Load-Balancing and Performance of a Gridless Particle Simulation on MIMD, SIMD, and Vector Supercomputers

Steve Plimpton, Isaac Shokair, John Wagner, Jeff Jortner*

Abstract. Our charged particle simulation models a relativistic electron beam for which the field solution is local and thus requires no grid. We have implemented the simulation on a CRAY and on two parallel machines, a nCUBE 2 and Connection Machine. We present implementation details and contrast the approaches necessary for the three architectures. On the parallel machines a dynamic load-balancing problem arises because the beam grows uniformly in one dimension from a few hundred to hundreds of thousands of particles as the simulation progresses. We discuss a folded Gray-code mapping of the processors to the length scale of the simulation that expands (or shrinks) as the beam changes length so as to minimize inter-processor communication. This improves the efficiency of the nCUBE version of the simulation which runs at 10x the speed of the vectorized CRAY version.

Introduction

Charged particle simulations are good candidates for parallelization because large numbers of particles move simultaneously for many timesteps. At each timestep forces on each particle are calculated (due to self-induced or externally applied fields) and the particles moved in accordance with the forces. For efficiency, fields are often calculated on a mesh or grid and interpolated to the particle positions. In a general particle-in-cell (PIC) code implemented on a parallel machine, a load-balancing problem often arises because the mesh and particles are stored as different data sets for computational efficiency and large amounts of data must be transferred back and forth between the two domains at each timestep to do field solves and particle pushes [1].

Our simulation models a highly relativistic electron beam as it propagates through a plasma channel. Because of the approximations used, the fields can be calculated directly from the particle positions without use of a grid. Hence it is simpler to parallelize than a general PIC code. We have

* Sandia National Laboratories, Albuquerque, NM 87185. This work was partially supported by the Applied Mathematical Science Program, U.S. Department of Energy, Office of Energy Research and was performed at Sandia which is operated for the DOE under contract No. DE-AC04-76DP00789.

Proceedings of "5th SIAM Conference on Parallel Processing for Scientific Computing" (SIAM Press), March 1991, Houston, TX, p. 507-512.

adapted a CRAY version of the simulation [2,3] written by two of the authors (J.W. and I.S.), for implementation on two parallel supercomputers - a MIMD nCUBE 2 and a SIMD Connection Machine (CM). One complicating factor is that as the simulation progresses the electron beam grows uniformly in length from a few hundred to many hundreds of thousands of particles. To achieve high performance on a parallel machine, processors must be continually remapped to this growing domain. In this paper we present a solution to this load-balancing problem that works well for MIMD computers where one has explicit control over the assignment of processors to the physical geometry of the problem. In general, it can be applied to any problem where the processors need to expand or shrink in one dimension of the simulation domain.

In the next section, we briefly describe the physics of the beam propagation being modeled by the simulation. We then give details of the implementation on the two parallel machines and discusses the load-balancing issue. Finally, timing and performance results are presented and conclusions offered.

Model

We are interested in the propagation of highly relativistic electrons once they are emitted from an accelerating device as in Figure 1. The goal of experimenters is to propagate an intense electron beam (10^3 amps) over long distances (10^2 meters) either in outer space (e.g. the ionosphere) or circulating in a second accelerator, while keeping the beam focused to within a few centimeters. Such a beam might have application as a space weapon or for studying beam interaction with plasmas and magnetic fields. The simulation is a tool for modeling actual experimental apparatuses and for examining several physical effects that conspire to produce defocusing and beam spreading.

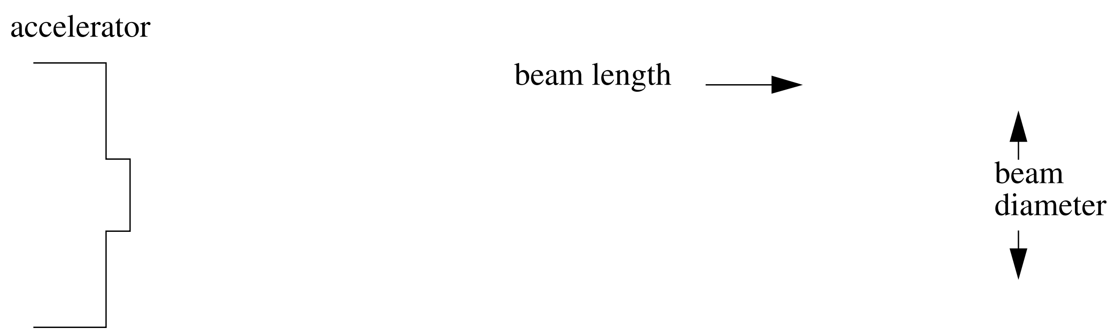

Figure 1. Schematic of a beam propagating (to the right) after being emitted by an accelerating device. The focusing of the beam at the front and oscillations at the back are discussed in the text.

