A Comparison of Particle Simulation Implementations on Two Different Parallel Architectures

Jeffrey D. McDonald[1]

Eloret Institute
3788 Fabian Way,
Palo Alo, CA 94303

Leonardo Dagum

Computer Sciences Corp.
M.S. T045-1
NASA Ames Research Center
Moffett Field, CA 94035

Abstract

Direct particle simulation is a powerful method for analyzing low density, hypersonic re-entry flows. The method involves following a large sample of representative gas molecules through motion and collision with other molecules or with surfaces in the simulated flow. In this paper, two very different parallel architectures are examined for their suitability in particle simulation computations, namely the Connection Machine CM-2 and the Intel iPSC/860. The difference in architectures has resulted in very different parallel decompositions. The two implementations are described and performance results are given. Both implementations achieve performance comparable to a single Cray-2 CPU, however, this performance is obtained at the cost of greatly increased programming complexity.

1 Introduction

Particle methods of simulation are of interest primarily for high altitude, low density flows. When a gas becomes sufficiently rarefied the constitutive relations of the Navier-Stokes equations (i.e. the Stokes law for viscosity and the Fourier law for heat conduction) no longer apply and either higher order relations must be employed (e.g. the Burnett equations [6]), or the continuum approach must be abandoned and the molecular nature of the gas must be addressed explicitly. The latter approach leads to direct particle simulation.

1.1 Particle Simulation Method

In direct particle simulation, a gas is described by a collection of simulated molecules thus completely avoiding any need for differential equations explicitly describing the flow. By accurately modelling the microscopic state of the gas the macroscopic description is obtained through the appropriate integration. The primary disadvantage of this approach is that the computational cost is relatively large. Therefore, although the molecular description of a gas is accurate at all densities, a direct particle simulation is competitive only for low densities where accurate continuum descriptions are not available.

For a small discrete time step, the molecular motion and collision terms of the Boltzmann equation may be decoupled. This allows the simulated particle flow to be considered in terms of two consecutive but distinct events in one time step, specifically there is a collisionless motion of all particles followed by a motionless collision of those pairs of particles which have been identified as colliding partners. The collisionless motion of particles is strictly deterministic and reversible. However, the collision of particles is treated on a probabilistic basis. The particles move through a grid of cells which serves to define the geometry, to identify colliding partners, and to sample the macroscopic quantities used to generate a solution.

The state of the system is updated on a per time step basis. A single time step is comprised of five events:

1. Collisionless motion of particles.
2. Enforcement of boundary conditions.
3. Pairing of collision partners.
4. Collision of selected collision partners.

[1]Mailing Address: M.S. 230-2, NASA Ames Research Center, Moffett Field CA, 94035.

5. Sampling for macroscopic flow quantities.

Detailed description of these algorithms may be found in [7] and [1].

1.2 Computational Performance

Although particle simulation methods are very powerful, they are computationally expensive when applied on a scale large enough to address general three-dimensional problems. Typically millions of particles are utilized in a simulation (with as many as possible being desirable) resulting in large memory and CPU requirements. Early efforts to make large scale particle simulation feasible focused on replacing the underlying algorithms to allow substantial vectorization. Vectorization coupled with improvements in other implementation details led to a speedup of about 100 on Cray-2 computers[7] when compared to other available simulation methods used at that time[4].

In view of the evident limits to single processor performance, it is of interest to the particle simulation community to examine the suitability of distributed memory parallel architectures to problems of this type. Such machines now offer supercomputer performance and greater local memory bandwidth making them attractive for use in particle simulation. In the remainder of this paper, the implementation of a particle simulation method on two distinct parallel machines, namely the Thinking Machines Corp. Connection Machine CM-2 and the Intel Corp. iPSC/860, is examined. The architectures of these machines are briefly outlined, the two implementations are presented, results from an example simulation are shown, and the performance and suitability of each machine to this application are discussed.

