Distributed Combusion Simulations

K. Sikorski* and Kwan-Liu Ma†

Department of Computer Science, University of Utah, Salt Lake City, Utah 84112

Philip J. Smith and Bradley R. Adams

Department of Chemical Engineering, University of Utah, Salt Lake City, Utah 84112

Received May 21, 1993. Revised Manuscript Received September 7, 1993®

This paper reports research in progress. Two types of domain decomposition have been used in distributed computing with networked workstations for the numerical modeling of full-scale utility boilers. The numerical model is a three-dimensional combustion code which couples turbulent computational fluid dynamics with the chemical reaction process and the radiative heat transfer. Two approaches, here called microscale parallelism and macroscale parallelism, are proposed to study the intrinsic parallelism of typical combustion simulations. We describe the implementation of the microscale parallelism as well as its performance on networked workstations.

Introduction

Pressures to reduce the environmental impact of fossil fuel combustion are placing increased demands on the designers and operators of combustion equipment. With increased knowledge of the fundamental mechanisms involved in combustion processes and with rapid improvements in computer capabilities, there has arisen a potential for numerical simulations to address these issues.

Over the past 15 years, faculty at the University of Utah have been involved in the development of combustion simulation software to predict the local behavior of flames in full-scale utility boilers. These power generation systems typically stand 300 ft high and 30 ft wide and generate 500 MW of electricity. Figure 1 show the labeled geometries for an industrial furnace configuration case. Coal (or other fuel such as oil or natural gas) enters the combustion chamber as dust that is blown through inlet burners near the base of the furnace. The fluid dynamical behavior of the gas and the dust flowing through the chamber are computed in order to determine the convective and diffusive mixing of energy and relevant chemical species. This flow and chemical reaction are influenced by the radiative heat transfer that is generated by the combustion process.

This computational problem involves discretization of the partial differential equation set using a nonuniformly spaced three-dimensional Cartesian and cylindrical mesh. Usually 10^6 nodes are needed to resolve important features of the combustion process and up to 60 variables (representing, e.g., components of velocity of the gas, gas pressure, and concentration of various chemical species) are computed at each node. Mathematically the system may be described as a coupled system of nonsymmetric elliptic partial differential equations with algebraic constraints.

The computational requirements of the full-scale simulations are proving to be significant. The numerical mesh needs to be fine enough to resolve physical processes and remove numerical error for full-scale furnaces over a spatial scales that are a few millimeters in some regions. The demands on computer resources (both computational speed and memory) are proving to be unrealistic on traditional serial computer architectures. We have shown that to simulate coal-fired flames in utility boilers to the degree of accuracy sufficient to address environmental emission formation and destruction mechanisms requires finite-difference meshes of several million node points.[4] This computational problem requires a few gigabytes of memory for code and data and runs for in excess of 360 cpu hours on an IBM 3090 vector supercomputer. Without some radical changes in computer architecture, this problem will not be attainable by the average utility or

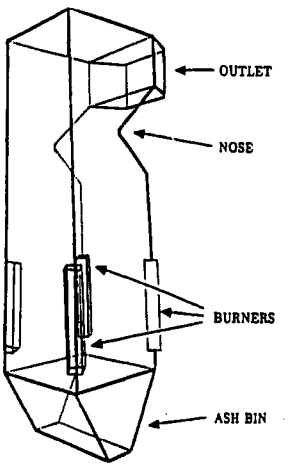

Figure 1. Labeled geometries for an industrial furnace configuration case.

† Current address is ICASE, Mail Stop 132C, NASA Langley Research Center, Hampton, VA 23681.

* Abstract published in *Advance ACS Abstracts*, October 15, 1993.

(1) Beguelin, A.; Dongarra, J.; Geist, A.; Manchek, R.; Sunderam, V. *A Users' Guide to PVM Parallel Virtual Machine*; Sept 1991, ORNL/TM-11826, Oak Ridge National Laboratory.

(2) Davis, M. E. *Numerical Methods and Modeling for Chemical Engineering*; John Wiley and Sons: New York, 1984.

(3) Gillis, P. A. Three-Dimensional Computational Fluid Dynamics Modeling in Industrial Furnaces; Ph.D. Thesis, Brigham Young University, Dec 1989.

