JOURNAL OF THERMOPHYSICS AND HEAT TRANSFER
Vol. 6, No. 4, Oct.–Dec. 1992

Three-Dimensional Direct Particle Simulation on the Connection Machine

Leonardo Dagum*
NASA Ames Research Center, Moffett Field, California 94035

This paper presents the algorithms necessary for an efficient data parallel implementation of a three-dimensional particle simulation. A general master/slave algorithm and a fast sorting algorithm are described, and the use of these algorithms in a particle simulation is outlined. A particle simulation using these algorithms has been implemented on a 32768 processor connection machine that is capable of simulating over 30 million particles at an average rate of 2.4 μs/particle/step. Results are presented from the simulation of flow over an aeroassisted flight experiment (AFE) geometry at 100 km alt.

I. Introduction

INTEREST in analyzing hypersonic flows has led to great activity in the field of direct particle simulation. Much of the current work has focused on designing algorithms targeted specifically to vector architectures in an attempt to improve the performance of this method on vector machines.[1–4] The results from this effort have been rewarding, and it can be said that the algorithms for implementing a fully vectorized particle simulation are well established. Not as well established, however, are the algorithms necessary for an efficient data parallel implementation of a particle simulation. The current trend towards massively parallel architectures makes the investigation of parallel algorithms for direct simulation both appropriate and timely.

This paper describes the data parallel algorithms necessary for the efficient implementation of a three-dimensional particle simulation on the connection machine. Results from a flow simulation for the aeroassisted flight experiment (AFE) geometry are presented.

II. Connection Machine Architecture

The thinking machines connection machine model CM-2 is a massively parallel single-instruction multiple-data (SIMD) computer consisting of many thousands of bit serial data processors under the direction of a front-end computer. The system at NASA Ames consists of 32,768 bit serial processors, each with 1 Mbit of memory and operating at 7 MHz. The processors and memory are packaged as 16 to a chip. Each chip also contains the routing circuitry that allows any processor to send and receive messages from any other processor in the system. In addition, there are 1024 64-bit Weitek floating point processors which are fed from the bit serial processors through a special purpose "sprint" chip. There is one sprint chip connecting every two CM chips to a Weitek. Each Weitek processor can execute an add and multiply each clock cycle, thus, performing at 14 Mflop/s, and yielding a peak aggregate performance of 14 Gflop/s for the system.

The connection machine can be viewed two ways, either as an eleven-dimensional hypercube connecting the 2048 CM chips, or a ten-dimensional hypercube connecting the 1024 Weitek processing elements. The first view is the "fieldwise" model of the machine that has existed since its introduction. This view admits to the existence of at least 32,768 physical processors (when using the whole machine) each storing data in fields within its local memory. The second is the more recent "slicewise" model of the machine that admits to only 1024 processing elements (when using the whole machine) each storing data in slices of 32 bits distributed across the 32 physical processors in the processing element. Both models allow for "virtual processing," where the resources of a single processor or processing element may be divided to allow a greater number of virtual processors (VP).

Regardless of the machine model, the architecture allows interprocessor communication to proceed in three manners. For very general communication with no regular pattern, the router determines the destination of messages at run time and directs the messages accordingly. This is referred to as general router communication. For communication with an irregular, but static pattern, the message paths may be precompiled and the router will direct messages according to the precompiled paths. This is referred to as compiled communication and is faster than general router communication. Finally, for communication that is perfectly regular and involves only shifts along grid axes, the system software optimizes the data layout by ensuring strictly nearest-neighbor communication and uses its own precompiled paths. This is referred to as NEWS (for NorthEastWestSouth) communication. Despite the name, NEWS communication is not restricted to two-dimensional grids, and up to thirty-one-dimensional NEWS grids may be specified. When processors are prescribed by a NEWS grid, it is possible to efficiently perform "scan" operations[5] across a grid axis. A scan operation with binary operator $\oplus$ across an ordered set $[a_0, a_1, \ldots, a_{n-1}]$ returns the ordered set $[a_0, (a_0 \oplus a_1), \ldots, (a_0 \oplus a_1 \oplus \cdots \oplus a_{n-1})]$.

