A 65+ Gflop/s Unstructured Finite Element Simulation of Chemically Reacting Flows on the Intel Paragon

(A submission for the Gordon Bell Prize in performance)

*John N. Shadid, Scott A. Hutchinson, Harry K. Moffat, Gary L. Hennigan**

Bruce Hendrickson and Robert W. Leland

Sandia National Laboratories
**New Mexico State University*

Abstract

Many scientific and engineering applications require a detailed analysis of complex systems with strongly coupled fluid flow, thermal energy transfer, mass transfer and non-equilibrium chemical reactions. Examples include combustion research for transportation and energy conversion system and the modeling of chemical vapor deposition (CVD) processing for advanced semiconductor materials. Here we describe the performance of a newly developed application code, SALSA, designed to simulate these complex flows on large-scale parallel machines such as the Intel Paragon. SALSA uses 3D unstructured finite element methods to model geometrically complex flow systems. Fully implicit time integration, multicomponent mass transport and general gas phase and surface species non-equilibrium chemical kinetics are employed. The implicit nature of the algorithm requires the solution of a coupled set of nonlinear PDEs on unstructured computational domains, a difficult task on the distributed memory architectures of modern large-scale parallel machines. To address these issues we have designed SALSA around general kernel routines. These include automated problem partitioning algorithms, efficient unstructured message passing communication, a distributed sparse-block matrix representation of the fully summed global finite element equations (as opposed to less efficient element-by-element techniques) and a parallel preconditioned Krylov iterative solver library. Using these techniques we have obtained over 65 Gflop/s on a minimal-flop solution of a 3D chemically reacting flow CVD problem for Silicon Carbide (SiC) deposition. This represents 46% of the peak performance of our 1904 node Intel Paragon, an outstanding computational rate in view of the required unstructured data communication and sparse matrix computations.

1. Introduction

Current state-of-the-art chemically reacting flow codes use either complex fluid dynamic with simple chemical reaction mechanisms or complex chemical kinetics with simple fluid mechanics models. This unfortunate dichotomy in resolution is due to the tremendous computational resources needed to solve large, real-world chemically reacting flow problems in complex flow geometries. To date, the solution of 3D problems with complex reaction chemistry and flow geometries has been impossible.

As an important example, designers of CVD reactors need detailed information on the complex flow structure, temperature distribution, chemical species distribution and uniformity of deposition rates. In a typical reaction mechanism there can be over thirty important chemical species undergoing more than fifty reactions. Consequently the computer analysis of these systems in the past has been limited to idealized 1D and 2D geometries with a moderate number of chemical species. Our results demonstrate that large parallel machines can provide the performance and memory necessary to solve chemically reacting flow problems in 3D with an equal emphasis on flow and reaction kinetics. However, achieving this performance requires addressing specific difficulties associated with unstructured FE implementations on distributed memory computers.

One such difficulty is determining how the problem domain should be decomposed and mapped to the individual processors for maximum efficiency. We abstract this to a graph partitioning problem and have developed a number of new and effective algorithms to address different needs. Other important performance issues result directly from choices made in implementing the FE method. In particular, the choice of efficient data structures for the parallel iterative solvers is critical to overall performance. The ideal structure combines efficient memory usage with high computational throughput and allows straightforward implementation of robust preconditioners. These issues have been addressed in SALSA, enabling unprecedented performance in the solution of implicit unstructured FE problems. Through state-of-the-art algorithms and efficient solution methods, problems of real importance to the semiconductor industry can now be solved.

2. Numerical Methods

SALSA computes the solution of the conservation equation for momentum, total mass, thermal energy, and individual gas and surface phase chemical species for low Mach number flows. As Table 1 shows, these equations form a complex set of coupled, nonlinear PDEs. Constituative relations for the stress tensor, $\mathbf{T}$, heat flux vector, $\mathbf{q}$, and species mass fluxes, $\mathbf{j}_k$, are based on non-equilibrium statistical mechanical theory of multicomponent, dilute polyatomic gases. Necessary transport properties, diffusion coefficients, kinetic rate constants and diffusion velocities are obtained from the CHEMKIN [5] subroutine library developed at Sandia. This library provides a rigorous treatment of dilute-gas multicomponent transport, including the effects of thermal diffusion. Chemical reactions occurring in the gas phase and on surfaces are also obtained through CHEMKIN.

