Efficient parallelization for 3D-3V sparse grid Particle-In-Cell: shared memory systems architectures

Fabrice Deluzet, Gwenael Fubiani, Laurent Garrigues, Clément Guillet, Jacek Narski

EFFICIENT PARALLELIZATION FOR 3D-3V SPARSE GRID PARTICLE-IN-CELL: SHARED MEMORY SYSTEMS ARCHITECTURES

F. DELUZET[†], G. FUBIANI[‡], L. GARRIGUES[‡], C. GUILLET[†‡⋆], AND J. NARSKI[†]

ABSTRACT. Particle-In-Cell (PIC) schemes are ones of the most broadly used numerical methods in kinetic simulation of plasmas. The contribution of the present paper is dedicated to the introduction of parallelization strategies specific to shared memory architectures tailored for Particle-In-Cell methods implementing sparse grid reconstructions. These strategies operate the different parallelisms genuine to Sparse-PIC methods to obtain speed-up exceeding 100 on 128 cores. On top of that substantial gains are introduced for sequential as well as parallel implementations thanks to the hierarchization procedure. It consists in decomposing the information carried by sparse grids onto hierarchical basis functions, entailing a significantly reduced number of operations.

Keywords. Plasma physics, Particle-In-Cell (PIC), sparse grids, sparse grid combination technique, parallelization , OpenMP

1. INTRODUCTION

Among the most successful models for computer simulation of kinetic plasma problems are particle methods, Particle-In-Cell (PIC) being one of the most used for years [8, 9, 30, 35], still being carried on [16, 17, 21, 38]. The method consists in a coupling between a Lagrangian discretization for the Vlasov equation, based on the integration of numerical particle trajectories and a mesh-based discretization of Poisson's equation for the computation of the self-consistent electrostatic field. The major hurdle of the method is the statistical error originating from the sampling of the distribution function by a limited number of numerical particles. Therefore, noise reduction strategies such as variance reduction methods [18], filtering methods [24], or more recently sparse grid techniques [17, 22, 23, 36, 39] have received a lot of attention. Sparse grids, originally developed for the interpolation of high dimensional functions [11, 12, 13, 29], then extended to the approximation of partial differential equations [26, 27, 41, 40] have lately been applied to PIC in the framework of the sparse grid combination technique [10, 25]. The use of sparse grids within PIC methods offers a significant mitigation of the statistical noise and a reduction of the grid operation complexity (*e.g.* the resolution of Poisson equation).

The present paper extends the work of [17] to three dimensional framework with a particular focus on optimization and parallelization of the Sparse-PIC methods on shared memory architectures. Sparse-PIC methods are particularly memory efficient, with respect to the size of the sparse grids used to accumulate the density and compute the electric field as well as the array used to store the particles properties. The number of those numerical particles is significantly reduced, compared to standard methods, owing to the better control of the statistical noise. This limited memory footprint calls for the development of parallel implementations on shared memory architectures. Though these platforms offer a limited amount of memory compared to distributed architectures, this limitation is not an issue for Sparse-PIC methods. Therefore implementations of these methods scalable on tens (and tomorrow thousands) of cores open the way to 3D-3V simulations on simple and inexpensive platforms. Beyond this first objective, an efficient implementation on shared memory architectures provide a building block for porting these methods on many cores (CPU) platforms consisting of multiple distributed nodes, one node being a shared memory platform with tens of cores.

The scope of the present paper is therefore dedicated to CPU shared memory architectures, the porting to GPU will be addressed in a forthcoming paper.

Standard PIC methods are not well suited for scalable implementations onto shared memory architectures. This is mainly due to the fact that PIC methods are globally memory bound. Thus the multiplication of the cores without any significant increase of the memory bandwidth brings a poor speed-up. This issue is analysed in [43] and received a lot of attention for years [4, 5, 7]. Different workarounds are proposed to favor the locality of the data, increasing the cache reuse, therefore mitigating the number of requests to the main memory. Strategies are also proposed for Non-Uniform Memory Acces (NUMA) architecture platforms. The purpose there is to take advantage of pieces of memory, local to a subset of cores, to enhance the scalability of the algorithm. This is achieved thanks to a decomposition of the population

⋆ CORRRESPONDING AUTHOR

[†] UNIVERSITÉ DE TOULOUSE; UPS, INSA, UT1, UTM,, INSTITUT DE MATHÉMATIQUES DE TOULOUSE,, CNRS, INSTITUT DE MATHÉMATIQUES DE TOULOUSE UMR 5219,, F-31062 TOULOUSE, FRANCE

[‡]LAPLACE, UNIVERSITÉ DE TOULOUSE, CNRS, INPT, UPS,, 118 ROUTE DE NARBONNE, 31062 TOULOUSE, FRANCE

of particles into samples, each of which being assigned to a NUMA domain and the related operations performed by the local subset of cores.

The purpose of the present paper is to propose a first efficient implementation of Sparse-PIC methods, combining sound memory management and parallelization strategies exploiting the genuine properties of Sparse-PIC methods. The sparse grid reconstructions require the operations of each particles with numerous (tens) anisotropic grids, with coarse discretizations and different sparsity patterns. These grids are referred to as *component grids*. The set of nodes of all the component grids is designated by the *sparse grid* terminology. The sparsity of these grids reduces their memory footprint to few kilobytes (KB) which is small enough to be contained in the highest level of the cache hierarchy common to any modern CPU core. Furthermore, the interaction of the particles with two different grids are independent and can therefore be processed concurrently. This defines a new level of parallelism, specific to Sparse-PIC methods, that is central in obtaining a good scalability. The issue there lies in the load balancing, the number of grids being not necessarily a multiple of the number of cores. A strategy is therefore proposed to preserve the scalability close to optimal and finally obtain speed-up larger than 100 on 128 cores.

On top of that a hierarchization of the data accumulated on the component grids is introduced. It amounts to decompose the information onto a basis of hierarchical functions, with in the end, a very significant reduction of the complexity of the Sparse-PIC algorithm. As an illustration, a non-linear Landau damping simulation with sparse grid reconstructions, equivalent in classical framework to a grid with 512^3 cells and more than 4000 particles per cell, is performed on a portable device with eight cores and 32GB of memory. This configuration is utterly unreachable on most hardware with standard methods, the number of particles necessary being then roughly 10^{12}, requiring 40TB.

The organization of the paper is the following. In section 2, the PIC scheme is presented in its classical formulation. After introducing a necessary background on sparse grids, the combination technique is specifically applied to Sparse-PIC methods in both nodal and hierarchical representations. The section 3 is devoted to the parallelization of the methods, chiefly the sparse grid scheme introduced in the previous section. Efficient strategies tailored to non-uniform memory architectures are developed in order to maximize memory re-use. Finally in section 4, the efficiency of the parallelization strategies and the hierarchical representation are assessed on three dimensional classical test cases: the non-linear Landau damping and the diocotron instability using three different hardware: an uniform memory architecture and two non-uniform memory architectures (cc-NUMA platforms).

2. Particle-In-Cell and sparse grid reconstructions

2.1. **Notations.** Let us introduce some definitions and notations for the rest of the paper. Let d be the dimension of the problem considered, $\mathbf{l} = (l_1, ..., l_d) \in \mathbb{N}^d$ a multi-index denoting the level, *i.e.* the discretization resolution in a multivariate sense and $\mathbf{i} = (i_1, ..., i_d) \in \mathbb{N}^d$ be a multi-index denoting spatial positions. We define the following order relations on multi-indices:

$$\mathbf{k} \leq \mathbf{l} \;\Leftrightarrow\; \forall j \in \{1, ...d\}\; k_j \leq l_j, \tag{1}$$

$$\mathbf{k} < \mathbf{l} \;\Leftrightarrow\; \mathbf{k} \leq \mathbf{l} \text{ and } \exists\, j \in \{1, ..., d\}\; s.t.\; k_j < l_j, \tag{2}$$

and discrete l^1 norm and l^∞ norm by:

$$|\mathbf{l}|_1 := \sum_{j=1}^{d} |l_j|, \quad \|\mathbf{l}\|_\infty := \max_{j \in \{1,...,d\}} |l_j|. \tag{3}$$

Let us consider the family of d-dimensional anisotropic grids on the unit interval $\Omega = [0, 1]^d$ called component grids, or sub-grids:

$$\Omega_{h_\mathbf{l}} := \left\{ \mathbf{i} h_\mathbf{l} \mid \mathbf{i} \in I_{h_\mathbf{l}} \right\}, \quad I_{h_\mathbf{l}} := [\![0, h_{l_1}^{-1}]\!] \times ... \times [\![0, h_{l_d}^{-1}]\!] \subset \mathbb{N}^d, \tag{4}$$

and parameterized by the multi-index level $\mathbf{l} \in L$:

$$L := \bigcup_{j \in [\![0, d-1]\!]} L_j, \quad L_j := \{\mathbf{l} \in \mathbb{N}^d \mid |\mathbf{l}|_1 = n + d - 1 - j,\ \mathbf{l} \geq \mathbf{1}\}, \tag{5}$$

where $h_\mathbf{l} := (h_{l_1}, ..., h_{l_d}) := 2^{-\mathbf{l}}$ is called the grid discretization. We consider also a regular isotropic grid, named Cartesian grid, denoted $\Omega_{h_n}^{(\infty)}$ corresponding to a component grid of level $\mathbf{n} = n \cdot \mathbf{1}$ with discretization h_n in all directions, which is typically the underlying grid in PIC methods. An illustration of the full grid, the component grid of level $\mathbf{l} = (5, 1, 1)$ and the sparse grid [13], composed of the set of nodes of all the component grids, is provided on figure 1.

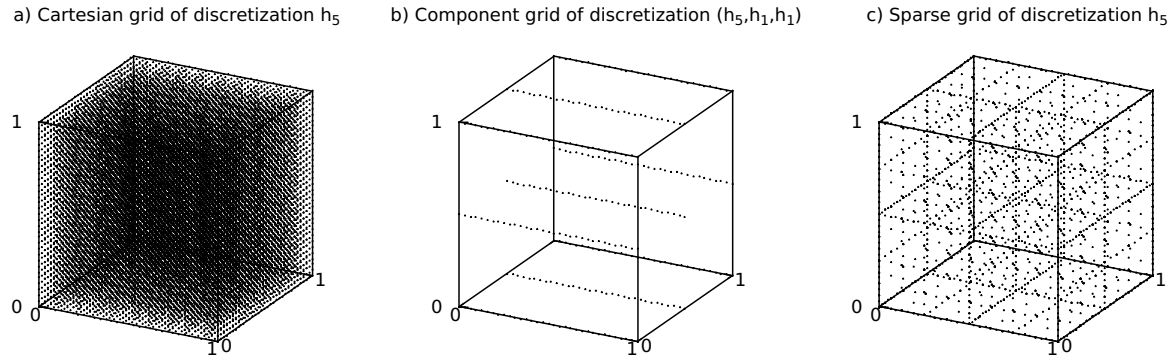

Figure 1. The discretization parameter is $n = 5$: Panel a) represents the Cartesian grid of discretization h_n; panel b) represents the component grid of level $\mathbf{l} = (5, 1, 1)$; panel c) represents the sparse grid, composed of all the nodes of the component grid.

2.2. **Particle-In-Cell scheme.** The reference scheme for particular methods applied to plasma physic is detailed in this section. Let us consider the non-relativistic Vlasov-Poisson system with external magnetic field $\mathbf{B}$:

$$\begin{cases} \dfrac{\partial f_s}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} f_s + \dfrac{q_s}{m_s}(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \nabla_{\mathbf{v}} f_s = 0, \\ \nabla \cdot \mathbf{E} = \dfrac{\rho}{\varepsilon_0}, \qquad \mathbf{E} = -\nabla \Phi, \end{cases} \tag{6}$$

where $f_s(\mathbf{x}, \mathbf{v}, t)$ is the phase-space distribution of the species s; q_s, m_s are the corresponding charge and mass, $\mathbf{E}$ is the electric field and ρ is the charge density obtained from the phase-space distribution of each species as follows:

$$\rho(\mathbf{x}, t) = \sum_s q_s \int f_s(\mathbf{x}, \mathbf{v}, t) d\mathbf{v}. \tag{7}$$

The particle distribution f_s is represented by a collection of N numerical particles whose positions and velocities are denoted $(\mathbf{x}_p, \mathbf{v}_p)$, for $p = 1, ..., N$ and with total charge Q. The Standard Particle-In-Cell scheme (PIC-Std) [14, 15] consists of four steps repeated at each time iteration, which are detailed in the algorithm 1. The charge density of each type of particle is accumulated onto the Cartesian grid according to the following equations:

$$\hat{\rho}_{h_n}(\mathbf{x}) = \frac{Q}{N} \sum_{p=1}^{N} \mathcal{S}_{h_n}(\mathbf{x} - \mathbf{x}_p), \quad \mathcal{S}_{h_n} = \left(h_n^{-1} \varphi\left(h_n^{-1} \cdot \right) \right)^{\otimes d}, \quad \varphi(x) = \max(1 - |x|, 0), \tag{8}$$

where h_n is the discretization of the Cartesian grid, defined in section 2.1.

