Progress in parallelizing XOOPIC

P. J. Mardahl and J. P. Verboncoeur
Cory Hall Box 173
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94720-1770 USA
peterm@langmuir.eecs.berkeley.edu
USA: (510) 642-1297
Acknowledgements: AFSOR/AASERT F49620-94-1-0387, AFOSR/MURI F49620-95-1-0253

1 Introduction

XOOPIC was developed as a non-parallel 2-d electromagnetic particle-in-cell simulation program. It was intended for microwave beam devices, but the object oriented design of the code made it relatively simple to extend the program to electrostatic models with gas chemistry.

The motivation for the parallelization is performance. Many problems, such as physically large devices, high density/high current devices, and 3d models, require days to simulate on a fast single-node workstation. Even a naive parallelization might enable a 10x speedup for such problems. However, an effort is being made to perform the parallelization carefully so that scaling to larger speedups (100x) is possible for certain problems.

The objectives of this project include portability and free distribution. The MPI library and GNU g++ compiler provide freely distributable cross-platform tools for accomplishing this task.

The non-parallel code had the following main features as of the last release: 2-dimensional, cylindrical or Cartesian geometries, electrostatic or electromagnetic field solve, stepwise oblique boundary conditions, time-dependent particle and field injection, metal, dielectric, and wave launching/absorbing boundary conditions, and multiple species with gas chemistry.

2 Parallelization Strategy

The strategy for parallelization is to partition the physical model into computational regions (Fig. 1), called SpatialRegions, and assign a CPU to each SpatialRegion. The computational domains are presently fixed, and do not vary with time or load.

SpatialRegions are connected to each other by "virtual" boundaries, called SpatialRegionBoundaries. These boundaries are "objects" in the C++ style, which means that the code for managing the data structure is conceptually bundled with the data structure. In this way, code is hidden from the rest of the program, or "encapsulated".

The SpatialRegionBoundaries take advantage of encapsulation to make sweeping changes to XOOPIC unnecessary in order to parallelize it. Only the SpatialRegionBoundaries "know" that XOOPIC is parallel. (Actually other parts must too, such as the initialization code.) SpatialRegionBoundaries handle the special field solve at the virtual boundaries, are responsible for passing particles across regions, and must perform the necessary communication between regions. Many of the existing physics models written into the non-parallel version of XOOPIC are immediately valid in the parallel version, without modification. One exception is the electrostatic field solve: because solving Poisson's equations requires global information, it is not straightforward to adapt the existing Poisson solve in XOOPIC to the parallel version. A parallel Poisson solve will therefore have to be added to XOOPIC to handle electrostatic problems.

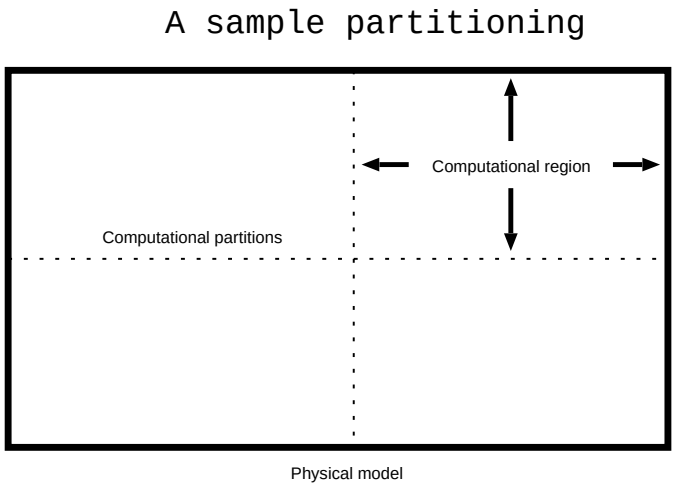

Figure 1: Schematic of computational partitioning of a physical model.

