AIAA-95-0152

**IMPLEMENTATION OF A PARALLEL ALGORITHM
FOR THERMO-CHEMICAL NONEQUILIBRIUM
FLOW SIMULATIONS**

**C. Channy Wong, F. G. Blottner, and Jeffrey L. Payne
Sandia National Laboratories
Albuquerque, NM**

**Moeljo Soetrisno
Amtec Engineering, Inc.
Bellevue, WA**

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

**33rd Aerospace Sciences
Meeting and Exhibit
January 9-12, 1995 / Reno, NV**

MASTER

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

IMPLEMENTATION OF A PARALLEL ALGORITHM FOR THERMO-CHEMICAL NONEQUILIBRIUM FLOW SIMULATIONS*

C. Channy Wong[†], Fred G. Blottner[‡], and Jeffrey L. Payne[†]

Sandia National Laboratories, Albuquerque, NM 87185-5800

Moeljo Soetrisno[#]

Amtec Engineering, Inc., Bellevue, WA 98009

Abstract

Massively parallel (MP) computing is considered to be the future direction of high performance computing. When engineers apply this new MP computing technology to solve large-scale problems, one major interest is what is the maximum problem size that a MP computer can handle. To determine the maximum size, it is important to address the code scalability issue. Scalability implies whether the code can provide an increase in performance proportional to an increase in problem size. If the size of the problem increases, by utilizing more computer nodes, the ideal elapsed time to simulate a problem should not increase much. Hence one important task in the development of the MP computing technology is to ensure scalability. A scalable code is an efficient code.

In order to obtain good scaled performance, it is necessary to first have the code optimized for a single node performance before proceeding to a large-scale simulation with a large number of computer nodes. This paper will discuss the implementation of a massively parallel computing strategy and the process of optimization to improve the scaled performance. Specifically, we will look at domain decomposition, resource management in the code, communication overhead, and problem mapping. By incorporating these improvements and adopting an efficient MP computing strategy, an efficiency of about 85% and 96%, respectively, has been achieved using 64 nodes on MP computers for both perfect gas and chemically reactive gas problems. A comparison of the performance between MP computers and a vectorized computer, such as Cray-YMP, will also be presented.

*This work was supported by the United States Department of Energy under Contract DE-AC04-94AL85000 through the Laboratory Directed Research and Development (LDRD) Program.

[†]Senior Member of Technical Staff

[‡]DMTS, Fellow AIAA

[#]Senior Engineer. Member AIAA

This paper is declared a work of the U. S. Government and is not subject to copyright protection in the United States

Nomenclature

$C_{\epsilon 1}$	turbulent coefficient constant 1 = 1.44
$C_{\epsilon 2}$	turbulent coefficient constant 2 = 1.92
e	internal energy per unit mass
e_{ve}	vibrational-electronic component of the internal energy per unit mass
E	total energy per unit volume = $\rho(e + \frac{1}{2}u_j u_j)$
G_k	$\mu_i \cdot \left(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i} \right) \cdot \frac{\partial u_i}{\partial x_k}$
k	turbulent kinetic energy
M_i	molecular weight of species i
NS	total number of species
P	pressure
q_i	total heat transfer rate
q_{ve}	heat transfer rate for the vibrational-electronic component of the internal energy
s	subscript s is the number of the chemical species being considered
u_i	velocity
w_i	chemical species source terms
y_j	moles of species j per unit volume
β_c	a relaxation parameter greater than one
ϵ	turbulent dissipation
ρ	density
τ_{ij}	the full viscous shear stresses,
v_{si}^d	diffusion velocity of species s .

I. Introduction

Massively parallel (MP) computers hold the promise to solve accurately and efficiently complex, three-dimensional compressible flow problems with nonequilibrium chemistry. At present, it is extremely expensive to analyze these chemically reactive flow problems accurately with a conventional supercomputer like the Cray-YMP. In most cases, simulations are not even possible without a host of simplifying assumptions. Massively parallel computers provide the potential for efficiently solving this important class of problems that appear in

aerospace, manufacturing, and combustion industries. Currently, fluid dynamics software that can take advantage of the existing massively parallel computer does not exist in the open literature.