Momentum	$\frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla \bullet (\rho\mathbf{u}\mathbf{u}) - \nabla \bullet \mathbf{T} - \rho\mathbf{g} = 0$
Total Mass	$\frac{\partial\rho}{\partial t} + \nabla \bullet (\rho\mathbf{u}) = 0$
Thermal Energy	$\hat{C}_p\left[\frac{\partial(\rho T)}{\partial t} + \nabla \bullet (\rho\mathbf{u}T)\right] = -\nabla \bullet \mathbf{q} + \Phi + \dot{Q} + \sum_{k=1}^{N_s} h_k \nabla \bullet \mathbf{j}_k - \sum_{k=1}^{N_s} h_k W_k \dot{\omega}_k$
Species Mass Fraction for Species k	$\frac{\partial(\rho Y_k)}{\partial t} + \nabla \bullet \rho\mathbf{u}Y_k = -\nabla \bullet \mathbf{j}_k + W_k\dot{\omega}_k,\ k = 1, 2, \ldots, N_S - 1$

Table 1: Governing Conservation Equations

The continuous problem defined by the governing conservation equations is approximated by a Galerkin finite element method for the spatial representation coupled with first and second order dynamically controlled time stepping methods. At each time step an implicit solution of the nonlinear PDEs is accomplished by a back-tracking, damped Newton method. The resulting linear systems are solved iteratively using preconditioned Krylov techniques. The nonlinear Jacobian entries are determined by both analytical construction and numerical differentiation. The strong coupling between chemical species at a particular FE node and the induced nonzero block structure motivate the use of the sparse block representation discussed later. Since the element integrations and chemical kinetics mechanisms depend only on local geometry and thermodynamic state, they can be computed independently on each processor, yielding nearly perfect parallel scaling.

3.1 Sparse Matrix Data Structures

Typical FE applications store the sparse coefficient matrix in one of two general ways - in an element-by-element (EBE) scheme or as fully summed equations. In the EBE case [2], each element's interaction matrix is stored separately and is not explicitly summed with contributions from neighboring elements. Rather, all matrix-vector operations are performed with elemental matrices where the global vector result is only obtained after summing over all elements. While this scheme simplifies parallel implementation, it substantially increases the required storage and flops. For example, in the case of 3D linear hexahedral elements, more than *three* times as many floating point computations are required with a corresponding increase in execution time. Although the larger block sizes associated with the EBE approach may yield an increase in flop rate, this is unlikely to compensate for the increased operation count.

For this reason we have chosen to store fully summed equations in a sparse matrix format which allows for a minimal flop solution. This approach sums the elemental coefficient matrices at the beginning of the linear solve rather than summing each matrix vector product and hence eliminates redundant computations. Further, it allows the design of more complex and robust preconditioning methods. We have implemented these fully summed equations using a variable block row (VBR) format [3]. In this approach the matrix has a sparse block structure due to the coupling of the physics at a FE node. These blocks are stored contiguously in memory so that the indirect addressing of other sparse matrix formats is replaced with directly addressed dense matrix vector multiplications, e.g. those in level 2 BLAS. These multiplications can be very efficient on CPUs which use vectorization and/or a cache memory hierarchy. In SALSA we have used special assembly coded routines supplied by Intel to enhance performance on problems with small block sizes. These routines sustain a computational rate in excess of 40 Mflop/s on an individual Intel Paragon processor for the SiC - CVD problem in which the blocks are of size 24x24.

3.2 Krylov Iterative Solvers

The parallel Krylov algorithms implemented in SALSA include CG, CGS, GMRES, CGSTAB and TFQMR [6]. The available preconditioners are row sum and block Jacobi scaling, block Jacobi preconditioning and Neumann series and least-squares polynomial methods. (A domain decomposition preconditioner using block ILU is under development.) The main kernels of the iterative methods are matrix-vector products, DAXPY type operations and vector inner products. The key to performance in these solvers is the efficiency of the matrix-vector multiply kernel, where interprocessor communication time must be minimized.

In forming the sparse matrix and vectors, each processor is assigned a set of unknowns corresponding to a set of rows in the sparse matrix and associated vectors [7]. This set is further subdivided into *border* and *internal* unknowns. Border unknowns are those which must communicate with neighboring processors to complete the matrix-vector multiply; the remaining unknowns on a processor are designated as internal. Those unknowns required for a processor's computations but assigned to a neighboring processor are designated *external*. Calculations on the internal nodes require no updated values, so they can proceed simultaneously with communication. Once a specific partition and assignment of the unknowns to internal, border and external sets has been defined, a distributed VBR sparse matrix storage scheme is constructed.

3.3 Partitioning

Much of the algorithm development for the parallel solution of PDE systems has focused on problems on regular domains. The partitioning of the resulting structured grids can be accomplished easily using simple heuristics which minimize the perimeter-to-area (or surface -to-volume) ratios of the subdomains. In addition, these subdomains can then be mapped directly to hypercube or mesh architecture parallel computers so that only nearest neighbor (and hence contention free) communication is required. In contrast, the partitioning and mapping problem for unstructured meshes is much more difficult. Indeed, the determination of a partition which minimizes communication between balanced sets is known to be an NP-hard problem. Furthermore, an unstructured mesh cannot generally be mapped with only nearest neighbor communication on a hypercube or mesh architecture. The resulting contention for interprocessor communication channels make near-optimal mappings even more important.