III. Implementation Details

Because of the SIMD nature of the connection machine, it is difficult to efficiently implement either the "time counter"[6] or the "no time counter"[7,8] collision selection rules of Bird's direct simulation Monte Carlo (DSMC) method. For this reason Baganoff and McDonald's[1] collision selection rule has been applied, and the algorithms of the Stanford particle simulation method have been implemented.

A single time step in the particle simulation can be considered comprised of six events 1) collisionless motion of particles; 2) enforcement of boundary conditions; 3) sorting of particles into cells; 4) pairing of collision partners; 5) collision of selected collision partners; and 6) sampling for macroscopic flow quantities. Detailed description of these algorithms may be found in Refs. 9 and 10.

In the implementation, one virtual processor is assigned to each particle, therefore, each physical processor is simulating

Presented as Paper 91-1365 at the AIAA 26th Thermophysics Conference, Honolulu, HI, June 24–26, 1991; received Aug. 12, 1991; revision received Nov. 18, 1991; accepted for publication Nov. 18, 1991.
*Research Scientist. Member AIAA.

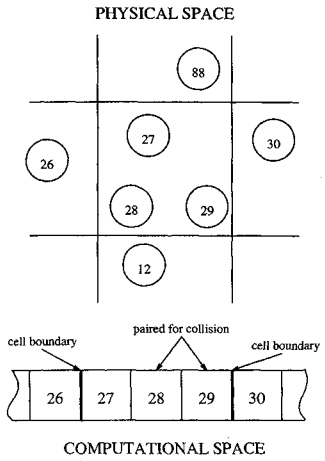

Fig. 1 Mapping of particle data to virtual processors in the connection machine. Each particle in physical space is represented by one virtual processor in computational space.

a fixed and identical number of particles. Figure 1 illustrates the parallel decomposition of the problem. The set of virtual processors (VP-set) storing the particle data is referred to as the "main" VP-set. Several other VP-sets are used in the simulation and all the VP-sets are one-dimensional NEWS grids. Note in Fig. 1 that particles occupying the same cell in physical space are represented by neighboring processors in computational space. When the particles are ordered in this manner, cell boundaries may be quickly determined by having every processor compare its particle's cell index with that of its neighboring processor. Cell boundaries are often used to delimit segments in scan operations. This convenient ordering is lost when particles move into different cells and the order must be restored by sorting. With the particle data sorted, it becomes a simple matter to match pairs for collision, as well as to sample for macroscopic flow quantities. The algorithms used to implement the six events comprising a time step are described below.

A. Enforcing Boundary Conditions

The Stanford particle simulation method currently employs a grid of cubic cells for deciding on collisions and sampling macroscopic flow quantities. Arbitrary three-dimensional geometries are defined in the grid of cells in the manner described in Ref. 9. The geometry information is stored in a distinct VP-set, referred to as the "geometry" VP-set, and is accessed through a master/slave scheme similar to that described in Ref. 10.

The master/slave scheme is a way of allowing a good load balance to persist in parts of the calculation that do not involve a large fraction of the particles; the enforcement of boundary conditions is such an example. Typically, only a small number of particles interact with the body on any given time step (for example, in the application discussed below, less than 4% of the total number of particles need to be considered for possible surface interaction). Since the work for each particle is being carried out by an individual virtual processor, a straightforward application of the boundary conditions would result in a large number of VP's remaining idle, with only a small number of VP's actually involved in the calculation. However, assume for now that, without accessing the geometry VP-set, one knows which particles must be considered for possible surface interaction. In such a case, it is profitable to carry out the surface interaction with a master/slave algorithm. The idea is to allocate a "slave" VP-set large enough to accommodate the active processors in the "master" VP-set, and then carry out the calculation from this smaller slave VP-set. There is an initial penalty in communicating the necessary data from the master to the slave, but this is offset by the faster execution of instructions in the smaller VP-set. Greater detail on this algorithm may be found in Ref. 11.