When engineers apply this new, massively parallel computing technology to large-scale problems, their major interest will likely be related to the characteristics of the problem and the procedure to obtain a solution. They are probably less interested with the hardware performance of the computer. A few typical MP simulation questions are: (1) what is the maximum problem size that a MP computer can handle? (2) how easy is it to work in a massively parallel computing environment? (3) how long (the total elapsed time) does it take to obtain a reasonably converged solution? (4) how many computational cells are needed to solve the problem? and (5) whether the grid resolution is fine enough?

To determine the maximum problem size for a given MP computer, it is important to address the code scalability issue. Scalability implies whether the code can deliver an increase in performance proportional to an increase in problem size. As the size of the problem increases, engineers should be able to utilize more computer nodes so that the total elapsed time to simulate a problem would not change much. Hence, one important task in massively parallel computing is to ensure that code performance scales accordingly with an increase in number of computer nodes. An inefficient code will result in an enormous increase in computer time when solving a large-scale problem with a large number of computer nodes.

This paper presents a newly developed, highly parallel, scalable version of a chemically reactive Navier-Stokes solver that works in the Intel/Paragon and Intel/Gamma MP computers. The computer code, known as PINCA^{1,2}, has been applied to simulate a hypersonic re-entry problem and a shock-induced combustion problem. Both perfect gas as well as thermo-chemical nonequilibrium gas models were considered. Results of these simulations have demonstrated that it is important to develop a massively parallel computing strategy and to minimize communication overhead in order to achieve a good scaled performance.

The first step to obtain a good scaled performance is to optimize the code for single node performance before proceeding onto a large-scale simulation with a large number of computer nodes. The following sections will discuss the implementation of a massively parallel computing strategy and the process of optimization to improve the scaled performance. Specifically, we have looked at domain decomposition, memory resource management, communication overhead, and problem mapping.

II. Description of the PINCA Code

The PINCA code, a parallel version of the INCA code³, is a three-dimensional, finite-volume, multiple-zone, upwind, implicit Navier-Stokes solver with many capabilities. It solves the Reynolds-averaged Navier-Stokes equations with user specified chemistry and an optional two-equation k-ε turbulence model. The governing equations in PINCA written in an integral form are as follows:

$$\frac{\partial}{\partial t} \int_{Vol} U dV + \int_S (\vec{P} \cdot \vec{n}) dS = \int_{Vol} N dV \quad (1)$$

$$\text{where } \vec{P} = F_1 \hat{i} + F_2 \hat{j} + F_3 \hat{k}$$

$$U = \begin{bmatrix} \rho_1 \\ \dots \\ \rho_s \\ \dots \\ \rho_{NS} \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ E \\ \rho e_{ve} \\ \rho k \\ \rho \epsilon \end{bmatrix}$$

$$F_i = \begin{bmatrix} \rho_1 u_i + \rho_1 v_{1i}^d \\ \dots \\ \rho_s u_i + \rho_s v_{si}^d \\ \dots \\ \rho_{NS} u_i - \rho_{NS} v_{NSi}^d \\ \rho u_i u_1 + p \delta_{1i} + \tau_{i1} \\ \rho u_i u_2 + p \delta_{2i} + \tau_{i2} \\ \rho u_i u_3 + p \delta_{3i} + \tau_{i3} \\ (E + p) u_i + u_j \tau_{ij} + q_i + \sum_s \rho_s v_{si}^d h_s \\ \rho e_{ve} u_i + q_{ve_i} + \sum_s \rho_s v_{si}^d e_{ve_s} \\ \rho k u_i - \left(\mu_k \frac{\partial k}{\partial x_i} \right) \\ \rho \epsilon u_i - \left(\mu_\epsilon \frac{\partial \epsilon}{\partial x_i} \right) \end{bmatrix}$$

$$N = \begin{bmatrix} w_1 \\ \dots \\ w_s \\ \dots \\ w_{NS} \\ 0 \\ 0 \\ 0 \\ 0 \\ w_{ve} \\ G_k - p\varepsilon \\ \left(C_{\varepsilon 1} \cdot \frac{\varepsilon G_k}{k} \right) - \left(C_{\varepsilon 2} \cdot \frac{p\varepsilon^2}{k} \right) \end{bmatrix}$$

The subscript s is the number of the chemical species being considered.

