Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 00 (2013) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2013

High Performance Solvers for Implicit Particle in Cell Simulation

Pawan Kumar[a,b,*], Stefano Markidis[c], Giovanni Lapenta[d], Karl Meerbergen[a], Dirk Roose[a]

[a]*Department of Computer Science, Celestijnenlaan 200A, 3001 Heverlee, Belgium*
[b]*Flanders Exascience Lab (Intel Labs Europe), Heverlee, Belgium*
[c]*PDC Center for High Performance Computing, KTH Royal Institute of Technology, Stockholm, Sweden*
[d]*Center for Plasma Astrophysics, Department of Mathematics, Celestijnenlaan 200B, Heverlee, Belgium*

Abstract

A three-dimensional implicit particle-in-cell (iPIC3D) method implemented by S. Markidis et. al. in ["Multiscale simulations of plasma with iPIC3D", Mathematics and Computers in Simulation, 80(2010), 1509-1519] allows time steps at magnetohydrodynamics time scale. The code requires the solution of two linear systems: a Poisson system related to divergence cleaning, and a system related to a second order formulation of Maxwell equation. In iPIC3D, the former is the most costly.

To reduce the cost of solving the Poisson system, a parallel matrix assembly and partitioning method are implemented, and conjugate gradient and algebraic multigrid (AMG) solvers from the Hypre library are called. The scalability of AMG as a solver is studied for 1D and 3D partitionings and compared to that of CG.

Keywords: Implicit PIC; Algebraic multigrid; Krylov solvers; Distributed memory;

1. Introduction

A three-dimensional implicit particle-in-cell (iPIC3D) code implemented by S. Markidis et.al [5]. allows time steps at magnetohydrodynamics time scale [11, 10]. The code requires the solution of two linear systems: a system with symmetric and positive definite matrix derived from Poisson equation corresponding corresponding to divergence cleaning, and a system with an unsymmetric and indefinite matrix corresponding to the second order formulation of Maxwell equations.

In iPIC3D [5], the conjugate gradient method (CG) is used to solve the Poisson system, and GMRES is used to solve the Maxwell system.

In this paper, we discuss the implementation of a parallel partitioning and assembly of the matrix corresponding to the Poisson problem. We explore efficient algebraic multigrid (AMG) schemes available in the `HYPRE` library [3]. We study the weak scalability of AMG with 1D and 3D partitioning schemes and compare it with CG.

The paper is organized as follows. In section 2, we briefly outline the steps involved in an implicit PIC cycle. In section 3, we describe the discretization schemes used, we explain the assembly and the partitioning scheme for zero and periodic boundary conditions, we also compare the communication overheads of 1D and 3D partitionings. In section 4, we compare the weak scalability of AMG and CG. Finally, section 5 concludes the paper.

*Corresponding author.
E-mail address: pawan.kumar@cs.kuleuven.be.

2. Model for plasma simulations

As in iPIC3D [5], the simulation of the collisonless charged particles such as electrons and ions in plasma is given by the coupling of Vlasov and Maxwell equations in CGS units as follows

Vlasov equation:

$$\frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \frac{\partial f_s}{\partial \mathbf{x}} + \frac{q_s}{m_s}\left(\mathbf{E} + \frac{\mathbf{v} \times \mathbf{B}}{c}\right) \cdot \frac{\partial f_s}{\partial \mathbf{v}} = 0. \tag{1}$$

Maxwell equation:

$$\begin{aligned} \nabla \times \mathbf{E} + \frac{1}{c}\frac{\partial \mathbf{B}}{\partial t} &= 0, \\ \nabla \times \mathbf{B} - \frac{1}{c}\frac{\partial \mathbf{E}}{\partial t} &= \frac{4\pi}{c}\mathbf{J}, \\ \nabla \cdot \mathbf{E} &= 4\pi\rho, \\ \nabla \cdot \mathbf{B} &= 0. \end{aligned}$$

where q_s and m_s are the charge and mass of the species respectively. Here, the distribution function $f_s(\mathbf{x}, \mathbf{v}, t)$ describes the number of particles of species s which have the velocity $\mathbf{v}$ at time t when they are near position $\mathbf{x}$. The symbols $\mathbf{E}$ and $\mathbf{B}$ denote the electric and magnetic fields respectively.