In order to effectively use the master/slave algorithm for enforcing boundary conditions, it is necessary to somehow identify which particles have entered so-called "geometry" cells. These are cells for which a boundary is defined in the geometry VP-set. It is not practical to let the processors in the main VP-set directly access the data in the geometry VP-set because of the enormous communication cost it would entail. Instead, for every cell in the simulation, there is stored a 28-bit descriptor of the region local to the cell. The first 27 bits of the descriptor map the cell itself and its 26 immediate neighbors, the last bit maps the region comprised by the 98 cells that are 2 away. If a bit in the descriptor has value 1, then the region it maps (either a neighboring cell or group of cells) includes a geometry cell. Conversely, if a bit has value 0 then the region it maps does not include a geometry cell.

The local region descriptors are stored in a separate VP-set that will be referred to as the "space" VP-set. The space VP-set also stores, for every cell, a pointer to the appropriate address in the geometry VP-set. At the beginning of every time step, descriptors are broadcast to every processor in the main VP-set as follows:

1) In the main VP-set, identify one processor for every occupied cell.

2) Send the processor's self-address to the appropriate processor in the space VP-set.

3) Processors in the space VP-set that receive a message from the main VP-set, return the local region descriptor.

4) The local region descriptors received by the processors active in step 1 are copied with a scan operation to the other processors, representing particles in the same cell.

Note that at the beginning of the time step the particle data in the main VP-set is sorted, therefore, steps 1 and 4 can be carried out using NEWS communication.

Processors in the main VP-set can now update their particle's position. At the end of the free motion, each particle's cell position is computed and its local region descriptor is consulted. Those particles, which may need to interact with a body surface, are identified and the master/slave algorithm is applied. The slave processors access the space VP-set to get pointers into the geometry VP-set from which they can then retrieve the geometry data. Because it is extremely unlikely that a particle will move more than two cell widths in one time step (see Refs. 9 and 10), the local region descriptor maps neighboring cells no further than two away.

B. Sorting Particles and Pairing Collision Partners

When the particle data is sorted, neighboring virtual processors in the main VP-set will store data for particles occupying the same cell (except, of course, at cell boundaries). Collision partners then may be paired on an even-with-odd basis, i.e., all even-numbered particles are considered for collision with their odd-numbered neighbor (see Fig. 1). Naturally, if the odd-numbered neighbor is occupying a different cell, then the two particles are not allowed to collide.

At the beginning of a time step the particle data is in an ordered state, allowing easy access to the information for particles occupying the same cell. However, diffusion and convection of the particles through a time step destroys the order, and it is necessary to sort the particles on every time step. Sorting is a communication-intensive process and can be very costly on a parallel architecture. However, knowledge of the mechanism behind the disordering can be used to greatly reduce this cost, and Ref. 11 describes a very efficient sorting algorithm for particle simulations.

Three fundamental observations on the mechanism behind the disordering may be used to reduce the problem of sorting to one of merging. In particular, one may observe:

1) At the beginning of a time step, the particle data is ordered, and becomes disordered only through the motion of the particles.

2) The range of motion of a particle over one time step is, to a very high probability, limited to less than two cell widths.

3) Most particles do not change cells over one time step. Furthermore, of those particles that do change cells, the majority make the same cell change.

The first observation indicates that the data is never very far out of order. Therefore, it is reasonable to expect that a specially tailored sorting algorithm may be substantially faster than a general one. This observation implies that disorder occurs when particles change cells. Trivially, if particles did not change cells then they would remain sorted, but also if all particles made the identical cell change then they would still remain sorted. All particles that undergo the same cell change represent an ordered set, and one can divide the particles into mutually exclusive sets based on their cell change. The problem of sorting becomes one of identifying and merging these ordered sets. The next two observations aid in that respect.