A wide range of inflow, outflow, and wall boundary conditions are available to model complex three-dimensional flow fields. The inviscid terms of the equations are approximated using either flux-vector⁴ or flux-difference splitting⁵ in combination with upwind biased or total variation diminishing (TVD) differencing⁶.

PINCA solves the field equations using an implicit finite-volume method with the resulting difference equation solved with a LU-SGS method. This type of algorithm has proven to be a robust and efficient relaxation procedure for obtaining steady state flow fields². It involves with the following three steps:

1. Calculate the residuals, $R_{i,j,k}$, for each cell using an explicit flux balance.

$$R_{i,j,k} = -\left(D_i \cdot \dot{P} \cdot \dot{S} + D_j \cdot \dot{P} \cdot \dot{S} + D_k \cdot \dot{P} \cdot \dot{S} \right)^n + N_{i,j,k}^n \quad (2)$$

2. Calculate the change in the solution by solving the block-linear system of algebraic equations. The rows of this systems are the block-linear algebraic relations resulting from the implicit flux balance applied to each cell.

$$\begin{aligned} & \frac{Vol_{i,j,k}}{\Delta t} \cdot \delta U_{i,j,k} + \frac{\partial N_{i,j,k}}{\partial U_{i,j,k}} \cdot \delta U_{i,j,k} \\ & + D_i \left[A_{i+1/2,j,k}^+ \delta U_{i,j,k} + A_{i+1/2,j,k}^- \delta U_{i+1,j,k} \right] \\ & + D_j \left[B_{i,j+1/2,k}^+ \delta U_{i,j,k} + B_{i,j+1/2,k}^- \delta U_{i,j+1,k} \right] \\ & + D_k \left[C_{i,j,k+1/2}^+ \delta U_{i,j,k} + C_{i,j,k+1/2}^- \delta U_{i,j,k+1} \right] \\ & = R_{i,j,k} \end{aligned} \quad (3)$$

Because of the choice of Jacobians, this resulting block matrix multiplying $\delta U_{i,j,k}$ is approximately diagonal when the source Jacobian, $\frac{\partial N_{i,j,k}}{\partial U_{i,j,k}}$ is zero.

3. Update the solution using

$$U_{i,j,k}^{n+1} = U_{i,j,k}^n + r \cdot \delta U_{i,j,k} \quad (4)$$

where r is a relaxation factor.

In Step 2, most implicit algorithms require solving a large sparse system of linear equations. The LU-SGS algorithm approximately solves this system using two sweeps of a point Gauss-Seidel relaxation. When there are no source Jacobian terms (i.e., perfect gas), the LU-SGS scheme is very efficient because the main block of this linear system of equations is a diagonal matrix and hence is easy to invert.

When there are chemical reactions, the source Jacobian is not zero. Therefore, the resulting matrix is not a diagonal matrix anymore and would be more expensive to invert. We therefore replace the source Jacobian due to the chemical source terms with an approximate diagonal chemical source Jacobian⁷.

If only the species terms are involved, the source Jacobian is of the following form.

$$Z = \frac{\partial N}{\partial U} = \begin{bmatrix} \frac{\partial w_1}{\partial p_1} \frac{\partial w_1}{\partial p_2} \dots \frac{\partial w_1}{\partial p_{n-1}} \frac{\partial w_1}{\partial p_n} \dots \dots \\ \frac{\partial w_2}{\partial p_1} \frac{\partial w_2}{\partial p_2} \dots \frac{\partial w_2}{\partial p_{n-1}} \frac{\partial w_2}{\partial p_n} \dots \dots \\ \frac{\partial w_3}{\partial p_1} \frac{\partial w_3}{\partial p_2} \dots \frac{\partial w_3}{\partial p_{n-1}} \frac{\partial w_3}{\partial p_n} \dots \dots \\ \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \end{bmatrix} \quad (5)$$