Algorithm 1 Std-PIC scheme

Require: Particle positions and velocities $(\mathbf{x}_p, \mathbf{v}_p)$, $p \in [\![1, N]\!]$, time step Δt, external fixed magnetic field $\mathbf{B}$.
for each time step Δt **do**
Accumulate the charge density onto $\Omega_{h_n}^{(\infty)}$ according to equations (8).
Compute the electric field from the charge density on $\Omega_{h_n}^{(\infty)}$ with finite differences according to:

$$\mathbf{E}_{h_n} = -\nabla_{h_n} \Phi_{h_n}, \quad \Delta_{h_n} \Phi_{h_n} = -\frac{\hat{\rho}_{h_n}}{\varepsilon_0}, \tag{9}$$

where ∇_{h_n}, Δ_{h_n} are standard finite differences operators (see for instance [17]).
Interpolate the electric field at the particle positions.
Update the particle positions and velocities by integrating Newton's law:

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{v}_p, \qquad \frac{d\mathbf{v}_p}{dt} = \frac{q_s}{m_s}(\mathbf{E} + \mathbf{v}_p \times \mathbf{B})|_{\mathbf{x}=\mathbf{x}_p}. \tag{10}$$

end for

The error between the density and its statistical estimator can be decomposed into the bias of the estimator and a centered random variable, denoted $\mathcal{V}_{N,h_n}$ corresponding to the error stemming from the variance of the sampling with a finite number of particles, the latter's being the cause of the statistical noise [17]:

$$(\hat{\rho}_{h_n} - \rho)(\mathbf{x}) = \text{Bias}(\hat{\rho}_{h_n})(\mathbf{x}) + \mathcal{V}_{N,h_n}(\mathbf{x}), \tag{11}$$

where

$$\text{Bias}(\hat{\rho}_{h_n})(\mathbf{x}) = O\left(h_n^2\right), \qquad \mathbb{V}(\mathcal{V}_{N,h_n}(\mathbf{x}))^{\frac{1}{2}} = O\left(\left(Nh_n^d\right)^{-\frac{1}{2}}\right). \tag{12}$$

The convergence of the estimator, *i.e.* the bias and the variance, requires respectively:

$$\lim_{n\to+\infty} h_n = 0, \qquad \lim_{n,N\to+\infty} Nh_n^d = +\infty, \tag{13}$$

and thus an extremely large number of particles N is necessary in order to achieve a low statistical error. Practically, the statistical noise is generally the most detrimental component of the error to the precision of the numerical approximation. In order to control the amount of statistical noise in the simulation, we refer to P_c, the average number of particles per cell of the grid. The number of numerical particles is given by:

$$N = P_c * h_n^{-d}. \tag{14}$$

2.3. **Sparse grid reconstructions.**

2.3.1. *Nodal basis or hierarchical basis representation.* As a preliminary step to sparse grid techniques, one shall introduce some interpolation tools. Let $\mathbf{l} \in \mathbb{N}^d$, $\mathbf{i} \in I_{h_\mathbf{l}}$ be multi-indexes and consider basis functions defined by tensor products of one-dimensional hat functions as follows:

$$\varphi_{h_\mathbf{l},\mathbf{i}}(\mathbf{x}) := \left(\bigotimes_{j=1}^{d} \varphi_{h_{l_j},i_j}\right)(\mathbf{x}), \quad \varphi_{h_{l_i},i_j}(x) := \varphi\left(h_{l_j}^{-1}(x_j - i_j h_{l_j})\right), \quad \varphi(x) = \max\left(1 - |x|, 0\right), \tag{15}$$

where $h_\mathbf{l}$ is the grid discretization of the component grid $\Omega_{h_\mathbf{l}}$ defined in section 2.1. These functions verify a partition of unit property:

$$\sum_{\mathbf{i}\in I_{h_\mathbf{l}}} \varphi_{h_\mathbf{l},\mathbf{i}}(\mathbf{x}) = 1. \tag{16}$$

The space of d-dimensional hat functions with respect to the component grid $\Omega_{h_\mathbf{l}}$, denoted $V_{h_\mathbf{l}}$, is defined by:

$$V_{h_\mathbf{l}} := \text{span}\{\varphi_{h_\mathbf{l},\mathbf{i}} \mid \mathbf{i} \in I_{h_\mathbf{l}}\}, \tag{17}$$

where $\{\varphi_{h_\mathbf{l},\mathbf{i}} \mid \mathbf{i} \in I_{h_\mathbf{l}}\}$ is called the nodal basis of the space $V_{h_\mathbf{l}}$ and $I_{h_\mathbf{l}}$ the nodal basis index set. Additionally we introduce hierarchical increments of $V_{h_\mathbf{l}}$ [13, 20], denoted by $W_{h_\mathbf{l}}$ and defined by:

$$W_{h_\mathbf{l}} := V_{h_\mathbf{l}} \backslash \bigoplus_{j=1}^{d} V_{h_{\mathbf{l}-\mathbf{e}_j}}, \quad \text{where } V_{h_\mathbf{l}} := 0 \ \text{ if } \ \exists j \in \{1,...,d\} \ s.t. \ l_j = -1, \tag{18}$$

and $\mathbf{e}_j \in \mathbb{N}^d$ is the unit vector with the j^{th} coordinate equal to one. The hierarchical increment contains all $\varphi_{h_\mathbf{l},\mathbf{i}} \in V_{h_\mathbf{l}}$ that are not included in smaller $V_{h_\mathbf{k}}$, with $\mathbf{k} < \mathbf{l}$. It can also be expressed in the following form:

$$W_{h_\mathbf{l}} = \text{span}\{\varphi_{h_\mathbf{l},\mathbf{i}} \mid \mathbf{i} \in \mathcal{B}_{h_\mathbf{l}}\}, \quad \mathcal{B}_{h_\mathbf{l}} := \left\{\mathbf{i} \in \mathbb{N}^d \mid \mathbf{0} \le \mathbf{i} \le h_\mathbf{l}^{-1}, \ \mathbf{i} \text{ odd}\right\}, \tag{19}$$

where $\{\varphi_{h_\mathbf{l},\mathbf{i}} \mid \mathbf{i} \in \mathcal{B}_{h_\mathbf{l}}\}$ is called the hierarchical basis of the space $V_{h_\mathbf{l}}$ and $\mathcal{B}_{h_\mathbf{l}}$ the hierarchical basis index set. The space of piecewise d-linear functions of level $\mathbf{l}$ with respect to $\Omega_{h_\mathbf{l}}$ can be represented with its hierarchical basis:

$$V_{h_\mathbf{l}} = \bigoplus_{k_1\le l_1} \cdots \bigoplus_{k_d \le l_d} W_{h_\mathbf{k}} = \bigoplus_{\mathbf{k}\le\mathbf{l}} W_{h_\mathbf{k}}, \tag{20}$$

Thus, each function $v_{h_\mathbf{l}} \in V_{h_\mathbf{l}}$ can be represented identically in the hierarchical (21) or nodal (22) basis of $V_{h_\mathbf{l}}$:

$$v_{h_\mathbf{l}} = \sum_{\mathbf{k}\le\mathbf{l}} \hat{v}_{h_\mathbf{k}} = \sum_{\mathbf{k}\le\mathbf{l}} \sum_{\mathbf{i}\in\mathcal{B}_{h_\mathbf{k}}} \alpha_{\mathbf{k},\mathbf{i}} \varphi_{h_\mathbf{k},\mathbf{i}}, \tag{21}$$

$$v_{h_\mathbf{l}} = \sum_{\mathbf{i}\in I_{h_\mathbf{l}}} \beta_{\mathbf{l},\mathbf{j}} \varphi_{h_\mathbf{l},\mathbf{i}}, \tag{22}$$

where $\alpha_{\mathbf{k},\mathbf{i}}$ are the coefficients of $v_{h_\mathbf{l}}$ in the hierarchical basis, called hierarchical surplus, that shall be defined later; and $\beta_{\mathbf{l},\mathbf{j}}$ are the coefficients of $v_{h_\mathbf{l}}$ in the nodal basis which are the nodal values of the function $v_{h_\mathbf{l}}$. We introduce the space of d-dimensional piecewise linear functions with respect to $\Omega_{h_n}^{(\infty)}$, denoted $V_{h_n}^{(\infty)}$.

$$V_{h_n}^{(\infty)} = \bigoplus_{|\mathbf{l}|_\infty \leq n} W_\mathbf{l} = \text{span}\{\varphi_{h_\mathbf{n},\mathbf{i}} \mid \mathbf{i} \in \mathbb{N}^d \mid \mathbf{0} \leq \mathbf{i} \leq h_\mathbf{n}^{-1}\}. \tag{23}$$

Eventually, we introduce the interpolation operators in nodal and hierarchical basis defined by:

$$I_{V_{h_\mathbf{l}}}^{\mathcal{N}} f = \sum_{\mathbf{i} \in I_{h_\mathbf{l}}} f(\mathbf{i}h_\mathbf{l})\varphi_{h_\mathbf{l},\mathbf{i}}, \qquad I_{V_{h_\mathbf{l}}}^{\mathcal{H}} f = \sum_{\mathbf{k} \leq \mathbf{l}} \sum_{\mathbf{i} \in \mathcal{B}_{h_\mathbf{l}}} \alpha_{\mathbf{k},\mathbf{i}} \varphi_{h_\mathbf{k},\mathbf{i}}, \tag{24}$$

2.3.2. *Sparse grid combination technique.* In this section, the sparse grid combination technique [25] is introduced. It is a method based on sparse grid representation of functions and using linear combination of contributions from coarse grids, the component grids, in order to reconstruct a solution on the refined Cartesian grid. The method shall provide within the PIC scheme a precise reconstruction of the electric potential initially computed with finite differences on the component grids. The major benefit of the method is a significant reduction of statistical noise resulting from the deposition of the charge density onto several coarse grids instead of an unique fine grid. An additional gain is to be expected on the complexity of the linear system issued from the Poisson equation.

Let us consider the set of d-dimensional component grids $(\Omega_{h_\mathbf{l}})_{\mathbf{l} \in L}$ defined in equation (5) and $\Phi_{h_\mathbf{l}}$ be the solution of the linear system associated to the Poisson equation on the grid $\Omega_{h_\mathbf{l}}$ defined by:

$$\mathbf{E}_{h_\mathbf{l}} = -\nabla_{h_\mathbf{l}} \Phi_{h_\mathbf{l}}, \quad \Delta_{h_\mathbf{l}} \Phi_{h_\mathbf{l}} = -\frac{\hat{\rho}_{h_\mathbf{l}}}{\varepsilon_0}, \tag{25}$$

where $\nabla_{h_\mathbf{l}}$, $\Delta_{h_\mathbf{l}}$ are the finite differences operators (see [17]) associated to the component grid $\Omega_{h_\mathbf{l}}$. From this, an interpolation onto the space $V_{h_\mathbf{l}}$, denoted $I_{V_{h_\mathbf{l}}}^{\mathcal{N}} \Phi_{h_\mathbf{l}}$, is constructed in its nodal basis representation according to the relation:

$$I_{V_{h_\mathbf{l}}}^{\mathcal{N}} \Phi_{h_\mathbf{l}} = \sum_{\mathbf{i} \in I_{h_\mathbf{l}}} \Phi_{h_\mathbf{l}}(\mathbf{j}h_\mathbf{l})\varphi_{h_\mathbf{l},\mathbf{i}}, \tag{26}$$

From all the contributions on each component grid $\Omega_{h_\mathbf{l}}$ for $\mathbf{l} \in L$, a fine reconstruction of the electric potential on the Cartesian grid $\Omega_n^{(\infty)}$, denoted $\Phi_{h_n}^C$, is constructed by linear combination:

$$\Phi_{h_n}^C = \sum_{\mathbf{l} \in L} c_\mathbf{l} I_{V_{h_\mathbf{l}}}^{\mathcal{N}} \Phi_{h_\mathbf{l}}, \tag{27}$$

where $c_\mathbf{l} = (-1)^j \dfrac{(d-1)!}{j!(d-1-j)!}$ for $\mathbf{l} \in L_j$ are the combination coefficients.

Proposition 2.1. *Let Φ and ρ be sufficiently smooth function, then the function Φ_{h_l} defined by equation (25) verifies the following point-wise error expression for each $\boldsymbol{l} \in L$:*

$$\Phi(\boldsymbol{x}) - \Phi_{h_l}(\boldsymbol{x}) = \sum_{m=1}^{d} \sum_{\substack{\{j_1,\ldots,j_m\} \\ \subset \{1,\ldots,d\} \\ j_i \neq j_k}} a_{j_1,\ldots,j_m}(\boldsymbol{x}; h_{l_{j_1}}, \ldots, h_{l_{j_m}}) h_{l_{j_1}}^2 \ldots h_{l_{jm}}^2 \tag{28}$$

with bounded $\|a_{j_1,\ldots,j_m}(\cdot; h_{l_{j_1}}, \ldots, h_{l_{j_m}})\|_\infty \leq \kappa$, and the sparse grid reconstruction $\Phi_{h_n}^C$ verifies:

$$\|\Phi_{h_n}^C - \Phi\|_\infty = O\left(h_n^2 \cdot |\log h_n|^{d-1}\right) + O\left((Nh_n)^{-\frac{1}{2}} |\log h_n|^{d-1}\right) \tag{29}$$

Proof. We refer to [17] for the proof of this proposition. □

One shall recall that, for the standard scheme, the approximation error of the electric potential scales with h_n^2 which is of the same order, save a negligible logarithmic term ($|\log h_n|^{d-1}$), than the sparse grid approximation according to the above theorem.

2.3.3. *Nodal basis Sparse-PIC scheme.* Let us now introduce a scheme taking advantages of these sparse grid techniques in order to achieve a reduction of the computational time. The scheme is referred to as the Nodal basis Sparse grid Particle-In-Cell scheme (PIC-NSg) [17] and presented in the algorithm 2. Within this scheme the grid operations (charge deposition and field solver) are considered separately and independently on the component grids $\Omega_{h_\mathbf{l}}$. After being computed on each component grid, the electric potential is reconstructed on the full grid $\Omega_{h_n}^{(\infty)}$ by combining all the contributions according to equations (26), (27). Eventually the electric potential is differentiated, interpolated from the full grid to the particles positions and the particle positions and velocities are updated similarly to the standard scheme.

Algorithm 2 PIC-NSg scheme

Require: Particle positions and velocities $(\mathbf{x}_p, \mathbf{v}_p)$, time step Δt, external magnetic field $\mathbf{B}$.
  for each time step Δt **do**
    for each component grid of index $\mathbf{l} \in L$ **do**
      Accumulate the charge density onto $\Omega_{h_\mathbf{l}}$ according to:

$$\hat{\rho}_{h_\mathbf{l}}(\mathbf{x}) = \frac{Q}{N}\sum_{p=1}^{N}\mathcal{S}_{h_\mathbf{l}}\left(\mathbf{x}-\mathbf{x}_p\right), \quad \mathcal{S}_{h_\mathbf{l}} = \bigotimes_{j=1}^{d}\left(h_{l_j}^{-1}\varphi\left(h_{l_j}^{-1}\cdot\right)\right), \quad \varphi(x) = \max\left(1-|x|,0\right) \tag{30}$$

      Compute the electric potential from the charge density on $\Omega_{h_\mathbf{l}}$ according to equation (25).
      Interpolate the electric potential onto $V_{h_\mathbf{l}}$ (26).
    end for
    Combine the electric potential onto $\Omega_{h_n}^{(\infty)}$ (27) in nodal basis.
    Differentiate the electric potential on $\Omega_{h_n}^{(\infty)}$.
    Interpolate the electric field at the particle positions.
    Update the particle positions and velocities (10).
  end for

It has already been established in [17] that the statistical noise on the charge density is bounded by $|\log h_n|^{d-1} \cdot (Nh_n)^{-\frac{1}{2}}$ which is a significant improvement relative to the standard scheme estimate $(Nh_n^d)^{-\frac{1}{2}}$.