The second observation indicates that there is a limited number of ordered sets to be identified. If particles move no more than two cell widths in a time step, then there can only be 125 possible cell changes and no more than 125 ordered sets to be found. In practice, many of these will be empty sets, and typically, only about 25 nonempty ordered sets will be found in the data. One can make use of the third observation to further reduce the number of sets to be merged. This observation indicates that most of the particles will belong to just two sets so that a master/slave algorithm may be used effectively in sorting the remaining particles. The algorithm is referred to as "flux-sort" because it operates on the principle that particles constituting a particular flux retain their order after motion. The algorithm proceeds as follows:

1) Identify the two largest ordered sets of particles and enumerate them.

2) Sort the remaining particles using a master/slave algorithm.

3) Merge the three ordered sets.

The enumeration in step 1 restarts at every cell boundary. The sorting of step 2 is very fast because only a small number of particles need to be sorted. The merging used for step 3 is described in detail in Ref. 11. It is of interest to note that the scan operations used in flux-sort are vectorizable (see Ref. 12), and flux-sort is a vectorizable algorithm. However, it is doubtful whether flux-sort would outperform bucket sorting (which is not vectorizable) on a vector architecture, since the operation count in flux-sort is much greater. Finally, it is important to note that flux-sort scales linearly with the number of particles in the simulation.

C. Statistical Independence

For the simulation to accurately reflect the expected collision frequency in a gas, it is necessary that the set of pairings considered for collision be statistically independent between time steps. A straightforward approach toward ensuring such independence would be to select pairs randomly from a cell. Unfortunately, such a scheme does not readily lend itself to implementation on a SIMD architecture like the connection machine. The alternative approach is to employ a regular pairing (for example, the even-with-odd pairing used here) but apply a shuffling of the in-cell ordering prior to the pairing. The regular pairing allows a regular communication pattern that leads to simple and efficient implementation. The difficulty then is to find efficient algorithms for shuffling the in-cell ordering. Two different shuffling algorithms are employed every time step and these are described in Ref. 11.

D. Colliding Particles

Collision mechanics are implemented as described in Ref. 10. Currently, only single specie monatomic or diatomic gases may be simulated. Thermal nonequilibrium between translational and rotational modes is handled according to the Borgnakke-Larsen model described in Ref. 9. Vibrational relaxation employs the discrete model also described in Ref. 9. Collision numbers are fixed at 5 for rotation and 50 for vibration. Eventually multiple reacting species will be allowed and it is expected that the vibration and chemistry models described in Ref. 13 will be implemented.

E. Sampling Macroscopic Flow Quantities

For diatomic gases, 7 macroscopic quantities are sampled, these include density, velocity, and three temperatures; these quantities are computed in the main VP-set for every occupied cell. Because the particles are sorted, scan operations may be used to compute the necessary averages. The results are stored in a separate "sampling" VP-set. A master/slave algorithm (with the sampling VP-set acting as the slave) is employed to transfer the sampled quantities from the main VP-set. Samples are collected every time step once steady state has been reached.

IV. Validation

Validation of the implemented algorithms was undertaken through test calculations for thermal relaxation of a gas, shock wave profiles, and shear-stress profiles in Couette flow. Figure 2 presents sample results from a shock-wave simulation. The normalized density and temperature profiles for a Mach 10 shock wave in a perfect diatomic gas as computed by McDonald[9] are given by the solid curves. Position along the profile is normalized by the mean free path before the shock. The symbols represent results obtained with the connection machine implementation described here. The shock tube was three-dimensional with size $60 \times 4 \times 4$.

