Progress towards Physics-Based Space Weather Forecasting with Exascale Computing

Giovanni Lapenta, Maria Elena Innocenti, Jorge Amaya, Alec Johnson, Jan Deca, Vyacheslav Olshevsky

Center for Mathematical Plasma Astrophysics, Departement Wiskunde
Leuven Centre for Aero & Space Science, Technology, and Applications (LASA)
Leuven Mathematical Modeling & Computational Science Research Centre (LMCC)
KULeuven, University of Leuven, Celestijnenlaan 200B - bus 2400, B-3001 Heverlee, België.

Stefano Markidis
High Performance Computing and Visualization Department, KTH Royal Institute of Technology, Stockholm, Sweden

Abstract

Particle in cell simulations applied to space weather modelling represent an excellent application for codesign efforts. PIC codes are simple and flexible with many variants addressing different physical conditions (e.g. explicit, implicit, hybrid, gyrokinetic, fluid) and different architectures (e.g. vector, parallel, GPU). It is relatively easy to consider radical changes and test them in a short time. For this reason, the project DEEP funded by the European Commission (www.deep-project.eu) and the Intel Exascience Lab (www.exascience.com) have used PIC as one of their target application for a codesign approach aiming at developing PIC methods for future exascale computers. The present work focuses on the efforts within DEEP. The starting point is the iPic3D implicit PIC approach. Here we report on the analysis of code performance, on the use of GPUs and the new MICs (Intel Phi processors). We describe how the method can be rethought for hybrid architectures composed of MICs and CPUs (as in the new Deep Supercomputer in Juelich, as well as in others). The focus is on a codesign approach where computer science issues motivate modifications of the algorithms used while physics constraints what should be eventually achieved.

Keywords: Space Weather, Particle-In-Cell, Adaptive, Implicit, Exascale, High Performance Computing

1. Introduction

Space weather is the fast-growing area of science that focuses on the conditions in the space amidst the Sun, the Earth and the other planets of our solar system. The Sun is an extremely dynamic source of energy. Besides the electromagnetic radiation (including of course light) that makes life possible, the Sun is a giant dynamo that produces a strongly magnetized plasma that flows away from the Sun at supersonic speeds, arriving at the Earth with velocities ranging from about 200 to 800 Km/s. This wind is very dynamic and varies greatly. Just like Earth winds, it also carries storms. The most powerful space storms are generated irregularly in singular events of mass and energy released by the Sun (called coronal mass ejections). The intensity and frequency of these storms oscillates over the course of a solar cycle, with a period of about 11 years. Peak activity is called solar maximum (occurring in 2000 and now in 2013) and is separated by periods of very low activity at solar minimum (2008-2009). Solar activity also changes on much longer scales. Generations of scientists have monitored the solar activity since the 17th century. The solar cycles of the space age, following the first space trip of Yuri Gagarin, have happened to be particularly active, but a period of prolonged quietness lasting several decades happened in the end of the 17th century (the so-called Maunder minimum corresponding to a period of exceptionally cold winters and insufficient crop production in Europe). Since this period, at least one exceedingly strong event happened in 1859 (the Carrington event), causing severe telegraph disruptions back in those early industrial revolution days.

Preprint submitted to AES October 28, 2013
similar event now would wreak havoc on our space technology and on our ground infrastructure (power and communication lines, transformers, pipelines and computing infrastructure). A Carrington-sized event now could not only render multiple satellites permanently inoperative but could damage beyond repair the largest transformers on the highest voltage long distance power lines. These transformers and satellites would require several months to replace. One does not need a crystal ball to see that a blackout lasting months would greatly disrupt the economy and social tranquility, especially if one takes into account that past blackouts caused by space weather events involved especially the northeast of the American continent (because of the geographical location of the magnetic north pole), an area that includes some of the world’s most active financial trading and data centers.

This brief summary explains why space weather has become a key area of scientific computing. Predicting space weather is a top priority to defend key societal infrastructures and to expand the human presence in space. All major industrialized and developing countries are developing a space weather forecasting capability, with both military and civil agencies contributing to the effort at national and international levels. All these activities center on one key ingredient: the reliance on physics-based high performance computing codes that can follow one of the space storms from its origin on the Sun to its consequences at the Earth. These models are very complex and must include many processes and aspects, including the sheer size of the vastness of the space involved and the great variety of conditions encountered from the hot solar corona to the cold depths of deep space. Density and temperature change over many orders of magnitude. In space, matter is in the form of plasma where atoms are ionized, a process that frees electrons from their bond to the nuclei of atoms. Electrons, being much lighter than the nuclei (for hydrogen the mass ratio between nuclei and electrons is 1836), respond on much smaller scales.

The physics-based description of space requires handling multiple scales in space and time and multiple physical processes. Dealing with multi-scale and multi-physics systems is the key challenge in modern high performance computing (HPC), common to many other engineering and science applications. Even the largest supercomputers can only cover a limited range of scales and physics, posing a great challenge to code-design: how to design new computer architectures and new algorithms so that the gaps are filled and a complete answer can be found for multi-scale and multi-physics challenges.

Here we detail the approach followed in our co-design effort for HPC applied to space weather modeling. The progress reported here follows two intertwined lines.

On the algorithmic line, starting from the implicit particle in cell approach, we report further development designed to handle multiple scales. The temporal discretization has been improved to reach exact energy conservation, an aspect of great practical importance. The spatial discretization has been expanded to allow the use of multiple levels of resolution required to describe different levels of physical refinement in different regions. Finally, a new hybrid approach that extends the ability to include multiple physical descriptions based on kinetic or fluid methods has been developed.

On the hardware side, we report our experience in deploying the methods of space weather modeling not only on the latest generation of petascale supercomputers but also on emerging technologies for HPC such as the now-established GPU and the newly released MIC (commercially known as Xeon Phi).

We conclude the description with a look forward towards the possibilities opened by an exascale computer, estimating the impact that such a new computer will have on space weather modeling and the prospects for the first fully resolved physics-based model of the Earth-space environment. Such a model can provide the future basis for all space weather forecasting systems.

2. Target Application: Kinetic modeling of space weather

The fundamental agents of a space weather simulation are the electric and magnetic fields that permeate space and the particles moving through them.
The field evolution is described by the equations of Maxwell,
\[
\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t},
\]
\[
\nabla \times \mathbf{B} = \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} + \mu_0 \mathbf{J},
\]
for the electric field \( \mathbf{E} \) and the magnetic field \( \mathbf{B} \), and the particle evolution is described by the equations of Newton,
\[
\frac{d\mathbf{x}_p}{dt} = \mathbf{v}_p,
\]
\[
\frac{d\mathbf{v}_p}{dt} = \frac{q_s}{m_s}(\mathbf{E}_p + \mathbf{v}_p \times \mathbf{B}_p),
\]
for the position \( \mathbf{x}_p \) and velocity \( \mathbf{v}_p \) of each particle labelled by \( p \). Particles are organized in species (e.g. electrons, ions) labelled by \( s \) and sharing common properties such as the ratio of charge to mass, \( q_s/m_s = q_p/m_p \). Particle motion feeds back into field evolution by means of the current \( \mathbf{J}(\mathbf{x}) := \sum_p q_p \mathbf{v}_p \delta(\mathbf{x} - \mathbf{x}_p) \) (i.e. the flux of charge per unit area per time); here \( \delta \) is the unit-volume impulse function of Dirac. Equations (1) maintain the divergence conditions
\[
\varepsilon_0 \nabla \cdot \mathbf{E} = \rho,
\]
\[
\nabla \cdot \mathbf{B} = 0,
\]
where \( \rho(\mathbf{x}) := \sum_p q_p \delta(\mathbf{x} - \mathbf{x}_p) \) is the charge density.

In some instances in space weather, relativistic effects need to be considered. Their inclusion does not introduce particular complexities, but for simplicity of presentation they are neglected here.

To solve this interacting system, where the sources of the Maxwell’s equations (the charge density \( \rho \) and current \( \mathbf{J} \)) are determined by the particles and the forces on the particles are determined by the fields, the most commonly used method is the particle-in-cell (PIC) approach. Time is discretized in fixed intervals of length \( \Delta t \), and a grid is introduced to discretize the field equations. The knowledge of the system is then provided by the position and velocity of all particles at the discrete time levels \( t^n = n\Delta t \) and of the fields at prescribed discrete grid locations and at prescribed times: \( \mathbf{E}^g, \mathbf{B}^g \). Here the superscript \( n \) is used to label time, and the subscript \( g \) labels the grid points. A single label is used regardless of the dimensionality of the grid.

