A high scalability parallel algebraic multigrid solver

T. Saad[1] and M. Darwish[2]

[1] AUB, FEA and P.O. Box 11-0236, Riad El Solh Street, Beirut 1107 2020, Lebanon tsaad@aub.edu.lb
[2] darwish@aub.edu.lb

Summary. In this paper we present the performance analysis of a parallel Algebraic Multigrid Solver (AMG) for a finite volume unstructured CFD code. The multigrid solver is part of an unstructured cell-centered finite volume code. The parallelization of the solver is based on the domain decomposition approach using the single program multiple data paradigm. The Message passing interface Library (MPI) is used for communication of data. An ILU(0) iterative solver is used for smoothing the errors arising within each partition at the different grid levels, and a multi-level synchronization across the computational domain partitions is enforced in order to improve the performance of the parallelized Multigrid solver. A number of Multigrid cycles (V, W and F-Cycle) and two strategies have been implemented. Tests on different grids have been conductied for a non-linear problems on up to 28 parallel processors. Results show, that synchronization plays across partitions for multigrid-levels plays an essential role in ensuring good scalability. Furthermore for large partition number gathering coefficients across partitions is important to ensure a convergence history on par with the sequential solver.

1 Introduction

Computational Fluid Dynamics (CFD) has become an essential engineering tool in many industries (aerospace, automotive, chemical processing, power generation, etc ...). At its basic level CFD involves the following fundamental steps: (i) a discretization step that translates a set of highly non-linear partial differential equations representing conservation principles (conservation of momentum, mass, energy, etc) into a sparse system of linearized algebraic equations, (ii) and a solution step to solve the system of algebraic equations iteratively. Usually iterative solvers are used for solving the system of equations (inner loop) because of their lower computational requirements in memory and cpu time [1]. Because of nonlinearities these two steps need to be performed repeatedly [2] (outer loop) until a final converged solution is reached. The converged solution is then analyzed for the needed engineering information.

To accelerate the convergence of the outer loop effort was expanded to develop more efficient algorithms [3] to treat the inter-equation coupling of the momentum and pressure equations either in a segregated manner [4] or more implicitly [5]. Effort to accelerate the inner loop, have led to the development of geometric multigrid methods in the 1970's and later in the 1980's to algebraic multigrid methods (AMG) [6] that extended the main ideas of geometric multigrid methods to purely algebraic settings, yielding more robustness and algorithmic simplicity. This dual approach sequential has been quite successful in tackling increasingly larger cfd simulations, but the reliance of many industries on using CFD to simulate large scale engineering problems has led to a more drastic approach to decrease simulation time, i.e parallel processing, which enables the scaling of the CFD problem through the decomposition of the problem into as many sub-problems as there are processors. The Parallelization of CFD codes can be achieved in several ways however when the target of the parallelization effort is clusters of interconnected processors [7] then the Domain Decomposition approach is remains the most effective. In Domain Decomposition the parallelization is enforced by dividing (partitioning) the domain of interest into a number of sub-domains or partitions (usually one for each processor). In this case inter-domain synchronization where neighboring sub-domains swap solution information is essential in ensuring a consistent solution. The synchronization can be performed on one extreme only at the finest mesh, with the multigrid solver basically playing the role of its sequential counterpart over the sub-domain or partition, or on the other extreme the synchronization can be performed at each multigrid level, with the multigrid solver then playing a additional role of smoother across partitions but at the expense of additional communication cost between partitions [8]. The subject of this paper is the development of a highly scalable Algebraic Multigrid solver. After a brief review of the different synchronization strategies possible across partitions results for the performance of the pAMG solver for different grid sizes and partition numbers are presented.

2 Parallel Agglomeration Strategies

There are various approaches of parallelizing the coarse grid agglomeration schemes The simplest strategy is to decouple the agglomerated grids across partitions. In this case agglomeration proceeds in each partition independently and the agglomerated levels do not communicate across partitions and thus synchronization across partitions become a non-issue. However, the benefits of multigridding are lost in this case, as the smoothing of the multigrid is no more applied throughout the computational domain. In the second approach denoted by Coupled Coarsening Strategy case the synchronization is performed across partitions for any coarsened grid level. This requires that the total number of multigrid levels across partitions be the same, and that agglomerated shadow be consistent with their agglomerated core elements.