The first effect is thermal noise in the beam itself which causes the beam to spread as it propagates. Beam confinement is accomplished using a laser (or low energy electron gun) to ionize a gas, creating a plasma channel. As the beam propagates along the channel, the plasma electrons are ejected by the net electrostatic field, leaving a positive ion channel which confines the beam. This mode of propagation is know as the ion-focused regime [4]. Beam propagation in this regime is well-behaved until instabilities associated with channel ion motion begin to grow.

One of these effects is known as the ion-hose instability [5]. The first electrons through the channel (at the front of the beam) displace the channel ions as they oscillate back and forth across the channel due to the restoring force of the ions. Any initial inhomogeneity in the beam will cause the channel to become “wavy”. Electrons passing by at a later time “see” this perturbed channel and are induced to greater amplitude oscillations. The effect cascades until at later times, a large

Proceedings of “5th SIAM Conference on Parallel Processing for Scientific Computing” (SIAM Press), March 1991, Houston, TX, p. 507-512.

hose-like wave is induced at the back of the beam with both channel ions and beam electrons undergoing large (many centimeters) oscillations. This is indicated at the left of the schematic in Figure 1. Additionally, electrons introduced into the channel at different lateral positions will oscillate with different characteristic wavelengths due to the non-linear restoring force of the channel ions. The phases of these different oscillations mix as the beam propagates forward, turning the lateral displacements of the ion hose wave into a broader and broader diameter beam.

The simulation models all of these effects [6]. It can also include additional ions that are created in the channel by a laser that is pulsed coincident with the beam. This is sometimes done in experiments to produce a higher channel charge at the back of the beam to help damp the undesired oscillations.

Implementation

A typical simulation consists of ~10^5 particles of each of several species (e.g. beam electrons, channel ions, channel electrons, laser-induced ions) and is run for one or two thousand timesteps. The beam and channel are divided into slices (perpendicular to the direction of the beam propagation) as in Figure 2, each consisting of a few hundred particles of each species.

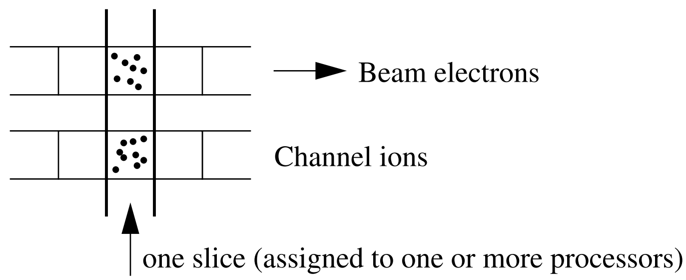

Figure 2. Partitioning of the beam electrons and channel ions into slices along the beam propagation direction. At one timestep only particles in the same slice interact with each other.

Because the beam is moving at essentially the speed of light and the wavelengths of interest are much longer than both the diameter of the beam and its amplitude of oscillation, the paraxial approximation can be invoked. This allows beam and plasma particles to be modeled as long filaments [6], so that the force on each particle is only the sum of all the beam-beam, channel-channel, and beam-channel interactions in its own slice. At each timestep, these interactions are calculated for each slice (from the relativistic Maxwell equations) and the particles moved accordingly in the transverse directions. Then the beam electrons shift forward one slice (while the channel particles remain in place) and at the next timestep the beam interacts with a new slice of the channel. Because the slice calculations are independent of each other, a natural decomposition of the problem is to have one or more processors assigned to each slice of the simulation. The force computation is then perfectly parallel; communication between processors only need occur among processors working on a single slice and when the beam shifts forward between timesteps.