In the diagonal method this matrix is replaced by

$$\tilde{Z} = \begin{bmatrix} \frac{1}{\tau_1} & 0 & 0 & \dots & \dots & \dots & 0 \\ 0 & \frac{1}{\tau_2} & 0 & \dots & \dots & \dots & 0 \\ 0 & 0 & \frac{1}{\tau_3} & \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & 0 & \frac{1}{\tau_n} \end{bmatrix} \quad (6)$$

where,

$$\frac{1}{\tau_i} = \frac{\beta_c}{M_i} \cdot \left[\sum_j \left(\frac{\partial w_i}{\partial y_j} \right)^2 \right]^{1/2} \quad (7)$$

This algorithm eliminates the expense of inverting large block matrices that arise in chemically reacting flows and allows more species and more reactions to be included in the calculation. Details of the equations and numerical methods can be found in References 2 and 3.

This LU-SGS algorithm with the approximate source Jacobian terms is suitable for use in an MP computing environment, since it requires less memory per computational cell and less communication-to-computation ratio compared to a direct inversion approach. Furthermore, the domain of dependence of the LU-SGS algorithm is smaller than the direct inversion approach, consequently, its convergence characteristics are not significantly altered by parallelization.

III. Massively Parallel Computing Strategy

A. Scalability

Scalability of the code addresses the scaled performance; whether the code can deliver an increase in performance proportional to an increase in problem size. In the following performance study, the scaled problem is set up as follows: locally, each computer node will process information within a subdomain of interest. Each subdomain will have a fixed and equal number of computational cells to maintain a well-balanced loading. As the problem size increases, more computer nodes will be utilized, though the number of computational cells per node will remain constant. In theory, the ideal performance of a scalable code should be independent of the problem size (Figure 1). In practice, such an ideal performance is very difficult to achieve.

Figure 1 Previous PINCA (Version 1a) Results for a Scaled Problem (Local Mesh per Node: 20x20x1).

Figure 1 also shows the poor performance of an inefficient code. These are the results of the earlier version of PINCA (version 1a) simulating a hypersonic flow over a conical body⁸. Efficiency of PINCA falls to 17% when utilizing 64 computer nodes. There is a substantial increase in the elapsed time when a large number of computer nodes are used. Further investigation reveals that communication time between computer nodes dominates the elapsed time. Hence one important task of code optimization is to minimize any increase in the communication time with an increasing number of computer nodes.

B. Domain Decomposition

One favorable approach to massively parallel computing in computational fluid dynamics is domain decomposition⁹. The basic idea of domain decomposition is to break up the overall spatial domain into subdomains (subzones) which are assigned to separate computer nodes (Figure 2). In this case, the number of

Figure 2 Grid Distribution and Domain Decomposition in the Global Domain of Interest.

subzones are equal to the number of computer nodes used in the calculations. The size of the subzones must be small enough to fit within the available local memory of a computer node attached to that subzone. Calculations are driven using a global time step. Subzones are connected by means of interzone patches which provide communication between adjacent subzones. This communication between the subzones on different nodes is done by copying the data from the sending subzone to an intermediate array on the same node. This array is then sent to an intermediate array on the receiving node and subsequently used as boundary conditions in the corresponding subzone on the receiving node.

There are many ways to decompose a spatial domain, for example, a strip or a patch decomposition scheme for a two-dimensional domain. Reference 10 has investigated different decomposition schemes and concluded that the patch decomposition scheme works well for the numerical algorithm in the PINCA code.

C. Resource Management

Before modifying any communication procedure, we have reviewed the resource management utility in the code. This relates to the work load of each computer node. One parameter of interest is the number of computational cells per node. Reference 10, a domain decomposition study, has demonstrated that for a given global mesh system, the decomposition scheme with the least number of subdomains will produce the best node-time performance. Node-time is the sum of the elapsed time for all nodes. Usually this node-time is associated with the cost of a simulation. Minimizing the number of subdomains implies maximizing the number of computational cells per computer node. Hence, it is very important to fully utilize each computer node and use a minimum number of computer nodes in the simulation. Since each computer node has limited resources, i.e. limited local memory, determining the size of the local mesh system for fully utilization is not a straight-forward matter. It requires a few iterations, depending on the characteristics of the problem and the code architecture. For example, a chemically reactive gas problem is different from a perfect gas problem because the reactive gas problem models the transport of chemical species. Hence the size of the equation set that must be solved for every cell in the subdomain is bigger for the reactive gas problem than the perfect gas problem.