Let us recall the average number of particles per cell for standard schemes, denoted P_c and introduced in equation (14). We are interested in determining the number of numerical particles required for Sparse-PIC simulation to set an equivalent amount of statistical noise with respect to the standard scheme. It can be estimated by considering the sum of all component grid cells as follows:

$$\begin{aligned} N &= P_c * \left(\sum_{\mathbf{l}\in L}|c_\mathbf{l}|h_{l_1}^{-1}\ldots h_{l_d}^{-1}\right) \\ &= P_c * h_n^{-1} * \left(\frac{9}{2}n^2 - \frac{3}{2}n + 1\right), \quad \text{for } d = 3. \end{aligned} \tag{31}$$

What follows is that the same amount of statistical noise as in the standard scheme is obtained for much less particles (because $\frac{9}{2}n^2 - \frac{3}{2}n + 1 \ll h_n^{-2}$ for large n), drastically reducing memory requirements of three dimensional simulations.

2.3.4. *Combination in hierarchical basis.* In the present paper, an alternative representation of the sparse grid based on the hierarchical decomposition of the basis functions, yielding the same reconstruction than the previous method, is introduced in order to reduce the complexity of the combination operations. The coefficients in the hierarchical basis $(\alpha_{\mathbf{k},\mathbf{i}})_{\mathbf{i}\in\mathcal{B}_{h_\mathbf{k}},\mathbf{k}\leq\mathbf{l}}$ of the function $\Phi_{h_\mathbf{l}} \in V_{h_\mathbf{l}}$ are determined by a transformation from the nodal basis (whose coefficients are plainly the values of $\Phi_{h_\mathbf{l}}$ on the nodes of the full grid) to the hierarchical basis and the transformation is called hierarchization. Hierarchization is done by applying a d-dimensional stencil constructed by tensor product of one-dimensional stencil [13]:

$$\mathcal{H}_{\mathbf{i}h_\mathbf{k},\mathbf{k}} = \bigotimes_{j=1}^{d}\mathcal{H}_{i_jh_{k_j},k_j}, \quad \mathcal{H}_{i_jh_{k_j},k_j} = \left[-\frac{1}{2} \quad 1 \quad -\frac{1}{2}\right]_{i_jh_{k_j},k_j}, \tag{32}$$

where the one-dimensional stencil stands for:

$$\mathcal{H}_{i_jh_{k_j},k_j}f = f(i_jh_{k_j}) - \frac{1}{2}\left[f\left(\frac{i_j-1}{2}h_{k_j-1}\right) + f\left(\frac{i_j+1}{2}h_{k_j-1}\right)\right]. \tag{33}$$

The hierarchical surpluses of a function are given by:

$$\alpha_{\mathbf{k},\mathbf{i}} := \mathcal{H}_{\mathbf{i}h_{\mathbf{k}},\mathbf{k}} f, \qquad \mathbf{i} \in \mathcal{B}_{h_{\mathbf{k}}},\ \mathbf{k} \in \mathbb{N}^d. \tag{34}$$

A matrix representation of the hierarchization is also provided in the A. The inverse operation consisting in a transformation from the hierarchical basis to the nodal basis is named dehierarchization [28] and is also done by applying a d-dimensional stencil:

$$\mathcal{D}_{\mathbf{i}h_{\mathbf{k}},\mathbf{k}} = \bigotimes_{j=1}^{d} \mathcal{D}_{i_j h_{k_j},k_j}, \quad \mathcal{D}_{i_j h_{k_j},k_j} = \begin{bmatrix} \frac{1}{2} & 1 & \frac{1}{2} \end{bmatrix}_{i_j h_{k_j},k_j}, \tag{35}$$

where the one-dimensional stencil is defined by:

$$\mathcal{D}_{i_j h_{k_j},k_j} f = f(i_j h_{k_j}) + \frac{1}{2}\left[f\left(\frac{i_j - 1}{2} h_{k_j - 1}\right) + f\left(\frac{i_j + 1}{2} h_{k_j - 1}\right)\right]. \tag{36}$$

Proposition 2.2. *The sparse grid reconstruction defined in equation (27) can be expressed in the following form:*

$$\Phi_{h_n}^C = \sum_{|\boldsymbol{k}|_\infty \leq n} \sum_{\boldsymbol{i} \in \mathcal{B}_{h_{\boldsymbol{l}}}} \gamma_{\boldsymbol{k},\boldsymbol{i}} \varphi_{h_{\boldsymbol{k}},\boldsymbol{i}}, \tag{37}$$

where:

$$\gamma_{\boldsymbol{k},\boldsymbol{i}} = \sum_{\boldsymbol{l} \in L} c_{\boldsymbol{l}} \beta_{\boldsymbol{k},\boldsymbol{i}}, \qquad \beta_{\boldsymbol{k},\boldsymbol{i}} := \begin{cases} \mathcal{H}_{\boldsymbol{i}h_{\boldsymbol{k}},\boldsymbol{k}} \Phi_{h_{\boldsymbol{k}}}, & \text{if } \boldsymbol{k} \leq \boldsymbol{l}, \\ 0 & \text{else}. \end{cases} \tag{38}$$

Proof. See A. □

The result of proposition 2.2 allows us to conceive an algorithm for the combination of the electric potential based on the hierarchical basis representation, which is presented in algorithm 3.

Algorithm 3 Combination in hierarchical basis (PIC-HSg)

Require: Approximation of the electric potential $\Phi_{h_{\mathbf{l}}}$ on the component grid.
  for each component grid of index $\mathbf{l} \in L$ **do**
    for each node of the component grid $\mathbf{i} \in I_{\mathbf{l}}$ **do**
      Apply the transformation into the hierarchical basis:

$$\alpha_{\mathbf{l},\mathbf{i}} \leftarrow \mathcal{H}_{\mathbf{i}h_{\mathbf{l}},\mathbf{l}} \Phi_{h_{\mathbf{l}}}. \tag{39}$$

      Determine the full grid index node $\mathbf{j}$ corresponding to the index node $\mathbf{i}$.
      Add the contribution $\gamma_{\mathbf{n},\mathbf{j}} := c_{\mathbf{l}} \alpha_{\mathbf{l},\mathbf{i}}$ to $\Omega_{h_n}^{(\infty)}$ at the node $\mathbf{j}h_n$.
    end for
  end for
  for each node of the full grid $\mathbf{j} \in I_n$ **do**
    Apply the transformation into the nodal basis:

$$\Phi_{h_n}^C(\mathbf{j}h_n) \leftarrow \mathcal{H}^{-1}_{\mathbf{j}h_{\mathbf{n}},\mathbf{n}} \gamma_{\mathbf{n},\mathbf{j}}. \tag{40}$$

  end for

2.3.5. *Undirectional principle.* The undirectional principle is a way to perform hierarchization of a multiple dimension function by a series of one-dimensional hierarchization and is detailed in [31, 32, 33]. We briefly recall the principle of the method in this section. The principle exploits the tensor structure of the basis functions; for the d-dimensional hierarchization is done by hierarchize each dimension one after the other. The hierarchization with the unidirectional principle is presented in d dimensions in algorithm 4. The outer loop iterates over the d dimensions and constitutes the unidirectional principle. For a specific dimension i, the data are split into one-dimension poles upon which the operations are made. A pole in dimension i consists of all points of the grid which only differ in the i^{th} component, that is which lie on a line parallel to the i^{th} coordinate axis. For each pole, the operation is solely the one-dimensional hierarchization introduced in equation (33).

Algorithm 4 Hierarchization with unidirectional principle

Require: $\mathbf{l} = (l_1, ..., l_d) \geq 0$ level, nodal coefficients stored in array[:]
Ensure: hierarchical coefficients stored in array [:]
  for i **from** 1 **to** d **do**
    for $p = l_i$ **downto** 0 **do**
      $\tilde{\mathbf{l}} \leftarrow (..., l_{i-1}, p, l_{i+1}, ...)$
      for all nodes $\mathbf{x}$ of level $\tilde{\mathbf{l}}$ **do**
        Let $\mathbf{x}_l$ be the left hierarchical ancestor of $\mathbf{x}$ in dimension i
        Let $\mathbf{x}_r$ be the right hierarchical ancestor of $\mathbf{x}$ in dimension i
        array[$\mathbf{x}$]$\leftarrow$ array[$\mathbf{x}$] $-\frac{1}{2}$(array[$\mathbf{x}_l$]+array[$\mathbf{x}_r$])
      end for
    end for
  end for

2.3.6. *Complexity of the different combinations.* In the previous section, the sparse grid reconstruction of the solution from the component grid contributions has been equivalently exposed in nodal or hierarchical basis. The methods, though providing equivalent results, differ on the operations involved and thus an investigation of the complexity of both approaches is provided in the present section. The combination technique in the hierarchical basis follows four steps:

- *Hierarchization:* For each component grid $\Omega_{h_\mathbf{l}}$, $\mathbf{l} \in L$, a transformation to the hierarchical basis is performed by applying $\mathcal{H}_\mathbf{l}$. According to the unidirectional principle, the hierarchization of each grid $\Omega_{h_\mathbf{l}}$ amounts to a number of operations $n_{op}(\mathbf{l})$[31]:

$$n_{op}(\mathbf{l}) = 2 \cdot \sum_{i=1}^{d} \left((2^{l_i+1} - 2 \cdot l_i - 2) \cdot \prod_{\substack{m=1 \\ m \neq i}}^{d} (2^{l_m} - 1) \right), \tag{41}$$

  then, the complexity of the hierarchization is:

$$\sum_{\mathbf{l} \in L} n_{op}(\mathbf{l}) = O\left(n^{d-1} \cdot 2^n\right). \tag{42}$$

- *Prolongation:* Since every component grid involved in the combination is included in the full grid, the hierarchical surplus of each component grid are prolongated onto the full grid. This step yields no computation operation.
- *Combination:* The hierarchical surplus from each component grid are combined onto the full grid.The number of operations $n_{op}(\mathbf{l})$ for each component grid $\Omega_{h_\mathbf{l}}$ is:

$$n_{op}(\mathbf{l}) = 2 \cdot \prod_{m=1}^{d} (2^{l_m} + 1),$$

  and the complexity is similar to equation (42).
- *Dehierarchization:* A transformation from the hierarchical basis to the nodal basis is performed on the full grid to recover the values of the sparse grid reconstruction. Applying equation (41) to the full grid, the number of operations for the dehierarchization on the full grid scales with $O\left(2^{nd}\right)$.

The complexities of the hierarchization and combination steps are negligible and thus the complexity of the method is dominated by the dehierarchization complexity, that is $O(2^{nd})$. On the other side, the combination in the nodal basis consists of the following steps:

- *Interpolation:* For each component grid, an interpolation of $\Phi_{h_\mathbf{l}}$ onto the space $V_{h_\mathbf{l}}$ is considered in nodal basis according to the relation (26). The number of operations n_{op} for each component grid is:

$$n_{op}(\mathbf{l}) = (6d + d2^d)(2^n + 1)^d.$$

  Since there are $O\left(n^{d-1}\right)$* component grids in the combination, the number of operations for all the grids scales with $O\left(n^{d-1} \cdot 2^{nd}\right)$.

*There are $\frac{n(n+1)}{2} + \frac{n(n-1)}{2} + \frac{(n-1)(n-2)}{2}$ component grids for $d = 3$.

- *Combination:* The contribution from each component grid is added to the full grid. The number of operations $n_{op}(\mathbf{l})$ for each component grid $\Omega_{h_\mathbf{l}}$ is

$$n_{op}(\mathbf{l}) = 2 \cdot (2^n + 1)^d,$$

and the complexity is similar to the interpolation.

From this investigation upon complexity, it is manifest that the hierarchical representation leading to $O\left(2^{nd}\right)$ operations shall provide a more efficient method than the nodal representation with $O\left(n^{d-1} \cdot 2^{nd}\right)$ operations.

3. Parallelization

3.1. **A non exhaustive overview of optimizations and parallelizations of PIC methods on shared memory architectures.** In PIC simulations, the implementations are usually memory-bounded rather than compute-bounded [4, 43]. The performance of the implementation, with respect to the computational time, is limited by the number of memory accesses. For these kind of applications, multiplying the number of cores involved in the computation increases the memory contention and in the end deteriorates the efficiency.

This feature of PIC methods lies in the interaction of the particles and the array used to accumulate their properties. A particle contributes to the values stored on a limited sub-set of array indices corresponding to the cell it is contained in. The particle being randomly distributed onto the computational domain and free to move during the simulation, a naive implementation of PIC methods does not secure a contiguous access to both the array containing the particle properties and the (grid) array accumulating the density (see figure 2). Indeed, the charge density accumulation, sketched in algorithm 5, consists mainly in loading one particle coordinates, computing the grid cell it is contained in, and accumulate the contribution to the 8 nodes of this cell. Two consecutive particles (with regard to their indices in the particle coordinate array) may contribute to different cells in the density array. This entails either a contiguous memory access in the particle array or in the density (grid) array. A similar issue characterizes the interpolation of the forces from the grid onto the position of the particles. Though, the density accumulation and the field interpolation are only two steps of the complete algorithm they account for a significant part of the computational time due to the tremendous number of particles mandatory to mitigate the statistical noise.

Different workarounds are proposed, mainly based on a so-called sorting [7] of the particles. To this end, each cell of the grid receive a rank, two cells contiguous in memory being associated to consecutive ranks. The particles properties (position, velocity, etc.) are then stored in the particle arrays by rank of the cell they are contained in. By this means, both the grid and the particle arrays may be accessed contiguously providing a better cache reuse for the density accumulation and the field interpolation. Nonetheless, the particles shall be periodically sorted, since during the computation they are likely to cross the cell boundary. This is thus a trade-off between the cost of re-sorting the particle population and increasing the cache-miss rate during iterations. Different elaborated data structures are proposed to alleviate the cost of the periodic particle sorting (see [4, 44] and [5, 6]). Another approach to mitigate the randomness of the memory accesses is to consider domain decomposition [19, 42, 45] with subdomains so small that they can fit in the cache system.