The simulation begins with only one beam and channel slice. At the next timestep there are two, then three, etc. After a few hundred or more timesteps the beam will reach its full length (several hundred slices) and then propagate forward through new channel particles for the remainder of the simulation. Effective parallelization requires that at early times many processors do work on a few slices and that at later times each processor work on several slices. The goal of a load-balancing scheme is to make a smooth transition from one regime to the other. Additional desirable properties of a processor decomposition are that (1) the assignment of processors to one slice be a subcube of the entire hypercube (for either the nCUBE or CM), so that the exchange of information within a slice can be done optimally, and that (2) between timesteps only beam particles need be shifted forward which is the minimum required communication.

A mapping of the processors which satisfies these criteria and which we have implemented on the nCUBE is illustrated in Figure 3. It is a 2-d Gray coding that unfolds as the beam length grows. At any instant processors in the same vertical column work on the same slice and are a sub-cube of the hypercube. When the number of slices grows larger than the number of columns in the 2-d mapping, the processors “unfold” as illustrated. The unfolding preserves the lateral Gray-coded geometry so that between any two timesteps beam particles can be shifted to the right to a nearest neighbor processor. We note that some other 2-d Gray-codings will meet these criteria as well [7]; this one is simple to visualize in terms of the unfolding property. Some mappings, however, do not satisfy criteria (2) and require both beam particles to be passed forward and channel particles to be passed backward as the simulation domain grows. When the beam has grown sufficiently, the processors are completely unfolded (to a 1-d Gray-coding) and each processor stores multiple slices if necessary.

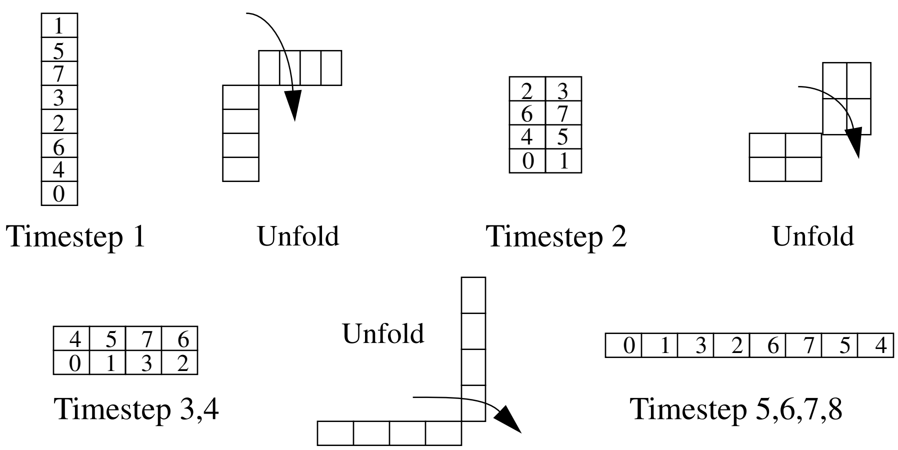

Figure 3. Schematic of a 2-d folded Gray-coding for 8 processors for the first 8 timesteps of the simulation. The processors unfold as the simulation progresses until they are in a 1-d Gray-coded order (bottom right).

Results

We have implemented the load-balancing scheme described above on the nCUBE 2 hypercube since it allows us to explicitly map processors to the problem domain. Our nCUBE has 1024 processors each with 4 Mbytes of memory. Our CM-2 has 16K processors (512 floating point processors) and 2 Gbytes of memory. On the CM, the particles and slices are stored as 2-d arrays. The force calculation is done by scrolling a copy of the particles past the original array so that each particle “sees” all the others in its slice. Similarly, the beam particles can be shifted ahead one array location (in the slice dimension) between timesteps. Dynamic memory allocation for arrays on the CM allows small slice-dimensioned arrays to be used at the beginning of the simulation and large slice dimensions to be used at the end. Transferring data between the growing arrays (analogous to unfolding the processors for the nCUBE) requires general router communication but only needs to take place a few times during the simulation.

Relative performance of the two parallel machines vs. the CRAY-YMP is shown in Table I. These are timings to perform one timestep of a simulation with 1024 slices. Each slice contains

1024 particles of various species for a total of ~10^6 particles. On the CRAY and nCUBE there are two force solvers implemented, a N^2 pairwise solver and a NlnN Fourier solver (where N is the # of particles in one slice), one of which may be more accurate or faster for a particular problem [3]. Only the N^2 solver has been implemented on the CM, because the NlnN solver as written needs all particles in a slice to reside on a single floating point processor which is not feasible for small slice dimensions.