Two parameters commonly used to measure the parallel performance of a computer program are speedup and efficiency. Speedup is defined as the ratio of the time required to run a particular application on one computer node to that required to run on N computer nodes. Efficiency is defined as:

$$Efficiency = Speedup/N,$$

or the fraction of the maximum possible speedup obtainable with N computer nodes. Amdahl's law relation¹¹ is often used to determine the maximum possible performance of an algorithm. Using Amdahl's relation, the speedup and efficiency are given by

$$Speedup = \frac{N}{p + (1-p)N} \quad (8)$$

$$Efficiency = \frac{1}{p + (1-p)N} \quad (9)$$

where p is the fraction of the calculation that is performed in parallel.

Figure 3 Comparing Efficiency Between 2 Cases. Each with Different Local Mesh System. (Results based on the Previous Version of PINCA).

Figure 4 Comparing Scaled Speedup Between 2 Cases. Each with Different Local Mesh System. (Results based on the Previous Version of PINCA.)

Figure 3 and Figure 4 compare the scaled performance of the earlier version of the PINCA code for two cases. Each case has a different local mesh system, i.e., the number of computational cells per node is different. The case with 60*60 mesh cells per node gives better performance. This result demonstrates that without any modification to the code, a better performance can be achieved by fully utilizing each computer node. Fully utilization implies maximizing the number of computational cells per node. Since the computation time is linearly proportional to the number of computational cells and the communication time is theoretically proportional to the square root of the number of computational cells, maximizing the number of computational cells per node will minimize the communication overhead, that is the ratio of the communication time to the computation time.

D. Communication Overhead

Besides analyzing the resource management in the code, most of our development effort is directed towards improving the node-to-node and global communication. One communication procedure that requires special attention is how the PINCA code calculates and broadcasts the minimum time step size. This procedure has several steps: (1) each computer node calculates its own local minimum time step size within its subdomain, (2) the i/o node collects all the time step information from all nodes, (3) the i/o node determines the global minimum, and (4) the i/o node propagates the global minimum to all nodes. The earlier version of the PINCA code uses sequential message passing calls (do-loop format) to perform this task. Performing the task in this matter is very inefficient (i.e., there are too many wait periods between communications).

Since the objective of this specific communication procedure is to calculate and broadcast the global minimum size of the time step, our first step to improve communication is to investigate a vendor-supplied utility for determining a global minimum. We have implemented this utility into the code (PINCA version 2a) and tested its performance. This utility called GDLOW is used to calculate the global minimum time step. The new PINCA results are encouraging. Incorporating the GDLOW utility does save substantial communication time. However, using this utility has a drawback; the number of computer nodes used in the simulation is restricted to the order of 2^n power for the Intel Gamma and to the factor of 2 for the Intel Paragon. Currently we are investigating other techniques to globally collect and broadcast the minimum time step information.

E. Problem Mapping

Problem mapping addresses the way to distribute the computational subdomains into the computer nodes. An effective guideline is to have the neighboring subdomains located in the neighboring computer nodes to minimize the routing and the communication time. Hence mapping depends on the architecture and the communication design of the parallel computer. It could become important and affect the time performance if two neighboring subdomains are located in different nodes physically far apart in the computer system. The Intel/Gamma has a hypercube topology and the Intel/Paragon has a two-dimensional (2D) mesh topology.

A MP computer with the hypercube topology uses cut-through routing for a node-to-node communication. Sometimes the latency time for communication may increase because the direct memory access communication channels can be blocked. Thus, it is better to utilize a gray coding for a hypercube parallel computer to enforce nearest neighbor communication to take advantage of the hypercube topology¹¹. For a similar reason, it is recommended to use a periodic node assignment for the 2D-mesh topology¹². Figure 5 compares the layout

					<i>node number</i>
					<i>numeric</i>
					<i>binary</i>
11	13 1100	14 1101	15 1110	16 1111	
10	9 1000	10 1001	11 1010	12 1011	
01	5 0100	6 0101	7 0110	8 0111	
00	1 0000	2 0001	3 0010	4 0011	
	00	01	10	11	

(a) Simple Node Assignment.

