FLOW-3D/MP

Parallelization Approach

High Performance Computing with Hybrid *FLOW-3D*/MP

The Hybrid version of ***FLOW-3D*** uses OpenMP and Message Passing Interface (MPI) to achieve parallelization. The OpenMP paradigm is based on shared memory architecture and on fine-grained or loop-level parallelism. MPI is based on a distributed memory, domain decomposition paradigm (coarse-grained). The physical domain is decomposed into blocks which are then associated with MPI processes and then assigned to nodes on a cluster. Within the node, the MPI process spawns threads and performs multi-threaded calculations using OpenMP directives. When required, the MPI process communicates with other MPI processes on other nodes over the network (GigE /Infiniband). The Hybrid paradigm is based on a THREAD_MASTER ONLY model [Rabenseifner et al., "Communication and optimization aspects of parallel programming models on Hybrid architectures," Int. J. High Perform. Comput. Appl. 17 (2003) 49-63]. In this model, the MPI task can be multi-threaded, but the MPI functions may be called ***only by the master thread, and only outside of parallel regions***. Figure 1 illustrates this concept on 2 nodes (each node has 4 cores).

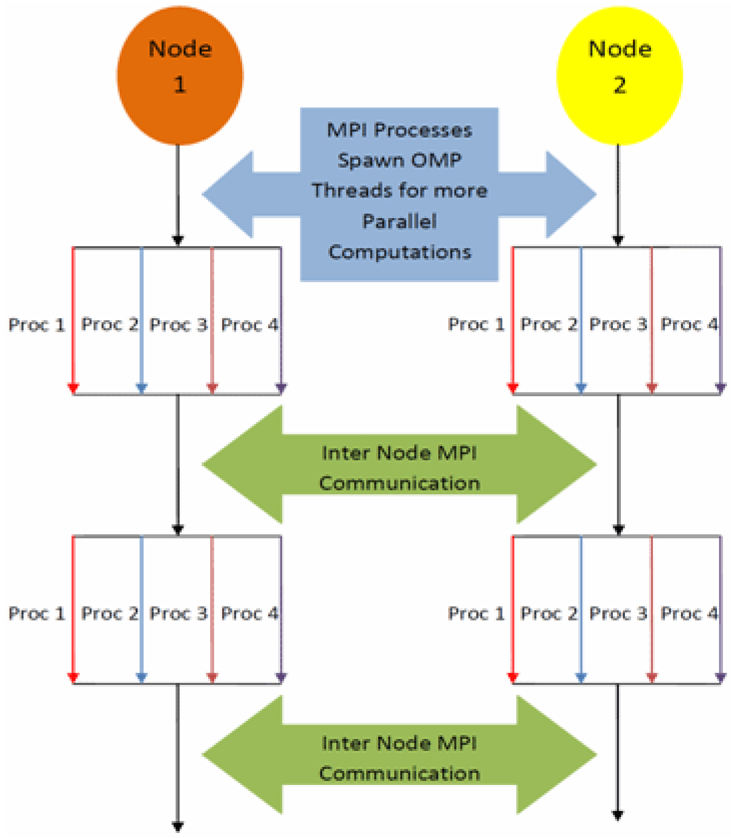

Node 1
Node 2
MPI Processes Spawn OMP Threads for more Parallel Computations
Proc 1
Proc 2
Proc 3
Proc 4
Inter Node MPI Communication

*Figure 1: Hybrid OpenMP/MPI paradigm for **FLOW-3D**.*

Advantages to the Hybrid Approach

The Hybrid approach has three main advantages over the standard MPI parallelization. Described below, they are:

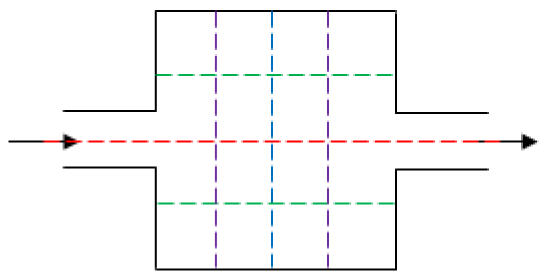
1. MPI Communication and Speed Up
2. Load Balancing
3. Mutual Benefit

1. MPI Communication and Speed Up

A 16-rank simulation using only MPI parallelization on a cluster where each node has 8 cores requires 16 blocks (as a minimum). MPI communication takes place between 16 processes (this may include inter-node and intra-node communication). A Hybrid simulation on 16 cores requires only 2 physical blocks or 2 MPI processes. Each MPI process spawns 8 OpenMP threads. The amount of MPI communication is therefore reduced (in terms of the number of MPI calls). It is always beneficial to send larger messages fewer times than smaller messages many times due to MPI overheads. Since fewer processes are communicating, time spent in synchronization is also reduced. Calls that require communication between all MPI processes will be faster (e.g., collective calls such as MPI_ALLREDUCE).

2. Load Balancing

The Hybrid approach, to an extent, can improve the *dynamic* load balance between parallel threads. This is best explained by the following example. Figure 2 shows a geometry where initially the domain is empty, and at t=0 fluid enters from the left boundary. If one uses the Automatic Decomposition Tool (ADT) to decompose the domain to be able to simulate on 16 ranks, the configuration might be like what is shown in Fig. 2. As one can see, the blocks near the exit section may be empty for more than 50% of the simulation time. Figure 3 shows the same configuration for the Hybrid simulation. The domain is decomposed into 2 blocks (Users can now choose not to decompose in a specific direction) and although the domain is initially empty, the fluid enters both blocks at the same time and then each block does the rest of the work using OpenMP threads. For this particular simulation, Hybrid computing on 16 cores has a *lesser dynamic load imbalance* since OpenMP threads will have cells with fluid in them for the most part of the simulation. In most scenarios the *dynamic* load balancing should be less of an issue than in a pure MPI simulation. The dynamic load balancing issue is further improved by the dynamic thread adjustment.