The object in making this, and other shock wave calculations, was to validate the shuffling algorithms for the three-dimensional simulation. Any statistical dependence between pairings made on subsequent time steps results in an incorrect collision frequency. Typically, this will widen the shock profiles and prevent thermal equilibrium from being reached behind the shock. These effects were not observed and the shock wave results compared favorably to accepted solutions. Results from thermal relaxation simulations are given in Ref. 10; results from Couette flow simulations are given in Ref. 14.

V. Calculations

To demonstrate the capabilities of the method, the results from a large scale three-dimensional simulation for the hypersonic flow about the AFE vehicle are presented. The geometry is identical to that described in Ref. 15. The freestream conditions corresponded to flight at 100 km alt where the mean free path is 10 cm and the temperature is 194 K. The Knudsen number based on the 4.25 m aeroshell diameter is 0.0235.

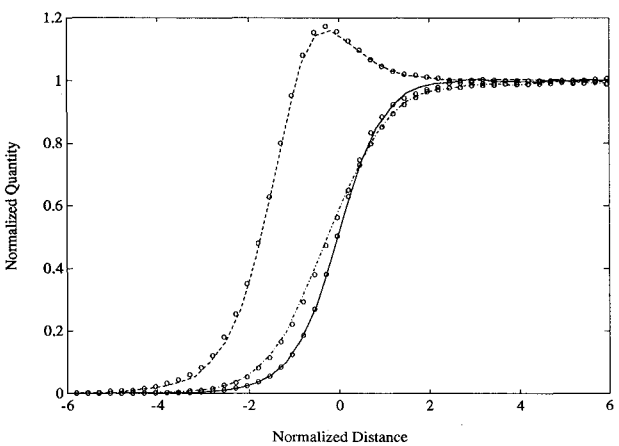

Fig. 2 Normalized density and temperature profiles across a Mach 10 shock wave in a perfect diatomic gas: — ρ; – – – T_{tr}; –.–.–. T_{rot}.

The simulation geometry had a diameter of 44 cell widths, and the hard sphere freestream mean free path was set at 1.035 cell widths in order to match Knudsen numbers. For the assumed inverse ninth power law potential, this required setting the simulation mean free path to 0.854 as described in Ref. 16. The freestream Mach number was 35.42 corresponding to a vehicle velocity of 9.9 km/s. The surface temperature was fixed at 1500 K. The simulated particles corresponded to molecular nitrogen with a characteristic temperature for vibration of 3390 K.

The geometry was placed in a wind tunnel with dimensions of 55 × 120 × 60 and the simulation was started with just 10^6 particles. The procedure used to reach steady state with 32 million particles was: run 700 steps, clone, run 300 steps, clone, run 10 steps, clone, run 10 steps, clone, run 10 steps, clone, run 50 steps. Time averaging was then carried out for 150 steps with samples collected on every step. Note that some number of time steps were taken after each cloning in order to reduce the statistical dependence created by cloning.

Figure 3 presents the temperatures along the stagnation streamline normalized by the freestream temperature. Unfortunately, these results cannot be compared to the Cray implementation results presented in Ref. 15 because of a known error in boundary conditions used in the Cray implementation. The peak normalized translational temperature of 266 corresponds to a 9% overshoot on the Rankine-Hugoniot jump value of 245 for this Mach number in a perfect diatomic gas. The rotational and vibrational temperatures lag behind the translational temperature with peak values of 84 and 20, respectively. With a collision number of 50, vibrational modes

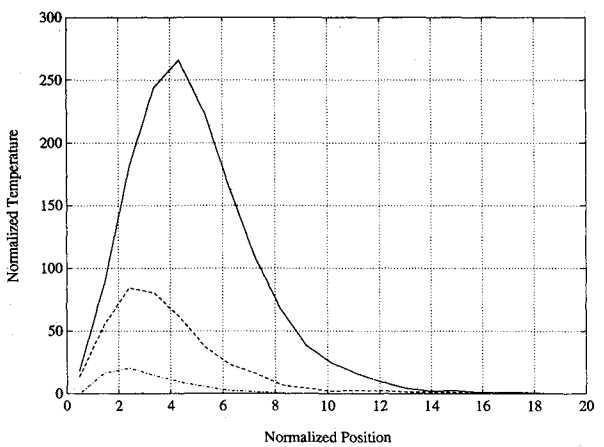

Fig. 3 Normalized temperatures along the stagnation streamline: — T_{tr}; - - - T_{rot}; -.-.-. T_{vib}. Position is normalized by the freestream mean free path.