The first order system of Maxwell's equation above could be formulated as a coupled system of second order equations for electric field $\mathbf{E}$ and magnetic field $\mathbf{B}$ [7]. For the electric field, the second order formulation is given as follows

$$\begin{aligned} \nabla \times \nabla \times \mathbf{E} + \frac{1}{c^2}\frac{\partial^2 \mathbf{E}}{\partial t^2} &= -\frac{4\pi}{c^2}\frac{\partial \mathbf{J}}{\partial t}, \\ \nabla(\nabla \cdot \mathbf{E} - 4\pi\rho) &= 0, \\ \nabla \cdot \mathbf{E} &= 4\pi\rho, \end{aligned} \tag{2}$$

along with the appropriate boundary conditions.

As mentioned in [6], there are two possible approaches: either the equation (2) is studied, or as shown in [5], the following identity is used

$$\nabla \times \nabla \times \mathbf{E} = \nabla(\nabla \cdot \mathbf{E}) - \nabla^2 \mathbf{E}. \tag{3}$$

Using identity (3) in (2), we have

$$\nabla^2 \mathbf{E} - \frac{1}{c^2}\frac{\partial^2 \mathbf{E}}{\partial t^2} = \frac{4\pi}{c^2}\frac{\partial J}{\partial t} + \nabla(\nabla \cdot \mathbf{E}). \tag{4}$$

Now using Gauss's law $\nabla \cdot \mathbf{E} = 4\pi\rho$, equation (4) becomes

$$\nabla^2 \mathbf{E} - \frac{1}{c^2}\frac{\partial^2 \mathbf{E}}{\partial t^2} = \frac{4\pi}{c^2}\frac{\partial J}{\partial t} + 4\pi\nabla\rho. \tag{5}$$

While using a numerical scheme, it is possible that Gauss's law $\nabla \cdot \mathbf{E} = 4\pi\rho$ does not hold and a correction is required, we shall come to this shortly. Once the electric field is computed from above, the magnetic field is given by the Faraday's law of induction

$$\frac{\partial \mathbf{B}}{\partial t} = -c\nabla \times \mathbf{E}. \tag{6}$$

Equations (5) and (6) lead to solutions that are consistent with the first order equations [6].

The distribution function corresponding to a species s is a collection of N_s computational particles each having a shape function S, velocity $\mathbf{v}_p$ and position $\mathbf{x}_p = (x_p, y_p, z_p)$, and is given as follows

$$f_s(\mathbf{x}, \mathbf{v}, t) = \sum_{p=1}^{N_s} S(x - x_p) S(y - y_p) S(z - z_p) \delta(\mathbf{v} - \mathbf{v_p}), \tag{7}$$

where δ is the Dirac's delta function. In iPIC3D, the shape function S is choosen as a B-spline function of order l in each direction. So, if Δx, Δy, and Δz are grid spacings in the $x-$, $y-$, and $z-$direction, the shape functions are given as follows

$$S(x - x_p) = b_l \frac{(x - x_p)}{\Delta x},\ S(y - y_p) = b_l \frac{(y - y_p)}{\Delta y},\ S(z - z_p) = b_l \frac{(z - z_p)}{\Delta z}.$$

One of the most popular choices for the shape functions is the cloud-in-cell (CIC) PIC scheme [4] which has first order linear B-splines as shape functions. Substituting (7) in (1), the equations for $\mathbf{x}_p$ and $\mathbf{v}_p$ are given as follows

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{v}_p, \tag{8}$$

$$\frac{d\mathbf{v}_p}{dt} = \frac{q_s}{m_s} \left(\mathbf{E}_p + \frac{\mathbf{v}_p \times \mathbf{B}_p}{c} \right). \tag{9}$$

The average electric field $\mathbf{E}_p$ and the magnetic field $\mathbf{B}_p$ are given as follows

$$\mathbf{E}_p = \int_V \mathbf{E}(\mathbf{x}) S(\mathbf{x} - \mathbf{x}_p), \tag{10}$$

$$\mathbf{B}_p = \int_V \mathbf{B}(\mathbf{x}) S(\mathbf{x} - \mathbf{x}_p) d\mathbf{x}. \tag{11}$$

The Maxwell's equations are solved on a grid and the interpolation functions $W(x_g - x_p)$ given as follows

$$W(x - x_p) = \int_{-\infty}^{\infty} S(x - x_p) b_0 \left(\frac{x - x_p}{\Delta x} \right) dx = \frac{b_1(x - x_p)}{\Delta x}$$

are used to transfer the information between particles and the grid. In the formula above, we used the general property of the B-splines $b_{l+1}(\xi) = \int b_l(\xi') b_0(\xi - \xi') d\xi'$. If $\mathbf{E}_g$ and $\mathbf{B}_g$ denote the field values corresponding to cell g, then using (10) and (11), the electric and magnetic fields acting on the particles are expressed as follows