The typical conceptual situation of a particle in cell approach is shown in Fig. 1.

A first fundamental feature of the PIC approach is evident: the particle positions are defined in the continuum of the 3D space and their location changes in time; the fields are defined at precise, immutable locations. In terms of data structures, this means that there is no a priori knowledge of where the particles will be relative to the grid.

A second key aspect of the method is that information regarding these two types of entities needs to be exchanged. Since the particles and the grid are not co-located, interpolations are needed. The sources of the Maxwell equations are defined by
\[
\rho(\mathbf{x}) := \sum_p q_p S(\mathbf{x} - \mathbf{x}_p),
\]
\[
\mathbf{J}(\mathbf{x}) := \sum_p q_p \mathbf{v}_p S(\mathbf{x} - \mathbf{x}_p),
\]
where the shape function \( S \) has unit volume and is typically a “top-hat” \( S_0 \) proportional to the indicator function of a mesh cell centered at zero or a smoothed version \( S_t \), defined by \( S_t := S_0 \ast \tilde{S}_{t-1} \), where \( \ast \) denotes
Figure 1: Conceptual representation of the PIC method. The fields are computed on a grid (shown here in 2D for visual simplicity but 3D in the code used here). The particles move in the continuum space. To guide the eye, only the center of the particles is shown, but each particle has a finite square (cubic in 3D) shape around the particle center. The electrons are shown in orange while the ions in yellow. To remind the reader that the ions are 1836 times more massive than the electrons, their dot is shown bigger. In a real simulation there are on average hundreds of particles per cell, but for clarity the figure only shows a few.

Convolution product. The interpolation function \( W := S \ast S_0 \) is then used to interpolate values via

\[
\rho_g^n = \frac{1}{V_g} \sum_p q_p W(x_g - x_p^n),
\]

\[
J_g^n = \frac{1}{V_g} \sum_p q_p v_p^n W(x_g - x_p^n),
\]

(5)

where \( x_g \) is the position of each node of the grid and \( V_g \) is the volume of each cell of the grid. We assume here uniform grids with grid spacing \( \Delta x, \Delta y \) and \( \Delta z \) (thereby \( V_g = \Delta x \Delta y \Delta z \)).

Similarly, the electromagnetic field information is interpolated to the continuum particle location from the discrete nodes of the grid as:

\[
E_p = \sum_g E_g W(x_g - x_p),
\]

\[
B_p = \sum_g B_g W(x_g - x_p).
\]

(6)

In iPic3D we choose \( S = S_0 \) and thus \( W = S_1 \). To be explicit, \( S_\ell(x) = b_\ell(x/\Delta x) b_\ell(y/\Delta y) b_\ell(z/\Delta z) \), where \( b_\ell \) is the B-spline of order \( \ell \) and in particular

\[
b_0(x) = \begin{cases} 
1 & \text{if } |x| < \frac{1}{2}, \\
0 & \text{otherwise}
\end{cases} \quad \text{and} \quad b_1(x) = \begin{cases} 
1 - |x| & \text{if } |x| < 1, \\
0 & \text{otherwise}
\end{cases}
\]

(7)

The evolution equations for the fields and the particles need to be discretized in time. The simplest approach is to solve the two sets of equations in sequence. This is called the explicit approach: the field equations are advanced over one time step \( \Delta t \) using the previous values of the particle properties, and then the particles are advanced by one time step using the fields previously computed. The explicit method is widely popular for its simplicity, allowing an efficient implementation in parallel architectures (e.g. see [2]).

The explicit approach, however, is not viable for space weather simulations. Due to the sequential nature of the approach, each component is temporarily frozen while the other is advancing: the particles stay fixed while the fields evolve, and conversely the particles advance in frozen fields. Needless to say, this is not how nature works. This distortion of the real behavior leads to a limited numerical stability.
In-depth analysis of the explicit scheme [1, 3] shows that the method is stable if three conditions are satisfied. First, the time step must resolve the fastest temporal scale; in space weather problems, this corresponds to the condition $\omega_{pe}\Delta t < 2$, where $\omega_{pe} = \left(\frac{4\pi n_e e^2/m_e}{m_e}\right)^{1/2}$ is the plasma frequency determined by the electron density, mass and charge. Second, the grid spacing is anchored to the smallest spatial scales. In space weather problems, this is determined by the electron thermal speed and electron plasma frequency as $\Delta x \omega_{pe}/v_{th,e} < \xi$. The parameter $\xi$ is of order unity, but its precise value depends upon the details of the method used. For cloud-in-cell interpolation, its value is approximately 3 [3]. Finally, the explicit solution of Maxwell’s equations requires the motion of light waves to be resolved in accordance with a Courant condition: $c\Delta t/\Delta x < 1$, but this latter condition is of lesser importance in space weather simulation since the first two are already more limiting.

Let us briefly consider the actual numbers typical of the Earth-space environment. Figure 2 shows typical values of the temporal and spatial scales encountered. The smallest scales are of the order of hundreds of meters and tens of microseconds. Simulating the whole Earth environment (the typical simulation uses a box of 100 Earth radii on a side, about 600,000 km) for hours or days is clearly not feasible using explicit PIC codes. This limitation is not caused by the intrinsic need to resolve those small scales but is caused only by the numerical limitations of the explicit method due to the three stability constraints listed above. The stability constraints of the explicit method are shown in Fig. 2 by the light blue area.

A space weather simulation would be exceedingly accurate, including all the physics processes of interest, if it were to limit itself to much larger scales. In Fig. 2, the smallest scales of actual interest are indicated by the yellow area.

To avoid the prohibitively large number of time steps and the exceedingly high resolution needed to run realistic problems with explicit particle methods, implicit methods must be preferred [4, 5].

![Figure 2: Visual representation of the typical scales observed in the Earth environment (conditions common in the magnetotail). The stability limit of the explicit PIC method forces the use of the resolution at the level indicated by the light blue box. The desired resolution, instead, is shown by the yellow box.](image)
2.1. Selected algorithm: Implicit PIC

The present study takes as a starting point the implicit approach deployed in the code iPic3D [6]. The implicit approach used in iPic3D retains the coupling between the Maxwell and Newton equations. The particles move in a field that is based on an average over the time step, and similarly the fields are advanced using an average position of the particles during the time step. Of course neither is known before the other is computed, and therefore it becomes impossible to solve one without the other. An iterative procedure becomes necessary.

The starting point of implicit particle methods [7, 8, 9, 10] replaces the explicit time differencing of the equations of motion with an implicit scheme. In our approach, the so-called $\theta$ scheme is used [11, 12]:

$$x_{p}^{n+1} = x_{p}^{n} + v_{p}^{n+1/2} \Delta t,$$

$$v_{p}^{n+1} = v_{p}^{n} + \frac{q_{s} \Delta t}{m_{s}} \left( E_{p}^{n+\theta}(x_{p}^{n+1/2}) + v_{p}^{n+1/2} \times B_{p}^{n}(x_{p}^{n+1/2}) \right),$$

(8)

where the flexibility of defining a decentering parameter $\theta$ is used to vary the properties of the scheme. The quantities at time level $n + \theta$ are computed as weighted averages of the values at the old and new time level, $\Psi^{n+\theta} = \Psi^{n}(1 - \theta) + \Psi^{n+\theta} \theta$. Note that the magnetic field does not introduce stability limits (because the magnetic field does no work) and therefore for simplicity can be considered at the old time level $n$.

The second of the equations (8) can be reformulated more efficiently from a computational point of view by a clever decomposition of the velocity. The velocity equation is rewritten as:

$$v_{p}^{n+1/2} = \hat{v}_{p} + \beta_{s} \hat{E}_{p}^{n+\theta}(x_{p}^{n+1/2}),$$

(9)

where $\beta_{s} = q_{s} \Delta t/2m_{p}$ (independent of the particle weight and unique to a given species $s$, electrons or ions). For convenience, we have introduced hatted quantities obtained by explicit transformation of quantities known from the previous computational cycle:

$$\hat{v}_{p} = \alpha_{p}^{n} \cdot v_{p}^{n},$$

$$\hat{E}_{p}^{n+\theta} = \alpha_{p}^{n} \cdot E_{p}^{n+\theta},$$

(10)

The transformation tensor operators $\alpha_{p}^{n}$ are defined as:

$$\alpha_{p}^{n} = \frac{1}{1 + (\beta_{s} B_{p}^{n})^{2}} \left( I - \beta_{s} I \times B_{p}^{n} + \beta_{s}^{2} B_{p}^{n} B_{p}^{n} \right),$$

(11)

and represent a scaling and rotation of the velocity vector.