From the parallelization point of view, the parallelization of the particle-grid interaction (density accumulation) may give rise to race conditions. The most obvious strategy consists in organizing the particles into clusters and distribute these clusters onto the available cores. Different particles are then likely to provide contributions to a same grid node which entails a race condition between different cores when writing at the same memory address. This issue is overcome thanks to private copies of grid arrays and reduction operations.

Regarding the field interpolation and the particle pusher the operations related to different particles are independent. Therefore the parallelization is quite straightforward but the scalability may be limited by the poor arithmetic intensity of these steps. On NUMA architectures, the multiplication of core number comes with an increase of the memory bandwidth: the cores are organized in NUMA domains associated to a local memory for which the access is faster compared to that of an other NUMA domain. The parallelization strategy is tuned to cope with the hardware topology. A SPMD (Single Program Multiple Data) is commonly deployed. The particle population is split into multiple samples, one sample being stored in the memory of one NUMA domain and the related operations computed by the cores with a fast access to this memory. The operations on each of the samples are finally reduced over the NUMA domains to recover the complete statistic.

3.2. **Sparse-PIC parallelization.** Sparse grid applications to PIC methods are rather recent [17, 22, 39] and, to date, no efficient implementation combining sound memory management and tailored parallelization strategies has been provided. It is therefore the purpose of the present article. From equation (31), it is manifest that the number of particles required to achieve a given statistical error is much smaller, by hundreds, for the sparse reconstruction. The consequence is that the particle operations (particle pusher) or the operations interacting between the full grid and the particles (field interpolation) are no longer the most time consuming operations. The interactions between the component grids and

Algorithm 5 Projection or charge accumulation

Require: Array particle[:]
Ensure: Array grid[:] containing the charge density
  for all particles **do**
    Read positions of particles in particle array [:]
    Determine $\xi_1, ..., \xi_8$ the eight nodes of the cell containing the particle
    for i **from** 1 **to** 8 **do**
      Determine the charge contribution ρ_i of the particle at the node ξ_i
      Add the contribution ρ_i in the grid array at position ξ_i: grid[ξ_i]$\leftarrow \rho_i$
    end for
  end for

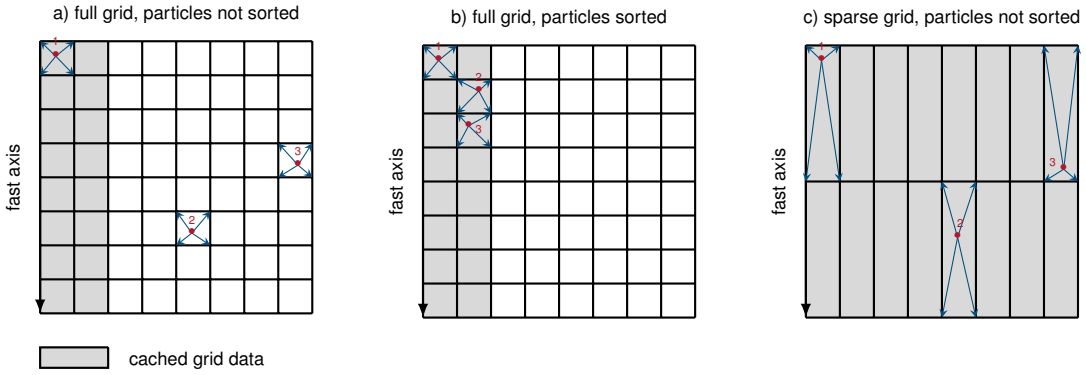

Figure 2. Cache memory management during charge deposition step [45]. The particles are accessed in order 1-2-3 . On panel a) the grid data are accessed non-contiguously and must be loaded from the main memory. On panel b) the grid data are accessed contiguously and can be loaded from the cache. On panel c) The grid data are accessed non-contiguously but can be load from the cache.

the particles (charge deposition) dominate the computational time because of the large number of these grids (about one hundred). As a result, the strategies exposed in this section are strongly motivated by the mitigation of the charge deposition computational load.

3.2.1. *Sparse-PIC memory management and shared memory parallelization.* The strategy proposed within the present paper takes advantage of the extremely reduced memory footprint of sparse grids. Indeed the component grids are stored in tiny array with 2^n, 2^{n+1} or 2^{n+2} nodes when a full grid requires 2^{3n} nodes, n defining the spatial discretization ($h_n = 2^{-n}$). To emphasize this huge memory savings, consider a discretization parameter n ranging from 7 to 9 (corresponding to Cartesian grids with 128^3 to 512^3 nodes), the storage of the Cartesian grid requires 17MB, 134MB, 1GB which hardly fit in the last level cache memory. Conversely, the storage of the larger component grid requires then 4KB, 8KB, 16KB which fits in the L1 data cache memory of the cores of modern CPUs, 32KB being the standard for the L1 data cache. Therefore two memory management policies, depicted in the algorithm 6, both deprived from any particle sorting, may be proposed. Contrariwise, the array used to store the component grids are assumed to fit in the highest level of cache memory, with random accesses to these arrays while the particle arrays are access contiguously. However, these random accesses will not generate cache misses since the whole component grid fits in the L1-cache (see figure 2).

The first strategy relies on computing the interaction of one particle with all the component grids at once, and then proceed with the next particle. For each particle, the contributions are written successively in every array and thus, in order to maximize memory reuse, the data of all arrays must fit in the cache memory. The size of all these arrays in bytes is:

$$(Total\ no.\ of\ grid\ nodes) \cdot (size\ of\ datatype) = 2^n \left(7n^2 - n + 2\right) \cdot 8 \tag{43}$$

The second strategy is the opposite: first the interactions of all the particles are computed for one component grid and then the algorithm proceeds with the following component grid. The size of the data that must fit in the cache memory to maximize memory re-use is bounded by the size of the largest component grid:

$$(no.\ of\ component\ grid\ nodes) \cdot (size\ of\ datatype) \leq 2^{n+2} \cdot 8, \tag{44}$$

which is smaller than in the case of the first policy.

Algorithm 6 Projection or charge accumulation (PIC-Sg)

Option 1: Particles-component grids loops

for all particles **do**
  Read positions of particles in particle array
  for all component grids **do**
    Determine $\xi_1, ..., \xi_8$ the eight nodes of the cell containing the particle
    for i **from** 1 **to** 8 **do**
      Determine the charge contribution ρ_i of the particle at the node ξ_i
      Add the contribution ρ_i in the grid array at position ξ_i: grid[ξ_i]$\leftarrow \rho_i$
    end for
  end for
end for

Option 2: Component grids-particles loops

for all component grids **do**
  for all particles **do**
    Read positions of particles in particle array
    Determine $\xi_1, ..., \xi_8$ the eight nodes of the cell containing the particle
    for i **from** 1 **to** 8 **do**
      Determine the charge contribution ρ_i of the particle at the node ξ_i
      Add the contribution ρ_i in the grid array at position ξ_i: grid[ξ_i]$\leftarrow \rho_i$
    end for
  end for
end for

The number of component grid ranges from 60 to more than 120 for n ranging from 7 to 10. The accumulation of the density on these component grids is therefore expected to be the most time consuming task of the Sparse-PIC algorithm. Nonetheless these tasks are arithmetic intensive and are therefore expected to offer a good scalability.

The parallelization strategy for shared memory architectures exploits the genuine parallelism of the accumulation onto the different component grids. The component grids are arranged into groups and distributed onto the cores. Each core executes successively the tasks (accumulation, field solver operations) on the component grid assigned to it; eventually a reduction operation between the cores is performed to complete the combination step. In order to mitigate the load imbalance for runs with a number of grids that does not match the number of cores, we consider clusters of particles (*i.e* sub-samples of the particle population): the population of particle is subdivided into as many clusters as the number of cores. It results in a number of tasks, a task being the accumulation of one particle cluster onto one component grid, which is a multiple of the number of cores. Then, the tasks, *i.e.* the couples component grid-particle cluster, are distributed onto the cores. Thanks to this procedure, the work load related to the density projection is distributed onto the cores with an ideal balance irrespective to the number of cores and component grids.

The anisotropy, corollarily the number of grid nodes, takes an important part in the computational time imbalance during the resolution of Poisson equation: the convergence of iterative methods necessitates more iterations for larger systems and maximum refinement level $\|\mathbf{l}\|_\infty$. As a workaround, the grids are arranged in a decreasing order according to their complexity in three groups (grid of complexity 2^{n+2}, 2^{n+1} or 2^n); then, in each group, the grids are arranged in a decreasing order according to their maximum level of discretization $\|\mathbf{l}\|_\infty$.

Different strategies may be considered for the resolution of the Poisson equation. Within the Sparse-PIC method, a Poisson problem shall be solved on each of the component grids. The first strategy consists therefore to distribute the linear systems issued from the dicretization of the Poisson problem on the component grids onto the cores. The advantage of this strategy lies in the very small size of the linear systems. Furthermore, the systems are independent. Nonetheless, the load balance may be poor when the number of component grids is not a multiple of the number of cores.

The second strategy consists in gathering all these problems inot a single (by block) linear system. Solving this single system is a more computational expensive task, however it offers a better tuning of the load balance at the level of the linear system solver.

Let us consider now the parallelization of the hierarchization (depicted in algorithm 4) and dehierarchization. Although the procedures are similar, different strategies are considered. The former operates on the component grids $\Omega_{h_\mathbf{l}}$ whereas the latter operates on the full grid $\Omega^{(\infty)}_{h_n}$. Since the hierarchization of the component grids is independent

of each other, the parallelization is straightforward similarly to the resolution of the Poisson equation. Concerning the dehierarchization, whose algorithm resembles algorithm 4, one shall notice, when proceeding a specific dimension i, that the operations are independent for each pole (see section 2.3.5). Thus an immediate parallelization, consisting of a distribution of the poles onto the cores, may be conceived from this observation. Though, the access of the data is not optimal for all but the innermost dimension. Indeed the grid nodes within a pole are potentially in different cache lines. Different methods such as unrolled unidirectional hierarchization algorithm [31] using blocks of poles or cache-oblivious hierarchization algorithm [32] subdividing the grid into smaller subproblems that completely fit into the cache has been conceived in order to circumvent the issue; however such refinements have not proven to be mandatory to obtain a good efficiency.

3.2.2. *Parallelization for NUMA architectures.* NUMA refers to non-uniform memory architectures where the memory access time depends on the memory location of the processors. The memory resides in separate regions, named NUMA domains, and is assigned to groups of cores. A core assigned to a NUMA domain accesses data from its local memory (data stored on the memory of its NUMA domain) much faster than the non-local memory (data stored on another NUMA domain). In this section, we present a second parallelization strategy, tailored to NUMA architectures and strongly inspired of the existing SPMD implementations for the standard method. It consists in a subdivision of the particle population into samples, each associated and bound to one unique core or subset of cores. Private arrays and reduction operations are used to avoid race conditions (update of the same memory address by different cores). The efficiency of the strategy is limited by the memory architecture and bandwidth of the hardware since each core accesses simultaneously to the particle data. Consequently the number of samples of particles shall be chosen accordingly to the number of NUMA domains and distributed into the memory banks of these NUMA domains. In order to make efficient use of this strategy, data should be as much as possible accessed within a NUMA domain. Therefore, several implementation policies shall be respected ∗.

3.2.3. *Embedding the different parallelization strategies.* The different parallelization strategies, though presented independently, shall be merged to define the most effective parallel implementation for a targetted hardware. The population of particles is subdivided into samples of particles of the same size. The number of samples is intended to equal the numbers of NUMA domains, each sample being assigned to a single NUMA domain. The parallelization strategy detailed in 3.2.2 is then implemented in any of the NUMA domains: the local particle sample is decomposed into clusters and the tasks cluster-component grids are distributed on the cores of the NUMA domain. The pros of this strategy lies in the good exploitation of the increased memory bandwidth brought by the addition of NUMA domains. On the opposite, this decomposition entails the reduction operation between different NUMA domains.

4. Numerical results

The domain is a periodic cube $\Omega = (\mathbb{R}/L\mathbb{Z})^3$, of dimension L depending on the Debye length $\lambda_D = (\varepsilon_0 k_B T_e/(q_e n_0))^{\frac{1}{2}}$, with the following charge, mass and temperature for the electrons $q_e = 1.602 \times 10^{-19}$ C, $m_e = 9.109 \times 10^{-31}$ kg, $T_e = 11600$ K, and the Boltzmann constant $k_B = 1.38 \times 10^{-23}$ m^2 kg s^{-2} K^{-1}. The electrons are immersed in a uniform, immobile, background of ions ($\rho_i = Q_e/\int d\mathbf{x}$). The time discretization depends on the plasma frequency: $t, \Delta t \propto \omega_p^{-1}$, $\omega_p = (q_e n_0/m_e \varepsilon_0)^{\frac{1}{2}}$.

To assess the performance of the parallelization strategies, we consider three different hardware:

- The fist hardware is a laptop equipped with Intel® Core™ i9-10885H CPU with 8 cores @2.40 GHz sharing a L3 cache memory of 16MB. Random-Access Memory (RAM) size is 32GB. The memory is uniformly shared by all cores with two memory channels and a maximum memory bandwidth of 45.8GB/s. The cache memory is divided into a first level (L1 cache) dedicated for instructions (L1 I) and data (L1 D), both of 32KB, an intermediate level cache for instructions and data (L2 I+D) of 256KB, both three specific for each core; and a last level cache memory (L3 I+D) of 16MB shared by all the cores.

∗The following implementation policies shall be respected:

- In order to access contiguously the particle data, the dimension dedicated to the ids of the particles inside the samples must be the fast axis, *e.g.* in Fortran, the particle array must have the following form:

  $$\text{double particle\_array}[1 : N_s, \ldots];$$

  where N_s stands for the number of particles in a sample.
- A thread shall be bound to an unique core throughout the simulation so it always has its data in cache and in the same locality region (NUMA domain). It is ensured by the OMP_PROC_BIND=TRUE environment variable.
- The data must be initialized in their respective NUMA domain according to the "first touch" data placement policy.
- The cores of a NUMA domain work on the data stored into the memory local to the NUMA domain thanks to the numactl -l command.

