ABSTRACT

Modified PIC algorithm for efficient multiprocessor simulations

Sasser, G., Havranek, J., †Colella, S., Luginsland, J., Merkle, L., †Watrous, J.
Air Force Research Lab, Kirtland AFB, NM 87117
†NumerEx, Albuquerque, NM 87106

Introduction

In this paper we present a method for recasting the electromagnetic particle-in-cell (PIC) algorithm in such a way that a general class of problems may be simulated efficiently on parallel computers [1]. The electromagnetic PIC algorithm is well suited for implementation on distributed memory parallel machines because all of the calculations require only local information. To carry out this task requires partitioning the physical space (cells) and the particles in some way and sharing border values at the appropriate time. One method to accomplish this is to divide the particles evenly among the available processes and to separately divide the cells among available processes. This method, known as general concurrent PIC (GCPIC), works best when the problem is dominated by particles [2]. However, the algorithm is not as efficient for a significant class of problems in which the particle and cell workload are comparable. This is due in part to the likelihood that such problems will have highly non-uniform particle distributions (such as in beam problems). Such problems may have large regions that are not populated with particles. In this case it is quite possible using GCPIC to have processes for which the cell partition has no overlap with the particle partition. This will require copying significant volumes of cell data to allow the particle position and velocity updates.

We propose a single partitioning (single domain decomposition) of the problem space so that the total workload is balanced. If viewed separately, both the particle and cell workloads may in general be out of balance in this case. In order to prevent poor parallel performance the PIC algorithm must be recast and maximum use of asynchronous message passing must be used.

PIC Algorithm

The PIC algorithm as shown in Fig. 1 has four points at which information is required to be shared between processors. Here we consider only the case of single domain decomposition; for GCPIC (dual domain decomposition) there would be an additional message after the field updates that would copy the fields of cells for particles residing outside of the cell partition. There are a number of ways in which we can improve upon this loop. Let us assume first that each send and recieve is accomplished as shown in Fig. 1 and that these messages are of the blocking type (i.e., the code may not proceed until the send or receive has been completed). High parallel efficiency is attained if the problem is balanced (the time required for computation between messages is the same for each process) and when the number and size of messages is minimized.

To accomplish field and current density updates it is necessary that there be a duplicate layer of cells (termed ghost cells) surrounding each partition. These ghost cells maintain a copy of the field values which must be obtained from the neighboring process which owns the cells. The send and recieve pair for the electric field may be combined with the send and recieve pair for the magnetic field if an extra layer of ghost cells is created to allow updating the electric field of the first layer of ghost cells. Because the time required to initiate a message can be much longer than the time required to actually transmit the information, combining these messages can result in significant savings of time (at the expense of performing a number of redundant electric field updates proportional to the number of ghost cells in the first layer). This reduces the number of synchronization points to three.

In observing the PIC loop in Fig. 1 we see that although particle information may be sent immediately following the particle position update, this information is not required until the velocity update which doesn't occur until after the electric and magnetic field updates. By separating the send and receive we can overlap communication with computation. This allows the process to continue computation while communication is taking place. More importantly, the number of synchronization points is reduced to two. In Fig. 2 the sends and receives are separated with the receives being delayed

until immediately before the next step for which they are required. We consider points in the routine for which there are receives to be synchronization points. When a process arrives at such a point it is necessary for all of its neighbors to have sent the required messages. If processes have different workloads between synchronization points then the process which finishes first will be forced into a wait state.

The time required for each computational step is proportional either to the number of cells or to the number of particles. In Fig. 2 we see that the computation between receives is proportional either to the number of particles or two the number of cells. This would be the optimal configuration for GCPIC, although as mentioned above, GCPIC would require an extra message which would be at the same point in the loop as the electric and magnetic field messages. For single domain decomposition this becomes a problem because the processes will not in general each have the same number of particles and the same number of cells. Processes with more cells will be forced into a wait state before the particle receive and processes with more particles will be forced into a wait state before the field receive.

The imbalance for single domain decomposition can be removed by performing the field updates for cells interior to a partition separately from those on the border and partitioning the problem by allocating a number of cells and particles with these weighted proportional to their respective computational workload. The algorithm using this technique is shown in Fig. 3. Note that all of the receives are now contained within the shaded box. In doing this we have reduced the number of synchronization points to two with almost no computation between these two so that there is essentially only one synchronization point.

The total computational workload for a problem may be represented by

$$W_{Total} = W_{Particle} N_{Particles} + W_{Cell} N_{Cells}$$

where W represents computational workload and N the number of particles or cells. We may cause the problem to be balance by partitioning it such that each process has the same workload as defined above. This may be limited by the resolution of the problem as one moves to high numbers of processes and the workload per process is small so that careful gridding may be required to achieve optimum efficiency.