$$\mathbf{E}_p = \sum_g \mathbf{E}_g W(\mathbf{x} - \mathbf{x}_p),$$

$$\mathbf{B}_p = \sum_g \mathbf{B}_g W(\mathbf{x} - \mathbf{x}_p).$$

The moments of the distribution function ρ_g^n, $\mathbf{J}_g^n$, and $\mathbf{P}_g^n$ are obtained by iterating over the N_s particles of the n_s species as follows

$$\{\rho^n, \mathbf{J}^n, \mathbf{P}^n\}_g = \sum_s^{n_s} \sum_p^{N_s} q_s \{1, \mathbf{v}_p^n, \mathbf{v}_p^n \mathbf{v}_p^n\} W(\mathbf{x} - \mathbf{x}_p^n). \tag{12}$$

The implicit time differenciation of (5) reads

$$\mathbf{E}^{n+1} - (c\Delta t)^2 \nabla^2 \mathbf{E}^{n+1} = \mathbf{E}^n + c\Delta t \left(\nabla \times \mathbf{B}^n - \frac{4\pi}{c} \mathbf{J}^{n+1/2} \right) - (c\Delta t)^2 \nabla 4\pi \rho^{n+1}. \tag{13}$$

Using (6), the magnetic field is advanced in time as follows

$$\mathbf{B}^{n+1} = \mathbf{B}^n - (c\Delta t)\nabla \times \mathbf{E}^{n+1}. \tag{14}$$

In iPIC3D, the charge ρ and current density $\mathbf{J}$ are extrapolated using a Taylor expansion as follows

$$W(\mathbf{x} - \mathbf{x}_p^{n+1}) \approx W(\mathbf{x} - \mathbf{x}_p^n) + (\mathbf{x}^n - \mathbf{x}_p^{n+1})\nabla W(\mathbf{x} - \mathbf{x}_p^n) + \dots$$

Using $(\mathbf{x}^n - \mathbf{x}_p^n) = \overline{\mathbf{v}}_p \Delta t$, we have

$$W(\mathbf{x} - \mathbf{x}_p^{n+1}) \approx W(\mathbf{x} - \mathbf{x}_p^n) + \overline{\mathbf{v}}_p \Delta t \nabla W(\mathbf{x} - \mathbf{x}_p^n) + \dots \tag{15}$$

where $\overline{\mathbf{v}}_p = (\mathbf{v}_p^n + \mathbf{v}_p^{n+1})/2$ is the average velocity of the particle. Expressing $\overline{\mathbf{v}}_p$ as a function of $\mathbf{E}^{n+1}$ and then using (12) to obtain the extrapolated values and then inserting these extrapolated values in (13) and keeping terms up to the second order in Δt, we obtain the following equation for $\mathbf{E}^{n+1}$

$$(\mathbf{I} + \mathcal{X}^n) \cdot \mathbf{E}^{n+1} - (x\Delta t)^2(\nabla^2 \mathbf{E}^{n+1} + \nabla\nabla \cdot (\mathcal{X}^n \cdot \mathbf{E}^{n+1})) = \mathbf{E}^n + c\Delta t \left(\nabla \times \mathbf{B}^n - \frac{4\pi}{c}\widehat{\mathbf{J}}^n\right) - (c\Delta t^2)\nabla 4\pi\widehat{\rho}^n \tag{16}$$

where $\mathbf{I}$ is the identity matrix and $\mathcal{X}$ is called implicit susceptibility, and is defined as follows

$$\mathcal{X}\cdot = \sum_{n_s} \mathcal{X}_s \cdot, \quad \mathcal{X}_s^n \cdot \equiv \frac{1}{2}(\omega_{ps}\Delta t)^2 \left(\mathbf{\Omega_s}\frac{\Delta t}{2}\right)\cdot, \tag{17}$$

where $\widehat{\rho}^n$ and $\widehat{\mathbf{J}}^n$ denote

$$\begin{aligned}
\widehat{\rho}^n &= \rho^n - \Delta t \nabla \cdot \widehat{\mathbf{J}}^n, \\
\widehat{\mathbf{J}}^n &= \sum_s^{n_s} R\left(\mathbf{\Omega_s}\frac{\Delta t}{2}\right) \cdot \left(\mathbf{J}_s^n - \frac{\Delta t}{2}\nabla \mathbf{P}_s^n\right),
\end{aligned}$$

and $R(\mathbf{\Omega_s}, \Delta t/2)$ is a rotation transformation defined as follows

