PLASMA SIMULATION USING THE MASSIVELY PARALLEL PROCESSOR

C. S. Lin, A. L. Thring, J. Koga, and R. W. Janetzke
Department of Space Sciences
Southwest Research Institute
San Antonio, Texas 78284

ABSTRACT

Two-dimensional electrostatic simulation codes using the particle-in-cell model are developed on the Massively Parallel Processor (MPP). The conventional plasma simulation procedure that computes electric fields at particle positions by means of a gridded system is found inefficient on the MPP. The MPP simulation code is thus based on the gridless system in which particles are assigned to processing elements and electric fields are computed directly via Discrete Fourier Transform. Currently the gridless model on the MPP in two dimensions is about nine times slower than the gridded system on the CRAY X-MP without considering I/O time. However, the gridless system on the MPP can be improved by incorporating a faster I/O between the staging memory and Array Unit and a more efficient procedure for taking floating point sums over processing elements. Our initial results suggest that the parallel processors have the potential for performing large scale plasma simulations.

Keywords: plasma simulation, gridded system, gridless system, discrete Fourier transform

INTRODUCTION

Plasma simulations have been used extensively in space physics and fusion energy research. Although the basic physical principles of plasma physics are well understood, the coupled physical phenomena are so complex that analytical studies cannot be easily carried out. With the advent of powerful supercomputers, computer simulations have been used to gain insight into physical phenomena.

One class of plasma simulation model is the particle-in-cell (PIC) model, which treats the plasma at the microscopic level by following the motion of a large number of particles. We have been using a particle-in-cell code to examine the instabilities of natural electron beams observed by the NASA Dynamics Explorer satellite at high altitudes ($> 10,000$ km) (Ref. 1). The simulation code typically uses over 400,000 simulation particles and a 32×32 spatial grid in the two-dimensional problem. It is difficult to improve the spatial resolution with our available resources, because a finer spatial grid would sharply increase the simulated particle number and would require unreasonable computer time. Likewise, three-dimensional simulations are beyond our capability at the present time. Buzbee (Ref. 2) examined the parallel properties of particle codes for fusion studies and concluded that a large percentage of plasma simulation programs can be processed in parallel. We investigated potential improvement in two-dimensional and three-dimensional plasma simulations by carrying out simulations on the Massively Parallel Processor (MPP).

Particle-in-cell simulation models have been developed for the parallel-architecture CHI computer by Dawson et al. (Ref. 3). Their results indicate that the system is cost-effective and offers a significant improvement in performance. The CHI plasma simulation system consists of six microprocessors: one array processor, one macro-processor and four I/O processors. The array processor does most of the calculations, the I/O processors move data around and the macro-processor handles scheduling and control. The parallel processing among these processors accounts for most of the performance advantage of the system. Buzbee (Ref. 2) has implemented a particle-in-cell code on two parallel processing devices–the UNIVAC 1100/84 and the Denelcor Heterogeneous Element Processor (HEP); results from these two parallel processors demonstrated that a large percentage of the total computation can be done in parallel.

For these previous simulation models, the number of parallel processors is less than eight. Plasma simulation models have not been developed for a single-instruction-stream and multiple-data-stream (SIMD) processor like the MPP which has 16,384 arithmetic processors configured as a 128 x 128 array. Without knowing how to take advantage of the unique structure of the MPP, we first attempted to transform our current particle code to run on the MPP. As might be expected, we encountered difficulties in achieving good performance on the MPP using the particle code optimized for the vector computer CRAY. Before

discussing the difficulties and the new approach, we present some background on the procedures of plasma simulation models.

PARTICLE-IN-CELL MODEL

The particle-in-cell simulation code usually represents the plasma as a large number of computational particles (usually greater than 100,000 particles for each species) which move according to classical mechanics in the self-consistent electromagnetic fields. The two-dimensional spatial system is then divided into fixed spatial cells or grids on which charge densities, potentials, and fields are defined. For the purposes of illustration, we will discuss only the electrostatic model which has no electric current density. In its simplest form, the procedures of the plasma simulation are:

1. computation of the charge density at cell centers. A particle is assumed to have a finite size comparable to the cell size to reduce numerical noise. The charge density of each cell is calculated by accumulating each particle's contribution according to its occupied area in the cell.