Equations (8) are implicit, because the new fields are needed before the equations for the particles can be solved. If the electric and magnetic fields were given, the equations of motion could be solved independently. Note that even this task is not trivial, because to solve for the particle velocity one needs to compute the electric and magnetic field at the particle position. Moreover, even if the value of the fields at the advanced time were given, the equations for particle velocity and position would still form a coupled set. Given that a B-spline interpolation is used, the equations are nonlinear, and the task of solving them is not trivial. Nevertheless, each particle can be treated as a system of 6 coupled nonlinear equations to be solved with any of the iterative schemes widely used. Newton’s method can be used [13], but in iPic3D this is accomplished by a simpler fixed number of iterations of the predictor-corrector scheme [6] (3 iterations are used in the present work).

But the real challenge of the implicit method is that the electric and magnetic fields are not given. Unlike the explicit method, the particle equations of motion need time-advanced fields that are not known yet. The implicit method reintroduces the coupling of the particle equations with the field equations. Implicit methods solve Maxwell’s equations also implicitly in time. Indicating with index $g$ the spatial discretisation of the
operators, the discretised Maxwell’s equations can be written as:

\[ B_{g}^{n+1} - B_{g}^{n} = -\Delta t \nabla \times E_{g}^{n+\theta}, \]
\[ E_{g}^{n+1} - E_{g}^{n} = \frac{\Delta t}{\varepsilon_{0} \mu_{0}} \left( \nabla \times B_{g}^{n+\theta} - \mu_{0} J_{g}^{n+\frac{1}{2}} \right), \]
\[ \varepsilon_{0} \nabla \cdot E_{g}^{n+\theta} = \rho_{g}^{n+\theta}, \]
\[ \nabla \cdot B_{g}^{n+1} = 0. \]  

(12)

Many spatial discretization schemes can be used, but their detail is beyond the scope of the present discussion. More details on the finite volume method used by iPic3D are presented elsewhere [6]. The point of crucial interest here is the temporal discretization.

Two issues need to be considered to reach an effective implementation of the implicit method: the divergence conditions and the coupling between the Maxwell and Newton equations.

2.2. The divergence conditions

There are in fact eight equations for only six unknowns. The electric field is determined by the three curl equations and also by the divergence equation (Gauss’s law), and similarly, the magnetic field is governed by three curl equations plus the zero-divergence condition. The latter is of no concern in PIC methods because the numerical discretization used automatically satisfies it (because the magnetic field is expressed as the curl of another vector, thereby imposing its zero divergence). But Gauss’s law requires careful consideration.

In the continuum, before a spatial discretization is introduced, the theory of electromagnetism ensures that any system governed by the two sets of curl equations and satisfying the two divergence conditions at the initial time will automatically preserve them at all times, provided that charge is conserved in the system, meaning that

\[ \frac{\partial \rho}{\partial t} = \nabla \cdot J. \]  

(13)

The divergence conditions become then just a constraint on the initial state.

In a discretized system, however, the charge conservation equation is not satisfied. Of course, the total charge in the system is on average maintained, as no charge is created or destroyed by the discretization, but the detailed balance in each cell implied by Eq. (13) may not be satisfied. Two types of approaches are commonly followed to overcome this difficulty.

The so-called charge conserving schemes can be used. The interpolation of charge and density is chosen in a way to conserve charge locally, ensuring that the discretised version of Eq. (13) is valid [3]. The approach has two serious drawbacks: to enforce this property, one is forced to sacrifice the conservation of momentum (see below), and reportedly, the simulations display an increased level of noise, requiring many more particles per cell (see Ref. [3] at page 360 and Ref. [14]). The advantage is that no elliptic solver is needed, as both divergence equations are automatically enforced. This feature leads to excellent parallel scaling when implemented on massively parallel computers [2].

Alternatively, divergence cleaning approaches can be used by simply correcting the electric field for its electrostatic part to enforce Poisson’s equation:

\[ \nabla^{2} \delta \varphi = \nabla \cdot E - \rho / \varepsilon_{0}, \]  

(14)

where the corrected field is \( E' = E - \nabla \delta \varphi \). Clearly, this achieves the goal by adding a correction, but at the cost of solving an elliptic problem, a CPU-intensive operation. To circumvent this difficulty, inexact versions have been proposed [15, 16] where the elliptic equation is not solved to convergence but is in fact in a sense converged over multiple time steps, reducing the computational burden at the cost of allowing a controlled but non-zero error in the divergence equation. Also, this method can be made efficient by choosing scalable modern solvers [17, 18].
In iPic3D, fortunately, the situation is simplified by the choice of discretization that ensures that any divergence error is damped in time and eliminated automatically [19]. For this reason, it is typically not necessary to apply either of the two techniques.

2.3. Implicit Moment Method

The second challenge is handling the coupling between the Maxwell and Newton equations. In iPic3D, this task is accomplished with the implicit moment method. The algorithm of implicit PIC requires dealing with the coupled equations (12) and (8). The first step is to consider the true nature of the coupling between the two sets. The time-discretized Maxwell equations can be conveniently rewritten in their second-order form by taking the curl of the first equation in System (12), solving for \( \nabla \times \mathbf{B}^{n+1} \), and substituting in the second equation to eliminate \( \mathbf{B}^{n+1} := \theta \mathbf{B}^{n+1} + (1 - \theta) \mathbf{B}^{n} \) [19]:

\[
\mathbf{E}^{n+\theta} = \mathbf{E}^{n} + c^{2} \theta \Delta t \left( \nabla \times \mathbf{B}^{n} - \theta \Delta t \nabla \times \mathbf{E}^{n+\theta} - \mu_{0} \mathbf{J}^{n+1/2} \right). \tag{15}
\]

The present discussion focuses on the temporal discretization, which is the defining aspect of the implicit moment method. The spatial discretization used is a straightforward application of finite difference and is described elsewhere [6].

For stability of the numerical scheme [19], the double curl is best expressed using vector identities and Gauss’s law as:

\[
\nabla \times \nabla \times \mathbf{E}^{n+\theta} = \frac{1}{\epsilon_{0}} \nabla \rho^{n+\theta} - \nabla^{2} \mathbf{E}^{n+\theta}. \tag{16}
\]

With this choice, the system for the field equations for the implicit PIC method becomes:

\[
\mathbf{E}^{n+\theta} - (c \theta \Delta t)^{2} \nabla^{2} \mathbf{E}^{n+\theta} = \mathbf{E}^{n} + c^{2} \theta \Delta t \left( \nabla \times \mathbf{B}^{n} - \frac{\theta \Delta t}{\epsilon_{0}} \nabla \rho^{n+\theta} - \mu_{0} \mathbf{J}^{n+1/2} \right), \tag{17}
\]

where it is now most evident that the new value of the electric field requires two sources: the density and the current computed at intermediate levels during the time step. These values are not available from the previous time step and require an iteration with the equations of motion. The current and densities are in fact defined by Eq. (5), which in turn requires to know the position and velocity of all particles at the advanced time level \( t^{n+1} \).

Two approaches are possible: a full non-linear iterative solver of the coupled particle and field equations or a semi-implicit solver that linearizes this iteration procedure. Recently, the first approach finally was attempted as reported in Section 3 below, but the current production version of iPic3D uses the latter, in the specific implementation called the implicit moment method [9].

