Hybrid Parallelization of Particle-in-Cell (PIC) Algorithm For Simulation Of Low Temperature Plasmas

Bhaskar Chaudhury[1], Mihir Shah[1], Unnati Parekh[1], Hasnain Gandhi[1], Paramjeet Desai[1], Keval Shah[1], Anusha Phadnis[1], Miral Shah[1], Mainak Bandyopadhyay [2,3], Arun Chakraborty [2]

[1]Group in Computational Science and HPC, DA-IICT, India - 382007
[2]ITER-India, Institute for Plasma Research (IPR), Gandhinagar, India - 382428
[3]Homi Bhabha National Institute (HBNI), Anushakti Nagar, Mumbai 400094

December 14, 2018

Hybrid Parallelization of Particle-in-Cell Algorithm

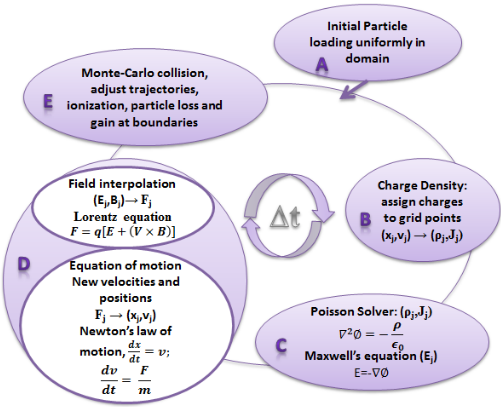
Particle in Cell Simulation (PIC)

Introduction

- PIC is a numerical approximation technique.
- Captures spatial and temporal dynamics of system of charged particles in presence of
  - Electro-Magnetic Field
  - Other forces such as collision
- Actual physical process is approximated by: [1]
  - Discretizing the continuous EM field over grid points.
  - Computational particles capture the behaviour of actual charged particles.
- Computationally expensive because
  - Large number of complex numerical operations, namely interpolations.
  - Large simulation timescale.
  - More computational particles for more accuracy and hence better understanding.
- Objective: Development of a hybrid parallel program (MPI + OpenMP) for accurate simulation of large scale Low Temperature Plasma systems.

Computational Model

- Iterative algorithm.
- Typically requires up to 10^6 iterations to generate meaningful results.



Initial Particle loading uniformly in domain
A
Monte-Carlo collision, adjust trajectories, ionization, particle loss and gain at boundaries
E
Δt
Charge Density: assign charges to grid points
B
$(x_j,v_j) \rightarrow (\rho_j,J_j)$
Field interpolation
$(E_j,B_j) \rightarrow F_j$
Lorentz equation
$F = q[E + (V \times B)]$
D
Equation of motion New velocities and positions
$F_j \rightarrow (x_j,v_j)$
Newton's law of motion, $\frac{dx}{dt} = v;$
$\frac{dv}{dt} = \frac{F}{m}$
Poisson Solver: (ρ_j,J_j)
$\nabla^2 \emptyset = -\frac{\rho}{\epsilon_0}$
C
Maxwell's equation (E_j)
$E=-\nabla\emptyset$

Figure 1: Flowchart of the PIC-MCC algorithm

Implementation Details

- The Electric and Magnetic field are grid quantities and are stored in arrays as double precision floating point values.
- Position (x, y, z) and velocity (v_x. v_y, v_z) for a specific computational particle is stored in a *Particle* structure.
- An array of *Particle* structure is used to maintain the details of all the particles.

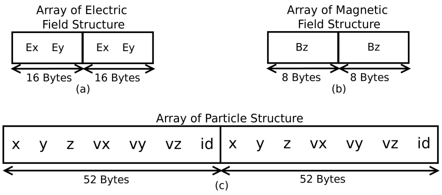


Figure 2: Visualization of Particle and Grid data structure.

Profiling of the Serial PIC Code

- *Mover* and *Charge Deposition* modules prevail the overall execution time.

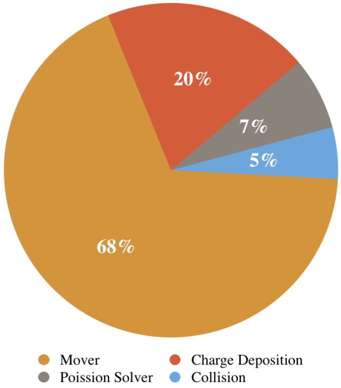


Figure 3: Module-wise contribution to the total serial execution time. Grid size of 1024 $\times$ 1024 with 20 PPC density simulation.

Shared Memory (OpenMP) Parallelization Strategy

Strategy for *Charge Deposition* Module:

- Involves **updating** the grid quantities using the existing particle quantities.
- Parallelization done by dividing equal number of particles among the threads:
  - Allowed because of principle of superposition.
  - Race conditions but better load balancing.
  - To avoid race conditions, each thread has its private copy of grid quantities and later all these private copies are aggregated.