2. computation of electric fields from the charge density using Poisson's equation. Poisson's equation is usually solved by using a finite-difference scheme or Fast Fourier Transform.

3. interpolation of the electric forces on the particles from the electric fields at the nearest grid points.

4. application of the electric force to advance the particle velocities and then positions in time using a simple leapfrog scheme.

The mathematical formulas of the electrostatic model are briefly described here. The model consists of finite-size particles, moving in a uniform and constant magnetic field $\vec{B}$ and interacting via a self-consistent electric field $\vec{E}$. The equations of motion are

$$d\vec{v_i}/dt = (q_i/m_i)(\vec{E} + \frac{\vec{v_i}}{c} \times \vec{B}) \quad (1)$$

$$d\vec{r_i}/dt = \vec{v_i} \quad (2)$$

Here $\vec{v_i}$, m_i and q_i are the velocity, mass, and charge of the ith particle with the center position denoted by $\vec{r_i}$. The particles have a rectangular shape with a width Δ. The charge density ρ is then

$$\rho(\vec{r}) = \sum_i q_i S(\vec{r} - \vec{r_i}) \quad (3)$$

The shape function $S(\vec{r} - \vec{r_i})$ is 1 when both $(x - x_i)$ and $(y - y_i)$ are negative, and is 0 otherwise. Here x and y are the two coordinates of r. Poisson's equation is

$$\nabla^2\phi = -4\pi\rho(\vec{r}) \quad (4)$$

where ϕ is the electric potential. The electric field $\vec{E}$ is defined as

$$\vec{E} = -\vec{\nabla}\phi \quad (5)$$

We will focus on the numerical method of obtaining the electric field $\vec{E}$, which is the key problem in implementing plasma simulation models on parallel computers. The algorithm of solving the electric field $\vec{E}$ from Equations (4) and (5) is usually based on Fourier Transforms. For an infinite continuous system, the Fourier Transforms of Equations (4) and (5) yield

$$\vec{E}(\vec{k}) = -i4\pi\vec{k}S(\vec{k})\rho(\vec{k})/k^2 \quad (6)$$

where $\vec{E}$, $S(\vec{k})$ and $\rho(\vec{k})$ are the transformed quantities. The infinite continuous transform is defined here as

$$\rho(\vec{k}) = \int d\vec{r}\rho(\vec{r})e^{-i\vec{k}\cdot\vec{r}} \quad (7)$$

An infinite continuous system has no grid; therefore, the charge density $\rho(k)$ becomes a summation over the particle positions

$$\rho(\vec{k}) = S(\vec{k})\sum_i q_i e^{-i\vec{k}\cdot\vec{r_i}} \quad (8)$$

When the particle number is large, $\rho(\vec{k})$ in Equation (8) is generally too inefficient for computation. Instead a fast algorithm involving a gridded system and Fast Fourier Transform is often used. Figure 1 shows the flow chart of computing electric field $\vec{E}(\vec{r_i})$ at the location of particles $\vec{r_i}$ in a gridded system. Because the Fast Fourier Transform of $\rho(\vec{k})$ is computed from a small number of $\rho(\vec{r_g})$ at the cell centers $\vec{r_g}$, the algorithm is very efficient. The saving in computation makes up for the extra steps in collecting cell charge densities and interpolating the electric field at particle positions.