- The second hardware is a node of the supercomputer OLYMPE from CALMIP composed of two processors Intel® Skylake 6140 @2.3 GHz with 18 cores. Each socket has a RAM of 192GB. The memory architecture is uniform for the cores of one CPU with two controllers, six memory channels and a maximum memory bandwidth of 119.21GB/s per socket (NUMA domain). The relative memory latency between the different NUMA nodes is 21 in the Advanced Configuration and Power Interface System Locality Information Table (ACPI SLIT), that is to say the latency between cores from different NUMA nodes is 2.1 times higher than the latency between cores from the same node. Within each CPU, the cache memory is divided into a level L1 cache memories of 32KB dedicated for instructions (L1 I) and data (L1 D), an intermediate level cache for instructions and data (L2 I+D) of 1MB, the two first level of cache are specific to a core; the last level cache memory (L3 I+D) of 24.75MB is shared by all the cores.
- The last hardware consists of a single computational server equipped with 128 cores in two AMD EPYC™ 7713 *Milan* CPUs and RAM memory of 512GB. Each *Milan* socket has a maximum memory bandwidth of 190.73GB/s (4 controllers, each controller has 2 memory channels). The memory architecture is non-uniform within a CPU. It consists of four NUMA domains per socket, each containing sixteen cores and 64GB of memory. The relative memory latency is at best 12 between NUMA domains in the same socket and 32 between domains from different sockets (in the ACPI SLIT). Each NUMA domain contains two Core Complexes (CCX) of eight AMD ZEN 3 cores @2.0 GHz (altogether 64 cores per socket). Within the CCX, the compute cores share a 32MB L3 cache, each core has an optimized 32KB L1 write-back cache and private 512 KB Unified (Instruction/Data) L2 cache.

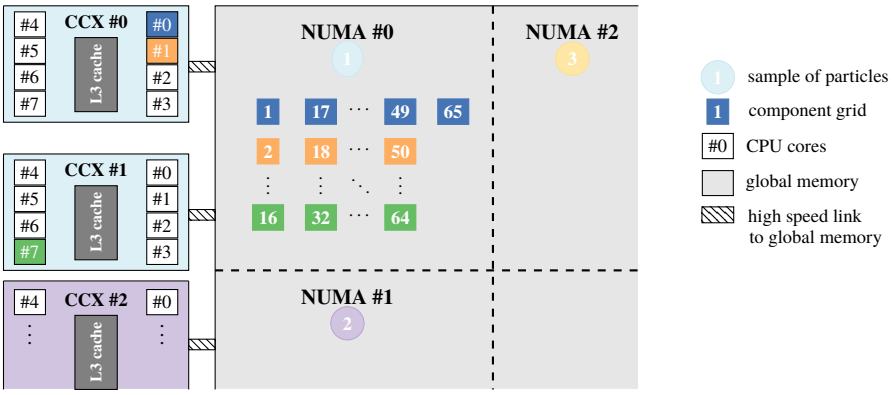

Figure 3. Embedded particle sample and component grids work sharing parallelization strategies applied to the AMD EPYC™ 7713 *Milan* architecture. The particles are subdivided into samples, as many as NUMA domains, and the 65 component grids (considered in this illustration) are distributed to the cores of the corresponding NUMA domain, e.g. the grids 1, 17, 49 and 65 are distributed to the core #0 of the CCX #0, one NUMA domain is composed of 2 Core Complex (CCX).

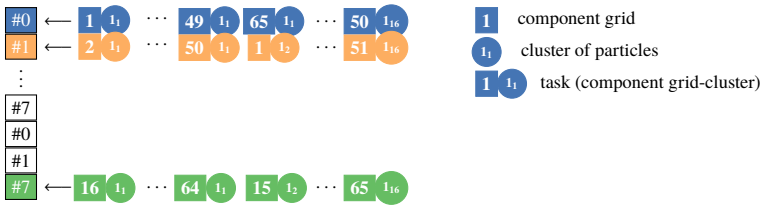

Figure 4. Load balance strategy within a NUMA domain (illustrated here for one NUMA domain within AMD EPYC™ 7713 *Milan* architecture composed with 2 CCX with 8 cores in a CCX). The particle sample associated to a NUMA domain is subdivided into as many clusters as core within the domain. The number of tasks, i.e. number of component grids*number of clusters (*e.g.* here $65 * 16 = 1040$ tasks), is a multiple of the number of cores in the domain. The work load is equally distributed onto the cores.

The compilers used for the three hardware are respectively GNU Fortran version 9.4.0, IFORT version 18.0.2 and GNU Fortran version 10.2.1 with options -fopenmp -cpp and optimizations -Ofast or -O3. Frequency boost and

hyperthreading are disabled. The code is executed with PETSc [3, 2] library for Poisson solver, using BiConjugate Gradient Stabilized (BiCGSTAB) method with Geometric Algebraic MultiGrid (GAMG) preconditioner or MUMPS [1] library with LU decomposition method. In this paper we consider two classical test cases:

- The 3D-3V non-linear Landau damping: the evolution in time of a perturbation known as the Landau damping [34] is considered. A perturbation in a maxwellian equilibrium state of the distribution is considered:

$$f_e(\mathbf{x},\mathbf{v}) = \frac{1}{2\pi}\prod_{i=1}^{3}\left(1+\alpha_i\cos\left(\frac{\beta_i 2\pi x}{L}\right)\right)e^{\frac{-\|\mathbf{v}\|_2^2}{2}}, \tag{45}$$

  where $\|\mathbf{v}\|_2^2 = v_1^2+v_2^2+v_3^2$, α_i is the magnitude and β_i is the period of the perturbation in the i^{th} dimension. Let $\alpha_i = 0.15$, $\beta_i = 3$, $i = 1,2,3$ in equation (45), let $L = 160\lambda_D$, $\Delta t = \frac{1}{20}\omega_p^{-1}$. The system is observed at time $T = 6\omega_p^{-1}$.
- The 3D-3V diocotron instability: a hollow profile is considered in the electron distribution, confined by a magnetic field $\mathbf{B}$ [37], with the following Maxwellian distribution of electrons :

$$f_e(\mathbf{x},\mathbf{v}) = \frac{\gamma e^{-\frac{\left(\left\|\mathbf{x}-\frac{L}{2}\right\|_2-\frac{L}{4}\right)^2}{2(0.03L)^2}}}{0.03L(2\pi)^2}e^{\frac{-\|\mathbf{v}\|_2^2}{2}}, \quad \gamma\ s.t.\ \iint_{\Omega\times\mathbb{R}^3} f(\mathbf{x},\mathbf{v})d\mathbf{x}d\mathbf{v} = 1 \tag{46}$$

  where $\left\|\mathbf{x}-\frac{L}{2}\right\|_2^2 = (x-\frac{L}{2})^2+(y-\frac{L}{2})^2+(z-\frac{L}{2})^2$. The external magnetic field is considered linear along the z-axis $\mathbf{B}(z) = (0,0,B_z+4z\times10^{-6})$ ($B_z = 2.2\times10^{-5}$ T) and strong enough so that the electron dynamics is dominated by advection in the self-consistent field $\mathbf{E}\times\mathbf{B}$. Let the parameters be $L = 22\lambda_D$, $\Delta t = 0.1\omega_p^{-1}$, the system is observed at time $T = 80\omega_p^{-1}$.

Throughout this section the different methods presented previously, namely the sparse grid schemes with hierarchical (PIC-HSg) or nodal combination (PIC-NSg), with first (PIC-HSg1) or second option (PIC-HSg2) (see algorithm 6) and the standard scheme (PIC-Std), are investigated and compared. In order to provide a fair comparison between the sparse grid methods and the standard methods, we introduce a naive implementation of the method (without sorting of particles) and an implementation based on sorted particle data beforehand. These implementations provide lower and upper bounds of the method efficiency. The numerical results upon the computational time consist of a mean of the first five iterations of the scheme at the end of which only 4% of the particles have moved to a different cell of the grid containing 128^3 cells with 75 particles per cell in the standard case. In the following, the scheme is divided into six steps, namely the projection of the density (charge accumulation) onto the grid (Proj), the resolution of the Poisson equation (Pois), the differentiation of the electric potential (Diff), the pusher of particles (Push), the interpolation of the electric field (F.Inter), the combination of the component grid contributions in either nodal or hierarchical basis (Comb). An additional step is performed within the scheme for the review of the numerical results; the charge density is reconstructed from all the component grid contributions and represented on the full grid.

4.1. **Sequential performance.** The performance of the different schemes carried out with one core (One AMD ZEN 3 core) is investigated. The reconstructions provided by the nodal basis and hierarchical basis are proved to be strictly similar as foreseen by the proposition 2.2. Therefore, only the hierarchical reconstruction will be provided in the sequel.

First, the results of the computational time of the Poisson solver for each scheme with different methods (direct, iterative) and configuration of the linear system (one large system or many small systems, see section 3.2.1) are represented on figure 5. The benefit resulting from the reduction of the grid nodes is manifest on the computational time of the linear system resolution. Indeed, dividing the mesh size by a factor 2, doubles the time of execution of the sparse grid scheme with LU decomposition whereas it is multiplied by ten for the standard scheme. It results in a significant difference between the two approaches, especially for fine spatial discretizations. The difference being of three order of magnitude for 256^3 grids. This efficiency deficit is predicted to deepen to four orders of magnitude for 1024^3 grids: the resolution lasts more than one hour for the standard method whereas the sparse grid resolution is done in a tenth of a second. The conclusion is that for the Sparse-PIC methods the most efficient configurations are the LU decomposition performed either on a large system or independently on each system.

Also, the computational time of one time iteration for the PIC-HSg and PIC-NSg schemes is represented on figure 5 for comparable configurations of grid and number of particles per cell, according to formulae (14), (31). The observation upon the complexity of the combination in nodal and hierarchical basis presented in section 2.3.6 is verified here: the computational load of the combination has been drastically mitigated (divided by 122 in the first and 240 in the second configuration) by the transformation into the hierarchical basis (from 12.5% to 0.11% and 91% to 4% of time of one iteration computational time). In the second test case, the combination in nodal basis takes a substantial amount of time because there are more grid nodes (in the Cartesian grid) than particles (about 12 times more grid nodes). As a result, the more efficient scheme on one core is the sparse grid scheme with the combination in hierarchical basis; and

the combination in nodal basis is put aside of considerations in the following. As predicted in the previous sections, the projection is by far the most costly operation within the sparse grid scheme (between 80% and 95 % of the time of the iteration) because of the large number of component grids involved in the charge deposition. The others steps are less time consuming than in the standard method because of the reduced number of particles and the smaller size of the linear systems. Theses observations confirm that our efforts should be concentrated into the optimization and parallelization of the projection.

The projection step is investigated in more details for the sparse grid schemes with both options of algorithm 6 on the one hand, and the standard scheme with sorted and unsorted particles on the other hand. The computational time of the step as a function of the grid discretization is represented on the panel a) of figure 6. The percentage of L1 data cache misses relative to the total number of data accesses during the projection is represented on the panel b) of the figure 6. An additional feature representing the storage requirements of the density array related to the memory size available on the different platforms is provided on the panel c) of figure 6. In order to extend the conclusions drawn in this section to the parallel implementation, in which each thread holds a copy of the (grid) to avoid the concurrent write accesses between the threads (see section 3.1), the storage requirements of these copies are also represented when the data of each thread exceed the L2 cache memory (limitation between private and shared memory of the threads). As a result of better memory reuse, the second option (see algorithm 6) is twice as fast as the first one for any grid discretization. The data of the second fit in the first level cache memory whereas the second one only fit in the second or last level cache memory. The number of cache miss during the projection step is a lot more important with the first option than the second one (especially for fine discretizations). In the latter case, the number of cache miss per memory access is negligible and corresponds to the ratio of the standard scheme with the particle sorting, in which the data are accessed contiguously. This result demonstrates that within our approach the cache memory management is close to optimal for grids up to 1024^3 cells.

In order to estimate the statistical noise in the simulation, we consider a frozen number of particles per cell and we examine the error of the density in L^2-norm at initial time. Indeed, it is possible to assess precisely the error of the density projected onto the grid for the different methods by comparison with the analytic expression of the initial electron density. For grids with more than 64^3 cells, the grid-based error is negligible compared to the particle sampling error (statistical noise), therefore the L^2-norm of the density error gives us an estimation of the square root variance of the statistical error $\mathbb{V}(\mathcal{V}_{N,h_1})^{\frac{1}{2}}$ defined by equation (12). The results of the electron density error are represented on the panel b) of figure 7. The results reveal that the relations given by equations (14), (31) does not provide an equivalent amount of statistical noise for both the standard and sparse grid schemes. It is indeed manifest on the figure 8 that, for equivalent configurations, the statistical noise is drastically reduced in the Sparse-PIC computations. Actually, the statistical error of the density is even slightly superior for the standard scheme with $P_c = 500$ (resp. $P_c = 100$) to that of the sparse grid scheme with $P_c = 63$ (resp. $P_c = 14$). The consequence of it being that, for an equivalent amount of statistical noise in the simulation, the number of particles in the sparse grid scheme can be drastically reduced (e.g. with a 256^3 grid and 100 particles per cell it can be reduced from 1.6×10^9 particles for the standard scheme to 9.9×10^5 particles for the sparse grid scheme, that is to say 1600 less particles). A section in z-direction of the electron density deposited after two oscillations of the damping, at $t = 6\omega_p^{-1}$ (60 time steps), on a 128^3 grid for the standard (panel a), sparse grid (panel b) schemes with $P_c = 500$ and for the sparse grid scheme (panel c) with $P_c = 63$ is represented on figure 9. The mitigation of the statistical noise is rather conspicuous for the sparse grid scheme and provides a sharper representation of the density. Indeed the error seems slightly inferior for the sparse grid scheme with $P_c = 63$ than the standard scheme with $P_c = 500$.

Table 1. Configurations and results of Landau damping test case; the standard PIC method with particles sorting (first line) is considered as a reference.