10	9 1000	10 1001	12 1011	11 1010
11	13 1100	14 1101	16 1111	15 1110
01	5 0100	6 0101	8 0111	7 0110
00	1 0000	2 0001	4 0011	3 0010
	00	01	11	10

(b) Gray Coded Node Assignment.

	16	15	14	13
11	1111	1110	1011	1010
10	9	10	11	12
	1000	1001	1010	1011
01	8	7	6	5
	0111	0110	0101	0100
00	1	2	3	4
	0000	0001	0010	0011
	00	01	10	11

(c) Periodic Node Assignment.

Figure 5 Layout of Three Different Mapping Topologies.

of these three topologies: simple node assignment, gray coding for a hypercube architecture, and periodic node assignment for a 2D-mesh architecture.

IV. Case Study I - Perfect Gas Problem

By incorporating those improvements as discussed in previous sections and adopting an efficient MP computing strategy, we apply and evaluate the PINCA code for two different fluid flow problems. The first case for the scaled performance study is a simulation of hypersonic flow over a generic body¹⁴ (Figure 6). In this problem, a

Figure 6 Pressure and Velocity Distributions of a Hypersonic Flow Over a Generic Body.
($P_\infty=461\text{N/m}^2$, $T_\infty=241\text{K}$, $u_\infty=6.2\text{km/s}$, perfect gas)

perfect gas model is assumed. Figure 7 and Figure 8 show the scaled performance of the modified PINCA code (version 2a) for an increasing number of computer

nodes. The modified code has achieved at least 75.2% efficiency with 64 computer nodes in the Intel/Paragon. This is much better than the previous results as shown in Figure 3. Hence, modifying the communication procedure and adapting a MP computing strategy substantially enhance the performance of the code.

Figure 7 Efficiency Versus Number of Computer Nodes to Illustrate the Scaled Performance of PINCA version 2a for a Perfect Gas Problem.

However, when more computer nodes are being utilized, the code performance especially for the Intel/Paragon with SUNMOS and the Intel/Gamma are far from perfect. Node-to-node communication begins to dominate the total elapsed time. This implies that the code has not been fully optimized nor fully utilized for the perfect gas problem. One area that will improve the par-

Figure 8 Scaled Speedup Versus Number of Computer Nodes to Illustrate the Scaled Performance of PINCA version 2a for a Perfect Gas Problem.

allel performance is to increase the number of mesh cells per node to fully utilize each computer node.

Figure 7 and Figure 8 also indicate a difference in the scaled performance of the PINCA code between two massively parallel computers. For most cases, running PINCA on the Intel/Paragon achieves a better performance than running on the Intel/Gamma. This difference may be caused by two factors: (1) a simple node assignment is used for the problem mapping scheme in the present study and (2) the computer architecture of these two MP computers are substantially different. The Intel/Paragon has a two-dimensional (2D) mesh topology while the Intel/Gamma has a hypercube topology. To enhance a better communication performance, it is important to enforce the nearest neighbor mapping strategy. Since PINCA uses a simple node assignment (Figure 5), this scheme is closer to the periodic scheme than the gray coding scheme. Hence the code performs better on the Intel/Paragon than on the Intel/Gamma.

Figure 9 compares the elapsed time performance between different supercomputers, Intel/Paragon, Intel/Gamma and Cray-YMP. As illustrated in the figure,

Figure 9 Comparison of Performance Between Different Supercomputers. (A Perfect Gas Problem).

when the problem size increases, it takes longer for a vectorized computer such as Cray-YMP or C-90 to obtain a solution because only 1 processor is being used. Since MP computers have multiple processors, as the problem size increases, engineers can utilize more computer nodes for simulation. Thus the total elapsed time does not change significantly and is relatively independent of the problem size. Hence the modified version of PINCA scales reasonably well with an increasing number of computer nodes. If the perfect gas problem has more 100,000 mesh cells, the Intel/Paragon with SUNMOS performs better than the Cray-YMP. This figure also reveals an important point in computational flow

simulation: MP computing really only becomes effectively for large-scale simulations. If the problem is relatively small, it is better to use a vectorized computer because the single processor speed in the Cray supercomputer is faster than in the Intel MP computers.