2 Target Architectures

2.1 Connection Machine

The Thinking Machines Connection Machine Model CM-2 is a massively parallel 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 32768 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 which 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 a multiply each clock cycle thus performing at 14 MFLOPS and yielding a peak aggregate performance of 14 GFLOPS for the system.

The Connection Machine can be viewed two ways, either as an 11-dimensional hypercube connecting the 2048 CM chips or a 10-dimensional hypercube connecting the 1024 processing elements. The first view is the "fieldwise" model of the machine which has existed since its introduction. This view admits to the existence of at least 32768 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 which 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.

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 pre-compiled and the router will direct messages according to the pre-compiled paths. This is referred to as compiled communication and can be 5 times faster than general router communication. Finally, for communication which 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 pre-compiled paths. This is referred to as NEWS (for "NorthEastWestSouth") communication. Despite the name, NEWS communication is not restricted to 2-dimensional grids, and up to 31-dimensional NEWS grids may be specified. NEWS communication is the fastest.

2.2 Intel iPSC/860

The Intel iPSC/860 (also known as Touchstone Gamma System) is based on the 64 bit i860 microprocessor by Intel [5]. The i860 has over 1 million transistors and runs at 40 MHz. The theoretical peak speed is 80 MFLOPS in 32 bit floating point and 60

MFLOPS for 64 bit floating point operations. The i860 features 32 integer address registers, with 32 bits each, and 16 floating point registers with 64 bits each (or 32 floating point registers with 32 bits each). It also features an 8 kilobyte on-chip data cache and a 4 kilobyte instruction cache. There is a 128 bit data path between cache and registers. There is a 64 bit data path between main memory and registers.

The i860 has a number of advanced features to facilitate high execution rates. First of all, a number of important operations, including floating point add, multiply and fetch from main memory, are pipelined operations. This means that they are segmented into three stages, and theoretically a new operation can be initiated every 25 nanosecond clock period. Another advanced feature is the fact that multiple instructions can be executed in a single clock period. For example, a 32-bit memory fetch, floating add and floating multiply can all be initiated in a single clock period.

A single node of the Touchstone Gamma system consists of the i860, 8 megabytes (MB) of dynamic random access memory, and hardware for communication to other nodes. For every 16 nodes, there is also a unit service module to facilitate access to the nodes for diagnostic purposes. The Touchstone Gamma system at NASA Ames consists of 128 computational nodes. The theoretical peak performance of this system is thus approximately 7.5 GFLOPS on 64 bit data.

The 128 nodes are arranged in a seven dimensional hypercube using the direct connect routing module and the hypercube interconnect technology of the iPSC/2. The point to point aggregate bandwidth of the interconnect system, which is 2.8 MB/sec per channel, is the same as on the iPSC/2. However the latency for the message passing is reduced from about 350 microseconds to about 90 microseconds. This reduction is mainly obtained through the increased speed of the i860 on the Touchstone Gamma machine, when compared to the Intel 386/387 on the iPSC/2. The improved latency is thus mainly a product of faster execution of the message passing software on the i860.

3 Particle Simulation Implementation

3.1 SIMD Implementation

Particle simulation is distinct from other computational fluid dynamics (CFD) applications in that there are two levels of parallel granularity in the method. There is a coarse level consisting of cells in the simulation (which are approximately equivalent to grid points in a continuum approach) and there is a fine level consisting of individual particles. At the time of the CM-2 implementation there existed only the fieldwise model of the machine, and it was natural to decompose the problem at the finest level of granularity[1]. In this decomposition, the data for each particle is stored in an individual virtual processor in the machine. A separate set of virtual processors (or VP set) stores the geometry and yet another set of virtual processors stores the sampled macroscopic quantities.

This decomposition is conceptually pleasing however in practice the relative slowness of the Connection Machine router can prove to be a bottleneck in the application. This motivated the introduction of several novel algorithms to minimize the amount of communication and improve the overall performance in such a decomposition[1]. In particular, steps 2 and 3 of the particle simulation algorithm require a somewhat less than straightforward approach.