The idea of the implicit moment method is that the two sources in the field equation (17), \( \rho^{n+\theta} \) and \( \mathbf{J}^{n+1/2} \), are provided by an average over all the particles in accordance with Eq. (4). Therefore, an approximation of these averages can be sufficient without having to move all the particles self-consistently. In practice, an approximation procedure can be followed where the expressions in Eq. (4) are expanded in Taylor series \( S(x - x^{n+\theta}) = S(x - x^{n}) - (\theta \Delta t) \mathbf{v}^{n+\theta} . \nabla S(x - x^{n}) + \cdots \) to obtain approximations in terms of initial quantities

\[
\mathbf{J}^{n+1/2} \approx \mathbf{J}^{n} + \frac{\mu_{0}}{\theta \Delta t} \cdot \mathbf{E}^{n+\theta},
\]

\[
\rho^{n+\theta} \approx \rho^{n} - \theta \Delta t \mathbf{v} \cdot \mathbf{J}^{n+1/2}.
\]

that can then be substituted in Eq. (17); in accordance with Eq. (9), hatted quantities approximate average values that would result in the absence of the electric field:

\[
\hat{\mathbf{J}}^{n}(x) := \sum_{p} q_{p} \mathbf{v}_{p}^{n} S(x - x_{p}^{n}) - \frac{\Delta t}{2} \mathbf{v} \cdot \sum_{p} q_{p} \mathbf{v}_{p}^{n} \mathbf{v}_{p}^{n} S(x - x_{p}^{n}),
\]

\[
\hat{\rho}^{n}(x) := \sum_{p} q_{p} S(x - x_{p}^{n}) - \theta \Delta t \mathbf{v} \cdot \hat{\mathbf{J}}^{n}(x). \tag{19}
\]
The response of the plasma to the changing electric field is summarized in the plasma response tensor:

$$\mu^n = \sum_s \frac{q_s \rho_s^n}{m_s} \left( I - \beta_s I \times B^n + \beta_s^2 B^n B^n \right) \frac{1}{1 + (\beta_s B^n)^2}. \quad (20)$$

Assuming that spatial gradients of the magnetic field are small in Eq. (19) yields an approximation that gives implicit moments per species in terms of initial moments:

$$\hat{J}_n^s(\mathbf{x}) \approx \alpha_s^n(\mathbf{x}) \cdot \frac{\Delta t}{2} \nabla \cdot \sum_{p \in s} q_p v_p v_p S(\mathbf{x} - \mathbf{x}_p^n),$$

$$\hat{\mathbf{J}}^n(\mathbf{x}) := \sum_s \hat{J}_n^s(\mathbf{x}).$$

This approximation is used in iPic3D.

Substituting in Eq. (17), the final form of the field solver for the implicit moment method follows as

$$\left( c \theta \Delta t \right)^2 \left[ -\nabla^2 E^n + \nabla \left( \mu^n \cdot E^n + \theta \right) \right] + \epsilon^n \cdot E^n + \theta = E^n + \left( c \theta \Delta t \right) \left( \nabla \times B^n - \mu_0 \hat{J}^n \right) - \left( c \theta \Delta t \right)^2 \nabla \rho^n,$$

where $\epsilon^n = I + \mu^n$ and $I$ is the identity tensor.

The equation (21) derived above is the field solver of the implicit moment method; from it the new electric field can be computed, and consequently, from the first of Eq. (12), the new magnetic field can be solved. And this without yet needing to move the particles. The information about the particle response to the changing fields during the time step is summarized by Eq. (20), obtained with a linearization procedure.

The computational advantage is great when compared with the fully implicit method, because the field solver needs only a linear iteration without involving the particles other than to compute the auxiliary quantities (??) once at the beginning of the time step. Compared with the fully implicit method, the semi-implicit method does not require any iteration between particles and fields and does not require any nonlinear solver. In iPic3D, Eq. (21) is solved with GMRES [6]. Of course, as discussed in Section 4, this simplification comes at a cost: the implicit moment method does not conserve energy exactly, while the fully implicit method does.

Provided that the condition that the particles on average do not travel more than one time cell per time step is satisfied, however, the semi-implicit method provides accurate results [9]. For a typical maxwellian plasma, this accuracy condition can be expressed as:

$$v_{th,e} \Delta t / \Delta x < 1. \quad (22)$$

Returning to the typical scales of space weather, reported in Fig. 2, this condition can be satisfied by arbitrarily large values of $\Delta t$ and $\Delta x$, provided their ratio is constrained by Eq. (22). In practice, the constraint means that the temporal and spatial resolutions can both be increased in the hourglass representation of Fig. 2, as long as the yellow bar is kept horizontal. $\Delta t$ and $\Delta x$ can be increased together by the same factor from the lowest values imposed by the explicit scheme (in light blue) to the desired higher value (in light yellow). The resolution bar must remain horizontal in semi-implicit methods, but this is exactly the position it is needed to have for space weather problems, causing no practical limitation to the user. Fully implicit methods do not have this limitation, and $\Delta t$ and $\Delta x$ can be chosen independently, a property that can be of value to other applications beyond space weather [20].

The typical computational cycle of the implicit moment method implemented in iPic3D is reported in Fig. 3.

2.4. Algorithm deployment on distributed memory computers: iPic3D

The moment implicit PIC method has been implemented in a parallel software package, called iPic3D, written entirely in C++ [6]. An Object-Oriented (OO) design has been followed in writing iPic3D using the
so-called lite OO approach presented in a previous work [21]. The variables related to particles are organized as arrays in Particle objects and divided depending on the species (electrons, ions, . . .). The electromagnetic field constitutes a whole object that consists of the electromagnetic field and field sources variables. OO paradigms, such as class inheritance and polymorphism, are used to make it easy for developers to add new code to iPic3D without modifying the existing code.

For moment implicit PIC, where the cost of particle-moving and of field-solving are of the same order (unlike conventional PIC codes, where most of the cost resides with updating particle values), it is crucial that both field-solving and particle-moving are parallelized efficiently. On distributed memory systems, the domain decomposition technique is used to divide the computational workload among the computing units. The simulation box is divided among processors using a generic cartesian virtual topology. Particles are distributed among processors based on their location and communicated to adjacent processors if exiting from the processor domain. In fact, an important aspect of efficiency is the need to retain the particles and cells belonging to a subdomain on the same processor. Large amounts of information are exchanged between grid and particles residing in the same physical domain because of the interpolation step, and therefore it is important to avoid that this information exchange results in inter-processor communication.

The parallelization of the code is based on MPI libraries and on blocking MPI\_Send/MPI\_Receive calls. Parallel communication is involved in all the stages of the implicit moment PIC code. At each mover iteration, the data for particles leaving the domain is collected in temporary arrays. Once the particle data is updated, the temporary arrays are exchanged among contiguous domains. Ghost cell data is communicated during each solver iteration and during the calculation of the densities in the interpolation from particles to the grid. The use of non-blocking MPI subroutines in combination with the OmpSs programming model is under development within the DEEP project.

2.5. Performance analysis of iPic3D

The very good scaling properties of iPic3D on as many as 16384 CPUs have already been demonstrated in, for example, [22]. In this section, the focus is on a more detailed analysis of the different logical blocks of Fig. 3, which are repeated every computational cycle. It is reminded that the logical blocks under investigation are the following: B5 (moments calculation), B1 (hatted moment calculation), B2 (field solver), and B4 (particle mover). The logical block B3 (calculation of fields on particles) is included in B4. A deep understanding of the characteristics of each of them is fundamental to address issues of co-design and of code porting to different architectures.

Figure 4 shows the percentage execution time (panel a) and the percentage bytes transferred (panel b) for the different logical blocks. The measures are obtained with Scalasca, an open-source toolset for
Table 1: Input parameters for the “judge” and “juqueen” test cases of Fig. 4. \( n_x, n_y \) and \( n_z \) are the cell number in the three directions, \( n_{ppc} \) the number of computational particles per cell, \( N_s \) the number of particle species (electrons, ions, background electrons and background ions), \( L_x/d_i \times L_y/d_i \times L_z/d_i \) the physical dimensions of the domain in the three directions normalized to the ion skin depth \( d_i \).

<table>
<thead>
<tr>
<th></th>
<th>JUDGE</th>
<th>JUQUEEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n_x \times n_y \times n_z )</td>
<td>64 \times 64 \times 64</td>
<td>128 \times 128 \times 128</td>
</tr>
<tr>
<td>( n_{ppc} )</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>( N_s )</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>( L_x/d_i \times L_y/d_i \times L_z/d_i )</td>
<td>10 \times 10 \times 10</td>
<td>20 \times 20 \times 20</td>
</tr>
</tbody>
</table>

scalable performance analysis of large-scale parallel applications [23]. Manual instrumentation is inserted in the source code around the entry points of the different logical blocks to reduce the overhead in the analysis coming from automatic instrumentation. The Scalasca overhead, provided as a metric, is bound to reasonable values. This confirms that the significance of the measures is not hindered.