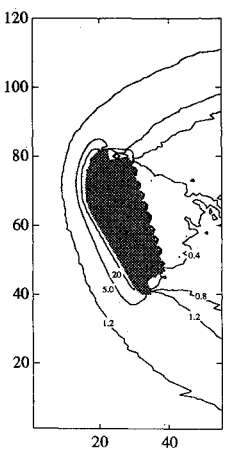

Fig. 4 Density contours in symmetry plane for flow over AFE. Contours shown at 0.4, 0.8, 1.2, 5.0, and 20 times the freestream density.

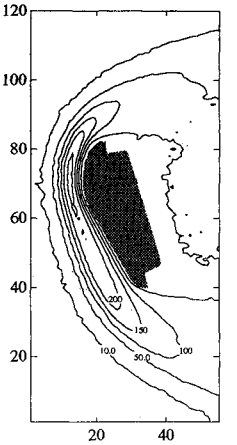

Fig. 5 Translational temperature contours in symmetry plane for flow over AFE. Contours shown at 10, 50, 100, 150, 200, and 250 times the freestream temperature.

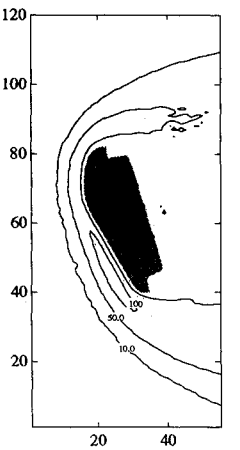

Fig. 6 Rotational temperature contours in symmetry plane for flow over AFE. Contours shown at 10, 50, and 100, times the freestream temperature.

are comparatively slower to activate, and greater vibrational temperatures are found downstream of the stagnation region.

Figures 4–7 present density and temperature contours in the symmetry plane of the vehicle. The figures were generated from a single plane of the solution. (Note that several planes could have been averaged to produce smoother contours, but part of the purpose in presenting these results is to demonstrate the quality of solution that can be obtained with 32 million particles in 150 time steps.) In Fig. 4 one can see a sharp rise in density through the boundary layer due to the relatively cold body surface. The accuracy of the simulation in this dense region of the flow is questionable, given that the mean free path in the free stream was only 0.86 cell widths, and a uniform cartesian grid was employed. Since the cell dimension fixes the sharpest gradient that can be captured by the simulation (where the local mean free path is much smaller than a cell dimension) it becomes impossible to accurately resolve gradients. Eventually, varying cell sizes will be supported so that the local mean free path and the cell dimension may be of comparable size everywhere in the flow. It is expected that varying cell sizes will be implemented over the uniform cartesian grid currently employed as cartesian grids have proved to be very computationally efficient for particle simulation.[2,9,10]

Comparing the density and translational temperature contours (Figs. 4 and 5), one can see the greater standoff distance for the latter. The peak translational temperature occurs before the body, and there is a sharp drop in temperature on approaching the body, because of the cold surface tempera-

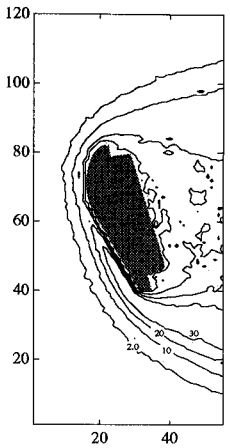

Fig. 7 Vibrational temperature contours in symmetry plane for flow over AFE. Contours shown at 2.0, 10, 20, and 30 times the freestream temperature.