Gridded System On the MPP

Plasma simulation models using grids have been tested on several parallel computers and found to be feasible (Refs. 2-3). Our first plasma simulation model on the MPP therefore uses the gridded system. The program assigns a particle to each processing element in the MPP. Because the Array Unit has 16,384 processing elements, the program updates positions and veloci-

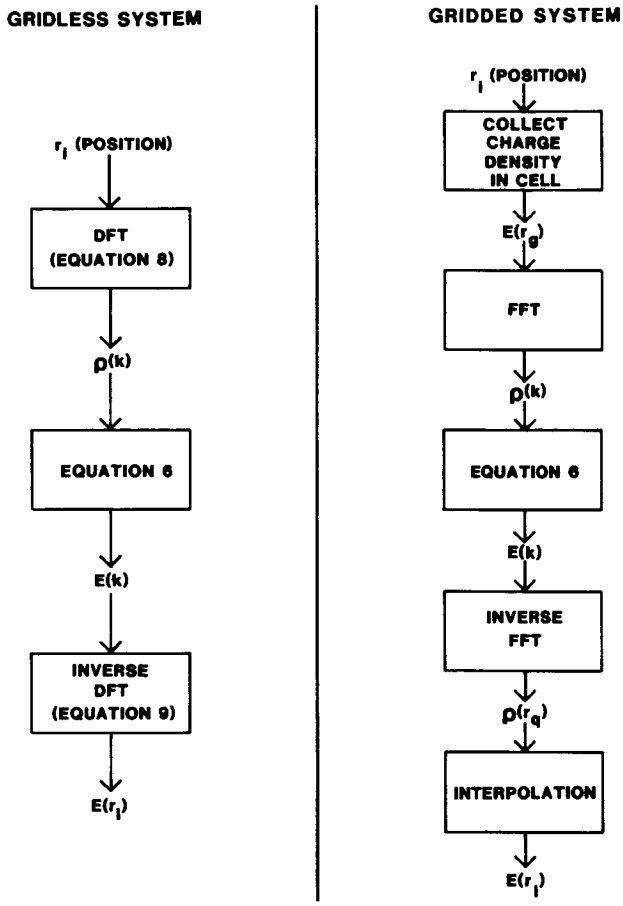

Figure 1. Flow chart of computing electric fields in the gridded and gridless systems.

ties of 16,384 particles at the same time. However, we soon realized that collection of cell charge density and interpolation of electric field at particle positions cannot be computed in parallel or vectorized.

We investigated the possibility of performing serial computations on the VAX, the frontend computer of the MPP. The serial calculations involved in computing charge density at cell centers (Step 1) were completed on the average in 7 seconds on the MPP and 14.5 seconds on the VAX. A separate timing of the number crunching portion on the MPP yields only 1.2 milliseconds, corresponding to a speed of approximately 230 Mflops (millions of floating point operations). Therefore the MPP time of performing Step 1 is essentially I/O time between the staging memory and the VAX. This approach is clearly not acceptable because of the excessive I/O time.

The problem is somewhat alleviated on some supercomputers, which can perform fast serial computation or do indirect indexing (Ref. 4). Some speedup in charge collection on the MPP might be achieved by using an efficient sorting algorithm. However, the interpolation cannot be performed efficiently on the MPP because it has only nearest-neighbor communication and a small memory in the Array Unit. We finally discarded the gridded system for the MPP and investigated the alternative approach described below.

GRIDLESS SYSTEM

The algorithm for the gridless system as shown in Figure 1 is much simpler than the gridded system. Equations (6) and (8) are used directly to compute the Fourier Transform of electric field $\vec{E}(\vec{k})$. No interpolation is needed to obtain the electric field at particle positions. Instead, the electric field at the particle position is computed by using the inverse Discrete Fourier Transform

$$\begin{aligned} \vec{E}(x,y) &= \int_0^\infty dk_x dk_y \vec{E}(k_x,k_y) e^{i(k_x \cdot x + k_y \cdot y)} \\ &= \sum_{n,m} \vec{E}_{nm} e^{i(2\pi/L)(nx+my)} \end{aligned} \tag{9}$$

Numerical evaluation of the integral in Equation (9) is inefficient. The integral is thus converted to discrete summations by assuming that the system is periodic and has a length of L. The summation is truncated by keeping only the low order terms; we have only kept terms with $n \leq 16$ and $m \leq 16$ in two dimensions. Since the right hand side of Equation (9) depends on only the local parameters (particle position), the calculation of $\vec{E}(x,y)$ can be parallelized.