Method	Figure	Nb. of particles	Particle memory footprint	Grid size	Computational time
Reference → PIC-SStd	9 a)	1.05×10^9	75 GB	128^3 ($P_c = 500$)	100% (306.4s)
PIC-Sg	9 b)	1.3×10^7 ($\div 80$)	936 MB	128^3 ($P_c = 500$)	3% (9.9s)
PIC-Sg	9 c)	1.7×10^6 ($\div 617$)	122 MB	128^3 ($P_c = 63$)	0.4% (1.2s)
PIC-Sg	9 d)	1.05×10^8	6.5 GB	512^3 ($P_c = 500$)	\
PIC-Std	\	9.01×10^7	7.5 GB	128^3 ($P_c = 50$)	4% (12s)

The computational time of one time iteration is compared between the standard and sparse grid schemes. The execution time of an iteration, as well as the proportion of each step is represented on the left panel of figure 7. The benefit of the sorting of particles is manifest on the grid/particles operations (i.e. the projection and the field interpolation) for the standard scheme and results in an acceleration of 1.4 upon the time iteration compared to the scheme without sorting. This is the result of a better cache memory reuse achieved by the sorting. Nonetheless this

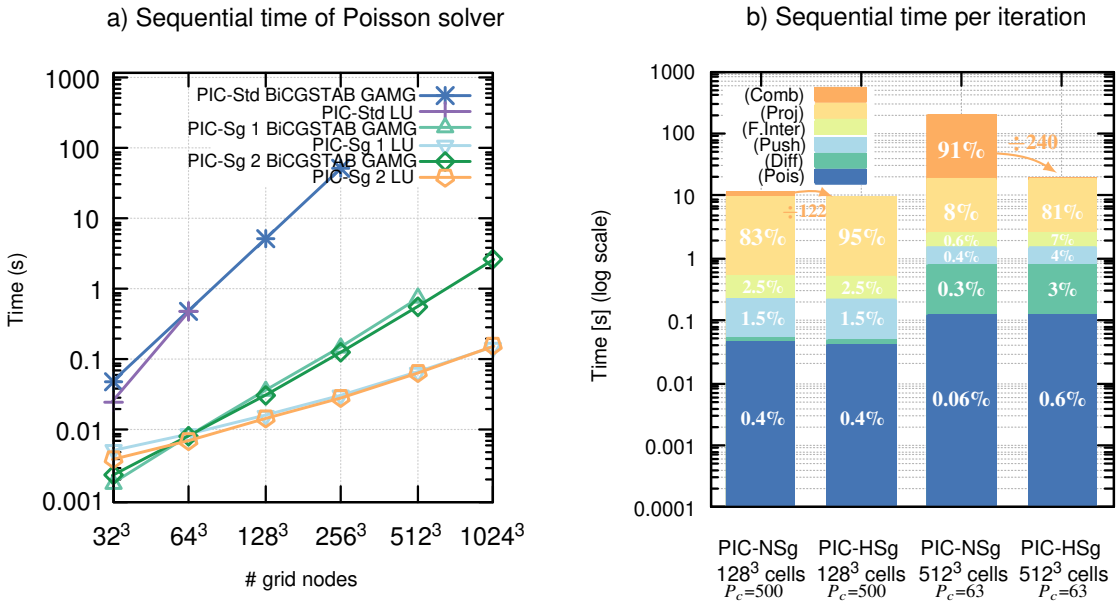

Figure 5. On panel a) the sequential time of Poisson solver: Direct (LU) and iterative BiConjugate Gradient STABilized (BiCGSTAB) with Geometric Algebric MultiGrid (GAMG) preconditioner methods. PIC-Sg 1: The resolution of all sub-grids has been gathered in one large linear system. PIC-Sg 2: The linear systems issued from the component grids are solved independently one after another. On panel b) the sequential time of one time iteration (in logarithmic scale) for the PIC-NSg and PIC-Hsg schemes. The use of the hierarchical basis shortens the combination step computational time by hundreds. One AMD ZEN 3 core @2GHz.

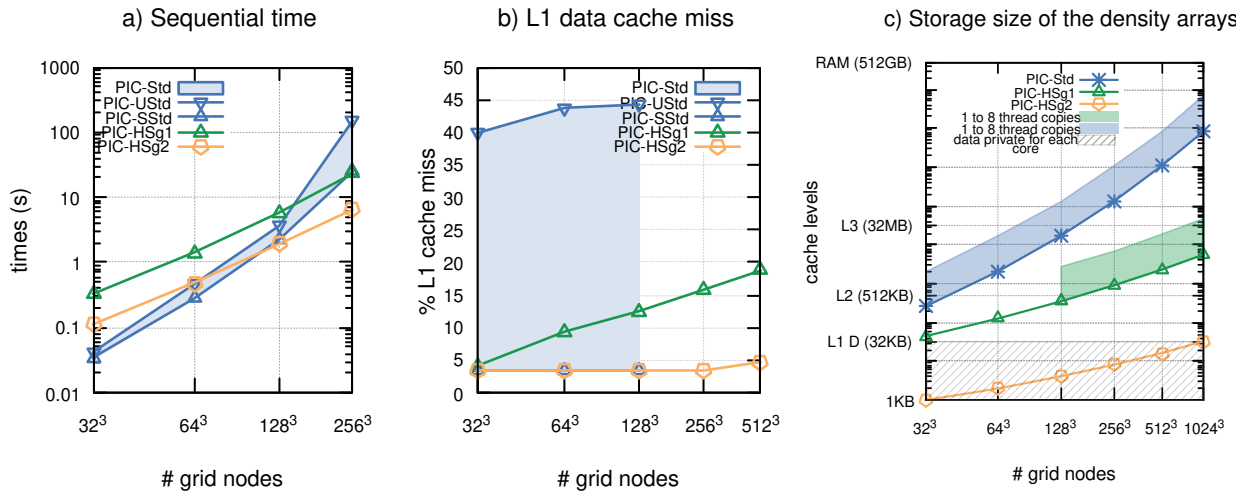

Figure 6. The sequential time (panel a), ratio of L1 data cache miss per total data access (panel b) and storage of grid data (panel c) for the projection step with $P_c = 100$ are represented. AMD ZEN 3 core @1.7GHz.

optimization concerns only the grid-particles operations, therefore it has no effects on the particle pusher which is the most costly operation. One of the reason of the large computational time of the standard schemes is the substantial amount of data to handle. Indeed, such a simulation with a 128^3 grid and 500 particles per cell results in more than 70GB of data to store which exceed the memory limitation of one NUMA domain and thus the core that executes the code has to access to remote memory (from other NUMA domains). As a comparison, a simulation with a 128^3 grid but only 50 particles per cell, so that the data entirely fit in the memory of a single NUMA domain, only requires 12s (with sorted particle data), which is about 20 times less than the simulation with ten times more particles.

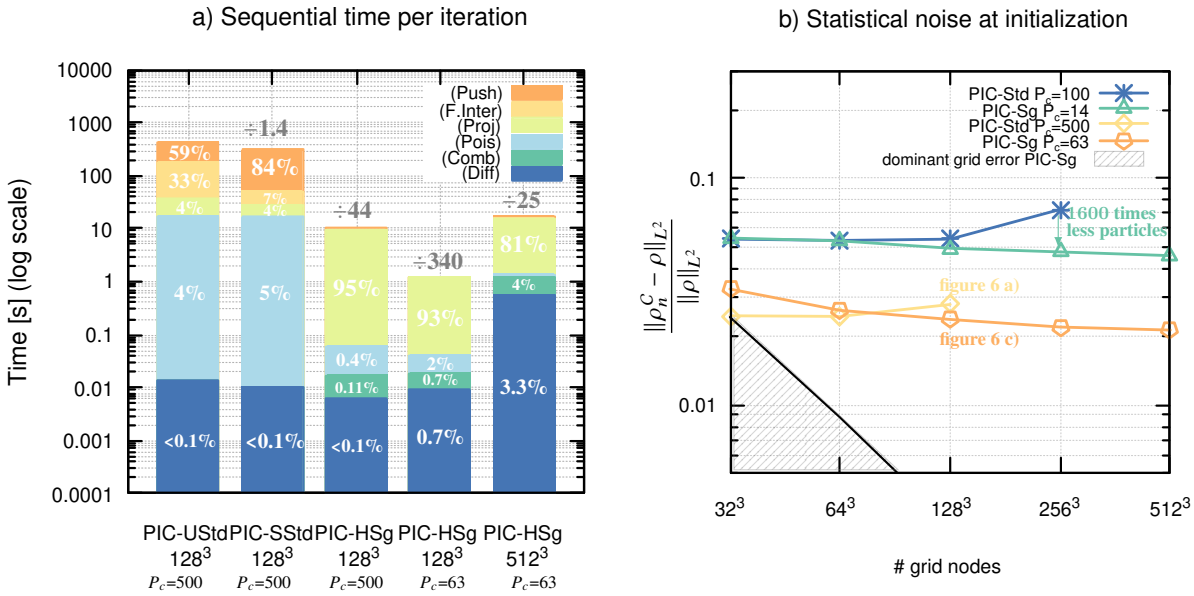

Figure 7. Computational time on 1 AMD ZEN 3 core @2GHz of one iteration for the Landau damping (panel a) and statistical noise at initialization (panel b). The statistical noise is roughly equivalent for PIC-Std scheme with $P_c = 500$ (figure 9 a) (resp. $P_c = 100$) and PIC-Sg with $P_c = 63$ (figure 9 c) (resp. $P_c = 14$).

The gain provided by the sparse grid technique on the computational time is outstanding. The execution time of the sparse grid scheme, with a statistical noise equivalent to the standard scheme, is reduced by more than 300 times. The substantial gain is due to the reduction of the number of particles in the sparse schemes (about 600 less particles). The computational time of particles operations, chiefly the field interpolation and the particle pusher, is negligible. In addition, the sparse grid schemes offer a significant reduction of the grid operations (roughly 100 less grid nodes for the sparse grid scheme). As a comparison, a simulation with a 512^3 grid and $P_c = 63$, necessitating 25 times less computational time than the standard one with a 128^3 grid, is performed. Apart from the utter impossibility of simulations in such configuration for standard schemes, because of memory limitations (it would require 5TB of particle data as illustrated by the left panel of figure 11), the gap in term of complexity and computational time between the standard and sparse grid schemes deepens as the grid is refined which is a promising feature for very demanding simulations.

However, as put forward in [17], sparse grid reconstructions may fail to reproduce solutions with localized support and steep gradients. As an illustration of this feature, we investigate the three-dimensional diocotron test case in which instabilities caused by the magnetic field lead to the formation of a discrete number of vortices exhibiting the weaknesses of the method. The three dimensional representation of the electron density, as well as a section in z-direction at the middle of the domain, is represented on figure 10 at $t = 80\omega_p^{-1}$, with a 128^3 grid and $P_c = 30$ for the standard scheme (panel a), with a 256^3 grid and $P_c = 5$ for the sparse grid scheme without (panel b) and with the offset combination technique (for the parameters $(\tau_0, \tau_1) = (3, 6)$) (panel c). The offset combination technique, introduced in [17], is an alternative to the classical combination technique presented in section 2.3.2, consisting in an elimination of the most anisotropic grids from the combination. Though the grid is more refined, the sparse grid scheme with the classical combination technique fails to reproduce the fine structure of the density. One can see that the sparse grid reconstruction has numerical diffusion and a tendency to flatten the steep gradients of the solution; yet, a finer reproduction of the instability could be obtained with a refinement of the grid. Nonetheless the offset combination technique provides a significant improvement of the sparse grid reconstructions and a mitigation of the statistical noise in comparison to the standard approach, though with a significantly reduced number of particles.

4.2. **Parallelization on uniform memory architecture.** Let us now consider the parallelization on uniform memory architecture: the first hardware considered in this paper: the Intel® Core™ i9-10885H CPU with eight cores. In this section, the different parallelization strategies, namely particle sample and component grid work sharing, are investigated separately and compared between them. In this section the resolution of the electric potential is not parallelized. It is assumed to be negligible (see figure 5 a). In this section, we consider few particles per cell in the

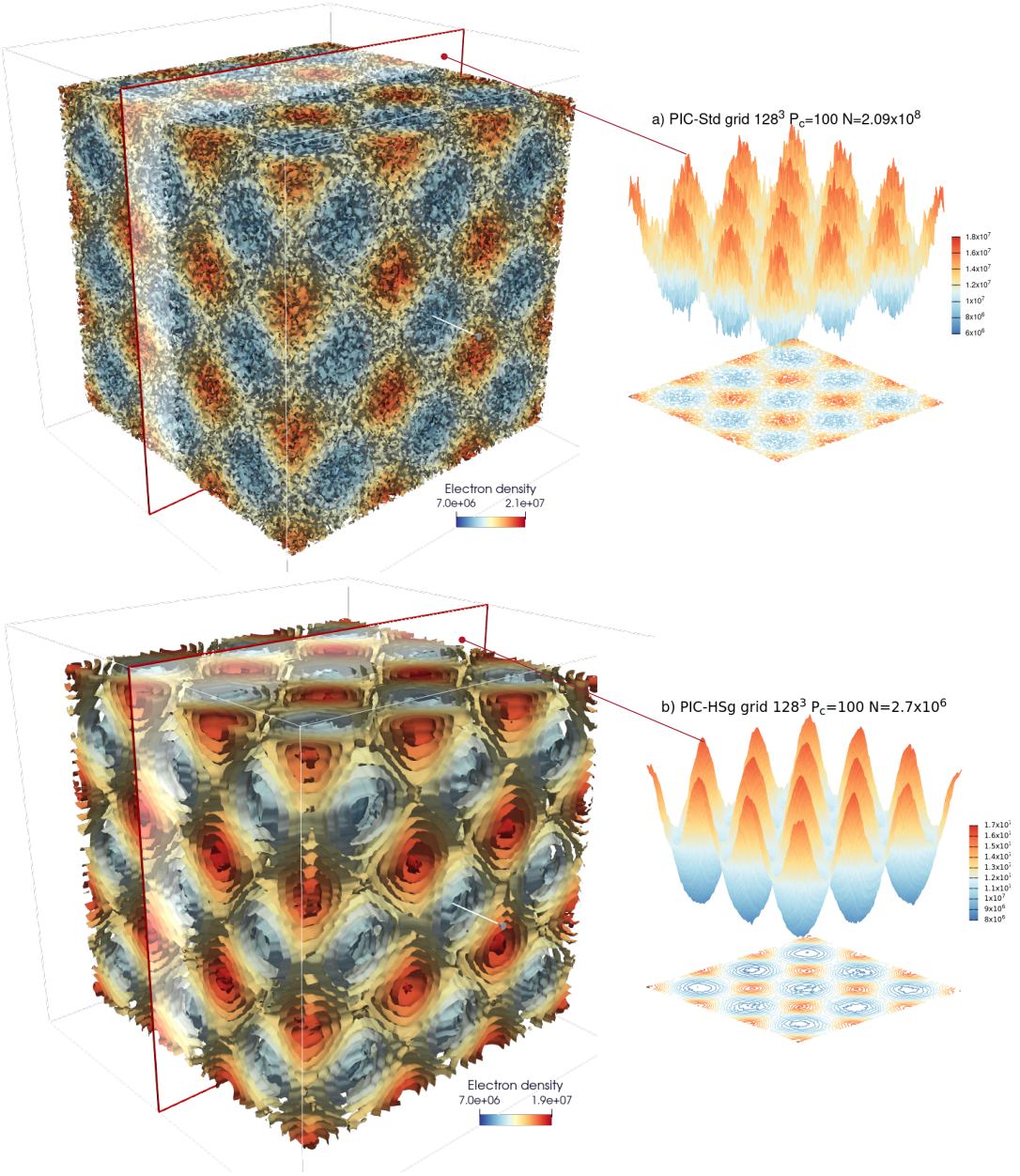

Figure 8. Representation of the electron density for the 3D-3V Landau damping simulation. Three dimensional representation and section in z-dimension ($\mathbf{x} = (x_1, x_2, \frac{L}{4})$) after two oscillations ($t = 6\omega_p^{-1}$, 60 time steps). The simulations have been performed with comparable configurations (grid with 128^3 cells and $P_c = 100$ particles per cell).

