assembly has been tested on the dynamic problem on the general Linux PC cluster system. The parallel performance of total work is somewhat poor because the network connection is not enough to reduce comunication time in domain connection routine.

A higher performance network system, for example the Myrinet, can reduce communication time in these routines. However, a larger number of processors is better than the expensive network system so far because the limitation of static load balancing tied to flow solver in grid assembly still remains and flow solver routine is over the 60% of total work.

## REFERENCES

[1] J.L. Steger, F.C. Dougherty and J.A. Benek, “A Chimera Grid Scheme,” In Advances in Grid Generation, pp59-69, *ASME FED*-Vol.5, NY, June 1985.

[2] N.C. Prewitt, D.M. Belk and W. Shyy, "Improvements in Parallel Chimera Grid Assembly," *AIAA Journal*, Vol.40, No.3, March 2002, pp.497-500.
[3] A.M. Wissink and R.L. Meakin, "On Parallel Implementations of Dynamic Overset Grid Methods." *SC97: High Performance Networking and Computing*, November 1997.
[4] S.H. Park, Y.S. Kim and J.H. Kwon, "Prediction of Damping Coefficients Using the Unsteady Euler Equations," *Journal of Spacecraft and Rockets*, Vol.40, No.3, 2003, pp.356-362.
[5] L.E. Lijewski and N.E. Suhs, "Time-Accurate Computational Fluid Dynamics Approach to Transonic Store Separation Trajectory Prediction," *Journal of Aircraft*, Vol.31, No.4, 1994, pp. 886-891.
[6] K.W. Cho, J.H. Kwon and S. Lee, "Development of a Fully Systemized Chimera Methodology for Steady/Unsteady Problems," *Journal of Aircraft*, Vol. 36, No.6, 1999, pp. 973-980.
[7] W. Gropp, E, Lusk and A. Skjellum, "Using MPI : Portable Parallel Programming with the Message-Passing Interface," The MIT Press, Cambridge, Massachusetts, London, England.