$$\begin{bmatrix}
1 + \left(\Omega_{s_x}\frac{\Delta t}{2}\right)^2 & \Omega_{s_z}\frac{\Delta t}{2} + \Omega_{s_x}\Omega_{s_y}\left(\frac{\Delta t}{2}\right)^2 & -\Omega_{s_y}\frac{\Delta t}{2} + \Omega_{s_x}\Omega_{s_z}\left(\frac{\Delta t}{2}\right)^2 \\
-\Omega_{s_z}\frac{\Delta t}{2} + \Omega_{s_x}\Omega_{s_y}\left(\frac{\Delta t}{2}\right)^2 & 1 + \left(\Omega_{s_y}\frac{\Delta t}{2}\right)^2 & \Omega_{s_x}\frac{\Delta t}{2} + \Omega_{s_y}\Omega_{s_z}\left(\frac{\Delta t}{2}\right)^2 \\
\Omega_{s_y}\frac{\Delta t}{2} + \Omega_{s_x}\Omega_{s_z}\left(\frac{\Delta t}{2}\right)^2 & -\Omega_{s_x}\frac{\Delta t}{2} + \Omega_{s_y}\Omega_{s_z}\left(\frac{\Delta t}{2}\right)^2 & 1 + \left(\Omega_{s_z}\frac{\Delta t}{2}\right)^2
\end{bmatrix}$$

, where $\mathbf{\Omega}_s \equiv q_s/m_s\mathbf{B}^n/c$, and $\omega_{ps} = \sqrt{(4\pi\rho_s q_s)/m_s)}$ are the cyclotron frequency vector and plasma frequency for species s.

Once the electric field is found, it must be corrected to ensure the charge density continuity equation is satisfied. In iPIC3D, we use a technique proposed by Boris [9].

$$\begin{aligned}
\tilde{\mathbf{E}}^{n+1} &= \mathbf{E}^{n+1} - \nabla\Phi, \\
\nabla^2\Phi &= \nabla \cdot \mathbf{E}^{n+1} - 4\pi\rho^n
\end{aligned} \tag{18}$$

Here ∇^2 is the Laplacian, and we shall frequently refer to $\nabla \cdot \mathbf{E}^{n+1} - 4\pi\rho^n$ as the right hand side. We refer the reader to [5] for further details on complete iPIC3D cycle.

3. Linear solvers in implicit particle in cell method

As shown in section 2, the implicit PIC method requires the solution of two linear s: one corresponding to the Maxwell system (13), and another one corresponding to the divergence cleaning (18). In iPIC3D, the solution of (13) is achieved by a GMRES iterative solver. Here we focus on the solution of (18). The Laplacian operator in

(18) is approximated by the classical second order finite difference approximation of the spatial derivatives at grid point $(i,\ j,\ k)$ as follows

$$\nabla^2\Phi|_{i,j,k} \approx \frac{\Phi_{i+1,j,k} - 2\Phi_{i,j,k} + \Phi_{i-1,j,k}}{{h_x}^2} + \frac{\Phi_{i,j+1,k} - 2\Phi_{i,j,k} + \Phi_{i,j-1,k}}{{h_y}^2} + \frac{\Phi_{i,j,k+1} - 2\Phi_{i,j,k} + \Phi_{i,j,k-1}}{{h_z}^2},$$

where h_x, h_y, and h_z are the discretization step sizes along the spatial directions X, Y, and Z respectively and $\Phi_{i,j,k} = \Phi(ih_x, jh_y, kh_z)$. For simplicity, we assume that $h_x = h_y = h_z = h$. Scaling both sides and absorbing the h^2 term in the right hand side, we have the following equation for each interior grid point

$$\Phi_{i+1,j,k} + \Phi_{i-1,j,k} + \Phi_{i,j+1,k} + \Phi_{i,j-1,k} + \Phi_{i,j,k+1} + \Phi_{i,j,k-1} - 6\Phi_{i,j,k} = h^2(\nabla \cdot \mathbf{E}^{n+1} - 4\pi\rho^n)_{i,j,k}$$

We obtain the following linear system

$$\mathbf{Ax} = \mathbf{b}$$

where $\mathbf{A}$ is the discrete Laplacian, $\mathbf{x}$ is the discrete potential, and $\mathbf{b}$ is a discrete representation of $h^2(\mathbf{E}^{n+1} - 4\pi\rho^n)$ and the boundary conditions. We now explain the parallel assembly of matrix $\mathbf{A}$.

3.1. Parallel Partitioning and Matrix Assembly

The iPIC3D code is written in the message passing programming model, using the MPI library for communication. Each MPI process is responsible for a part of the regular grid. Hence the grid must be partitioned.

