



Code_Saturne Optimizations in Preprocessing

A. Turk^{a,*}, C. Moulinec^b, A.G. Sunderland^b, C. Aykanat^a

^aBilkent University, Comp. Eng. Dept., Ankara, Turkey

^bSTFC Daresbury Laboratory, Warrington WA4 4AD, UK

Abstract

Code_Saturne is an open-source, multi-purpose Computational Fluid Dynamics (CFD) software, which has been developed by Electricite de France Recherche et Developement (EDF-R&D). *Code_Saturne* has been selected as an application of interest for CFD community in The Partnership for Advanced Computing in Europe First Implementation Phase Project (PRACE-IIP) and various efforts towards improving the scalability of *Code_Saturne* have been conducted. In this whitepaper, the efforts towards understanding and improving the preprocessing subsystem of *Code_Saturne* are described, and to this end, the performance of different mesh partitioning software packages that can be used are investigated.

1. Introduction

A current trend in HPC architectures is the expansion of memory resources not match that of cpu resources. That is, memory per core is often relatively low on these systems. Meanwhile, scientific goals are continuously becoming more and more ambitious, trying to solve more complex physical systems. For CFD applications, one major consequence of these trends is that the parallel performance of partitioning software is becoming increasingly important.

Code_Saturne is an open-source, multi-purpose Computational Fluid Dynamics (CFD) software, which has been developed by Electricite de France Recherche et Developement (EDFR&D). *Code_Saturne* is based on a co-located Finite Volume Method (FVM) that accepts three-dimensional meshes built with any type of cell (tetrahedral, hexahedral, prismatic, pyramidal, polyhedral) and with any type of grid structure (unstructured, block structured, hybrid). It is able to handle either incompressible or compressible flows with or without heat transfer and turbulence.

Previous studies [1] show that sequential graph partitioner MeTiS [2] provide the best results in terms of reducing the average time spent in a timestep of *Code_Saturne*. However, MeTiS has problems in partitioning into 64K parts or more. Furthermore, for sufficiently large problems, the memory requirements of MeTiS far surpasses the memory available even in fat nodes.

In this document, efforts towards understanding and improving the preprocessing subsystem of *Code_Saturne* are described, and to this end, the performance of different mesh partitioning software packages that can be used are investigated. Analysis reported in this document reveal that, for medium sized meshes, if the time spent for partitioning is not important, the usage of sequential SCOTCH [4] can be considered, since it provides reasonably good partitions with relatively low memory requirements and can easily scale up to 128K parts and beyond on a single fat node.

As the mesh sizes that are to be partitioned reaches to billion-cell meshes, even the mesh generation has to be performed in parallel via parallel mesh generators. Thus, both in order to adjust to memory constraints and to avoid migration of data, the partitioning has to be done in parallel as well. To fit such large meshes in the memory of cluster nodes, the mesh has to be partitioned into very large number of cores. Unfortunately the partitioning performance of parallel graph partitioning packages such as Par-MeTiS [3] and PT-SCOTCH [4] decline with increasing number of cores used in the partitioning process. To address this problem we propose to utilize a two-level hierarchical partitioning scheme that enables the usage of different partitioning schemes in each level. We investigate the different partitioning packages that are supported in *Code_Saturne* and also propose a hierarchical partitioning scheme utilizing Zoltan [5]. We analyze aspects such as memory requirements, load-balancing performance of partitioners and the effect of partitioning quality over the runtime of *Code_Saturne*.

*Corresponding author.

tel. +903122901776 fax. +903122664047 e-mail. atat@cs.bilkent.edu.tr

2. Proposed Scheme

A two-level hierarchical partitioning scheme that enables the usage of different partitioning schemes in each level is investigated as an alternative to the partitioning tools supported by *Code_Saturne*. The proposed hierarchical partitioning scheme utilizes the Zoltan partitioning framework for combining different partitioning schemes. In this scheme, the graph is first partitioned into a fraction of the desired number of parts in the first level, and then, in the second level, each subgraph is separately partitioned further to obtain the final desired number of parts. In the first level of the hierarchical framework, the domain is partitioned using all the available cores into the number of available nodes via a graph partitioning tool such as Par-MeTiS or PT-SCOTCH. This way, all the memory in the nodes can be used. In the second level of the hierarchy, each node independently partitions the data allocated to itself into the number of cores in the node. In this second level partitioning, graph partitioning tools such as Par-MeTiS or PT-SCOTCH can be utilized as well as cheap geometrical or random partitioning schemes since the communication between the cores of a node is much faster.

3. Obtained Results

Two different sets of experiments on two different datasets (see Table 1 for properties) are conducted to compare the partitioning schemes and tools. In the first set of experiments we check the partitioning results of the tools and in the second set of experiments we compare the running time of *Code_Saturne* when utilizing these partitions.