V. Case Study II - Chemically Reactive Gas Problem

Shock induced combustion phenomena, ranging from decoupled shock-deflagration systems to overdriven oblique detonation waves, were experimentally investigated in the mid 1960's and early 1970's. Experiments primarily consisted of firing spheres and other projectiles through explosive mixtures such as H_2 and Air. Calculations for this type of chemically reactive flow are very demanding in terms of numerical robustness and accuracy, since the reactions usually occur very fast with significant energy release that takes place in a very short distance.

This test case involves a supersonic premixed stoichiometric hydrogen-air mixture which is ignited by a bow-shock wave¹⁵. The combustion model includes 6 reacting species (H_2 , O_2 , H_2O , H , O , and OH) and 1 assumed inert species (N_2) which undergo 8 reaction steps. The results of PINCA simulation of this reactive flow problem are shown in Figure 10.

Figure 10 Temperature Contours of a Decoupled Shock-Deflagration System. ($P_\infty=42662$ N/m, $T_\infty=250K$, $u_\infty=1.685km/s$, reactive gas).

The parallel performance for this chemically reactive gas problem is better than the perfect gas problem. The efficiency increases to 86.5% from 70% on the Intel/

Figure 11 Efficiency Versus Number of Computer Nodes to Illustrate the Scaled Performance of PINCA version 2a for a Chemically Reactive Gas Problem. (7 species, 8 reactions).

Gamma computer and to 96.7% from 75.2% on the Intel/Paragon computer (Figure 11).

Figure 12 shows the scaled speedup of the chemically reactive gas problem. Both the efficiency and scaled speedup results demonstrates that the PINCA code can achieve a reasonable good scaled speedup. This implies that the code would scale well for a large-scale chemically reactive gas problem.

The improvement in parallel performance occurs because (1) the reactive gas problem is more computational intensive than the perfect gas problem. Note that the total elapsed time is the sum of computing time and communication. If the communication time is relatively

Figure 12 Scaled Speedup Versus Number of Computer Nodes to Illustrate the Scaled Performance of PINCA version 2a for a Chemically Reactive Gas Problem. (7 species, 8 reactions).

Figure 13 Comparison of Performance Between Different Supercomputers. (A Chemically Reactive Gas Problem; 7 Species and 8 Reactions are Modeled.)

small compared to the computing time, an insignificant increase in communication time may not affect the overall performance much. (2) Each computer node is probably fully utilized for the chemically reactive gas problem while it is not fully utilized for the perfect gas problem. In this study, the local mesh setup per node for both problems is the same. However, in the reactive gas problem, 7 species are modeled. This increases the complexity of the problem that requires additional memory and computing work. This will easily utilize the full capacity of each node.

Figure 13 compares the performance between different supercomputers. The results for the reactive gas problem is similar to the results for the perfect gas problem. As the problem size is increasing, with MP computers, engineers can utilize more computer nodes to simulate the problem. Thus the elapsed time to solve the problem is relatively constant provided the code is scalable. PINCA can deliver this scalable performance. However for a supercomputer with only one processor like Cray-YMP, the elapse time will increase. With multiple processors, for this reactive flow problem with almost half a million computational cells, the performance of the Intel/Paragon is five times faster than the Cray-YMP. In summary, our results has demonstrated that very high parallel performance can be achieved for a chemical reactive gas problem.

VI. Conclusions and Recommendations

To accurately and efficiently simulate complex three-dimensional compressible flow problems with nonequilibrium chemistry, a supercomputer with an enormous memory capacity and a super computing speed is need-

ed. Massively parallel computers satisfy these requirements. In order to have a computer code execute efficiently in these MP computers with a large number of computer nodes, the code has to be optimized and scalable. The scalability issue is also very important to determine the maximum problem size that a MP computer can handle.