Partitioning leads to a renumbering of the grid points which affects the convergence of the SOR smoother, hence affects the multigrid solver. To make a comprehensive study, we consider the following two parallel partitioning strategies

- 1D partitioning: In this simple partitioning each partition consists of a set of consecutive planes and each partition belongs to one MPI process. Notice that this partitioning retains the lexicographic numbering of the original unpartitioned grid.
- 3D partitioning: Here the partitions are geometrical cubical portions of the original grid, where each of these cubes are assigned to one MPI process as indicated below. A 3D partitioning into cube subdomains leads to minimal communication volume and better scalability.

Let T_{seq} be the execution time for the sequential algorithm, and let T_{par} be the execution time for the parallel algorithm with p processors, then the speedup is given by $\frac{T_{seq}}{T_{par}}$. Let T_{cal} be the time spent in computation, and let T_{com} be the time spent in communication, then we have $T_{par} = T_{cal} + T_{com}$. Assuming a perfect load balance and no other overheads, we can write $T_{seq} = pT_{cal}$. The parallel efficiency E_p is defined as follows

$$E = \frac{S}{p} = \frac{pT_{cal}}{p(T_{cal} + T_{com})} = \frac{1}{1 + \frac{T_{com}}{T_{cal}}} = \frac{1}{1 + f_c}$$

where f_c is the communication overhead. In a stencil based computation such as a sparse matrix vector product the number of grid points on the surface of the partition represents the amount of communication involved. Thus we have

$$f_c = O\left(\frac{\text{Number of grid points on the surface}}{\text{Number of grid points in the interior}}\right)$$

Let M be the total number of grid points and let $n_x \times n_y \times n_z$ be the number of grid points in each partition. For a 1D partitioning along X-direction $n_x = \frac{M^{1/3}}{p}, n_y = n_z = M^{1/3}$, and $n_x = n_y = n_z = \frac{M^{1/3}}{p^{1/3}}$ for the 3D partitioning. The communication overhead for the 1D partitioning is $f_c = O\left(\frac{p}{M^{1/3}}\right)$ and for the 3D partitioning, we have $f_c = O\left(\frac{p^{1/3}}{M^{1/3}}\right)$. Thus the communication overhead for a 3D partitioning grows at a much slower rate with increasing number of partitions compared to the 1D partitioning. See also [8].

In Figure (1), we show the assembly of the matrix in the serial case for the lexicographic traversal of the grid. A grid point numbered i and its neighbor connections are placed on the row numbered i. If we assume a 3D