(4) Gillis, P. A.; Smith, P. Three-Dimensional Computational Fluid Dynamics for Industrial Furnace Geometries. In *the 23rd Symposium on Combustion*; The Combustion Institute: 1990.

0887-0624/93/2507-0902$04.00/0

industrial user that would like to use these tools to solve their problems.

Vector supercomputers have had much influence on large-scale computational fluid dynamics problems because of their significant processing power and memory capacity. However, vector supercomputers, such as the Cray series or the IBM 3090, are expensive to maintain and run and usually must be shared between many users. Recently the availability of high-performance RISC workstations and high-speed communication links made networked workstation computing also adequate for practical large-scale scientific computing; in particular, they are inexpensive enough that they may be dedicated to individual users; and more importantly, they are easily extensible in both processing power and memory capacity for larger-scale problems. Currently, as in most other computing environments, the local area network connecting a group of hundreds of workstations has become the dominant computing paradigm at the University of Utah. We estimate that the accumulated computing power is at several gigaflops and memory capacity several gigabytes. When considered on a 24 h/day basis, most of the power of most of the machines goes to waste. Therefore, it is worth investigating the utilization of these unused workstation cycles for large-scale scientific simulations like the combustion simulation described above.

In this paper, we describe our experience with distributing a full-scale three-dimensional combustion simulation among networked workstations. The distributed computing scheme is based n an algorithm that we have developed for a three-dimensional Navier–Stokes solver.[6,7] The simulation domain is partitioned into subdomains, which are then treated independently and solved in parallel. Load balancing is achieved by preprocessing the computational mesh. Boundary values along the interfaces between the subdomains are determined to ensure smoothness of the solution across these interfaces. We experiment this divide-and-conquer strategy in two ways: microscale parallelism and macroscale parallelism. While the first approach, microscale parallelism, distributes only the most computing-intensive portion of the simulation, the second approach, macroscale parallelism, distributes the entire calculation.

The distributed computing part of the simulation has been implemented in PVM (parallel virtual machine),[1] which permits the utilization of a *heterogeneous* network of parallel and serial computers as a unified general concurrent computational resource. PVM consists of two parts: a daemon process that initiates processes on all machines used, and a user library that contains routines for initiating processes, for communicating between processes, and for synchronizing processes. PVM provides libraries for both C and Fortran. We have also developed a parallel volume visualization algorithm implemented in PVM.[5] This parallel visualization scheme can be integrated into the parallel simulation code for tracking the states of the simulation in place without moving the data. The explicit parallel programming library provided by PVM is very suitable for our parallel simulation as well as visualization work because of its portability and support to heterogeneous programming. Recently, porting PVM to new massively parallel computers like the Connection Machine CM-5 and the Intel Paragon has been under way.

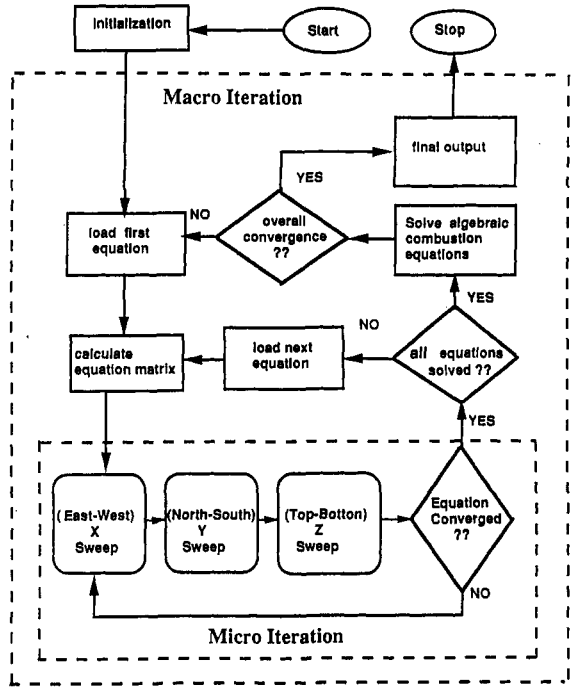

Figure 2. Flow diagram of iterative procedure in the combustion code.