The partitions used to produce the results presented in Section 4 were generated using Chaco [4], a general graph (or mesh) partitioning code which was developed in conjunction with SALSA and which supports a variety of new and established graph partitioning heuristics. These include spectral techniques, geometric methods, multilevel algorithms and the Kernighan-Lin method. All of these approaches may be applied in bisection, quadrisection or octasection mode to recursively partition general graphs for mapping onto hypercubes and mesh architectures of arbitrary size. The input graphs describing the application's communication pattern may be edge or vertex weighted, allowing accurate modeling of inhomogeneous computation and communication loads. The partitions generated by Chaco were processed further to increase the number of internal nodes (enhancing communication/computation overlap), and to improve the mapping of regions onto the Paragon architecture. Using these techniques, a problem mapping was constructed with low communication volume, good load balance, few message start-ups and a small amount of congestion, which also facilitated the overlapping of communication and computation.

4. Results

We have used SALSA to investigate a problem of current interest in the CVD community - the deposition of SiC. The reaction mechanism for this problem has 19 species and over 40 gas phase reactions, a simplified 1D simulation of which has recently appeared in the literature [1]. In our simulation we use a false transient time-stepping algorithm to solve for the three fluid velocities, pressure, temperature and 19 chemical species (resulting in 24 total unknowns) per FE node. The Reynolds number based on the inner disk diameter is 120 and the characteristic temperature difference between the heated disk and the inlet gasses is 700^{o}K. We simulated both a horizontal reactor and a vertical disk reactor depicted in Figure 1. Table 2 contains results for the horizontal reactor configuration with 176,000 FE nodes. This problem has 4.2 million unknowns and 2.5 billion nonzeros in the global coefficient matrix.

	Number of Processors		
	476	952	1904
#Unknowns	1.1 x 10^6	2.1 x 10^6	4.2 x 10^6
#Nonzeros	6.3 x 10^8	1.3 x 10^9	2.5 x 10^9
Matrix Fill (sec.)	41.3	42.1	40.2
Gflops	16.3	32.8	65.7
Percent of Peak*	46%	46%	46%

Table 2: Scaling Performance of Linear Solvers, CGS - LS9: Horizontal reactor 175,000 FE nodes

*Based on 75 Mflops/node.

We begin our discussion by considering the overall scaling of the two principal kernels, the matrix fill routine and the linear solver. For these results we select a representative solver (CGS-Conjugate Gradient Squared) and several least-squares preconditioners. The results in Table 2 correspond to three reactor models in which the number of unknowns per processor has been held constant and the number of processors has been scaled. We do not present a flop rate for the fill portion of the application since it involves a library of chemical kinetics software with an undocumented flop count. However, since each processor computes values for the nodes it owns, this operation is perfectly parallel. The linear solver results in the table indicate almost perfect scaling. They sustain nearly 50% of the peak performance of the i860 despite the unstructured communication and sparse matrix computations, achieving over 65 Gflop/s on 1904 processors. The partitions for these problems were generated using a multilevel Kernighan-Lin partitioning algorithm with post-processing as discussed in Section 3.3. We have observed that lower quality methods like inertial partitioning reduce performance by up to 20%.

In Table 3 we present the performance of the solvers using two basic preconditioners, the block Jacobi (Blk_Jac) and an "n" term least squares polynomial. The polynomial preconditioners have an overall higher Gflop rate than the block Jacobi preconditioners because they rely more heavily on the efficient VBR matrix vector multiplication. Although high flop rates do not always mean faster solutions [6], our studies of performance and convergence indicate that the simple polynomial preconditioners are quite effective when combined with appropriate matrix scaling. They are straightforward to implement and perform well on the transport part (flow) of the time dependent PDE system. However, we have also observed that the block Jacobi preconditioner has very desirable properties with respect to the strongly coupled local chemistry portion of the PDEs.

	Preconditioner			
	Blk_Jac	LS1	LS5	LS9
Gflops	50	55	61.3	65.7
Percent of Peak*	35%	39%	43%	46%

Table 3: Performance of Preconditioners on Linear Systems: Horizontal reactor 175,000 FE nodes

In Table 4 we give results for a representative Newton iteration at 0.3 microseconds into the solution of SiC deposition in the vertical CVD reactor. This problem contains 3.8 million unknowns and 2.4 billion nonzeros in the Jacobian matrix. At this point in the simulation the solution of the linear system takes more than 90% of the total time which is representative of the proportion of fill-time to solve-time of difficult timesteps in the simulation, validating our emphasis on the solution routines.