Strategy for *Mover* Module:

- Parallelization done by dividing particles equally among threads.
- Involves updating the particle quantities by **reading** grid quantities, hence no race conditions.
- Efficient load balance using OpenMP scheduling constructs.

Hybrid Parallelization Strategy

The hybrid programming paradigm implements OpenMP based thread-level shared memory parallelization inside MPI based node-level processes.

- Each MPI node has a fixed number of OpenMP threads to exploit shared memory parallelization.
- MPI processes communicate among each other outside the shared memory parallel region.

In our case,

- Each node has its private copy of grid quantities.
- These copies are updated using the thread-level parallelization.
- After individual updates, these node-level private copies are aggregated.

Hybrid Parallelization Strategy

A Pictorial Representation

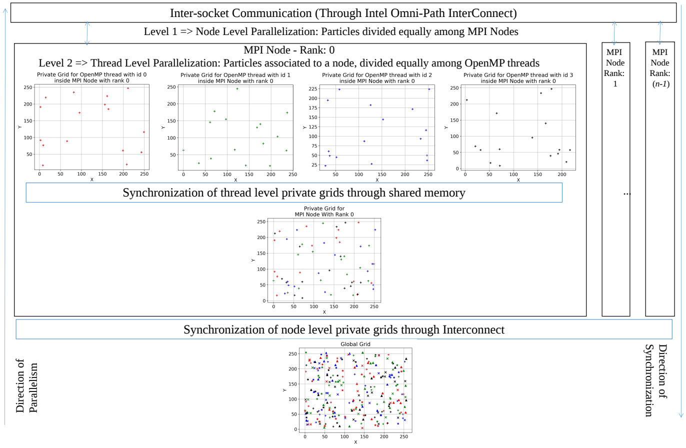

Inter-socket Communication (Through Intel Omni-Path InterConnect)
Level 1 => Node Level Parallelization: Particles divided equally among MPI Nodes
MPI Node - Rank: 0
Level 2 => Thread Level Parallelization: Particles associated to a node, divided equally among OpenMP threads
Private Grid for OpenMP thread with id 0 inside MPI Node with rank 0
Private Grid for OpenMP thread with id 1 inside MPI Node with rank 0
Private Grid for OpenMP thread with id 2 inside MPI Node with rank 0
Private Grid for OpenMP thread with id 3 inside MPI Node with rank 0
Synchronization of thread level private grids through shared memory
Private Grid for MPI Node With Rank 0
MPI Node Rank: 1
MPI Node Rank: (n-1)
...
Synchronization of node level private grids through Interconnect
Global Grid
Direction of Parallelism
Direction of Synchronization
X
Y

Figure 4: Level 1(node-level parallelization): Particle distribution per node and the aggregation of private grids of all nodes and Level 2(thread-level) parallelization inside a node)

Results

Memory Consumption

- The biggest advantage: its memory consumption.
- For large grid sizes, the memory used to store all the particles increases but the memory required per node for the hybrid code is far less than its shared memory parallelization counterpart.

Grid size	OpenMP with 4 cores	Hybrid with 4 nodes, 4 core each	Hybrid with 8 nodes, 4 core each
	Particle array size (MB)	Particle array size (MB)	Particle array size (MB)
512x512	1040	260	130
1024x1024	4160	1040	520
2048x2048	16640	4160	2080

Table 1: Memory consumed in MBs by the particle and grid data structures per MPI node - in both the parallelization strategy for different problem size and fixed particle per cell(PPC) value of 80.

Results

Overall Speedup

Comparison of the speedup for hybrid system with different number of MPI nodes for 100 iterations of the simulation with different grid sizes and particle per cell(PPC) values. Here, Speedup = execution time of OpenMP based code on a multi-core processor with 4 cores / execution time of corresponding hybrid system

Grid size	Speedup for 40 PPC		Speedup for 80 PPC	
	Hybrid 4 nodes, 4 cores per node	Hybrid 8 nodes, 4 cores per node	Hybrid 4 nodes, 4 cores per node	Hybrid 8 nodes, 4 cores per node
512 x 512	2.42	3.46	2.69	4.51
1024 x 1024	3.13	4.92	3.82	6.16
2048 x 2048	3.35	5.57	3.00	7.00

References

Birdsall, C., Langdon, A.. “Plasma Physics via Computer Simulation”. McGraw-Hill, 1991.

R. Rabenseifner, Hybrid Parallel Programming on HPC Platforms,Fifth European Workshop on OpenMP,EWOMP 03, Aachen, Germany, Sept. 22-26, 2003.

V. K. Decyk and T. V. Singh, Particle-in-Cell algorithms for emerging computer architectures,Compute Physics Communications, vol. 185, no. 3, pp. 708719, 2014.

J. Derouillat, A. Beck, F. Perez, T. Vinci, M. Chiaramello, A. Grassi, M. Fle, G. Bouchard, I. Plotnikov, N. Aunai, J. Dargent, C. Riconda, and M. Grech, Smilei: A collaborative, open-source, multi-purpose particle-in-cell code for plasma simulation, Computer Physics Communications, 2018.

Thank you