Perspectives on asynchronous computations for fluid flow problems

Daniel B. Szyld [*]

Department of Mathematics, Temple University, 1805 N. Broad Street, Philadelphia, PA 19122-6094, USA

Abstract

Instances where asynchronous parallel computations can be used for the solution of fluid flow problems are discussed. The use of asynchronous portions of solvers can effectively counteract the consequences of the inefficiencies associated with high latency in the interprocessor (or intercomputer) communication. This is especially so in the case of irregular regions, irregular partitions, or in clusters of inhomogeneous computers. Another situation where asynchronous methods can yield important gains is in the cases of moving boundaries, or adaptive grids. Asynchronous methods can be seen as an alternative to load balancing.

Keywords: Asynchronous communication; Parallel computing; Numerical solution; Fluid dynamics

1. Introduction

In a perfect parallelizable world all computational problems would be divided in tasks of approximately equal difficulty, achieving perfect load balancing. At the same time, communication times between processors of a MIMD machine would be negligible. The same would hold true for communication times between computers in a cluster, or even between machines in remote locations (in the case of metacomputing).

Of course, in most applications, load balancing is not automatic, and considerable attention is given to try to achieve it; see, e.g., the references given in Section 2. Another issue that preoccupies code production is the latency of a message-passing operation, i.e., the amount of time it takes to prepare and set up a message.

The load balancing paradigm is associated with synchronous computations: the processors need to exchange information at a synchronization point, and it is desirable that all processors reach the synchronization point at the same time, thus avoiding processor idle time. If the latency is high, the processors would remain idle during the data exchange period even if there is perfect load balancing.

One counterpart to the load balancing paradigm is to try to use numerical algorithms so that data exchange needs not be synchronized, i.e., to use asynchronous parallel methods, where data is sent to the other processors as soon as it is produced, without waiting for any synchronization point. There is an extensive theoretical foundation for the convergence of these methods; see the references cited is Section 3.

In this contribution, we concentrate on parallel methods for the numerical solution of fluid flow problems; see, e.g., the recent proceedings [6]. In this study, we look at a few instances where the idea of asynchronism has been used, and then propose some other computational situations, where we believe this concept can be used advantageously. This paper is in part a bibliographical tour, and in part an exploration of an alternative to the load balancing paradigm. This alternative has been successfully used in other computational applications.

2. Latency and load balancing

The goal of achieving load balancing has received considerable attention, and is far from trivial. One can appreciate this effort in many publications, e.g., in [29,30], and the references given therein; see also [23] for the case of metacomputing. Other references, where this issue is specifically addressed for fluid calculations include [10,15,21,22,28].

---

[*] Tel.: +1 (215) 204-7288; Fax: +1 (215) 204-6433; E-mail: szyld@math.temple.edu

One issue commonly addressed in these references is the need for runtime load balancing (or rebalancing) to maintain efficiency. This is especially needed in the case of moving boundaries and/or dynamically adaptive grids [29]. One approach is to repartition the grid once the load becomes unbalanced [21]. A more sophisticated approach consists not only of a redistribution of the variables, but also a reconfiguration of the parallel tasks [19]. Of course these strategies add considerable overhead and delays to the computations.

Another tool used to achieve load balancing, as well as latency-hiding, is the use of performance modeling [24], also called timing modeling [3], whereby a mathematical model of the computation and communication appearing in a code is used to study performance. These models can then be used to reconfigure the work distribution and thus obtain a shorter runtime.

3. Asynchronous approaches

Asynchronous parallel iterative methods have been successfully used in many applications of science and engineering; see, e.g., the recent survey [12] and the extensive bibliography therein. Other references worth mentioning include [1,2,5,8,16,18,20,25]. For example, in [7,11], asynchronous solutions of linear systems of several million variables are reported with execution times of about half to two-thirds of those reported for synchronous parallel times on the same architecture. The experiments reported in [11] correspond to a discretization of a second order elliptic boundary value problem, and were run on different CRAY multiprocessors. The problem treated in [7] is that of a singular linear system of equations modeling Petri nets for performance evaluation of computer systems, and the architecture used is a cluster of inhomogeneous workstations connected via Ethernet.

Typically, in the iterative solution of a large linear or nonlinear system of equations, an asynchronous method uses in each processor whatever information is available in local memory at the time the information is needed. (In the case of a shared memory architecture this information would be in the global memory.) When each processor completes a local iteration, it sends the new information it produced to the other (appropriate) processors (or the global memory). Thus, conceptually, the information used by each processor in its computations is older (or less current) than if all processors wait for the synchronization point for the data exchange. Nevertheless, under suitable general assumptions, such asynchronous methods converge to the solution; see the references already cited.