The enforcement of boundary conditions requires particles which are about to interact with a boundary to get the appropriate boundary information from the VP set storing the geometry data. Since the number of particles undergoing boundary interaction is relatively small, a master/slave algorithm is used to minimize both communication and computation. In this algorithm, the master is the VP set storing the particle data. The master creates a slave VP set large enough to accommodate all the particles which must undergo boundary interactions. Since the slave is much smaller than the master, instructions on the slave VP set execute much faster. This more than makes up for the time that the slave requires to get the geometry information and to both get and return the particle information.

The pairing of collision partners requires sorting the particle data such that particles occupying the same cell are represented by neighboring virtual processors in the one dimensional NEWS grid storing this data. Dagum [2] describes different sorting algorithms suitable for this purpose. The fastest of these makes use of the realization that the particle data moves through the CM processors in a manner analogous to the motion of the particles in the simulation. The mechanism for disorder is the motion of particles, and the extent of motion of particles, over a single time step, is small. This can be used to tremendously reduce the amount of communication necessary to re-order the particles.

These algorithms have been implemented in a two-dimensional particle simulation running on the CM-2. At the time of implementation, the CM-2 at NASA

Ames had only 64k bits of memory per processor which was insufficient to warrant a three-dimensional implementation. Furthermore, the slicewise model of the machine did not exist and the machine had the slower 32-bit Weitek's which did not carry out any integer arithmetic. Nonetheless, with this smaller amount of memory and fieldwise implementation, the code was capable of simulating over 2.0×10^6 particles in a grid with 6.0×10^4. Performance of the implementation is discussed later. Details on the fieldwise implementation and performance of a three-dimensional particle simulation may be found in [3].

3.2 MIMD Implementation

The MIMD implementation differs from the SIMD implementation not only because of the difference in programming models but also because of the difference in granularity between the machine models. Whereas the CM-2 has 32768 processors, the iPSC/860 has only 128. Therefore on the iPSC/860 it is natural to apply a spatial domain decomposition rather than the data object decomposition used on the CM-2.

In iPSC/860 implementation, the spatial domain of the simulation is divided into a number of subdomains or regions equal to the desired number of node processes[8]. Communication between processes occurs as particles pass from one region to another and is carried out asynchronously, thus allowing overlapping communication and computation. Particles crossing region "seams" are treated simply as an additional type of boundary condition. Each simulated region of space is surrounded by a shell of extra cells that, when entered by a particle, directs that particle to the neighboring region. This allows the representation of simulated space (i.e. the geometry definition) to be distributed along with the particles. The aim is to avoid maintaining a representation of all simulated space which, if stored on a single processor, would quickly become a serious bottleneck for large simulations, and if replicated would simply be too wasteful of memory.

Within each region the vectorized particle simulation algorithm discussed above are applied. This decomposition allows for great flexibility in the physical models used since node processes are asynchronous and largely independent of each other. Recall that communication between processes is required only when particles cross region seams. This is fortuitous since the particle motion is straightforward and fully agreed upon. The important area of research has to do with the modelling of particles and the various interactions they undergo, and since this part of the problem does not directly affect communication, particle models can evolve without requiring algorithmic changes complicated due to the use of multiple processors. Also the machine dependent elements of the implementation can be (and are) encapsulated in a separate machine dependent code module providing a certain degree of portability. Performance of the implementation is discussed below.

4 Example Application

An example solution from the Intel iPSC/860 implementation is presented in Figures 1 and 2. The problem concerns the hypersonic, low density flow about a two-dimensional circular cylinder. The freestream gas is pure diatomic oxygen (O_2) at Mach 25 having a temperature of $200K$ and a density of 2.18×10^{-6} Kg/m^3. The cylinder has a diameter of 1.44 meters while the upstream molecular mean free path length is 0.036 meters.