On the vector computer, the algorithm for the gridless system is much slower than that for the gridded system. Figure 2 shows the timing of the algorithms run on the CRAY X-MP at the San Diego Supercomputer Center. The algorithm of computing electric fields for the gridless and gridded systems in one dimension is shown in Figure 1. For particle number $N = 16384$, the gridless system requires about 15 times more CPU time than the gridded system to obtain comparable results on the CRAY X-MP. The reason for this remarkable difference in speed is simply that the gridless system uses a Discrete Fourier Transform whereas the gridded system uses a Fast Fourier Transform. The number of operations to be performed for the gridless system is proportional to NN_k, where N_k is the number of Fourier modes. For the gridded system the number of operations is proportional to $N_g \log N_g$, where N_g is the number of cells and is usually taken to be 32 for a model. Since N_k is chosen to to be $N_g/2$ and $N \gg N_g$, the gridless system has many more arithmetic operations than the gridded system.

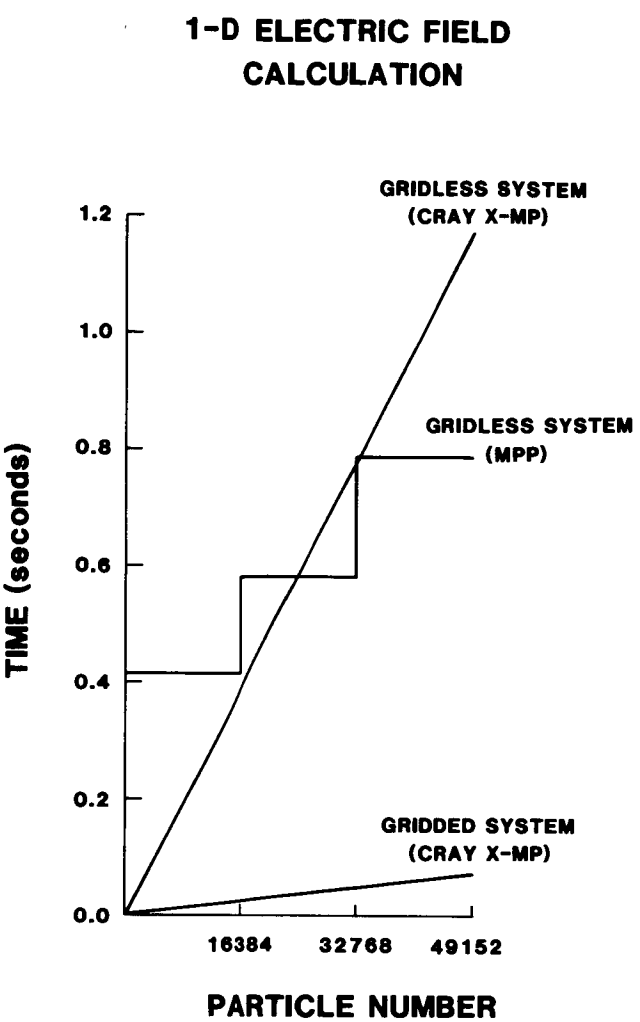

Figure 2. Algorithm timing of electric field calculations in one dimension.

We next discuss the algorithm timing for the gridless system run on the MPP. The MPP CPU time we present includes only processing times of the Array Unit for arithmetic operations and the Main Control Unit for controlling I/O. The I/O time between the staging memory and the Array Unit is not yet included in our algorithm timing because we have not yet optimized the I/O and learned how to determine the I/O time accurately. It appears that the I/O time is many times the MPP CPU time.

In CPU time required to compute electric field in the one-dimensional gridless system, the MPP is comparable to the CRAY X-MP in speed (Figure 2). The MPP is more efficient than the CRAY X-MP for the gridless system because it operates on 16,384 particles simultaneously and thus the number of operations is proportional to $(N/16384)N_k$. Note that the gridless system algorithm is highly parallelized on the MPP.

We next compare the timing of computing electric field in two dimensions between the gridless system on the MPP and the gridded system on the CRAY X-MP (Figure 3). The MPP is about 24 times slower than the CRAY X-MP in the case of 16,384 particles. When $N = 3 \times 16384$, the MPP is about 16 times slower than the CRAY X-MP.