Different test cases are examined in Fig. 4. The aim is to check if the behavior registered is dependent on the problem size, the number of processors used and the hardware used. All cases are 3D simulation of magnetic reconnection [24]. The test cases labeled as “judge” refer to quite small simulations performed on the cluster JUDGE [25], an IBM iDataPlex cluster hosted at Forschungszentrum Jülich. The test case “juqueen” is a rather bigger simulation performed on JUQUEEN [26], an IBM BlueGene/Q cluster and currently Europe’s fastest computer according to the most recent TOP500 list [27]. The input parameters for the two test cases are listed in Table 1, second and third column. The number of processors used in the three simulated directions is listed in the caption of Fig. 4 for the different test cases. The number of processes equals the number of cores.

Figure 4: Scalasca measurements of magnetic reconnection simulations performed with different input parameters and with a varying number of processors (i.e. cores) on the clusters JUDGE and JUQUEEN. Panel (a) shows the percentage execution time and panel (b) the percentage bytes transferred for the four main logical blocks which constitute iPic3D.

Figure 4 shows that the results obtained both for the percentage execution time and the percentage bytes transferred are rather independent of the problem size and of the number of processors used. The block B4 (particle mover) is consistently the most time-consuming, due to the need to iterate over all particles and perform a computationally expensive implicit solve on each particle. Another very time-consuming block is B5 (moments calculation), which cycles over all particles to extract particle moments (density, currents, pressures) to deposit on the grid. Note that the percentage execution times of these two blocks increases with the number of particles per cell used in the simulation; the value used here, 27, the same as in the sample application of Section 5.1 but is still rather low when compared to the number of particles often
employed in production simulations, typically \( n_{ppc} \approx 100 \). Using a higher number of particles helps to control numerical noise, but of course increases the computational cost of the simulation. Blocks B4 and B5 can be collectively defined as “particle blocks,” since they are dominated by computation proportional to the number of particles. Blocks B1 (hatted moments calculation) and B2 (solver), instead, operate on vectors sized according to the computational grids and are therefore “grid blocks.”

Fig. 4 (b) shows that block B2 (solver) is always responsible for most of the communication. In fact, the block solves Equation (21) and (14) with two iterative solvers, the Generalized Minimal RESidual method (GMRES) and the Conjugate Gradient (CG) [28]. At each iteration of the two solvers, Peer-to-peer (P2P) communication is performed to update the values of the grid ghost nodes of each process, and global communication is done to calculate vector moduli and scalar products.

The only other block performing global communication in iPic3D is B4 (mover), as part of a routine which communicates particles residing in a process to another one: if at the end of the particle move the center of a particle residing in one process is located in the grid area mapped to another process, the particle is moved. However, most of the global communication is done in B2. This is confirmed by Fig. 5, which shows how much of the bytes transferred on the network during the test cases described are part of global communications (blue bars) and how much of these global communications are initiated in B2 (red bars). The measures are again done with Scalasca. Observe that the percentage of bytes on the network transferred with collective communication reasonably explodes with the number of processors. It is also consistently evident that B2 is primarily responsible for global communication.

Figure 5: Scalasca measurements of the communication footprints of the test case simulations described before. The blue bars are the percentage of the total bytes transferred which are part of global communication. The red bars show how much of this global communication is initiated within B2.

3. Algorithmic developments

3.1. Fully implicit PIC

The fully implicit PIC method requires the concurrent solution of the equation of motion for each particle and of the field equations for the electric and magnetic fields at each grid point. The solution of the implicit numerical equations of PIC method implies the computation of a nonlinear system. Nonlinearity arises from the coupling between particles and field variables through the interpolation functions of the PIC method. In the fully implicit PIC scheme, the nonlinear equations are solved by a Newton-Krylov solver [29]. Despite the belief that the iterative solution of such equations could hardly converge, it has been proven that such PIC methods are convergent [30]. In addition, the fully implicit PIC scheme requires the solution of a very large matrix whose rank is of the order of the number of particles (the number of particles is considerably higher than the number of grid points in typical PIC simulations). For this reason, implementations of fully implicit PIC methods are based on matrix-free Jacobian-free solvers to avoid the storage of the matrix and Jacobian coefficients [30].
A fully implicit method can be formulated so that the PIC method conserves the total energy of the system exactly [31, 32, 20]. This quantity is not conserved in most PIC formulations. Energy conservation is of crucial importance when acceleration mechanisms need to be described correctly. Compared the moment implicit method, the fully implicit PIC has much simpler formulation and results in easier implementation.

The main disadvantage of the fully implicit PIC method is that it requires the solution of a very large system whose size increases with the number of particles. The number of particles is easily more than a million in a typical PIC simulation, leading to matrix to be inverted whose rank size is of the order of million. To reduce the size of this matrix, it is possible to use a technique called kinetic enslavement [31, 32, 20]. In this method, the only unknowns of the problem are the value of the electric magnetic field at the grid points at the new time level, and the JFNK solver computes only the field equations. The particle equations of motion are calculated by an ODE Newton-Raphson method and are embedded in the field solver as function evaluations.

3.2. Multi Level Multi Domain (MLMD)

The aim of the IMM Multi Level Multi Domain (MLMD) algorithm is to reduce the computational costs of space weather simulations algorithmically rather than by relying on more powerful or innovative hardware. The algorithm is of particular relevance in the field of space weather, since, as recalled in Section 2 and Fig. 2, multiple scales are present in the problem to simulate. If different scales are of relevance in different sub-domains of the simulation, notable computational resources can be saved if the different subdomains are resolved with the local resolution of interest, rather than with the highest resolution needed in the domain. Examples of such adaptive PIC codes are Adaptive Mesh Refinement (AMR) codes [33, 34] and the already cited MLMD technique [35, 36].

The MLMD represents the domain as a collection of subdomains resolved with increasing resolution. If smaller spatial scales are of interest in a subsection of the grid, that area is resolved both with the initial resolution and with increased resolution as a new level of the MLMD system. The two levels are labeled as coarse and refined respectively. All levels are self-similar; that is, they are fully simulated in fields and particles. Contrary to standard AMR implementations, coarse-level particles are present in the coarse grid area even where it is also resolved with a refined grid and fine particles. This is done for two reasons. First, under the numerical point of view, this technique allows to avoid a number of issues related to the change of the shape function of computational particles which cross boundaries between resolution levels (refer to [35] for a discussion). Second, the self-similarity between the levels results in an easier transition from a standard, single-level IMM PIC code to its MLMD version. Since the same particle and grid operations are performed at each level, the levels can be represented as instances of the same class. The difference lies, of course, in the physical dimension of the subdomains and in the resolutions used.

Fig. 6 shows how the MLMD method modifies the sequence of logical operation of Fig. 3, which refers to a single-level PIC IMM code. For simplicity, from now on just one coarse (level $\ell$) and one refined (level $\ell + 1$) grids are assumed.

The operations underlined need to be performed additionally in a MLMD system; refer again to [35] for details. Performance considerations are also provided:

- **MLMD initialization:** This series of operations is to be performed just as an initialization if the position of the refined grid is constant in time with respect to the coarse grid. Most of them are the construction of the “communication maps” between the levels. Each process assigned to the refined level calculates from the input data the rank of the coarse level processes it interacts with for the MLMD operations described below. This information is then broadcast to all coarse-level processes. Each coarse-level process involved in MLMD operations then receives extra information (e.g.: the number of grid points to send/receive) through P2P communication.

Notice that the same number of processes is assigned to the coarse and the refined grid and that a fraction $L_x/RF \times L_y/RF$ of the coarse grid is simulated with increased resolution and the same number of grid points and computational particles as the coarse grid. $L_x$ and $L_y$ are the dimensions of the domain in the $x$ and $y$ directions. RF is the Refinement Factor between the grids. Thus, the number of coarse grid processes interacting with the refined grid becomes smaller and smaller with
increasing RFs. Conversely, only the processes located at the boundary of the refined grid perform MLMD operations. This is a source of computational imbalance between the processes.

- **Boundary Condition (BC) interpolation**: Before the refined grid starts the computation of block B2 (field solver), boundary conditions for the electric and magnetic field at time $n + 1$ must be received from the coarse grid. This introduces a bottleneck in the execution, since the refined grid has to wait idle for the coarse grid solution to be computed before starting its own block B2.

  Notice however, from Section 2.5, that block B2 is a grid-related block, that is not one of the most time consuming ones. Additionally, coping strategies (e.g.: using as BC for the refined grid an intermediate solution of the iterative GMRES solver, rather than the fully converged result) can be devised to reduce the waiting time of the refined grid.