Fixed and Scaled Parallel Efficiency

To test the efficiency of this implementation of the PIC algorithm we consider a simple test case. This is the case of a long metallic box with small transverse cross section. One measure of the efficiency of a parallel code is termed scaled efficiency. In this measure the size of the problem is scaled linearly with the number of nodes used in a simulation. For scaled efficiency runs the problem has a mesh of 10x10x(X*1700) where X is the number of nodes with particles streaming in the z-direction. The problem is formulated such that there are approximately 1000 particles crossing a partition boundary per time step and there are 242 border cells at each border. There are approximately 100 particles per cell for a total of 1.7×10^7 particles per node. The simulations were performed on an IBM SP computer at the Army Corps of Engineers Waterways Experiment Station (CEWES) Major Shared Resource Center (MSRC). For 128 nodes, the problem size was 2×10^9 particles and 2×10^7 cells. This simulation completed 100 time steps in 5 hours with an efficiency of greater than 95% where scaled efficiency is defined as:

scaled efficiency = (Time on a single node) / (Time on X nodes)

A similar problem was used to test fixed efficiency where the problem size is held fixed as it is executed on greater numbers of nodes. For this problem the mesh was 10x10x3400 with 3.4×10^7 particles again streaming in the z-direction. For this case we achieved 81% efficiency on 128 nodes where the efficiency is defined in this case as

fixed efficiency = (Time on a single node) / [X * (Time on X nodes)]

81% efficiency on 128 nodes corresponds to a speed-up of 104. These numbers are preliminary. We have not yet performed a thorough evaluation of the timing for each part of the code and we may be able to improve these numbers with some fine tuning.

Discussion

For problems with uniform particle distributions and large particle to cell ratios this algorithm performs as well as GCPIC. Differences in performance are more likely due to details of implementation such data structures and sorting than in choice of partitioning schemes. However, the algorithm presented here is also expected to perform well for more complex problems such as those with low particle to cell ratios and those with highly non-uniform particle distributions. Results will be presented of timings for such a problem as well as a discussion of load balancing using this technique.

Acknowledgements

Much of this work was funded through the Air Force Office of Scientific Research, Mathematics and Geosciences Directorate. Major contributions to the ICEPIC code were made by B.J. Smith and M.G. Anderson. The authors have benefited greatly from discussions with R.E. Peterkin, Jr., D. McGrath, and U. Shumlak during the course of this research. We acknowledge use of the IBM SP parallel computers at the Maui High Performance Computing Center, at the Aeronautical Systems Center, and at the Corps of Engineers Waterways Experiment Station.

[1] J. Havranek, G. Sasser, "Dynamic load balancing for general 3D parallel electromagnetic PIC," *J. Comp. Phys*., submitted Apr 1997.
[2] J. Wang, P. Liewer, and V. Decyk, "3D electromagnetic plasma particle simulations on a MIMD parallel computer," *Comp. Phys. Comm*. **87**, 35-53 (1995).

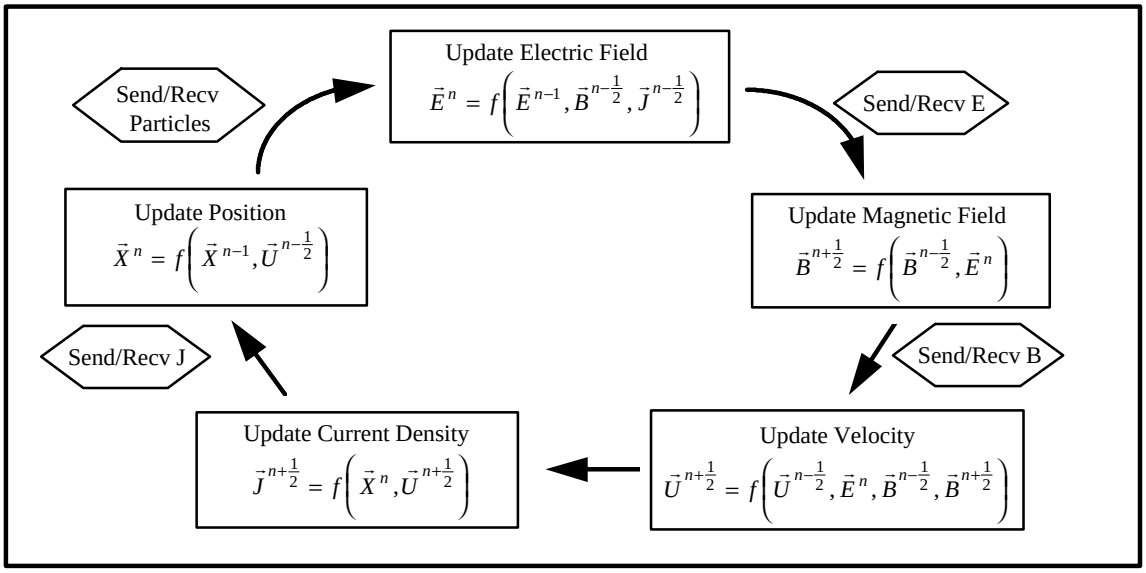

Figure 1. The electromagnetic PIC loop showing data dependencies and required message passing.