Model Description

The three-dimensional combustion code is an iterative solver. A flow diagram of this iterative procedure is shown in Figure 2. The differential equations for each variable, such as momentum, energy, dissipation rate, or enthalpy, are solved in succession in a decoupled manner. These equations can be classified as Eulerian, steady-state, second-order, nonlinear, elliptical, and partial differential equations (pde's). A series of algebraic equations which describe chemical reaction and turbulent mixing are also solved within the iteration after all of the pde's. This procedure usually begins with a component of velocity and ends with a combustion model species variable. A macroiteration is completed after the complete set of variables has been calculated once; this is illustrated in Figure 2. Coupling of the equation set is achieved through these macroiterations. Several hundred macroiterations are usually required to completely converge the reacting flow field.

The pde's are represented by a large system of linear, algebraic finite-difference equations, for which many solution techniques are available. The technique presently used is an efficient tridiagonal matrix algorithm (TDMA). The TDMA applies to the X–Y–Z sweeping procedure, as shown in Figure 2, to converge each variable, termed microiteration which is performed in a plane-by-plane manner. The TDMA X–Y–Z sweeping is a variation of the alternating direction implicit (ADI) method[2] which usually take over 45% of computational time in the three-dimensional combustion code to arrive at steady state solutions.[3] This single subroutine often requires the largest cpu usage.

Microscale Parallelism

Our initial model does not include the solver for algebraic combustion equations, thus the iterative TDMA solver

(5) Ma, K.-L.; Painter, J. S.; Hansen, C.; Krogh, M. A Data Distributed, Parallel Algorithm for Ray-Traced Volume Rendering. To appear in Parallel Rendering Symposium, 1993.
(6) Ma, K.-L.; Sikorski, K. A Distributed Algorithm for the Three-Dimensional Compressible Navier-Stokes Equations. In *Transput. Res. Appl.* **1990**, *4*, 46–56.
(7) Ma, K.-L.; Sikorski, K. A Distributed 3D Navier-Stokes Solver in Express. *Energy Fuels*, in press.

Table I. Measurements of Microscale Parallelism

grid size	sequential total	1 node		2 nodes		4 nodes	
		total	comp + ex	total	comp + ex	total	comp + ex
50 × 50 × 20	10.9	20.7	11.6	13.9	5.8	15.7	3.9
50 × 50 × 40	29.3	41.7	26.2	27.8	13.4	24.6	7.1
50 × 50 × 60	48.4	74.6	50.2	45.9	25.4	33.5	11.5
50 × 50 × 80	65.7	98.7	64.3	60.7	33.8	45.4	16.5
50 × 50 × 100	82.4	123.2	82.1	72.1	39.1	53.6	18.9
50 × 50 × 120	99.2	138.4	100.8	95.9	55.6	67.4	26.0
50 × 50 × 140	116.5	165.5	110.5	101.5	55.0	76.0	28.5
50 × 50 × 160	133.9	205.0	140.3	117.0	64.9	89.2	35.5
50 × 50 × 180	150.8	206.4	133.4	130.7	72.3	97.8	37.8
50 × 50 × 200	166.5	249.3	169.1	147.6	82.0	107.9	41.6
50 × 50 × 400	332.0	569.3	297.5	300.2	171.5	217.5	88.4

alone counts for over 80% of the total computational time. Therefore, we started with parallelizing the X–Y–Z sweeping procedure for improving overall performance of the algorithm. The complete solver then becomes a hybrid computational model in which the macroiteration runs on a host workstation, and the microiteration is distributed to other workstations. One limitation of this approach is that the main memory capacity of the host workstation restricts the size of the simulation since all distributed solutions must be returned to the host for testing overall convergence and further processing of the combustion submodel. That is, this approach is only good for relaxing the computation load.

Considering a linear array network topology, we evenly subdivide the computational domain along one of the main coordinate axes for this TDMA solver into as many subdomains as we have processors. Using networked workstations, each subdomain is loaded to a separate workstation and X–Y–Z sweeps are applied locally. Since the solution computed at each grid point depends on its six immediate neighbors, such subdomain need only replicate the boundary cells of its immediate left and right neighboring subdomains. However, after each iteration, those values at the boundary cells must be updated accordingly. This requires each workstation to communicate with only two other workstations that are its immediate neighbors in the linear array connection.