The development and testing of a scalable, parallel version of a chemically reactive flow solver is a challenging task. This paper presents our experience based on implementing a massively parallel computing strategy and developing an optimal parallel algorithm to achieve a good scaled performance. The areas being addressed are: domain decomposition, memory resource management, communication overhead, and problem mapping. By incorporating a better communication procedure and adopting an efficient MP computing strategy which fully utilizes each computer node, an efficiency of more than 96% can be achieved using 128 computer nodes for a chemically reactive gas problems (Intel/Paragon).

A comparison of performance between MP computers and vectorized computers such as Cray-YMP, has also been made. Our results show that the MP computers are very effective for use in large-scale simulations, while vectorized computers, like a Cray-YMP, are better for small-scale simulations. For a chemically reactive gas problem with almost half a million computational cells, the performance of the Intel/Paragon is five times faster than the Cray-YMP.

VII. References

1. PINCA: A Massively Parallel Version of the Integrated Code Analysis, a joint-development project between Sandia National Laboratories, Albuquerque, NM, and Amtec Engineering Inc., Bellevue, WA, August 1993.
2. Wong, C. C., Soetrisno, M., Blotner, F. G., Payne, J. L., and Imlay, S., "PINCA: A Scalable Parallel Program for Compressible Gas Dynamics with Nonequilibrium Chemistry," SAND94-2436, Sandia National Laboratories, Albuquerque, New Mexico, January 1995.
3. INCA Version 1.1, Amtec Engineering Inc., Bellevue, Washington, January 1993.
4. Steger, J. L., and Warming, R. F., "Flux Vector Splitting of the Inviscid Gasdynamic Equations with Applications to Finite Difference Method," J. of Computational Physics, Vol. 40, p. 263-293, 1981.
5. Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," J. of Computation Physics, Vol. 43, 1981, p. 357-372.
6. Yee, H. C., Warming, R. F., Harten, A., "Implicit Total Variation Diminishing (TVD) Schemes for Steady State Calculations," J. of Computational Physics, Vol. 57, p. 327-360, 1985.
7. Eberhardt, S., and Imlay, S., "Diagonal Implicit Scheme for Computing Flows with Finite Rate Chemistry," J. of Thermophysics and Heat Transfer, Vol. 6, No. 2, April-June 1992, pp. 208-216.
8. Wong, C. C., et al., "A Domain Decomposition Study for Massively Parallel Computing in Gas Dynamics," Fluid Mechanics Working Group Meeting, Sandia National Laboratories, Albuquerque, New Mexico, February 17, 1994.
9. Keyes, D., "Domain Decomposition: A Bridge Between Nature and Parallel Computers," in ASME AMD-Vol. 157: Adaptive, Multilevel, and Hierarchical Computational Strategies, presented at the Winter Annual Meeting of ASME, Anaheim, CA, Nov. 8-13, 1992, pp. 293-334.
10. Wong, C. C., et al., "A Domain Decomposition Study for Massively Parallel Computing in Compressible Gas Dynamics," AIAA-95-0572, to be presented at the 33rd Aerospace Sciences Meeting, Reno, Nevada, January 9-12, 1995.
11. Amdahl, G., "Validity of the Single-Processor Approach to Achieving Large-Scale Computer Capabilities," AFIPS Conference Proceeding, 30, 1967, p. 483-485.
12. Womble, D. E., et al., "A Tutorial on nCUBE Programming," SAND91-2903C, Sandia National Laboratories, Albuquerque, New Mexico, March 1992.
13. Workshop on Massively Parallel Computer Programming, private communication with D. E. Cline, Sandia National Laboratories, Albuquerque, New Mexico, October 1992.
14. Wong, C. C., et al., "A Maneuvering Vehicle for the Mars Environmental Survey (MESUR) Project," SAND93-0078, Sandia National Laboratories, Albuquerque, New Mexico, March, 1993.
15. Lehr, H. F., "Experiments on Shock-Induced Combustion," Astronautica Acta, Vol. 17, 1972, p. 589-596.