Asynchronous methods do not offer any advantage if there exists already an inherent load balance. Conversely, if there is a load imbalance, and/or the processors have different computational speeds, then asynchronous parallel

methods can perform very well; see, e.g., the experiments in [11]. In these cases, the full utilization of the processing capabilities, i.e., the elimination of the idle time, takes precedence over the fact that the information is less current than in the synchronous case. The load imbalance appears for example in the case of domain decomposition, when not all the domains have the same shape or size, or when one uses irregular meshes or moving boundaries. We should point out that asynchronous methods can be advantageous both in the case of low and high latency, though different considerations have to be taken in each case [18].

There are of course many different numerical methods for the solution of fluid equations; see, e.g., the diverse papers in [6]. Almost always, at the core of such methods there are some large sparse linear or nonlinear systems to be solved, for a given time step. These can be solved with some asynchronous method.

This approach was taken in [14], where a two-dimensional problem was solved using the vorticity–stream-function formulation. The Navier–Stokes equations are rewritten as two coupled boundary value problems for each time step. One is a Poisson equation with Dirichlet boundary conditions and the other is an evolution convection–diffusion equation, also with Dirichlet boundary conditions. The discretized equations are solved with variants of the Schwarz alternating method using a domain decomposition with overlap. The corresponding linear system for the Poisson equation is solved asynchronously. For the other equation, a nonlinear system is solved asynchronously. Other parts of the codes are synchronous; see [14] for more details. Execution times were presented for Reynolds numbers in the range 10–1000, for up to eight processors of an IBM-SP2. The asynchronous times reported are always more efficient than the synchronous counterparts. The increase of efficiency observed is relatively small, and this is probably due to the fact that the domain decomposition chosen has all domains of the same size.

In [4], in a metacomputing environment using two Cray T3E machines across the Atlantic, computations of three-dimensional supersonic flows around space vehicles are reported on a grid of almost a million cells. Asynchronism is used, both within each multiprocessor and across machines.

Other potential for the use of asynchronism is the application of ideas from asynchronous minimization, such as those in [27] to nonlinear elliptic problems as treated, e.g., in [26]. From the theoretical point of view, we highlight the fact that while in domain decomposition methods for discretized differential equations, convergence rates are shown to be independent of the mesh size, provided appropriate coarse subspaces are used, in [27] it is shown that this property is maintained for asynchronous versions of these methods within the general framework of nonlinear minimization problems, including nonlinear elliptic boundary value problems.

These examples confirm the potential for the use of asynchronous methods for portions of fluid dynamics calculations. This potential is bound to become more evident with the advent of the new generation of massively parallel computers and clusters of symmetric multiprocessors, where total communication costs are higher. The need for asynchronism is often brought up in connection with Grand Challenge problems; see, e.g., [9,17].

We believe that the potential for gains using asynchronous parallel iterative methods for linear or nonlinear systems in irregular regions or inhomogeneous clusters can be achieved with a small investment. For example, one could replace the (synchronous) SOR solver for the pressure in the three-dimensional code NaSt3DGP (based on the description in [13]) with an asynchronous linear solver, without much change in the rest of the code. Similarly, when many subdomains of different size appear naturally using a domain decomposition method, an asynchronous version of the type proposed in [11] can be used. In the case of moving boundaries, a subdomain can be enlarged or reduced without fear of a load imbalance. The processor assigned to such a subdomain will simply compute its task and send the resulting information to the other processors. The change in size implies a change in the local computational time, and thus in the delay in getting new information. Nevertheless, all processors can continue with their computational tasks, and no significant degradation of performance should be expected.

Acknowledgements

References

[1] Amitai D, Averbuch A, Israeli M, Itzikowitz S, Turkel E. A survey of asynchronous finite-difference methods for parabolic PDEs on multiprocessors. Appl Numer Math 1993;12:27–45.

[2] Bertsekas DP, Tsitsiklis JN. Parallel and Distributed Computation. Englewood Cliffs, NJ: Prentice Hall, 1989.

[3] Blathras K, Szyld DB, Shi Y. Timing models and local stopping criteria for asynchronous iterative algorithms. J Parallel Distr Comput 1999;58:446–465.

[4] Bönisch TP, Rühle R. Adaptation of a 3-D flow-solver for use in a metacomputing environment. In: Lin CA, Ecer A, Satofuka N, Fox P, Periaux J (Eds), Parallel Computational Fluid Dynamics, Development and Applications of Parallel Technology. Amsterdam: North Holland, 1999, pp. 119–125.

[5] Bru R, Migallón V, Penadés J, Szyld DB, Parallel, synchronous and asynchronous two-stage multisplitting methods. Electr Trans Numer Anal 1995;3:24–38.