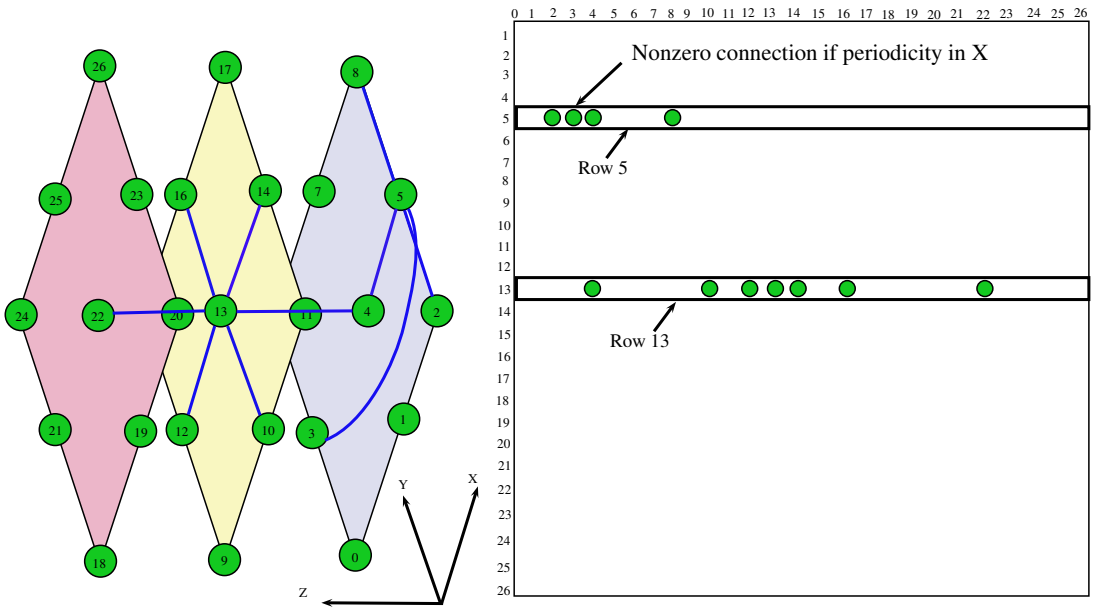

Fig. 1. Sequential assembly of matrix

processor grid of size $p_x \times p_y \times p_z$, then each processor owns a cube with roughly $\frac{M_x}{p_x} \times \frac{M_y}{p_y} \times \frac{M_z}{p_z}$ grid points. All the grid points of the interior cubes has full 7-point stencil regardless of the boundary conditions. For grid points on the boundary of non-interior cubes, appropriate connections are made corresponding to the given boundary conditions. For example, in Figure (2) due to periodicity along the Z-direction, grid point number 57 is connected to grid point number 246. Here we consider lexicographic ordering of the cubes on the processor grid, and a lexicographic ordering of the grid points within the cube.

4. Implementation of Linear solvers

Once the matrix is assembled, we pass the matrix to the HYPRE preconditioning library [3] which contains both preconditioners and solvers including algebraic multigrid. Each processor sets its own matrix entries row by row using the "HYPRE_IJMatrixSetValues" routine of HYPRE. This interface allows to experiment with several coarsening and smoothing schemes of HYPRE.

Algebraic multigrid (AMG) forms a hierarchy of matrices using prolongation operators $\mathbf{I}_{k+1}^{k}$ and restriction operators $\mathbf{I}_{k}^{k+1}$. Let the matrices corresponding to the hierarchy be denoted by $\mathbf{A}^1 = \mathbf{A}, \mathbf{A}^2, \cdots, \mathbf{A}^M$, then the typical V-cycle (adapted from [1]) is shown in Algorithm (1). The setup phase includes the determination of the

Algorithm 1 VCycle(k, $\mathbf{x^k}$, $\mathbf{r^k}$)

if $k = M$ **then**
  Set $\mathbf{x}^M = (\mathbf{A}^M)^{-1}\mathbf{r}^M$.
else
  Relax μ_1 times on $\mathbf{A}^k\mathbf{x}^k = \mathbf{r}^k$.
  Perform coarse grid correction:
  Set $\mathbf{x}^{k+1} = 0$, $\mathbf{r}^{k+1} = \mathbf{I}_k^{k+1}(\mathbf{r}^k - \mathbf{A}^k\mathbf{x}^k)$.
  Solve on level $k+1$ with VCycle($k+1$, $\mathbf{x}^{k+1}$, $\mathbf{r}^{k+1}$).
  Correct the solution by $\mathbf{x}^k \leftarrow \mathbf{x}^k + \mathbf{I}_{k+1}^k x^{k+1}$.
  Relax μ_2 times on $\mathbf{A}^k\mathbf{x}^k = \mathbf{r}^k$.
end if