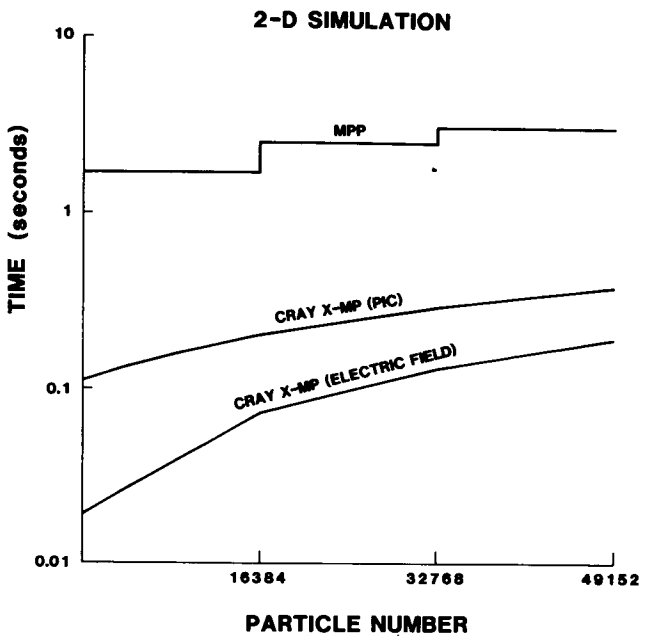

Figure 3. Timing comparison of two-dimensional electrostatic simulation codes between the CRAY X-MP and the MPP. Both the gridded and gridless systems are timed on the CRAY X-MP. Only the gridless system is timed on the MPP.

In breaking down the CPU timing on the MPP, we found that the MPP spends about 60% CPU time for $N = 16384$ and 30% CPU time for $N = 3 \times 16384$ in obtaining the floating point reduction sum, which is a summation taken over 16,384 processing elements. The amount of time spent on reduction sums is excessive since the number of reduction sums is quite small, only 16 for one dimension and 256 for two dimensions. We were able to determine that the routine in the MPP library for summing 32 bit floating point numbers over the processing elements had not been optimized. Specifically, a summation of a 32 bit parallel array took about 2.04 milliseconds, which is about 55 times the CPU time for addition of two 32 bit floating point numbers.

The plasma simulation procedure outlined in the particle-in-cell model includes advancing particle positions and velocities in addition to solving the electric field. Particle positions are checked to determine if

particles advance outside the system boundary. The periodic boundary condition is used to adjust the positions of particles advanced outside the boundary

$$x = x - L \quad if\ x > L \tag{10}$$

and

$$y = y - L \quad if\ y > L \tag{11}$$

The procedure of implementing the boundary condition cannot be fully vectorized on the CRAY X-MP. Furthermore, the MPP has an advantage over CRAY X-MP because it can advance 16,384 particles simultaneously instead of 64 particles as processed by CRAY X-MP.

Figure 3 presents the timing for the two-dimensional electrostatic simulation procedure outlined in the particle-in-cell model. The CRAY X-MP uses 0.2 seconds CPU time for simulating a plasma of 16,384 particles, about three times the CPU time needed to compute the electric field. Thus the CRAY X-MP spends about one-third of the CPU time on Steps 1-3 and two-thirds of the CPU time on Step 4. In contrast, the additional CPU time needed for Step 4 on the MPP is negligible ($\approx$ 1 millisecond).

For $N = 16384$, the timing ratio of the two-dimensional electrostatic PIC code is about nine. The ratio remains constant when N increases to 3×16384.