We present a graphical display of a representative load balance of the vertical reactor geometry in Figure 1. In this figure the varying colors indicate the distinct processor assignments. Interprocessor boundaries, across which communication is required, are shown in red. A representative plot of flow streamlines in the reactor along with a temperature distribution on the center plane is shown in Figure 2. The steamlines identify recirculating flow which, coupled with the reaction chemistry, may reveal undesirable deposition patterns. This is precisely the kind of design flaw that we can now address with SALSA.

	Matrix Fill (sec.)	Linear Solver	Iterations	Total Time (sec.)
CGS - LS5	36.7	378.0	357	414.7

Table 4: Time to solution for a Newton Step, Vertical CVD Reactor - CGS Solver

5. Conclusions

We have developed a parallel code to simulate chemically reacting flows on complex geometries, with the specific goal of modelling chemical vapor deposition reactors. The code solves complex 3D flow, heat transfer, mass transfer and nonequilibrium chemical reaction equations on unstructured grids. Through a combination of innovative algorithms and data structures, this code has obtained 65 Gflops/s and an efficiency of nearly 50% on the Intel Paragon, while simulating a problem of great interest to Sandia and the semiconductor industry.

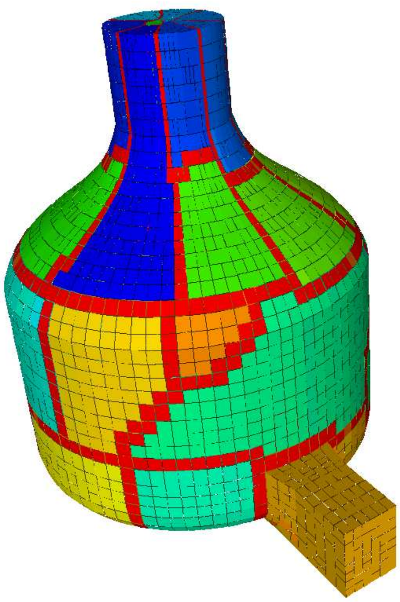

Figure 1. Load balance of vertical CVD reactor for 28 processors constructed with Chaco using spectral quadrisection and KL refinement.

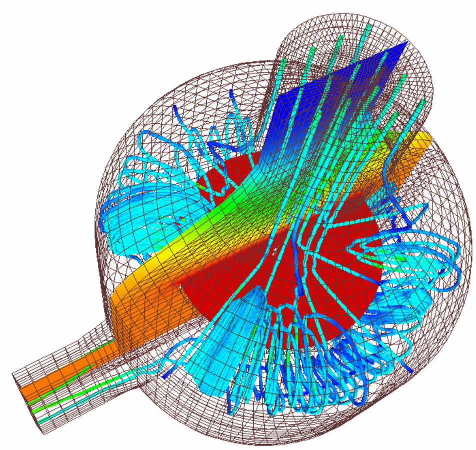

Figure 2. View of streamlines showing detrimental recirculation and a temperature profile in a vertical CVD reactor.

Acknowledgments

The authors gratefully acknowledge the work of Satya Gupta at Intel Supercomputer Systems Division for providing key assembly-coded routines. We would also like to thank Stephen Wheat, Rolf Riesen and the rest of the SUNMOS team at Sandia, as well as Ted Barragy, Mike Proicou and Mack Stallcup of Intel SSD for their help with SUNMOS and the Intel Paragon. We are also grateful to Mike Heroux for valuable discussions concerning the VBR sparse matrix format.

References:

1. M. Allendorf and R. Kee, “A model of silicon carbide chemical vapor deposition”, J. Electrochemical Soc., Vol. 138, No. 3, 841-852, (1991)
2. M. Behr, A. Johnson, J. Kennedy, S. Mittal and T. Tezduyar, “Computation of incompressible flows with implicit finite element implementations on the Connection Machine”, Technical Report 92-045, Army High Performance Computing Research Center, (1992)
3. S. Carney, M. Heroux and G. Li, “A proposal for a sparse BLAS toolkit”, SPARKER Working Note #2, Cray Research, Inc., Eagen, MN, (1993)
4. B. Hendrickson and R. Leland, “A user’s guide to Chaco, Version 1.0”, Technical Report SAND93-2339, Sandia National Laboratories, Albuquerque, NM, (1993)
5. R. Kee, G. Dixon-Lewis, J. Warnatz, M. Coltrin, J. Miller, Sandia National Laboratories Report, SAND 86-8245 (1986).
6. J. Shadid and R. Tuminaro, “A comparison of preconditioned nonsymmetric Krylov methods on a large-scale MIMD machine”, SIAM J. Sci. Comput., Vol. 15, No. 2, pp 440-459, March 1994
7. J. Shadid, S. Hutchinson, and H. Moffat, “Parallel Performance of a Preconditioned CG Solver for Unstructured Finite Element Applications”, Proceedings of the Colorado Conference on Iterative Methods, Breckenridge CO, April 5-9, 1994