simulation and thus a poor statistical resolution, especially for the standard scheme, as a consequence of hardware memory limitation. The memory requirements of the methods are illustrated on panel a) of figure 11.

Let us focus on the parallelization strategies for both standard and sparse grid schemes. The strong scaling of the projection, field interpolation and particle pusher up to eight cores is represented on the panels b) and c) of figure 11 for different configurations of grid ranging from 32^3 to 512^3 grid cells. For the standard scheme, the scalability of the projection seems rather good for grids with less than a hundred nodes in each dimension, for which it drastically deteriorates for the non-sorted data. This is caused by the size of all the thread copies of the density array exceeding the size of the last level (L3) cache memory (see panel c) of figure 6). Indeed, since the data array is accessed with a random pattern and the data does not fit in the cache, the number of cache misses is large and the multiplication of the cores increases the memory contention to access the data stored in the RAM. Conversely, the sparse grid scheme (with option 2 from alg. 6) demonstrates a scalability close to ideal. The grids involved in the projection fit in the first

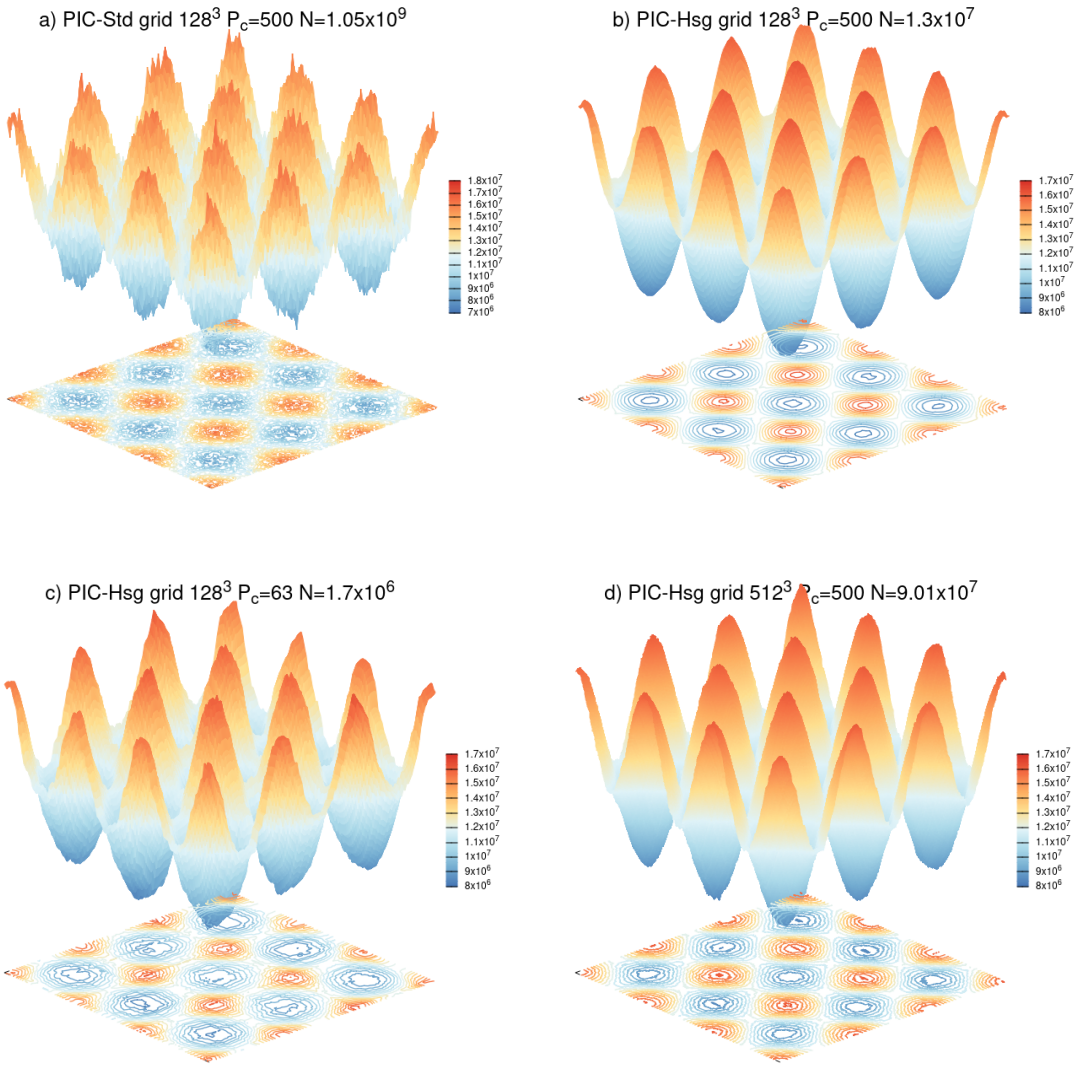

Figure 9. Representation of the electron density for the 3D-3V Landau damping simulation. Section in z-dimension ($\mathbf{x} = (x_1, x_2, \frac{L}{4})$) after two oscillations ($t = 6\omega_p^{-1}$). Figures a) and c) have the same statistical error but figure c) has more than 600 less particles than figure a). Figure a) requires 306.4 s per iteration (with sorted data), figure b) 9.9s per iteration and figure c) 1.2s per iteration, both on one AMD ZEN 3 core @2GHz. Figure d) requires 31.8 s on eight cores Intel® Core™ i9-10885H.

level cache memory (private to each core). Actually the size of the data barely exceeds the first level cache memory for discretization equivalent to 1024^3 grid cells (see panel c) of figure 6). The choice of the first option in alg. 6 reveals once more to bet he most effective since within the second option the grid data do not fit in the shared L2 cache memory for discretizations above 128^3 grid nodes. The scalability of the other steps, namely the pusher, combination and field interpolation is less favorable, similar to the standard scheme, but these operations are negligible for the sparse grid scheme due to the reduced number of particles and grid nodes. The loss of scalability for the combination step derived from the access of non-contiguous data within a pole (as explained in section 3.2.1) and small number of poles to parallelize (roughly 16 000). For the pusher of the particles the poor scalability is due to low arithmetic intensity characterizing these operations (nine writes and eighteen reads, as a comparison the projection step has only three reads upon the particle data).

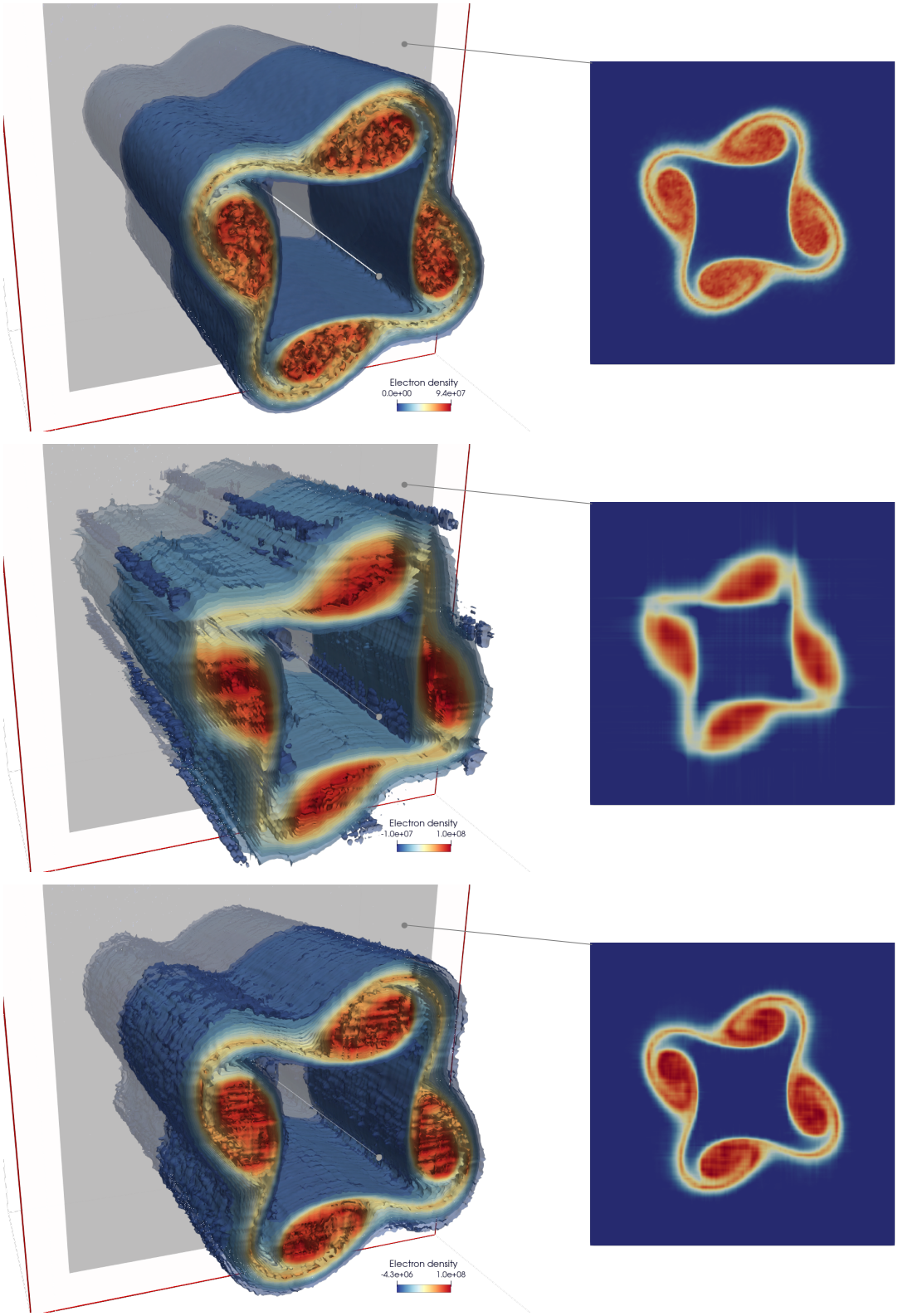

Figure 10. Representation of the electron density for the 3D-3V diocotron instability simulation at $t = 80\omega_p^{-1}$. A section in z-dimension ($\mathbf{x} = (x_1, x_2, \frac{L}{2})$) is also represented. The offset combination technique [17] is used on figure c) with parameters $(\tau_0, \tau_1) = (3, 6)$. Figure a) requires 10s (4.5 GB, $N = 6.2 \times 10^7$) per iteration, figure b) 0.49s per iteration (25MB, $N = 3.5 \times 10^5$) and figure c) 5.1s per iteration (619MB, $N = 8.6 \times 10^6$), both on one AMD ZEN 3 core @2GHz.

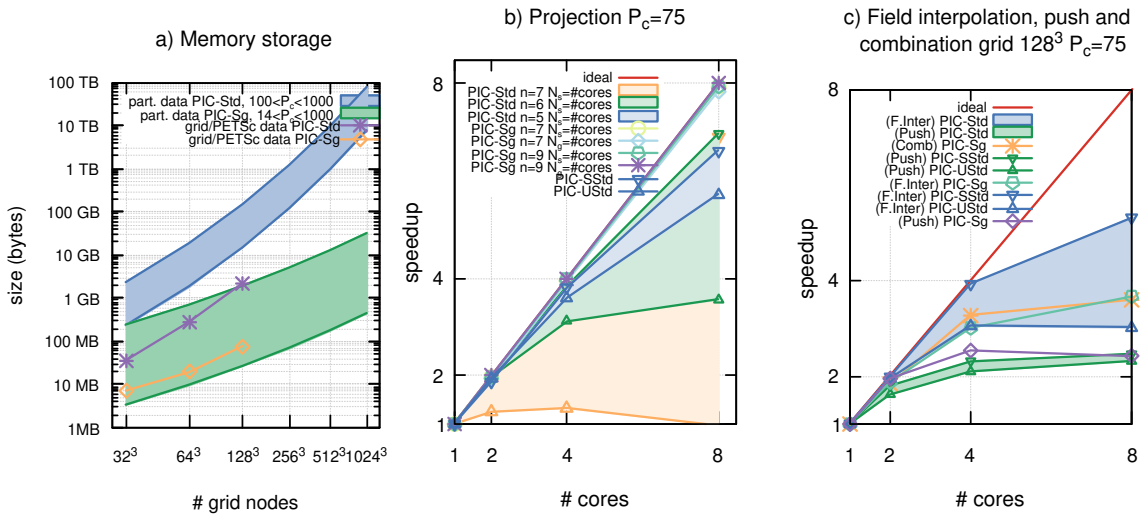

Figure 11. On panel a), the storage size of data (particle data, grid data, PETSc objects, etc.) is represented as a function of the number of grid nodes and particles per cell. On panel b) and c), the strong scaling of the projection, field interpolation and push steps up to 8 cores on a uniform memory architecture (Intel® Core™ i9-10885H) are represented.