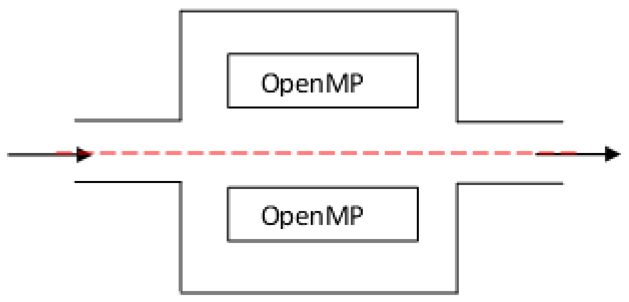


Figure 2: MPI Decomposition into 16 blocks. Figure 3: MPI Decomposition into 2 blocks for Hybrid.

Each MPI rank running Hybrid ***FLOW-3D*** may begin with the same number of threads. The number of threads, however, should ideally be dependent on the computational work for a given rank. ADT decides the number of active cells for each rank. However, depending on the amount of fluid present in each MPI rank, the computational load may vary. Consider the example of a sloshing problem common to aerospace and maritime applications. Figure 4 shows a 2D schematic of the free surface in 3 different situations. In Fig. 4a, both MPI ranks have the same amount of fluid. In Fig 4b, rank 0 has more fluid and will have more computational work than rank 1 and vice versa for Fig 4c. If the simulation is run on a single compute node with 8 cores, with 2 MPI ranks having 4 OpenMP threads each, because of the dynamics of the flow during the simulation, the computational load changes from one rank to the other. The dynamic thread adjustment accounts for this load imbalance by changing the number of threads during the simulation, e.g., in the situation of Fig 4b, rank 0 may have 6 threads and rank 2, 2 threads.

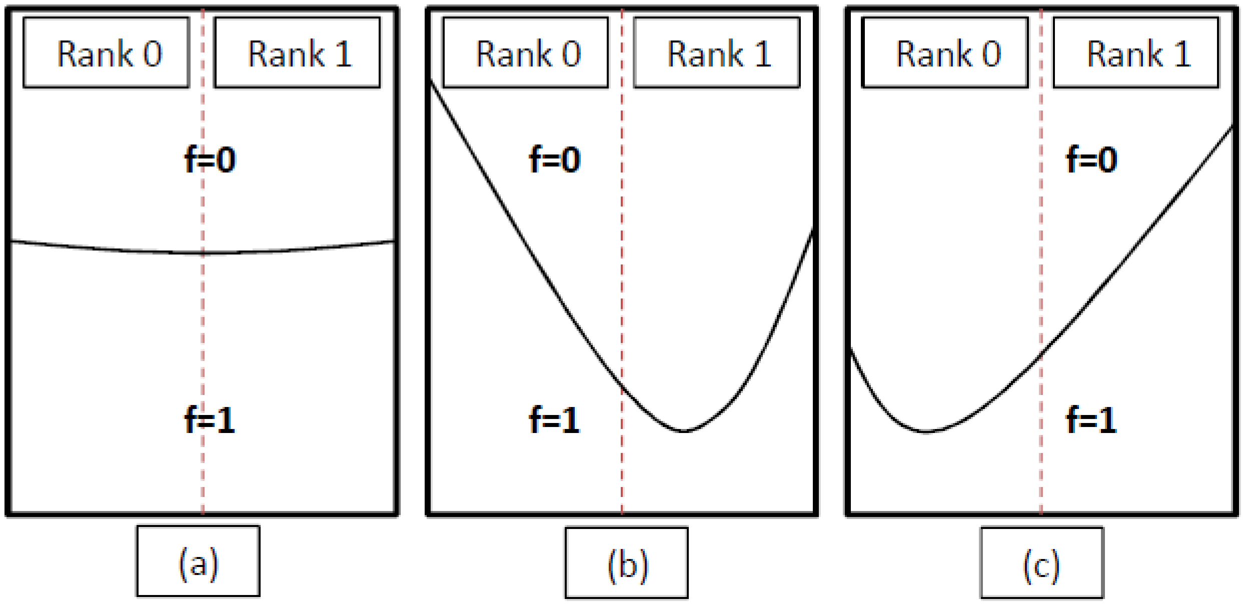


Figure 4: 2D Schematic of fluid sloshing in a tank. The solid line is the free surface. The red line indicates the demarcation for the 2 MPI ranks.

3. Mutual Benefit

Currently, pure MPI does not scale beyond 64 cores because of the *dynamic* load balancing issues as well as communication overheads, while pure OpenMP does not scale because it is limited by serialization and by the number of cores and memory on a single node. These two negative effects tend to cancel each other in the Hybrid approach thereby improving performance.

Conclusion

In conclusion, the hybrid version offers better scalability with lower memory requirements and control for dynamic load balance for ***FLOW-3D* /MP**. For more information about ***FLOW-3D* /MP**, please contact the Flow Science sales team at sales@flow3d.com.