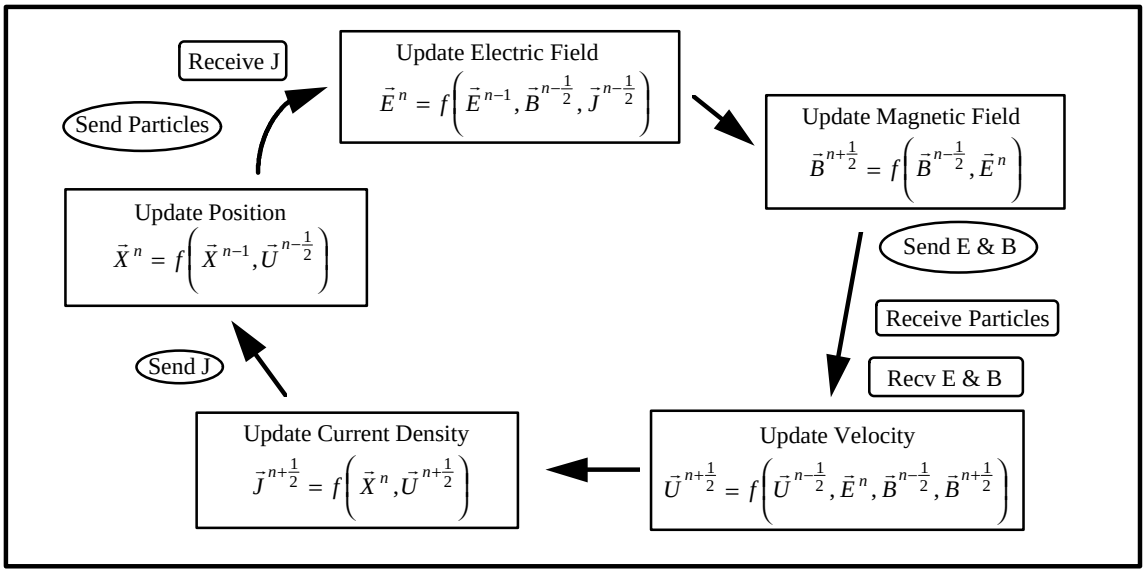

Figure 2. The electromagnetic PIC loop with asynchronous communication and combined field messages.

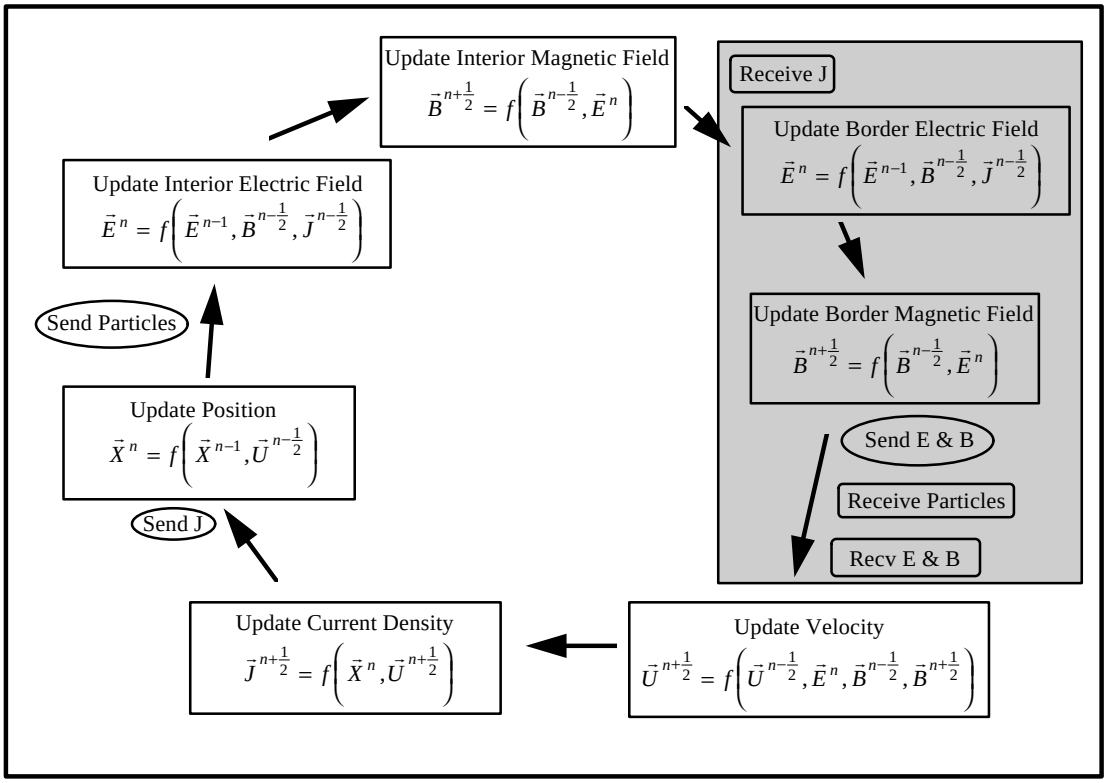

Figure 3. The electromagnetic PIC loop with the fields in interior and border cells updated separately.

Jerry Sasser
3550 Aberdeen Ave
Kirtland AFB, NM 87117
(505) 846-9105
sasser@ppws03.plk.af.mil