In this approach the agglomeration is restricted to core elements and starts after the partitioning step, and is performed in parallel in the different partitions, this is repeated until the coarsest grid level is reached [9]. At each level, partitions exchange information at the interface regarding the agglomerated shadow and sender elements.

3 Coarse Grid Parallelism

The solver on the coarsest grid can limit the ultimate speedup that can be attained in a parallel computation for two related reasons. First the linear system at this level is generally small and the time required for communication may be higher that the time required to solve the system on a single processor. Second, the coarsest grid may couple all pieces of the global problem, and thus an accurate solution at this level is important, as is the global communication of the right hand side. If the coarse grid is small enough instead of solving in parallel, the coarsest grid problem may factored and solved on a single processor with the right hand side gathered and solution scattered to the other processors. Another option would be to solve the coarses grid problem on all processors, this redundant form of the calculation does not require communication to distribute the results [10].

As the number of processors is increased the coarsest grid problem may become too large, in this case it might become preferable to do a parallel computation. However communication complexity can be reduced by solving with only a subset of the processors. Solving redundantly with a subset of the processors is again an option.

4 Test Problems

To test our parallelization, we choose to solve a nonlinear diffusion problem governed by

$$\nabla \cdot (\Gamma \nabla \phi) = 0 \tag{1}$$

$$\Gamma = \phi^{0.1} \tag{2}$$

where the various boundary conditions are shown in Figure 1. Four unstructured meshes were used for the geometric discretization; 99,454 - 199,222 - 298,154 - 397,393 elements. The runs were performed on a cluster of ten G5 dual processor machines running at 2.0 GHz each with a memory of 512 MB per node. The interconnection network is a 100-MBits Ethernet switch. The runs were made until the residual was reduced to a value of 10^{-4}.

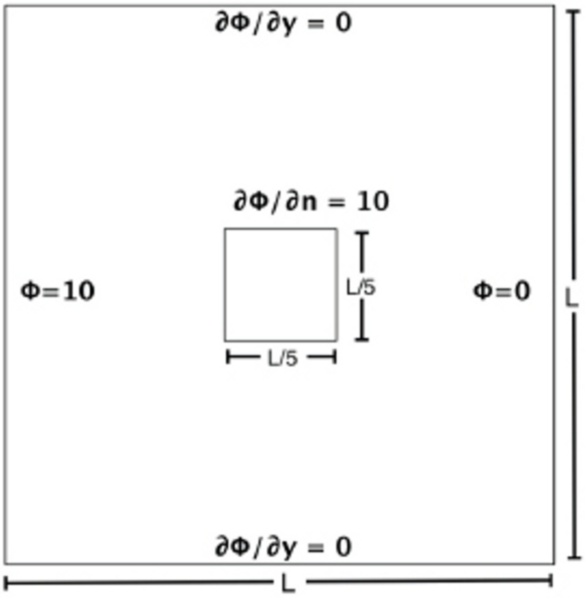

Fig. 1. Test problem geometry

4.1 speedup

The speedup, for a given number of processors P, is defined as the ratio of the solution time of the sequential algorithm to the solution time on P processors.

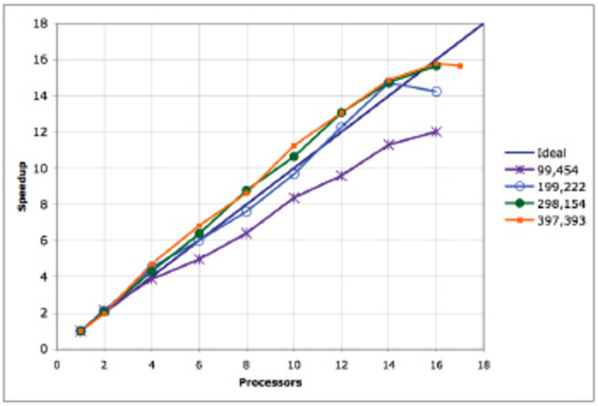

Fig. 2. Speedup

Figure 2 shows the speedup obtained for the four meshes using up to 16 processors. For the 99,454 elements mesh, the scalability starts deteriorating as of 4 processors. For the 199,222 elements mesh, the behavior is linear up to 14 processors. The scalability then starts to deteriorate as of where the ratio of the number of shadow elements to the number of core elements (shadow-to-core ratio hereafter) exceeds 0.02. For the 298,154 elements mesh, it is noticed that the behavior is super linear up to 16 processors at which the shadow-to-core ratio remains almost the same but the communication cost increases. The 397,393 elements mesh displays the same behavior as the latter mesh.