DISCUSSION

The objective of this study is to develop an efficient MPP program to simulate beam plasma interactions in three dimensions. The initial results presented here show some difficulties in reaching this goal. The efficient method typically employed on the CRAY X-MP simulates plasmas in a gridded system, first computing electric fields at the grid points via Fast Fourier Transforms and then interpolating electric fields at particle positions. Because the MPP has only nearest-neighbor communication and a limited memory in the Array Unit, the gridded system is awkward and very slow on the MPP. We have thus adopted a gridless system that assigns a particle to a processing element and computes electric fields at particle positions directly via Discrete Fourier Transform. Currently, the gridless model in two dimensions on the MPP is about nine times slower than the gridded system on the CRAY X-MP. Improvements in the CPU times for the MPP are still possible, since our MPP programs have not been fully optimized. In three-dimensional simulations, the MPP is expected to be much slower than the CRAY X-MP.

It is obvious that the speedup factor for the gridless model on the MPP relative to the CRAY X-MP is large since the gridless model is highly parallelized. Indeed, when the performance of the one-dimensional gridless system on both computers is compared, the MPP is as fast as the CRAY X-MP (see Figure 2). However, a much more efficient method using the gridded system can be adapted on the CRAY X-MP but not the MPP.

So far we have only tested the basic ideas of the gridless simulation model and shown that the unique structure of the MPP is suitable for processing this model. We have not yet completed the program for conducting plasma simulations. One critical area unresolved is the transfer of diagnosis quantities from the MPP to the VAX for post processing. In order to minimize the total run time, the program being designed will process I/O between the staging memory and the VAX in parallel with the Array Unit (Figure 4). The

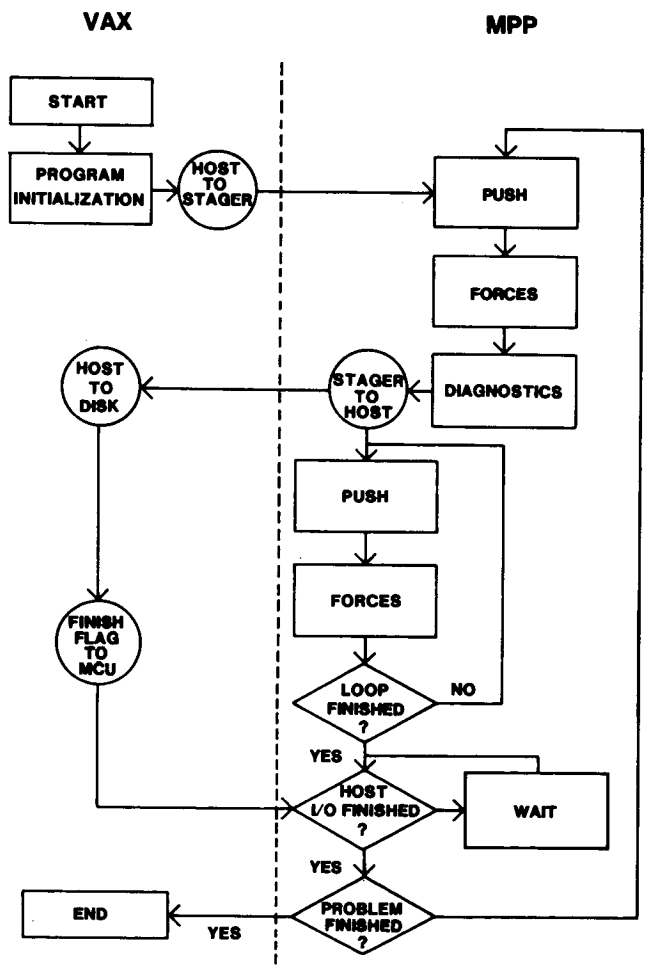

Figure 4. Program design of the plasma simulation code

simulation diagnosis usually needs to examine only a portion of particles for about every hundred time steps or longer. We estimate that the MPP uses about 2.5 seconds to transfer positions and velocities of 16,384 particles. Simulating 16,384 particles in one hundred time steps, the Array Unit will spend about 200 seconds, which is more than enough time to output the diagnosis results. The program will check to see that the I/O is completed before computations continue. These features of our program should ensure that the I/O time between the staging memory and the VAX will not contribute significantly to the total run time.