[6] Bruneau CH (Ed), Proceedings of the 16th International Conference on Numerical Methods in Fluid Dynamics, Lecture Notes in Physics. Heidelberg: Springer, 1998.

[7] Buchholz P, Fischer M, Kemper P. Distributed steady state analysis using Kronecker algebra. In: Plateau B, Stewart WJ, Silva M (Eds), Numerical Solution of Markov Chains (NSMC'99), Prensas Universitarias de Zaragoza, 1999, pp. 76–95.

[8] Chazan D, Miranker W. Chaotic relaxation. Linear Algebra Appl 1969;2:199–222.

[9] Department of Energy/National Science Foundation. Report on the National Workshop on Advanced Scientific Computing, July 30–31, 1998.

[10] Evangelinos C, Karniadakis GE. Communication patterns and models in PRISM: A spectral element-Fourier parallel Navier–Stokes solver. In: Proc. Supercomputing'96, Pittsburgh, Nov. 17–22, 1996, 1996. Available online at http://www.supercomp.org.

[11] Frommer A, Schwandt H, Szyld DB. Asynchronous weighted additive Schwarz methods. Electr Trans Numer Anal 1997;5:48–61.

[12] Frommer A, Szyld DB. On asynchronous iterations. J Comput Appl Math 2000;123:201–216.

[13] Griebel M, Dornseifer T, Neunhoeffer T. Numerical Simulation in Fluid Dynamics. A Practical Introduction. Philadelphia: SIAM, 1998.

[14] Guivarch R, Spiteri P, Boisson HC, Miellou JC, Schwarz alternating parallel algorithm applied to incompressible flow computation in vorticity stream function formulation. Parallel Algorithm Appl 1998;11:205–225.

[15] Hendrickson B, Devine K. Dynamic load balancing in computational mechanics. Comput Methods Appl Mechan Eng 2000;184:485–500.

[16] Kaszkurewicz E, Bhaya A. Matrix Diagonal Stability in Systems and Computation. Boston: Birkhäuser, 2000.

[17] Keyes DE. Trends in algorithms for nonuniform applications on hierarchical distributed architectures. In: Salas MD, Anderson WK (Eds), Proceedings of the Workshop on Computational Aerosciences for the 21st Century, Kluwer, 2000, pp. 103–137.

[18] Miellou J-C, El Baz D, Spiteri P. A new class of asynchronous iterative algorithms with order intervals. Math Comput 1998;67:237–255.

[19] Moreira JE, Naik VK. Dynamic resource management on distributed systems using reconfigurable applications. IBM J Res Dev 1998;41:303–330.

[20] Pereyra V. Asynchronous distributed solution of large scale nonlinear inversion problems. Appl Numer Math 1999;30:31–40.

[21] Persson I. Performance analysis of a CFD-code on the IBM-SP2. Technical report, Center for Computational Mathematics and Mechanics, Royal Institute of Technology, Stockholm, Sweden, August 1995. Available online at http://www.nada.kth.se/~ipe.

[22] Resch M, Babovsky H. Workstation clustering: a powerful tool for numerical simulation in flight science and space research. J Flight Sci Space Res 1995;19:253–258.

[23] Resch M, Beisel T, Berger H, Bidmon K, Gabriel E, Keller

R, Rantzau D. Clustering T3Es for metacomputing applications. In: Cray User Group Proceedings, 1998. Available online at http://www.cug.org.

[24] Steinberg SG, Yang J, Yelick K. Performance modeling and composition: a case study in cell simulation. In: Proceedings of the 10th International Parallel Processing Symposium, Honolulu, Hawaii, April 1996, Los Alamitos, CA, 1996. IEEE Computer Society Press, pp. 68–74.

[25] Szyld DB. Different models of parallel asynchronous iterations with overlapping blocks. Comput Appl Math 1998;17:101–115.

[26] Tai X-C, Espedal M. Applications of a space decomposition method to linear and nonlinear elliptic problems. Numer Methods Partial Differ Equat 1998;14:321–336.

[27] Tai X-C, Tseng P. Convergence rate analysis of an asynchronous space decomposition method for convex minimization. Math Comput, to appear.

[28] Touheed N, Selwood P, Jimack PK, Berzins M. A comparison of some dynamic load-balancing algorithms for a parallel adaptive flow solver. Parall Comput 2000;26:1535–1554.

[29] Wats J, Taylor S. A practical approach to dynamic load balancing. IEEE Trans Parallel Distr Syst 1998;9:235–248.

[30] Willebeek-LeMair MH, Reeves AP. Strategies for dynamic load balancing on highly parallel computers. IEEE Trans Parallel Distr Syst 1993;4:979–993.