4.2 Communication and Computational Costs

The communication and computational costs are directly related to the number of shadow elements and the number of core elements since the computation takes place over the core elements and the communication takes place over the shadow elements respectively.

Figure 3 shows the efficiency versus the maximum shadow-to-core ratio.This graph basically characterizes the relation between the maximum shadow-to-core ratioand the efficiency of the parallel solver. For the 99,454 elements mesh, as the shadow-to-core ratio exceeds 0.008, the efficiency starts to decrease. For the 199,222 elements mesh, the efficiency starts to degenerate at a shadow-to-core ratio exceeding 0.02. The same happens for the other two meshes. In general, it is noticed that when the shadow-to-core exceeds - on average - a value of 0.02, the efficiency starts to deteriorate.

5 Conclusion

In this paper we have presented the performance of a pAMG solver used in a Finite Volume cell-centered unstructured CFD code. The performance of the parallel solver was compared to that of the sequential solver and it was shown that a substantial improvement in the solution time was gained as long as the computational load is well distributed amongst processors and the ratio of parallel time to the computational time does not exceed 0.25 which is equivalent to a value of 0.02 for the ratio of shadow to core elements. The solver scalability was shown to exceed the theoretical limit in some cases, which is encountered in practice, as the usage of the cache memory is more efficient for small problems. The solver behaves extremely well for a wide range of processors.

6 Acknowledgment

The authors wish to acknowledge the generous support of the Lebanese National Council for Scientific Research (LNCSR).

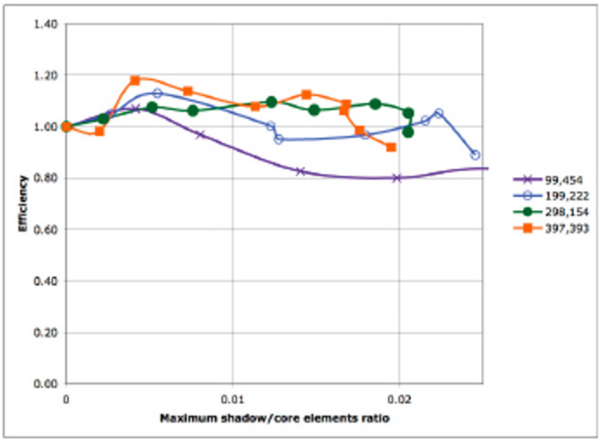

Fig. 3. Relation between efficiency and shadow-to-core ratio

References

1. Bucker H.M.: Iteratively solving large sparse linear systems on parallel computers. NIC Serices, John Von Neumann Institute for Computing, Julich, **10**, 521-548 (2002)
2. Patankar S.V.: Numerical heat transfer and fluid flow. Washington; New York: Hemisphere Pub. Corp.; McGraw-Hill, 1980
3. Moukalled F., Darwish M.S.: A Unified Formulation of the Segregated Class of Algorithms for Fluid Flow at All Speed, Num. Heat Transfer, **37**, 103-139 (2000)
4. Issa, R.I.: Solution of the Implicit Discretized Fluid Flow Equations by Operator Splitting, Mechanical Engineering Report, FS/82/15, Imperial College, London, (1982)
5. Webster R.: Iteratively solving large sparse linear systems on parallel computers. Int. J. Numer Meth. Fluids, **36**, 743-773 (2001)
6. Brandt A. 'Algebraic Multigrid Therory: THe symmetric Case' Appl. Math. COmput, vol 19, pp.24-56, 1986.
7. V. Dolean and S. Lanteri, "Parallel multigrid methods for the calculation of unsteady flows on unstructured grids: algorithmic aspects and parallel performances on clusters of PCs," Parallel Computing, vol. 30, pp. 503-525, 4. 2004
8. Mitchell W. 'A parallel multigrid methoid using the full domain partition', Electron. Trans. Numer. Anal., vol. 6, pp. 224-233, 1998
9. A. Krechel and K. Stüben. Parallel algebraic multigrid based on subdomain blocking. Parallel Computing, 27:1009-1031, 2001
10. Gropp W. D., 'Parallel computing and domain decomposition', in T. F. Chan, D. E. Keyes, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., Fifth Conference on Domain Decomposition Methods for Partial Differentia equations, pp. 349-362, SIAM, 1992.