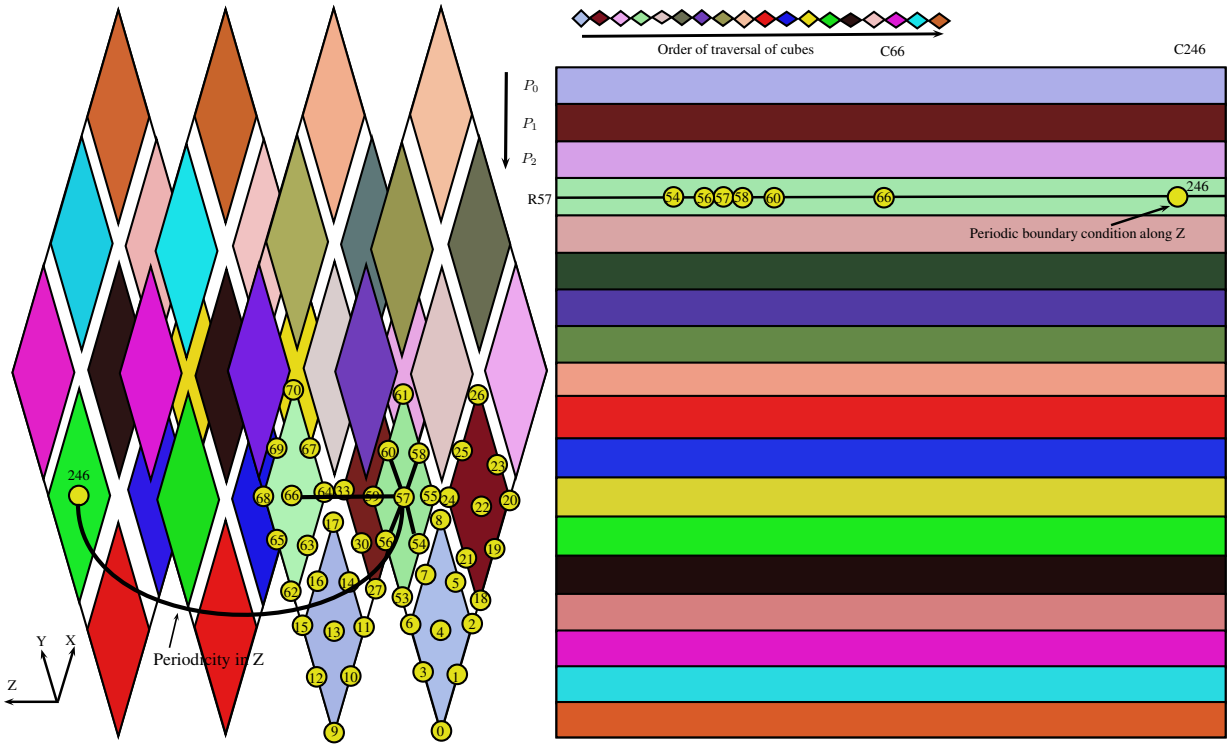

Fig. 2. Left: $9 \times 9 \times 4$ computational grid on $3 \times 3 \times 2$ processor grid, Right: Corresponding Matrix, rows of same color belong to same processors

transfer operators $\mathbf{I}_{k+1}^{k}$ and $\mathbf{I}_{k}^{k+1}$ by partitioning the vertices of the underlying graph into coarse (C-) and fine (F-) vertices [1]. The coarse grid matrices are constructed using Galerkin projection by setting $\mathbf{I}_{k+1}^{k} = (\mathbf{I}_{k}^{k+1})^T$ and $\mathbf{A}^{k+1} = \mathbf{I}_{k}^{k+1}\mathbf{A}^{k}\mathbf{I}_{k+1}^{k}$. For the Poisson problem related to divergence cleaning, the most effective combination we find is the following

- **Hybrid Gauss-Siedel or SOR**: Each processor performs a lexicographic SOR smoother, but neglecting updates from data that belongs to the subdomains handled by another process. One pre- and one post-smoothing is applied, that is, we set $\mu_1 = 1$ and $\mu_2 = 1$.
- **HMIS coarsening**: One pass of Ruge-Stueben coarsening executed independently on each processor (or partition) followed by a parallel coarsening algorithm using independent sets [1].

5. Numerical experiments

The numerical experiments were performed in double precision arithmetic, the iPIC3D and `HYPRE` codes are written in C++ and C respectively. The code is compiled and run on Nehalem processors, quad core Xeon 5560 with 24GB of RAM.

The iterative procedure for solving the Poisson problem is stopped whenever the relative residual norm

$$\frac{||\mathbf{b} - \mathbf{A}\mathbf{x}_k||}{||\mathbf{b}||} < 10^{-3}.$$

The initial solution is set to zero and the source term $\mathbf{b}$ changes in each time step of iPIC3D. In our tests, we test the weak scalability of the solvers by keeping the problem size (i.e., the number of grid points) per processor to be 32^3. We compare CG and AMG with HIMS coarsening denoted by AMG(10) [3] with both 1D and 3D partitioning. For all the test cases, we choose the initial boundary conditions to be zero. However, with other set of boundary conditions, the performance of the linear solvers is similar.

For AMG, the coarsest matrix dimensions are 3, 6, and 9 for problem sizes 32^3, 64^3, and 128^3 on 1, 8, and 64 cores respectively. The system with coarsest matrix is solved by Gaussian elimination.

In Table 1, we first present the execution time for one iteration of the CG and one V-cycle of AMG methods. For CG, the communication overheads are both due to stencil based operations such as sparse matrix vector product, and reduction operations in calculating dot products. On the other hand, for AMG, there are communication overheads solely due to stencil based computations. We notice that AMG with 3D partitioning has lower communication overhead and scales better than the 1D partitioning.