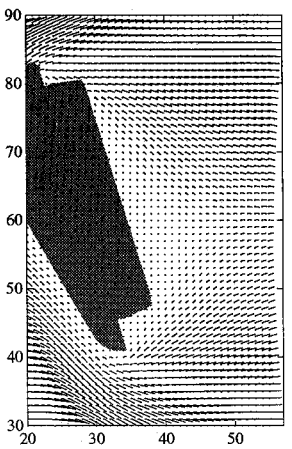

Fig. 8 Velocity field in symmetry plane around the AFE.

ture. The rotational temperature (see Fig. 6) reaches its peak downstream from the stagnation region. This is a result of the greater number of collisions required to thermalize rotational modes in the gas. Vibrational modes are even slower to thermalize, and the peak vibrational temperature is not reached until well downstream of the stagnation region (see Fig. 7). The subsequent expansion of the flow around the shoulder of the vehicle lowers the collision rate and the activated vibrational modes are frozen into the wake. Figure 8 presents the velocity field in the symmetry plane about the vehicle. The turning of the flow about the body, and the stagnant region directly behind the afterbody, are clearly evident.

VI. Performance

The code used for the calculation described in the previous section was written completely in C/Paris and run on all 32k processors of the connection machine at the NASA Ames Research Center using a Sun 4/90 as the front end. The average time to advance one particle over one time step at steady state was 2.4 μs and the entire calculation was completed in less than 6 h. This performance is comparable to that obtained from a fully vectorized implementation running on a single CPU of a Cray 2. However, the greater amount of memory available on the CM allows a greater number of particles to be simulated (specifically, 32 million particles on a 32k processor CM, compared to 20 million particles on a Cray 2 with 2 GB of memory). Both the execution time and memory requirements of the code have been shown to scale linearly with the number of processors up to 32k (see Ref. 17), and it is expected that a fully configured 64k processor CM would be capable of simulating 64 million particles at an average rate of 1.2 μs/particle/time step. This performance is comparable to that of a single CPU of a Cray YMP.[17]

Table 1 Distribution of computational time for particle simulation

Move	20%
Sort	38%
Reorder	26%
Collide	10%
Sample	6%

The distribution of computational time in the code is given in Table 1. The fraction of time listed for sorting includes the time to arrive at the rank for each particle, and the time to shuffle the in-cell ordering. The "reorder" time is simply the time to permute the particle data into its ordered arrangement. It is this reorder time that makes the "collide" and "sample" times so low, since once the data is reordered, the memory references for these two events are primarily local or nearest-neighbor. In a sequential implementation the reordering time would not appear explicitly in this manner, but would essentially be accounted for in terms of memory references in the collide and sample events.

VII. Discussion and Conclusions

The applicability of the current work to hypervelocity flow is somewhat limited since chemistry has not yet been incorporated. However, although the sample calculation of the previous section is physically meaningless, the aim in presenting those results is not to provide an accurate description of the real flow about the AFE, but rather to demonstrate the capacity of the connection machine to execute large scale particle simulations. It is expected that extending this work to include multiple reacting species will not require defining any new data parallel algorithms, but may be accomplished by applying the algorithms outlined in the first part of this paper.

The results presented in this article give some indication of the possibilities open to particle simulation through distributed memory architectures. The low computational cost per simulated particle of the Stanford method makes memory the bounding resource as opposed to CPU time. In this respect, distributed memory architectures are ideally suited to particle simulation, since they are most capable of providing extremely large memory capacities.

An important result of this investigation is a quantitative measure of the performance of the connection machine for direct particle simulation. From this result it is possible to conclude that the massively parallel architecture of the connection machine is quite suitable for this type of calculation, with performance comparable to that of a single processor Cray 2. The main advantage of the connection machine is a large memory that allows the simulation of over 30 million particles. However, there are difficulties in taking full advantage of this architecture because of the lack of a broad-based tradition of data parallel programming. An important outcome of this work is new data parallel algorithms, specifically for use in direct particle simulation, but that also expand the data parallel diction.