The simulation involves a three-dimensional cell network of dimension 128 by 128 by 4 finite volume cells to represent the physical domain. This domain is divided into 128 regions distributed over the full 128 nodes on the iPSC/860. The diameter of the cylinder in the non-dimensional simulation is 40 cells and the mean free path length is 1 cell. This provides the same Knudsen number as the dimensioned parameters. The free stream simulation number density is 25 particles per cell.

Two sets of translational temperature contours are presented in Figures 1 and 2 corresponding to simulation without and with chemistry modeling respectively. Note the lower temperatures in the chemistry case due to the dominant O_2 dissociation reaction. The gas is heated through the bow shock wave and when the effects of chemistry are included, dissociation occurs as collisions become more energetic. Since the dissociation of O_2 is an endothermic reaction, this is accompanied by a drop in the gas temperature in the post shock region.

5 Performance

5.1 Connection Machine

Dagum's implementation is two-dimensional. The performance of the code for a Mach 25 flow over a double-ellipse (see [1]) is given in Table 1.

Note that peak performance on a 32K machine processes particles at a rate of 1.88μsec/particle/timestep

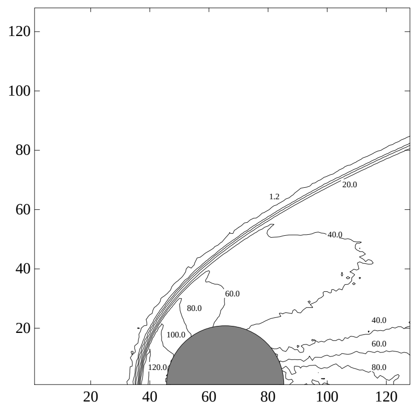

Figure 1: Temperature contours for Mach 25 flow with no chemistry over a circular cylinder. Contours are shown for temperatures of 0.2, 0.8, 1.2, 5, 10, 20, 40, 60, 80, 100, and 120 times the freestream temperature.

Table 1: **Performance of Particle Simulation on the CM-2**

Processors	μs/prt/step	MFLOPS	efficiency(%)
8,192	6.52	13.2	100
16,384	3.64	23.6	89
32,768	1.88	45.7	87

using all 32k processors (see [1]). By comparison, a fully vectorized equivalent simulation on a single processor of the Cray YMP runs at 1.0μsec/particle/timestep and 86 MFLOPS as measured by the Cray hardware performance monitor. Note that a significant fraction of a particle simulation involves integer arithmetic and the MFLOP measure is not completely indicative of the amount of computation involved. These performance results are illustrated in Figure 3 along with similar iPSC/860, Cray-2, and Cray-Y/MP results. (The Cray calculations employed 64-bit precision which is unnecessary for particle simulation.) Since a 64K Connection Machine is considered fully configured, the 32K result is plotted at the 64 node result for the iPSC/860. Performance is measured as the average amount of time it takes to fully update a single particle over the period of a single simulation time step. Currently, work is being carried out to extend the simulation to three dimensions using a parallel decomposition which takes full advantage of the slicewise model of the machine.

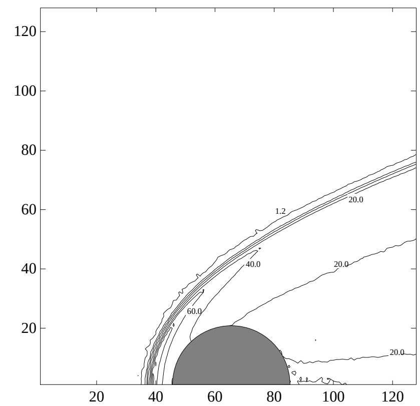

Figure 2: Temperature contours for Mach 25 flow with chemistry over a circular cylinder. Contours are shown for temperatures of 0.2, 0.8, 1.2, 5, 10, 20, 40, 60, 80, and 100 times the freestream temperature.