Table 1. Weak scalability execution time per V-cycle of AMG and per iterationof CG. Time shown in seconds.

procs	CG one iteration	AMG(10)-1D V-cycle	AMG(10)-3D V-cycle
1	0.00097	0.005	0.005
8	0.0046	0.024	0.028
64	0.0079	0.047	0.035

We need to solve a sequence of linear systems with same coefficient matrix and different right hand side. Since the setup phase of AMG depends only on the coefficient matrix, we do the setup only once during the first iPIC3D cycle and keep the setup data for subsequent iPIC3D cycles. In Table (2), we compare the weak scalability of the setup phase and the time taken by CG to do one Poisson solve. We observe that the setup time for AMG is of the same order of magnitude to the time taken by CG for one Poisson solve. It is well known that the number of

Table 2. Weak scalability of the setup phase of AMG and one Poisson solve with CG. Time shown in seconds.

procs	CG Poisson solve	AMG(10)-1D Setup	AMG(10)-3D Setup
1	0.04	0.15	0.14
8	0.41	0.73	0.49
64	1.15	1.2	1.2

CG iterations to reach a certain accuracy grows with the problem size. In Table (3), we compare the number of

iterations needed by CG and AMG to solve the system. The iteration count for CG almost doubles with eight fold increase in the problem size. On the other hand, the iteration count for AMG is significantly less, in particular, for AMG with 3D partitioning the number of iterations remains constant on 8 and 64 processors. Although both AMG schemes use the same set of parameters for the same problem, we observe a difference in iteration count. This is explained by the fact that both 1D and 3D partitioning leads to a renumbering (or coloring) of the grid points [2] that affects relaxation schemes such as SOR.

Table 3. Number of iterations needed by CG and AMG solvers

procs	CG	AMG(10)-1D	AMG(10)-3D
1	41	2	2
8	88	5	6
64	144	13	6

In Table (3), we compare the weak scalability of the execution time needed by CG and AMG to solve the system. On 64 cores, AMG(10)-3D is the fastest, and it is roughly 5 times faster than CG and three times faster than AMG(10)-1D. The slow convergence of AMG(10)-1D compared to AMG(10)-3D on 64 cores is both due to communication overhead and higher iteration count, see Table 3). However, on 8 cores, AMG(10)-1D is slightly faster than AMG(10)-3D due to lower iteration count, see Table (3).

Table 4. Weak scalability of the solve phases. Time shown in seconds.

procs	CG	AMG(10)-1D	AMG(10)-3D
1	0.04	0.01	0.01
8	0.41	0.12	0.17
64	1.15	0.62	0.21

References

[1] Van Emden Henson, Ulrike Meier Yang, *BoomerAMG: A parallel algebraic multigrid solver and preconditioner*, Applied numerical mathematics, 41, 155-177, 2002.
[2] L. M. Adams, H. F. Jordan, *Is SOR color blind?*, SIAM J. Sci. Comp., 7(2), 490-506, 1986.
[3] HYPRE, available online at: http://acts.nersc.gov/hypre/
[4] R. Hockney, J. Eastwood, *Computer simulation using particles*, Taylor & Francis, Bristol, 1988.
[5] S. Markidis, G. Lapenta, and R. Uddin, *Multi-scale simulations of plasma with iPIC3D*, Mathematics and Computers in Simulation, 80(2), 1509-1519
[6] P. Ricci, G. Lapenta, and J. U. Brackbill, *A simplified implicit Maxwell solver*, Journal of computational physics, 183, 117-141, 2002.
[7] B. N. Jiang, J. Wu, and L. A. Povinelli, *The origin of spurious solutions in computational electromagnetics*, Journal of computational physics, 183, 117-141, 2002.
[8] D. Roose, R. Van Driessche, *Distributed memory parallel computers and computational fluid dynamics*, Lecture Notes on Computational Fluid Dynamics, LS 1993-04 Von Karman Institute for Fluid Dynamics (VKI, Belgium), 1993.
[9] J. P. Boris, *Relativistic plasma simulation-optimization of a hybrid code*, Proc. Fourth Conf. Num. Sim. Plasmas, Naval Res. Lab, Wash. DC, 3-67, 1970.
[10] G. Lapenta, J. U. Brackbill, P. Ricci *Kinetic approach to microscopic-macroscopic coupling in space and laboratory plasmas*, Physics of plasmas, 13, 055904, 2006.
[11] J. U. Brackbill, D. W. Forslund, *An implicit method for electromagnetic plasma simulation in two dimensions*, Journal of Computational Physics, 46(2), 271-308, 1982.

6. Conclusion

We have implemented a parallel assembly and partitioning scheme for the matrix arising in the Poisson equation corresponding to the divergence cleaning in the implicit particle in cell code iPIC3D. The convergence and scalability of algebraic multigrid schemes is studied for 1D and 3D partitionings. We find that a combination of a block Jacobi with lexicographic SOR as a smoother and HIMS coarsening scheme with 3D partitioning is the best strategy; this combination for AMG is roughly five times faster than the conjugate gradient method.

Acknowledgements

This work is funded by Intel and by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT). We thank Arnaud Beck (Ecole polytechnique) for help with the iPIC3D code.