Acknowledgments

The author thanks Jeff McDonald for supplying shock profiles for comparison and Bill Feiereisen for supplying the AFE geometry. This work was sponsored under NASA contract NAS 2-12961.

References

[1]Baganoff, D., and McDonald, J. D., "A Collision-Selection Rule for a Particle Simulation Method Suited to Vector Computers," *Physics of Fluids A*, Vol. 2, No. 7, 1990, pp. 1248–1259.

[2]Boyd, I. D., "Vectorization of a Monte Carlo Scheme for Non-

equilibrium Gas Dynamics," *Journal of Computational Physics*, Vol. 96, No. 2, 1991, pp. 411–427.

[3]Ploss, H., "On Simulation Methods for Solving the Boltzmann Equation," *Computing*, Vol. 38, 1987, pp. 101–115.

[4]Pryor, D. V., and Burns, P. J., "Vectorized Monte Carlo Molecular Aerodynamics Simulation of the Rayleigh Problem," *Journal of Supercomputing*, Vol. 3, No. 4, 1989, pp. 305–330.

[5]The Connection Machine System: Paris Reference Manual Version 5.0, Thinking Machines Corp., Cambridge, MA, 1989.

[6]Bird, G. A., *Molecular Gas Dynamics*, 1st ed., Clarendon Press, Oxford, England, UK, 1976.

[7]Hermina, W. L., "Monte Carlo Simulation of High Altitude Rocket Plumes with Nonequilibrium Molecular Energy Exchange," AIAA Paper 86-1318, June 1986.

[8]Bird, G. A., "Perception of Numerical Methods in Rarefied Gas Dynamics," *Rarefied Gas Dynamics*, edited by E. D. Muntz, D. P. Weaver, and D. H. Campbell, Vol. 118, Progress in Astronautics and Aeronautics, AIAA, Washington, DC, 1989, pp. 211–226.

[9]McDonald, J. D., "A Computationally Efficient Particle Simulation Method Suited to Vector Computer Architectures," Ph.D. Thesis, Dept. of Aeronautics and Astronautics, Stanford Univ., Stanford, CA, 1989.

[10]Dagum, L., "On the Suitability of the Connection Machine for Direct Particle Simulation," Ph.D. Thesis, Dept. of Aeronautics and Astronautics, Stanford Univ., Stanford, CA, 1990.

[11]Dagum, L., "Data Parallel Sorting for Particle Simulation," *Concurrency: Practice and Experience*, (to be published).

[12]Chatterjee, S., Blelloch, G. E., and Zagha, M., "Scan Primitives for Vector Computers," *Proceedings Supercomputing '90*, Nov. 12–16, IEEE Computer Society Press, Los Alamitos, CA, 1990, pp. 666–675.

[13]Haas, B. L., "Thermochemistry Models Applicable to a Vectorized Particle Simulation," Ph.D. Thesis, Dept. of Aeronautics and Astronautics, Stanford Univ., Stanford, CA, 1990.

[14]Dagum, L., "Lip Leakage Flow Simulation for the Gravity Probe B Gas Spinup," AIAA Paper 92-0559, Jan. 1992.

[15]Feiereisen, W., McDonald, J. D., and Fallavollita, M. A., "Three Dimensional Discrete Particle Simulation about the AFE Geometry," AIAA Paper 90-1778, June 1990.

[16]Bird, G. A., "Definition of Mean Free Path for Real Gases," *Physics of Fluids*, Vol. 26, No. 11, 1983, pp. 3222–3223.

[17]McDonald, J. D., and Dagum, L., "A Comparison of Particle Simulation Implementations on Two Different Parallel Architectures," *Proceedings of DMCC6*, April 29–30, Portland, OR, 1991.