- **Field Projection**: After completion of block B2 on the refined grid, the increased-resolution solution is projected to the coarse grid. This introduces a second bottleneck, since this time the coarse grid has to wait for the refined grid solution.

- **Particle Repopulation**: To provide optimal particle boundary conditions for the refined grid, the native refined grid particles sitting in a small area around the grid boundary, indicated as Particle Repopulation Area (PRA), are deleted after the last iteration of the particle mover. New refined
particles are created from the corresponding coarse-grid particles with the splitting algorithm described in [37]. This operation is performed after both grids have completed block B4 (particle mover).

The operation adds a minimum overhead on the coarse-grid processes overlapping the PRA area, which store on the fly, while moving particles, the particle information to be sent to the refined grid. Higher overhead befalls the refined grid processes, which do the particle splitting and communicate the split particles to the appropriate refined process, according to the position of the center of the new particle.

Notwithstanding the node imbalance and bottleneck issues just underlined, the performances of the MLMD code are very promising due to two facts. First, the IMM method is used as baseline algorithm, while all AMR codes in use rely on explicit PIC algorithms. This means that the MLMD implementation proposed benefits from the laxer stability constraints of implicit methods, described in Section 2. To that, the benefits coming from the MLMD method itself have to be added.

Table 5.1 in [38] shows a comparison of the expected computational requirements for the simulation of the same physical process, magnetic reconnection [24], performed with a single-level explicit, a single-level IMM and a three-level IMM MLMD code. The simulation of one physical second is calculated to require \(10^5\) processes and 100 minutes with the first code, 451 processes and 2 minutes with the second and 1 process and 2 minutes for the third, the IMM MLMD code. The IMM allows to have very high RF jumps between the levels, while still being in the stability range of the method.

Such high refinement jumps are also possible due to the particle repopulation method used. With other particle boundary conditions, the RF jumps between the levels are limited also to keep the errors in the update of the particle positions and velocities at the boundaries between the levels bounded to reasonable values (see [35] for a discussion). Such necessity is removed with the splitting algorithm used.

Fig. 7, from [36], shows performance measures actually registered. The two lines refer to the execution times in seconds for the simulation of the same problem, a Maxwellian plasma relaxation over a relatively small domain, \(L_x = L_y = 21.0 \, d_e\), where \(d_e\) is the electron skin depth [39]. All simulations are performed using 128 processes for the same number of cycles. The red line depicts the execution times for a two level MLMD system with increasing RF between the two grids. The blue line shows the execution times for the simulation of the entire domain with the resolution of the refined grid of the corresponding MLMD case.

![Figure 7: Comparison of the execution times of two-level MLMD simulations (red line) and one-level simulations (blue line) performed with the same resolution of the MLMD refined grid as a function of the Refinement Factor RF. From [36].](image)

Notice, primarily, that the saving in computing time achieved with the MLMD method are astonishing, if a fraction \(L_x/RF \times L_y/RF\) of the total domain needs to be simulated with a resolution per side \(RF\) times higher than the coarse grid. In the case with \(RF = 14\), the execution times in the two cases are 46.97 versus 3287.46 seconds! \(RF = 14\) is chosen as final value in the series not due to a performance breakdown, but because higher factors would violate the IMM accuracy constraint on the refined grid. Secondly, the execution times of the MLMD simulations with increasing \(RF\)'s can be compared in Table 2.
Table 2: Execution time in seconds for two grid MLMD Maxwellian plasma simulations with varying RF. From [36].

<table>
<thead>
<tr>
<th>RF</th>
<th>Exec Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>41.25</td>
</tr>
<tr>
<td>6</td>
<td>43.98</td>
</tr>
<tr>
<td>10</td>
<td>45.87</td>
</tr>
<tr>
<td>14</td>
<td>46.97</td>
</tr>
</tbody>
</table>

At the first glance, the timing are remarkably similar, because the same number of computational particles is present in all the simulation, notwithstanding the RF. The slight increase with the RF is traceable to the increasing cost of the MLMD operations with increasing refinement. In particular, the particle repopulation operations are expected to scale with $RF^2$. However, this overhead is bounded to very reasonable values and does not constitute a major issue at this stage of the code development, when performance issues have not been addressed yet.

3.3. Fluid-Kinetic PIC

A new fluid-kinetic method that solves concurrently the multi-fluid and Maxwell’s equations has been recently developed [40]. The governing equations for the fluid-kinetic PIC solver are the multi-fluid equations, Maxwell’s equations and particle equations of motion. The multi-fluid equations and Maxwell’s equations are solved on the grid, while the equation of motion are solved for each computational particle.

The fluid continuity and momentum equations for the species $s$ are:

\[
\begin{align*}
\frac{\partial \rho_s}{\partial t} + \nabla \cdot \mathbf{J}_s &= 0, \\
\frac{\partial \mathbf{J}_s}{\partial t} &= \left( \frac{q}{m} \right)_s (\rho_s \mathbf{E} + (\mathbf{J}_s \times \mathbf{B})/c - \nabla T_s);
\end{align*}
\]

$T_s = \mathbf{J}_s \cdot \mathbf{J}_s / \rho_s + p_s$ and $p_s$ are respectively the stress and pressure tensors. Different closure equations can be provided to calculate the stress tensor $T_s$ in Eq. 23. In the fluid-kinetic method, we intend to retain the kinetic effects by calculating the stress tensor using the computational particles of the PIC method by interpolation. Therefore, the closure equation for the stress tensor is provided by particles in this way:

\[
(T_s)_g = \sum_p N_p q_s v_p v_p W(x_g - x_p).
\]

The fluid-kinetic PIC solver includes both fluid and kinetic models in the same framework, making the fluid-kinetic PIC solver an ideal candidate for coupling fluid-kinetic models. It shows that fluid-Maxwell solvers for plasma simulations that neglect kinetic effects can be equipped with computational particles to introduce kinetic effects in the simulation. Like other implicit PIC methods, the fluid-kinetic PIC solver allows to use large time steps enabling kinetic simulations over time scales typical of the fluid simulations. Finally, it simplifies the formulation of the implicit moment PIC models, making the implementation of such schemes less prone to errors.

4. Code Design Issues

Achieving exascale requires keeping as many as one billion threads busy. Such massive parallelization requires that all significant aspects of communication scale. We first discuss the parallel and communication properties of the GPU and MIC and then discuss how each block of the iPic3D algorithm can scale on these architectures.
4.1. Vectorization on GPUs and MIC

Vectorization promises to accelerate compute-intensive parallelizable parts of our algorithm. The MIC and the GPU are two competing technologies that offer alternative mechanisms for vectorization. Both the MIC and GPU offer a platform involving at least one coprocessor (or "accelerator") attached to a host processor. Efficient use of the coprocessor for massively parallel processing requires respecting constraints on the organization and movement of data.

A MIC coprocessor has a large number of cores (e.g. 50 or 60) each of which supports the full x86 instruction set and each of which has a vector unit capable of operating on eight double-precision numbers.

The GPU allows creating a large team of lightweight threads that operate on parallel data and that all simultaneously execute the same stream of instructions taken from a limited instruction set. The smallest unit in which these teams of threads are organized is a warp, a team of typically 32 GPU "threads" executing identical instructions. Memory access is very fast on the GPU if the data accessed by each warp is aligned and in sequence as if for a conventional vector processor with 32 elements in each vector. In contrast to vector units, such perfect coalescence of memory is not a requirement of the GPU, but the time required to access memory can increase by as much as a factor of 32 if restrictions on localization, alignment, and sequence of data are violated; more advanced GPU architectures are less restrictive on the alignment and sequencing required for fast access [41, 42].

4.2. Communication on GPUs and MIC

On massively parallel architectures such as GPU and MIC clusters, it is generally important to minimize communication and to delay synchronization. Specifically, even though GPUs are usually connected as peripheral devices on PCI Express, the available bandwidth is still slow compared to the computing capabilities of the GPU. It is not surprising then that one of the main limiting factors for multi-GPU architectures is the latency of transfer between CPU and GPU memory [43].