Currently, this distributed TDMA X–Y–Z sweeping has been implemented in PVM. Table I shows the performance of the distributed linear system solver on up to four IBM RS/6000-520 workstations. Since the workstations in our distributed computing laboratory are connected with a token-ring network, which only allows one-way data transmission, the nearest-neighbor data exchange is not as efficient as a similar implementation of a Navier–Stokes solver that we have tested on transputer array, which has direct node-to-node links.[6] The most significant problem of this microscale approach is the high overhead of data distribution which must be done for each variable in the macroiteration. In Table I, we show the total time in seconds for the linear system solver to solve one variable and the time with data distribution time removed. The column titled "comp + ex" is the computation time with boundary-cell exchange time. Considering only the computation and data exchange time, high parallel efficiency can be obtained. Thus the use of a high-speed network and a host with large main memory can still make this approach attractive.

Macroscale Parallelism

The high communication overhead in the microscale approach shadows the overall parallel efficiency obtained. In particular, because the current model incorporates the solver for algebraic combustion equations, the microscale approach, now only taking about half the total computation time, becomes even less practical. Another approach, here called macroscale parallelism, relaxes the communication overhead and thus is more efficient for traditional networks like Ethernet. As indicated by its name, the entire macroiteration is distributed among processors (workstations) instead of only the microiteration. The domain decomposition is similar to that of the microscale approach, but we are developing a more sophisticated decomposition scheme to ensure load balancing. One major advantage of macroscale parallelism is that no one machine ever needs all of the data for the entire process. Instead each computational unit retains the memory locally for its subdomain and only passes boundary cell information at each iteration.

This concept was tested using a combustion simulation in a long, axial flow furnace. The furnace was divided first into two and later into three domains to test numerical convergence and stability. Convergence for the test involving two domains was stable and rapid. The number of iterations required for convergence in each domain was approximately 15% greater than that required for the single-domain solution, but the memory required for each domain (i.e., the memory for each processor) was reduced by nearly a factor of 2 and the overall wallclock time required for convergence was around 70% of that required for the single-domain computation. The performance could be improved even further by fine tuning boundary condition updates. The computed primitive variable results (i.e., velocities, temperatures, etc.) were equivalent to those obtained when modeling the entire furnace as a single domain. There were negligible differences in results when updating boundary conditions using PVM in synchronous or asynchronous mode.

Convergence for the test involving three domains was not stable and thus primitive variable results were different from the single furnace simulation. This divergence may be explained as follows. The overall combustion simulation algorithm employs an implicit solver for the fluid flow, chemical reaction, and radiative heat transfer equations. The algorithm is designed to solve elliptic, steady-state, coupled mass, momentum, species, and radiative transfer equations using an implicit, iterative approach. As a problem is divided into multiple domains, equations for each domain may be solved implicitly, but the solution of the overall problem (all domains) becomes explicit due to the segregated nature of the domain solutions. As the solution becomes more explicit, the efficiency of the overall implicit solver is impaired and at some point, such as in the three-domain simulation here, the solution may diverge. Judicious selection of domain boundaries to include the elliptic portion of the solution within one

domain may minimize this ill-behavior. Thus, there is a trade-off between the increased efficiency offered by smaller memory and cpu times and the decreased convergence efficiency caused by the explicit nature of the macroscale parallelism approach. This type of trade-off suggests that each class of problems simulated may have an optimal range of domains which minimize computational resource requirements and maximize convergence efficiency.

Conclusions

The emergence of parallel computers and the software tools to easily access them provides the opportunity to take combustion simulation software tools from the academic realm to the sphere of the industrial user. Societal problems associated with environmental air pollution, incineration of hazardous and municipal waste and issues of global warming all originate in chemical physical processes occurring in utility and industrial furnaces. We have explored parallelization algorithms on a system of networked workstations. Workstation environments such as this are becoming more generally available. Two approaches were explored for solving the large system of simultaneous elliptic pde's and associated constitutive equations modeled in combustion simulations. Our initial test results indicate that a microscale parallelism approach is easy to implement but is computationally efficient only when fast communication links are available. A macroscale parallelism is currently under development to eliminate the communication overhead in the microscale approach. Other future work includes the study of problems in load balancing and grid generation.

Acknowledgment. This work has been supported in part by NSF and the University of Utah under ACERC and an IBM grant for Distributed Computing with clusters of workstations for Geophysical Modeling and Combustion Engineering.