Table 1. Dataset properties

	# of vertices	# of edges
SMALL	5.780.335	23.064.296
SUBMARINE	107.673.905	427.107.558

The results for MeTiS and Scotch are taken on a Sun AMD Opteron machine with 128GB of memory. The results for PTSCOTCH, PARMeTiS, and Hierarchical (HIER) are taken on the IBM Blue Gene/P (BGP) systems in Jugene and STFC Daresbury. All runs up to 4K cores are taken in STFC and runs larger than 4K cores are taken in Jugene. MeTiS version 5.0.0, PAR-MeTiS version 3.1.1 and Scotch version 5.1.11 are used and in HIER, Zoltan version 3.5 is utilized.

Table 2. Properties of partitions obtained via partitioning the SUBMARINE dataset with MeTiS, SCOTCH and PAR-MeTiS. Reported time values are in seconds and memory utilization is in GB.

# of parts	MeTiS			SCOTCH			ParMeTiS	
	cut	time	memory	cut	time	memory	cut	time
4096	5.899.387	397,30	20,84	6.365.021	2748,80	14,48	6.550.107	28,27
8192	7.569.441	430,51	21,02	8.134.073	3249,69	14,48	8.338.153	32,26
16384	9.703.853	595,28	21,85	10.332.241	3973,43	14,48	10.610.241	26,28
32768	12.405.125	792,76	24,95	13.242.850	4995,55	14,48	X	X
65536	X	X	X	16.820.681	4990,01	14,48	X	X
131072	X	X	X	21.300.171	4983,38	14,48	X	X

In Table 2 we present the partitioning properties of MeTiS, SCOTCH and PAR-MeTiS. As seen in the table, sequential MeTiS produces the best cut values and it runs much faster than sequential SCOTCH. However, in terms of memory utilization SCOTCH is more successful and SCOTCH can partition into much larger number of parts. Finally, in terms of running time, PAR-MeTiS is the clear winner. It runs more than 10 times faster than MeTiS and around 100 times faster than SCOTCH. It also produces better cuts than SCOTCH. However, when the number of parts to be partitioned into increase, PAR-MeTiS has problems just like MeTiS.

To be able to utilize the fast and relatively nice partitioning capability of PAR-MeTiS in partitioning into much larger number of processors, the hierarchical scheme is utilized.

The tests for hierarchical scheme are conducted on the BGP in STFC since we wanted to preserve our quota in Jugene for the time we will have much larger (e.g. 2 Billion) meshes. In the runs for HIER, PAR-MeTiS is used in the first level to divide the graph in $\text{Number_of_parts}/4$ and then in the second level, $\text{Number_of_parts}/4$ separate calls are made to PAR-MeTiS to divide each subgraph into 4.

An analysis of Table 3 shows that the hierarchical scheme has slightly higher runtime per timestep values when compared to other schemes, but it has the potential to scale to much larger number of cores than PAR-MeTiS and PT-SCOTCH so it is still interesting to investigate this scheme.

4. Conclusion

Collected experimental results indicate that proposed hierarchical scheme performs slightly worse than classical schemes but its advantages in partitioning into larger number of parts still makes it a viable approach. Our future work includes improvements in the quality of the partitions obtained via our hierarchical scheme and analysis on much larger number of cores.

Table 3. Runtime per timestep (seconds) of *Code_Saturne* via utilizing partitions obtained by partitioning the SMALL dataset with MeTiS, SCOTCH, PARMeTiS and PT-SCOTCH.

# of parts	HIER	MeTiS	SCOTCH	ParMeTiS	PT-SCOTCH
256	13,45	12,37	12,10	12,76	12,51
512	7,64	5,92	7,04	7,23	7,14
1024	4,96	3,87	4,64	4,91	5,01
2048	3,59	3,21	3,26	3,38	3,45
4096	3,07	2,52	2,66	2,71	2,60

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-211528 and FP7-261557. The work is achieved using the following PRACE Research Infrastructure resources: The Gauss Center for Supercomputing IBM Blue Gene/P system JUGENE (Jueliche Blue Gene/P) at Forschungszentrum Juelich (FZJ), Germany.

References

1. Y. Fournier, J. Bonelle, C. Moulinec, Z. Shang, A. G. Sunderland, J. C. Uribe, Optimizing *Code_Saturne* Computations on Petascale Systems, in *Computers & Fluids*, vol. 45(1), pp. 103-108. 2011.
2. METIS: unstructured graph partitioning and sparse matrix ordering system. <http://glaros.dtc.umn.edu/gkhome/views/metis>
3. ParMetis: parallel graph partitioning and fill-reducing matrix ordering. <http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview>
4. PT-SCOTCH, Software package and libraries for sequential and parallel graph partitioning. <http://www.labri.fr/perso/pelegrin/scotch>
5. ZOLTAN, Parallel Partitioning, Load Balancing and Data-Management Services. <http://www.cs.sandia.gov/Zoltan/>