The program flow diagram for the design is shown in Figure 2. Note the messages which must be passed, denoted as "msg #x", and the dashed arrows indicate blocking points where the processors must wait for receipt of the messages. Upon computing the charge density or current density, prior to the field solve, the SpatialRegions must communicate the fields at the facing edges to neighboring SpatialRegions. The fields elements which are passed are shown in Figure 3; those contained in the dashed box are solved redundantly by both SpatialRegions. Once the communication is initiated, the SpatialRegions can solve for the volumetric fields. For systems with a small surface to volume ratio, the message containing the neighboring fields should arrive well before the internal field solve has completed, resulting in zero wait for this message. The next message is sent when particles cross boundaries during the particle advance. This message must arrive before the next step, emission of particles from boundaries, can occur. This is because the particles passed from one SpatialRegion to another via VirtualBoundaries are treated as particles collected and emitted in the respective regions. This message may incur some wait as currently structured.

3 Present implementation status

The present implementation of parallel XOOPIC has the following characteristics: simple 1d partitions, manual partitioning of the problem, electromagnetic models only, diagnostics by computational region, and effective transmission of both fields and particles across virtual boundaries.

Some improvements are needed before the parallelized XOOPIC will be generally useful. Firstly, 2-d partitioning needs to be added. Secondly, partitioning needs to be automated. It is a considerable effort to manually partition the physical model. Thirdly, a glitch in how the third component of the magnetic field is applied to the particles at the SpatialRegionBoundaries causes the particles to see nearest-grid-point fields there, rather than fields bilinearly weighted from the grid points. Fourthly, global diagnostics have to be added; presently diagnostics for each SpatialRegion are shown independently.

The present implementation has been tested, and scales by number of CPU's as shown in Figure 4. The slope is nearly linear in each case, indicating good scaling. For the 4-processor run on the Intel 200MHz Pentium Pro (tm) machine, the 2-CPU run actually ran faster than 2x the 1-CPU run. Possibly, this is because of improved caching of data: the caches for Pentium Pro(tm) processors are tightly coupled to the processor, and so a 1-CPU run is also using only 1/2 the cache of a 2-CPU

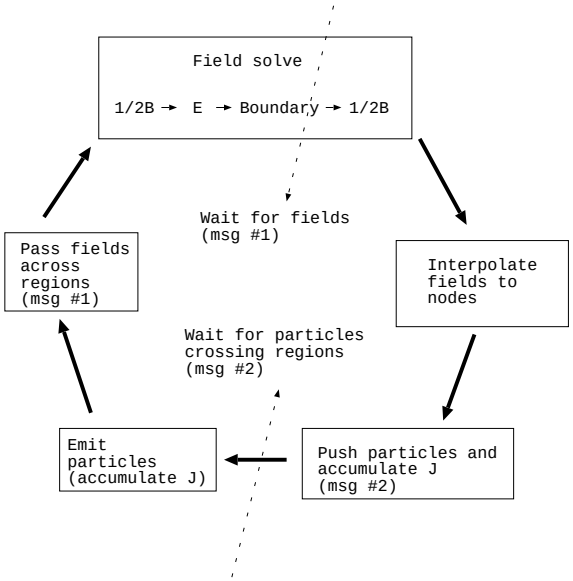

Figure 2: Program flow diagram for a parallelized XOOPIC.

run.

4 Future Work

The following features are unimplemented as yet in parallel XOOPIC: 2-d partitions, bilinear weighting of all fields to particles near SpatialRegionBoundaries, automatic partitioning of the physical model, global diagnostics, automatic balancing of computational load among nodes, partitioning according to network topology, and a parallel electrostatic field solve.

References

[1] J. P. Verboncoeur, A. B. Langdon, and N. T. Gladd, "An object-oriented electromagnetic PIC code." Computer Physics Communications **87** (1995) 199-211. Codes available via http://ptsg.eecs.berkeley.edu.