We have compared only the CPU time between the MPP and the CRAY X-MP. As mentioned before, the I/O time between the staging memory and the Array Unit is not yet timed. Since the I/O is very slow, the MPP performance is misleading without considering the I/O. By simulating only a small number of particles ($N < 49152$), we do not need to use staging memory. Plasma simulation for realistic problems will have many more particles than were used here. Because the Array Unit has a very limited memory (32 floating point words), it is necessary to transfer temporary variables to the staging memory. Our preliminary estimates indicate that tansferring a parallel array between the staging memory and the Array Unit takes from 35 to 200 milliseconds. The I/O time is very slow and highly variable because the I/O initiated by the standard MPP routines is controlled by the frontend VAX computer. Recently we learned that the I/O time can be reduced by using efficient routines without involving the VAX. But the number of I/O operations for using the staging memory as auxilliary memory for the Array Unit is estimated to be at least 3000 for $N = 3 \times 16384$. Unless the I/O speed is improved significantly, we do not believe that the MPP can compete with the CRAY computers in speed.

Another difficulty with the plasma simulation on the MPP is the floating point reduction sum over 16,384 processing elements. We were unable to investigate an efficient algorithm for computing the reduction sum, but were told that the current algorithm can be improved considerably (private communication with E. Seiler).

Having mentioned the disadvantage of the MPP, we feel it is only fair to discuss some advantages of the gridless model on the MPP. Implementation of the simulation procedure is simplified by assigning a particle to a processing element (see flow chart in Figure 1). Without using the grid, the simulation code also avoids numerical noises due to the grid. Finally, little effort is needed to extend the code to three dimensions. The experience of developing plasma simulation codes on the MPP has proved to be more challenging than anticipated. Our original attempt, implementing the conventional particle-in-cell code on the MPP, was fruitless. Because we assigned particles to processing elements, we encountered the difficulty of indirect addressing in the interpretation of forces at particle posistions from the nearest-neighbor grid points. The MPP currently cannot address indirectly other processing elements. Indirect addressing can be avoided in the gridded system if the cells are mapped to processing elements (Ref. 5).

We have investigated the gridless system approach that uses extensive parallel computation and minimal communication among parallel processors. Our results so far indicate that the MPP, with improvements in both hardware and software, might be acceptable for large scale scientific computation. The needed improvements include a faster I/O between the staging memory and the Array Unit, more memory in the Array Unit, and a more efficient procedure for the reduction sum. With these improvements the gridless system of particle simulations has the potential to be a useful and cost-effective method for simulating plasma phenomena on parallel computers.

ACKNOWLEDGEMENT

We thank E. Seiler of the MPP User Support Office for his valuable assistance in learning to use the MPP. We acknowledge San Diego Supercomputer Center for the use of the CRAY X-MP. We also acknowledge the joint grant of Digital Equipment Corpotion and National Science Foundation for providing a MicroVAX II, which is used for get access to the CRAY and the MPP. This work is supported by Southwest Research Institute Internal Research Program, NSF grant ATM-8405536, and NASA contract No. NAS8-32488.

REFERENCES

1. Lin, C. S., D. Winske, and R. L. Tokar, "Simulation of the Electron Acoustic Instability in the Polar Cusp," J. Geophys. Res., 1985, 90, pp.8269-8280.

2. Buzbee, B. L., "Plasma Simulation and Fusion Calculation," High-Speed Computation, ed. by J. S. Kowalik, Springer-Verlag, 1984, pp.417-424.

3. Dawson, J. M., R. W. Huff, and C. C. Wu, "Plasma Simulation on the UCLA CHI Com-

puter System," (Proceeding of the National Computer Conference) 1978, pp. 395-407.

4. Nishiguchi, A., S. Orii, T. Yabe, "Vector Calculation of Particle Code," Journal of Computational Physics, 1985, 61, pp. 519-522.

5. Gledhill, I. M. A., and L. R. O. Storey, "Particle Simulation of plasmas on the Massively Parallel Processor," (Proceedings of the First Symposium on the Frontier of Massively Parallel Scientific Computation) NASA/Goggard Space Flight Center September 24-25, published in December 1986.