On GPUs, current state-of-the-art MPI libraries can only access host memory, implying that if data needs to be transferred between GPUs on different nodes then one needs first to transfer it explicitly from the device to the host. There has been much effort to overcome this problem by explicitly pipelining the data movement and kernel execution at the application level [44, 45, 46, 47, 48, 49]. Since collective communication routines are internal to MPI, there is only so much one can do, and ultimately efficient collective communication will require MPI implementations that can access directly the device memory such as cudaMPI and glMPI [50]. This is not an issue on the MIC, since the coprocessor runs independently of the host processor and can issue its own MPI calls.

4.3. Parallelization and communication in iPic3D

All parts of the implicit moment method have the potential to benefit from parallelization. The part that has the greatest potential for vectorization is moving particles (block B4), since each particle moves independently of the others. The main potential bottleneck to computation is that calculating particle movement requires random-access interpolation of field data at the location of each particle. This means that particles should be sorted by location into "neighborhoods" so that the field data for each neighborhood fits in cache. Each particle needs to be transferred if necessary to the appropriate neighborhood after it is moved, which requires communicating particles between processors responsible for adjacent neighborhoods or subdomains. Because of the large number of particles, it is beneficial for particle data to remain on the coprocessor.

Computing moments (block B5) requires random write-access to a grid. If multiple threads simultaneously accumulate particle moments on the same mesh, then one must use atomic writes to guard against the possibility that two threads may occasionally attempt to write simultaneously to the same grid node. Fortunately, the GPU provides efficient atomic writes for single-precision floating point numbers. Note that use of single-precision arithmetic for particles can be sufficiently accurate, since moments are obtained by summing over a large number of particles and particle positions can be represented relative to the local mesh cell [51].
Computing fields (B2) is the most challenging block to parallelize, due to the use of an implicit field solver. We execute the GMRES algorithm to perform each Newton iteration. In each cycle of the GMRES algorithm, the implicit field solver must repeatedly evaluate a residual, perhaps \( m = 20 \) times. Evaluating the residual can be vectorized in the same way as for an explicit field solver and involves communication only of face data between processors responsible for adjacent subdomains. A greater challenge to communication scaling is the global communication needed for convergence. The GMRES algorithm involves computing on the order of \( m^2/2 \) (perhaps 200) inner products, which require blocking global reductions. More sophisticated versions of GMRES, however, have recently become available: \( \ell^1 \)-GMRES requires only \( m \) global reductions, and a pipelined version called \( p(\ell) \)-GMRES permits nonblocking global reductions which allow synchronization to be delayed by \( \ell \) iterations and thus can be interleaved with computation [52].

Effective preconditioning avoids the need for a large number of GMRES iterations and is of more primary importance than improving the computational efficiency of later iterations. An effective preconditioning strategy for Maxwell’s equations is to use geometric multigrid at the refined level (which avoids the need to explicitly form large, sparse matrices) combined with algebraic multigrid at the coarse level (which avoids the long-range communication bottlenecks that arise in architectures that fail to implement a communication network that scales in a self-similar way to the coarsest levels).

4.4. Eliminating Poisson Solves to Reduce Global Communication in iPic3D

In high performance computing, codes often have to be adapted in order to avoid the criticalities which prevent good performances or scaling on particular architectures. Performance bottlenecks can be mitigated through software modifications (e.g., optimizations) or attacked in a more radical way using algorithmic changes. An example of such possibility will be provided here regarding global communication.

As already argued in Section 2.5 and Fig. 5, block B2 is responsible for most of the global communication in the code. Of the two iterative solvers present in B2, the GMRES used for Eq. (21) and the CG for the Poisson’s correction of Eq. (14), the second is the main offender with respect to the amount of global communication.

In fact, with reasonable values for the convergence criterion, Equation (21) converges in a very low number of iterations, usually around 10, due to the fact that the Left Hand Side (LHS) term which poses the highest convergence problems, \( \nabla^2 \mathbf{E}_{n+\theta} \), has low magnitude when compared to the other LHS members [53]. Equation (14), however, converges in a number of iterations closer to 100. Alternative ways to enforce Gauss’s law, less intensive under the point of view of global communication, are thus investigated, following the tutorial in [54].

Recall that Gauss’s law is automatically enforced if the continuity equation for charge, Eq. (13), is respected when depositing currents at the grid points. This is not the case with the PIC moment-deposition scheme described earlier in Section 2. A number of schemes [55, 56, 57, 58, 20] have been devised to deposit currents while respecting Eq. (13). In practice, the motion of a particle from a start to an end point is broken into multiple intermediate segments, whose number and spatial orientation change with the scheme. Charge flux parcels are deposited at the end of each segment and not just at the end point of the motion. Since the coordinates of multiple relay points (i.e., the intermediate points at which the particle changes direction or crosses a cell boundary) and several particle to grid interpolation operations may need to be calculated, charge conservation methods can be computationally intensive. Preliminary considerations on the computing speed of the different charge conservation methods are provided in [54], based on the number of segments the particle motion is broken into and on the number of IF statements in each algorithm. It can be safely stated, however, that each of the methods, being compute bound but not requiring any global communication, is an interesting alternative to the Poisson’s correction to enforce Gauss’s law in architectures like MICs or GPUs.

4.5. Implementation strategy on Booster-Cluster

The DEEP project is implementing a Cluster-Booster architecture consisting of the Cluster, a traditional cluster of Xeon nodes, and the Booster, a collection of Xeon Phi coprocessors connected by high-speed interconnects to form a 3D torus. The Cluster and Booster are connected by an Infiniband network.
The intent of the DEEP architecture is to provide a mechanism to offload parts of an algorithm that scale well to the Booster while leaving parts on the Cluster whose scaling would be more difficult to improve. In keeping with this intent, we intend to represent particles on the Booster and to perform the implicit field solve on the Cluster. This requires that we transfer particle moments (consisting of 10 scalar fields) from the Booster to the Cluster in order to advance the fields and that we transfer the field data (consisting of 6 scalar fields) from the Cluster to the Booster in order to advance the particles with each cycle of the algorithm.

Our rationale for solving fields on the Cluster is that the implicit field solve involves a large number of global reductions, which should be more efficient on the Cluster. Particles, however, can be advanced completely independently of one another, and the only communication needed between processors is local communication to transfer particles; this communication pattern is well-suited to the Booster’s 3D torus interconnect.

In plasma simulation, use of a large number of particles per mesh cell helps to reduce particle noise. The DEEP architecture promises to allow us to significantly increase the number of particles per mesh cell in iPic3D.

While we do not aim to achieve full scaling of the field solve with this division on the DEEP architecture, we expect to be able to handle an amount of field data sufficiently large for our purposes. While field data must be transferred to and from the Booster, the amount of data will be much less than the particle data, and the transfer needs to be made only once per cycle. Furthermore, the independence of particles means that communication to and from the Booster can be spread out over time and done concurrently with moving particles or accumulating moments.

5. Sample Applications

For illustration of the type of goals of the space weather application we report below two typical examples of iPic3D simulations. The first is aimed at studying processes developing in many space weather events: the conversion of magnetic energy to kinetic energy. The second is the global model of the interaction of the Earth space environment with incoming solar plasma.

5.1. Simulations of null point reconnection in 3D

Using iPic3D, we have simulated the evolution of proton-electron plasma in a three-dimensional fully periodic box with 8 uniformly spaced null points [59]:

\begin{align*}
B_x &= -B_0 \cos \left( \frac{2\pi x}{L_x} \right) \sin \left( \frac{2\pi y}{L_y} \right), \\
B_y &= B_0 \cos \left( \frac{2\pi y}{L_y} \right) \left[ \sin \left( \frac{2\pi x}{L_x} \right) - 2 \sin \left( \frac{2\pi z}{L_z} \right) \right], \\
B_z &= 2B_0 \sin \left( \frac{2\pi y}{L_y} \right) \cos \left( \frac{2\pi z}{L_z} \right),
\end{align*}

where \( B_0 \) is the magnetic field amplitude; \( L_x, L_y, \) and \( L_z \) are the sizes of the simulation domain in the corresponding directions. It is easy to show that this configuration satisfies the condition \( \nabla \cdot \mathbf{B} = 0 \).