[2] Havranek, J.J.; Sasser, G.E. Large-scale parallel numerical simulations of the relativistic klystron oscillator. IN: IEEE Conference Record - Abstracts. 1996 IEEE International Conference on Plasma Science (Cat. No.96CH35939). (IEEE Conference Record - Abstracts. 1996 IEEE International Conference on Plasma Science (Cat. No.96CH35939)IEEE Conference Record - Abstracts. 1996 IEEE International Conference on Plasma Science, Boston, MA, USA, 3-5 June 1996). New York, NY, USA: IEEE, 1996. p. 166.

[3] Decyk, V.K.; Naitou, H.; Tokuda, S. Particle-in-cell simulation on the Fujitsu VPP500 parallel computer. Supercomputer, Dec. 1996, vol.12, (no.4):28-32.

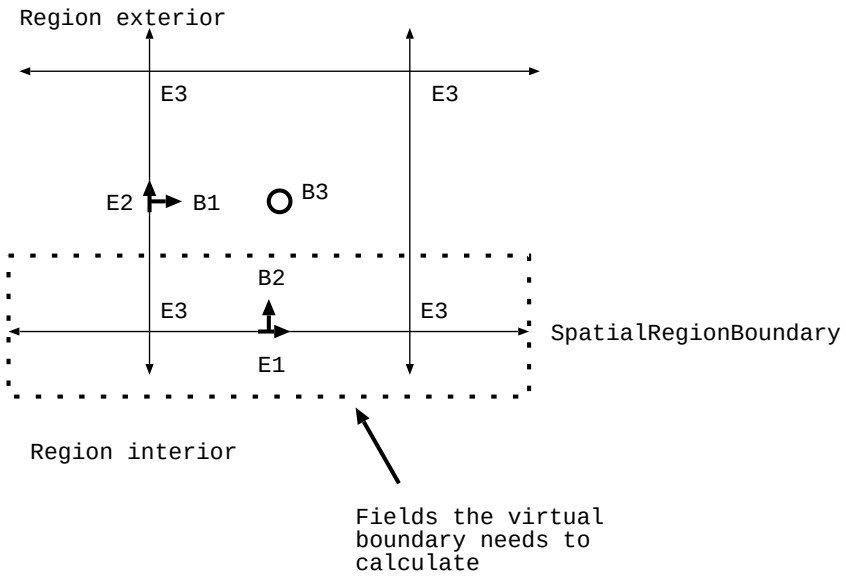

Figure 3: Schematic of fields in one cell of a SpatialRegionBoundary.

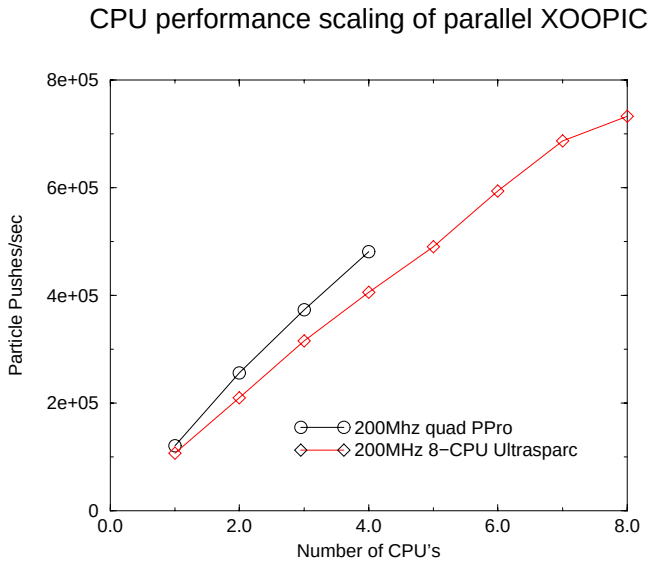

Figure 4: This figure shows the performance gain as a particular physical model is subdived among more and more CPU's. The number of particles per SpatialRegion was 941,000/#CPU's.