	CRAY-YMP (1 processor)		nCUBE 2 (1024 nodes)		CM-2 (16K)
	N^2	NlnN	N^2	NlnN	N^2
Time	121.9	36.0	6.71	2.82	32.0
Speed-up	---	---	18.2x	12.8x	3.8x

Table I. Timings (in seconds) for one timestep of 1024 slices each containing 1024 particles on each of three machines. The different solvers for the CRAY and nCUBE are discussed in the text.

A few comments on the timings in Table I are in order. First, the CM times suffer from the fact that the calculation of a few dozen constants for the force and motion equations is not a SIMD computation. Hence we pre-compute them and store them with the appropriate particles. This means a lot of extra data must be shifted with the particles. Additionally, the problem sizes we are interested in (~1000 slices) are not quite large enough to take advantage of the new slicewise compiler on the CM which would potentially speed-up the force computation considerably by allowing all particles in a slice to be stored on one floating point processor.

Secondly, the timings represent “best-case” speed-ups. The N^2 solver on the CRAY is well vectorized and runs at ~110 Mflops. Each node of the nCUBE is running at 2.0 Mflops on the N^2 solver for a peak performance of ~2 Gflops. In practice, there are inefficiencies during the beam growth phase (the folded Gray-code technique is 75% efficient on average), from not always running a # of slices that is a multiple of the # of processors, and from slices having unequal numbers of particles (e.g. from laser-induced ionization). The nCUBE version of the code is our production parallel version (because of its NlnN solver capability), and for the largest runs we have done (~2000 timesteps, ~10^6 particles) it is more typically 50-60% efficient for a 10x speed-up over the single processor YMP. This has given us the capability to do overnight runs on the nCUBE that are the equivalent of 50-100 hours of CRAY time.

A snapshot of a single timestep from a simulation run is shown in Figure 4. Five hundred slices are displayed each with beam electrons and channel ions. The beam is the dark bunch of particles in the center of the picture; the ions can only be distinguished at the left of the figure where they appear as streaming lines of particles. The scale of the figure has been skewed; the beam diameter is less than 2 centimeters while it is 10’s of meters long. Similar to the schematic in Figure 1, the beam has been focused at the front (right side of figure) by the ion channel, gradually grows in diameter due to phase mixing, and at the back (left side of figure) both the beam and channel have become unstable and are oscillating through several centimeters amplitude. The graph plots the radius of the beam and channel from front to back and shows oscillations corresponding to those in the snapshot.

Conclusions

Both the MIMD nCUBE and SIMD CM have proven to be fast machines (competitive with the CRAY) for performing particle simulations that require no grid for field computations. This particular application turned out to be a somewhat better match for the MIMD architecture because it allowed greater flexibility in choice of solvers and mapping of processors to the beam length. The folded Gray-coding technique employed here could be useful in other kinds of simulations, where one would like to remap processors to a simulation domain that expands or shrinks dynamically in one (or more) of its dimensions.

Figure 4. Snapshot and graph of a simulation run after the 500th timestep. There are 500 slices and a total of 1/4 million ions and electrons.

References

[1] P. C. Liewer, V. K. Decyk, "A general concurrent algorithm for plasma particle-in-cell simulation codes", J Comp Phys, 85 (1989), p. 302.

[2] J. S. Wagner, "Buckshot: description and comparison to experiment", Proc. of the SDIO/DARPA/Services Ann. Prop. Review, Monterey, CA (1987).

[3] I. R. Shokair, J. S. Wagner, "Gridless electrostatic field solver for particle simulation codes in cylindrical geometry", Sandia Report SAND90-1249, available from Sandia National Labs, Albuquerque, NM 87185.

[4] H. L. Buchanan, "Electron beam propagation in the ion-focused regime", Phys. Fluids, 30 (1987), p. 231.

[5] R. J. Lipinski, *et.al.*, "Measurement of the electron-ion-hose instability growth rate", Phys. Fluids B, 2 (1990), p. 2704.

[6] I. R. Shokair, J. S. Wagner, "Analysis and benchmarking calculations for the Buckshot code", Sandia Report SAND87-2015, available from Sandia National Labs, Albuquerque, NM 87185.

[7] G. Fox, *et.al.*, Solving Problems on Concurrent Processors, Vol I, Prentice Hall, NJ, 1988, p. 260.

Proceedings of "5th SIAM Conference on Parallel Processing for Scientific Computing" (SIAM Press), March 1991, Houston, TX, p. 507-512.