Table 2: **Performance of Particle Simulation on the Intel iPSC/860**

Processors	μs/prt/step	MFLOPS	efficiency(%)
1	47.3	1.8	100
2	24.4	3.5	97
4	12.5	6.9	95
8	6.35	13.5	93
16	3.25	26.5	91
32	1.63	52.8	91
64	0.85	101	87
128	0.42	215	88

5.2 Intel iPSC/860

McDonald's implementation is fully three-dimensional. The performance of the code on a 3D heat bath is given in Table 2 and illustrated in Figure 3 along with CM-2 and Cray performance results.

At the present time the domain decomposition is static, however work is being carried out to allow dynamic domain decomposition thus permitting a good load balance to exist throughout a calculation. The geometry and spatial decomposition of the heat bath simulation *exaggerated* the area to volume ratio of the regions in order to exercise the worst case communication expected in a real application with dynamic load balancing. The most promising feature of these results is the linear speed up obtained, indicating that the performance of the code should continue to increase with increasing numbers of processors. This apparent

Figure 3: Comparison of performance of various machines for particle simulation. Cray results are for a single processor and employing 64-bit precision.

scalability will soon be tested on the new Intel Delta prototype having 512 i860 processing nodes.

6 Concluding Remarks

Performance of a particle simulation method as implemented on both the Thinking Machines Corp. CM-2 and the Intel Corp. iPSC/860 rivals that of highly optimized Cray implementations. The slicewise architecture of more recent Connection Machines promises improved performance on that machine. The linear speedup behavior observed in early iPSC/860 results indicates scalability beyond the 128 processors currently available.

Implementation on both machines is complicated by the required programming models and portability to other machines is made difficult due to proprietary support mechanisms for these models. In the case of the iPSC/860 this complexity is more easily encapsulated into a separate module from physical modeling elements, thus making portability to other MIMD architectures possible through simple replacement of the machine dependent module. In order to fully utilize the Connection Machine, such encapsulation is not feasible making for poor portability.

Although the Connection Machine has a higher theoretical peak performance, the iPSC/860 has proved to be a more suitable platform in the context of a particle simulation. The domain decomposition allowed by the medium grain parallelism of the iPSC/860 architecture results in less interprocessor communication and a higher percentage of the peak performance is attained. It is expected that altering the CM-2 implementation to fully utilize the slicewise model will improve its performance.

Acknowledgements

J.D. McDonald acknowledges support from NASA contract NCC2-674. L. Dagum is supported through NASA contract NAS 2-12961. Computer resources for both the were provided by NASA Ames research center.

References

[1] L. Dagum. *On the Suitability of the Connection Machine for Direct Particle Simulation.* Technical Report 90.26, RIACS, NASA Ames Research Center, Moffett Field, CA 94035, June 1990.

[2] L. Dagum. Sorting for particle flow simulation on the connection machine. In Horst D. Simon, editor, *Research Directions in Parallel CFD*, MIT Press, Cambridge (to appear), 1991.

[3] L. Dagum. *Three-Dimensional Direct Particle Simulation On the Connection Machine.* AIAA-91-1365, 1991.

[4] V.K. Dogra, J.L. Moss, A.L. Simmonds, *Direct Simulation of Aerothermal Loads for an Aeroassist Flight Experiment Vehicle*, AIAA-87-1546, 1987.

[5] Intel Corporation. *i860 64-Bit Microprocessor Programmer's Reference Manual.* Santa Clara, California, 1990.

[6] F.E. Lumpkin. *Development and Evaluation of Continuum Models for Translational-Rotational Nonequilibrium.* PhD thesis, Stanford University, Dept. of Aeronautics and Astronautics, Stanford CA 94305, April 1990.

[7] J. D. McDonald. *A Computationally Efficient Particle Simulation Method Suited to Vector Computer Architectures.* PhD thesis, Stanford University, Dept. of Aeronautics and Astronautics, Stanford CA 94305, December 1989.

[8] J. D. McDonald. *Particle Simulation in a Multiprocessor Environment.* AIAA-91-1366, 1991.