The simulation domain represents a periodic cube with dimensions \( L_x \times L_y \times L_z = 20d_i \times 20d_i \times 20d_i \) and with 400\(^3\) cells and 27 particles of each species per cell. Our plasma consisted of electrons and ions with mass ratio \( m_i/m_e = 25 \) and temperature ratio \( T_i/T_e = 5 \); the particles were initialized with Maxwellian velocity distributions with electron thermal velocity \( \mathbf{v}_{th,e}/c = 0.0346 \). The initial particle density was uniform with \( n_0 = 1 \); the initial magnetic field amplitude was \( B_0 = 0.02 \). It is important to note that with such parameters the electron spatial scales are resolved well enough in our simulation: the electron skin depth \( d_e = 0.2d_i = 4\Delta x \), where \( \Delta x \) is the grid spacing. The time step was set relative to ion plasma frequency: \( \Delta t = 0.15/\omega_{pi} \), and ion cyclotron frequency was \( \Omega_{ci} = eB_0/m_i c = 0.035 \omega_{pi} \).
Figure 8: Three-dimensional simulations of magnetic reconnection with iPic3D, just before two null points are dissolved. The magnetic field lines are shown in grey; grey isocontours show regions of low magnetic field $B = 0.2B_0$; blue are electron streamlines, demonstrating electron jets entering the null points along separatrices and being scattered away along spines. Note that only two of eight null points are shown here.

We used 16000 CPUs on a grid of $20 \times 20 \times 40$ to run this simulation for 10000 iterations ($\Omega_{ci} t = 50$), and it took 9 hours for it to finish, which gives an efficiency of $0.2 \text{ ms/particle/iteration}$. The simulation was performed on Curie supercomputer “thin” nodes. For parallel data reduction, routines written in Python were used that convert code output (HDF format) into parallel VTK files. Data visualization was done on a desktop computer using ParaView (running on a single CPU).

In our simulation, 82% of initial magnetic field energy was transferred to particle kinetic energy: the decay rate was almost 5 times higher than in traditional two-dimensional current sheet reconnection. Before a magnetic null point is dissolved by reconnection, electron beams stream towards the nulls along separatrices; electrons are scattered away along the spine, as shown in Fig. 8 for two null points located at $[5, 5, 5]$ and $[5, 15, 5]$.

5.2. Global scale interactions between the solar wind and the magnetosphere

When the hot plasma ejected from the Sun interacts with a solid body in space, the trajectory of the plasma particles is modified: the plasma concentration, the electromagnetic field and the current systems around the object changes. A large number of complex physical interactions can be observed in this system.

Up until recently, measurements were performed using isolated spacecrafts at discrete locations in time and space. The correlations between the different spatial and temporal plasma fluctuations were difficult to analyse using only one data point. Data from multiple spacecrafts were used to better understand unsteady phenomena. But with the advent of multi-point missions like Cluster II and THEMIS, the measurement of local spatial gradients and temporal correlations in the magnetosphere of the Earth were improved. Today we have a network of spacecrafts that constantly record measurements in the solar wind, in all the zones of the Earth’s magnetosphere and around other planets of the solar system. However, obtaining an instantaneous picture of the full system at the global scale using such heterogeneous sources is still difficult [60].

A perfect addition to understand how the Sun affects the environment of solar system objects is to use powerful supercomputers and modern numerical methods to model the full system. Simulations at the scale of a planet have often been performed using MHD tools. These models have shown good general
agreement with measurements but rely on assumptions that may not be valid for the plasma flows observed in the solar wind [61]. Kinetic effects that may not be detected by MHD simulations include diffusion instabilities, magnetic reconnection, energy damping and electron runaway, all of which play an important role in the energy transfer between the solar wind and the planetary environment, one of the main concerns in space weather research. A fully kinetic approach, where the ions and the electrons are transported as independent particles, is the best solution to attack the shortcomings of MHD simulations. The Particle-in-Cell technique, where particles are transported using Newton’s laws and electromagnetic fields are modeled using a discretization of Maxwell equations on a grid, is a powerful tool, but very CPU-intensive. To study the global scale interactions presented in this section, it is necessary to use implicit numerical methods on massively parallel computers.

Different authors have previously performed simulations of the kinetic phenomena in planetary magnetospheres. Omidi et al. [62] performed simulations on a Mercury-size magnetosphere using a hybrid code, where ions were treated as particles and electrons were considered a fluid. Cai et al. [63] used an explicit PIC code to perform simulations of a 3D magnetosphere. In the first case, the resolution of the electron kinetic scales was avoided, and in the second case, the physical parameters used were strong enough to avoid numerical instabilities of the explicit PIC code. The simulations presented in this section solve the global interaction at the electron, ion and planetary scales with physical parameters closer to nature. The preliminary simulations discussed in this section were performed using the resources from the Tier-1 cluster in the Flemish Supercomputer Center (VSC) and the Curie and Fermi supercomputers from the PRACE infrastructure. Computations were performed using 512 to 4096 processors. Thanks to a new PRACE grant, we are planning to perform in the near future simulations using up to 100000 cores.

The numerical experiment shown in Fig. 9 is composed of a two-dimensional domain discretized with a cartesian grid with a spatial resolution of $\Delta x = \Delta y = 0.06325 d_i$. The total length of the box is $L_x = 64.768 d_i$ in the $x$ direction and $L_y = 36.432 d_i$ in the $y$ direction. A total of $N = 1024 \times 576 = 589824$ cells were used, each one of which contained 50 particles at the initialization of the simulations, for a total of $N_p = 29491200$ particles.

A particle-absorbing sphere is located at $(x_c, y_c) = (L_y/2, L_y/2)$, and a dipolar magnetic field is imposed at the same location. This field is superimposed on a southward Interplanetary Magnetic Field (IMF) of a strength of $B_{IMF} \approx 100$ nT.

At each iteration, particles reaching the right boundary are eliminated from the simulation (outflow condition), and in the remaining boundaries new particles are injected with a drift velocity in the $x$ direction (inflow condition) using a Maxwellian distribution.

Boundary fields are fixed so as to impose a southward IMF and a consistent solar wind velocity of $V_{sw}/c = 0.015$, where $c$ is the light speed. Different thermal velocities are imposed in the ions and electrons.
with values of $v_{b,c}/c = 7.4 \times 10^{-3}$ and $v_{b,c}/c = 0.05$. With these conditions, we ensure that the Alfvén Mach number is $M_A = 5$ and that the plasma beta is $\beta \approx 1$. A mass ratio of $m_i/m_e = 250$ is used.

Figure 9a shows a general view of the simulation box colored by the electron concentration (the normalized scale goes from low concentration in white to high concentration in light blue). Magnetic field lines were plotted using a stream tracer, each line originating at 100 equidistant points in the $y = L_y/2$ axis. The field lines show the creation of a tail in the solar wind flow direction. Due to the antiparallel nature of the closed field lines, two points of reconnection can be observed: one in the subsolar point and one in the tail. A transition between the IMF and the magnetospheric field can be observed in the line that crosses at the center of the planet, showing that the simulation captures the initial state of a shock-like feature. In Fig. 9b, the field lines were traced in order to highlight the structure of the magnetical field inside the magnetosphere. The transition between the IMF and the closed field lines is more obvious. It is also shown how the solar wind pressure wraps the IMF around the tail.

Strong variations in electron concentration can be observed in both images, the most important of which is observed in the north and south cusp regions. Electron concentration is also enhanced between the magnetopause and the upstream solar wind, and a cavity of low electron concentration, typical of the magnetotail lobes, can be observed behind the planet. An interesting feature captured by the simulation is the formation of a high-density zone of trapped particles in the closed magnetic lines close to the planet.

Our objective now is to extend this initial simulation to planetary scales comparable with the Earth, without compromising resolution, for the study of the detailed diffusion instabilities near the magnetopause in the equatorial plane and the reconnection points. We are currently performing simulations with the same spatial resolution but with a box ten times bigger in both axes compared to the previous simulation ($L_x = 647.68\,d_i$ and $L_y = 259.072\,d_i$). We are particularly interested in the capture of the bow shock, the tail formation and reconnection features and the cusp currents. Three-dimensional cases are also being developed using lower resolutions.

Acknowledgments

The present work is supported by the Onderzoekfonds KU Leuven (Research Fund KU Leuven) and by the European Commission’s Seventh Framework Programme (FP7/2007-2013) via the Deep project (www.deep-project.eu) and the SWIFF project (www.swiff.eu). The simulations were conducted in part on the resources of the NASA Advanced Supercomputing Division (NAS), of the NASA Center for Computational Sciences Division (NCCS) and of the Vlaams Supercomputer Centrum (VSC). Support from the PRACE allocation number 2011050747 and from the PRACE project SWEET allowed the use of the Curie and Fermi supercomputers for the space weather examples reported in Section 5.

The Bibliography