4.3. **Parallelization on NUMA architecture.** In this section, the different parallelization strategies, namely the particle sample and component grid work sharing, are investigated for the sparse grid scheme. Based on the hardware architecture and the reflections of section 3, the population of particles is distributed onto the NUMA domains and the component grids onto the cores of a NUMA domain. The set of component grids is replicated to match the number of particle samples and accumulate the density carried by the sample. In the following, we denote by N_g the number of groups of component grids and N_s the number of samples of particles. A series of simulations is performed in order to assess the strong scaling (when adding cores to a fixed discretization) of a time iteration.

Let us now consider a first non-uniform memory architecture: the two sockets Intel® Skylake 6140 @2.3 GHz of 18 cores (36 cores). The numerical results for the choice parallelization strategy on the the projection, as well as other configurations and the other most costly operations (field interpolation, particle pusher, differentiation) up to 36 cores for grids with 128^3 and 256^3 cells, 500 particles per cell are presented on panel b) of figure 12. The parallelization strategies providing the best efficiency on the projection are the ones that do not stress the memory bandwidth: configurations with two samples of particles (*i.e.* as many as NUMA domains) and the one without any sampling of particles. Indeed, when the number of samples of particles exceeds the number of NUMA domain, the threads access the memory in a competing way and the scalability is deteriorated. Although the scalability of the projection is close to ideal on 36 cores, the density projection remains by far the most costly operation within the scheme. As a result, the limited scalability of other steps such as the pusher, the differentiation or the sequential execution of the Poisson solvers has a marginal impact on the total speedup (of 27.3).

Then, a second non-uniform memory architecture is considered: the two sockets AMD EPYC™ 7713 *Milan* with 64 cores per socket (128 cores). With this increased number of cores, the sequential Poisson solver is no longer negligible. Therefore, the parallelization strategy based on the component grid work sharing is implemented. The linear systems issued from the different component grids are distributed onto the cores. Though this strategy is not optimal, since it entails a load imbalance, it permits to reduce the cost of this step of this algorithm and obtain a good scalability of the Sparse-PIC method for tens of cores. A series of simulations is performed in order to assess the strong scaling of the projection, which is by far the most costly operation of the sparse grid scheme as evidenced by the computational time of an iteration on one core (see panel a) of figure 13). On this panel, the computational time of the PIC-HSg scheme with one core and 128 cores, as well as the PIC-SStd scheme with 128 cores are represented. The projection remains the most costly step (nearly 80 % of the time iteration) of the PIC-HSg scheme on one hundred of cores. Nonetheless, for the PIC-Std scheme the resolution of the Poisson equation is sequential and counts for 70 % of the time iteration. Therefore the comparison between the two methods with respect to the computational time of one complete iteration is not fair in this configuration.

The scalability of the most costly steps within the Sparse-PIC method and the PIC-Std projection up to 128 cores for grids with 128^3 and 256^3 cells, 500 particles per cell are presented on panel b) of figure 12. The configuration that

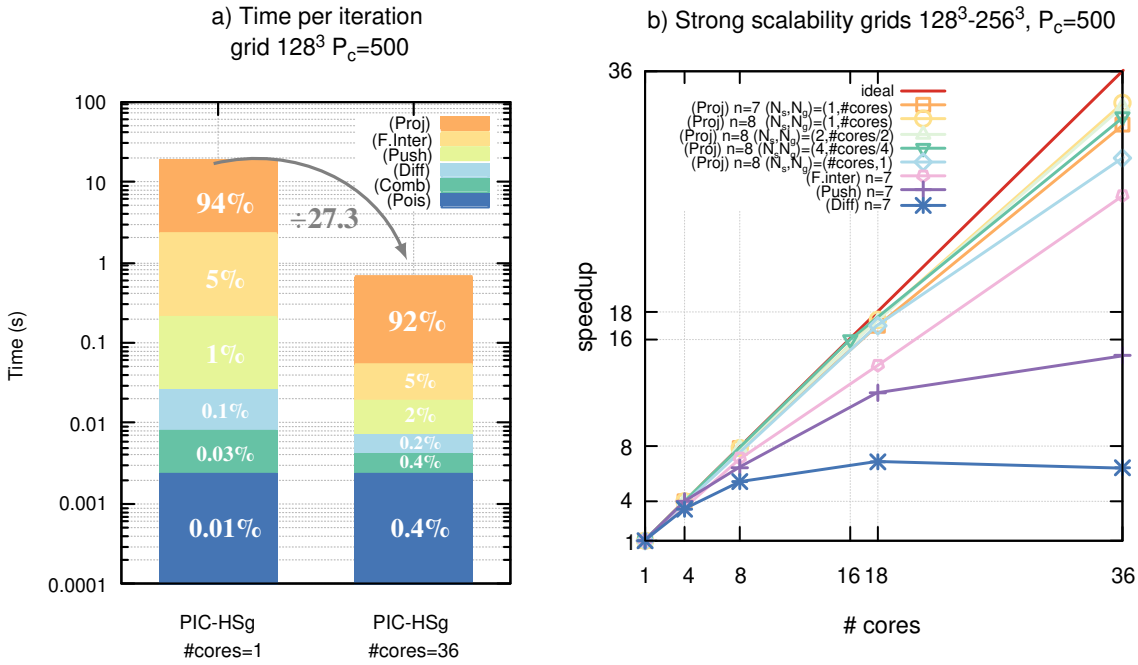

Figure 12. On panel a) the computational time per time iteration for the PIC-HSg scheme on one and 36 cores is represented. On panel b) the strong scaling of the projection step up to 36 cores is represented. Different configurations of groups of grids and samples of particles are represented(#cores denotes the number of cores). For the other steps than the projection, the configuration (N_s, N_g) is not given since all provide similar results. Two Intel® Skylake 6140 CPUs.

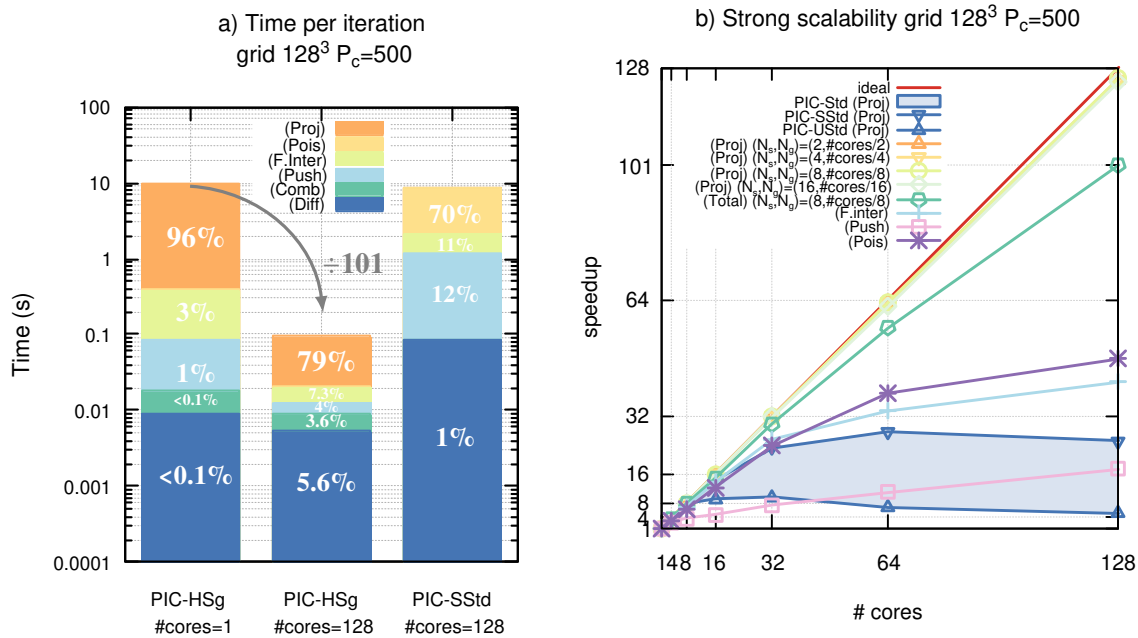

Figure 13. On panel a) the computational time per time iteration for the PIC-HSg and PIC-SStd schemes on one and 128 cores is represented. The total iteration has a speedup of 101. On panel b) the strong scaling of the most costly steps of the PIC-HSg scheme and the projection of the PIC-Std scheme up to 128 cores is represented. Different configurations of groups of grids and samples of particles are represented(#cores denotes the number of cores). For the other steps than the projection, the configuration (N_s, N_g) is not given since all provide similar results. Two AMD EPYC™ 7713 CPUs.

gives the best results and achieves a speedup of 126 is the one with eight samples of particles (as many as NUMA domains). For different configurations of samples, the speedup is still close to ideal, which lead to the conclusion that the parallelization strategy has not reach its scalability limit with one hundred of cores. The total speedup of one time iteration is 101 with 128 cores. For the standard scheme, the scalability of the projection above 32 cores is poor; the speedup ranges between 10 without sorting and 28 with sorted particles. The substantial difference between the standard and the Sparse-PIC schemes can be explained from two observations: first, as already highlighted, the particle sample work sharing parallelization strategy on its own may lead to increase of memory contention for a large number of cores sharing the same sample; second, the sparse grid reconstructions offer a more favorable trade-off between memory accesses and compute operations, resulting in an increased arithmetic intensity. Indeed the charge of one particle is deposited onto each component grid, the array used to store these grids being nursed in the first level of the cache memory, the contention of the RAM is alleviated and the scalability of Sparse-PIC algorithms is favored.

The same conclusions than in the uniform memory architecture case can be drawn for the pusher and field interpolation steps. The scalability of the pusher is limited by the number of memory channels, and thus increases with the number of NUMA domains used. The scalability of the resolution of Poisson is deteriorated by the load imbalance between the grids (complexity, anisotropy).

5. Conclusions

In this paper, we have proposed parallelization strategies tailored to Sparse-PIC methods on shared memory architectures. The better control of the statistical error offered by Sparse-PIC reconstructions permits a substantial reduction of the number of particles (up to 3 orders of magnitude). Conversely, the interactions of one particle (restricted to the density projection for the algorithm implementing the hierarchization), shall be computed for tens to more than one hundred component grids containing a very small number of nodes. Compared to standard PIC methods these features bring significant changes in the ranking of the most consuming steps of the algorithm. The Poisson problem is discretized on each of the component grid issuing linear system of tiny sizes which dramatically reduces the memory footprint as well as the computational effort attached to the electric potential approximation. The reduced number of particles, in addition to reduce massively the memory consumption, lowers the computational cost of the particle pusher to a marginal contribution. In the end, the very majority of the computations are dedicated to the projection of the particle properties onto the component grids. These operations are arithmetic intensive. The strategy proposed within this paper takes profits of the particularly reduced footprint of the arrays storing each of the component grids. These small arrays are entirely nursed in the first cache level of memory, providing fast access to non contiguous addresses, thus removing the mandatory particle sorting used in standard PIC methods. The genuine parallelism of sparse grid reconstructions permits an implementation with a scalability close to optimal (the efficiency reaches 126/128 on 128 cores) providing speed-ups of the global algorithms up to 100 on 128 cores.

From the numerical investigations carried out in this paper, it is manifest that the mean number of particle per cell (P_c) does not furnish an accurate estimator of the noise reduction. The reduction of the number of particles within Sparse-PIC methods may be significantly strenghen to obtain a comparable numerical noise to that of standard PIC methods. Therefore, the mathematical analysis of the statistical error within the sparse grid scheme and the number of particles required to guarantee an equivalent noise in both schemes shall be part of further works.

Acknowledgements

Clément Guillet benefits from a Université de Toulouse/Région Occitanie PhD grant.
This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No 101052200 — EUROfusion). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.
This work has been supported by a public grant from the "Laboratoire d'Excellence Centre International de Mathématiques et d'Informatique" (LabEx CIMI) overseen by the French National Research Agency (ANR) as part of the "Investissements d'Avenir" program (reference ANR-11-LABX-0040) in the frame of the PROMETEUS project (PRospect of nOvel nuMerical modEls for elecTric propulsion and low tEmperatUre plaSmas).
Support from the FrFCM (Fédération de recherche pour la Fusion par Confinement Magnétique) in the frame of the SPARCLE project (SParse grid Acceleration for the paRticle-in-CelL mEthod) is also acknowledged.
Jacek Narski was supported by the ANR project MUFFIN (ANR-19-CE46-0004).
This work was granted access to the HPC resources of CALMIP supercomputing center under the allocation 2022-2022-1125.
Clément Guillet is grateful to Pierre Jolivet for his advice and fruitful discussions.

REFERENCES

[1] P.R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. SIAM Journal on Matrix Analysis and Applications, 23(1):15–41, 2001.

[2] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil M. Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. PETSc Web page. https://petsc.org/, 2022.

[3] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, Modern Software Tools in Scientific Computing, pages 163–202. Birkhäuser Press, 1997.

[4] Yann Barsamian, Arthur Charguéraud, Sever A. Hirstoaga, and Michel Mehrenberger. Efficient Strict-Binning Particle-in-Cell Algorithm for Multi-core SIMD Processors. In Marco Aldinucci, Luca Padovani, and Massimo Torquati, editors, Euro-Par 2018: Parallel Processing, volume 11014, pages 749–763. Springer International Publishing, Cham, 2018. Series Title: Lecture Notes in Computer Science.

[5] Yann Barsamian, Sever A. Hirstoaga, and Eric Violard. Efficient Data Structures for a Hybrid Parallel and Vectorized Particle-in-Cell Code. In 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pages 1168–1177, Lake Buena Vista, FL, May 2017. IEEE.

[6] Yann Barsamian, Sever A. Hirstoaga, and Eric Violard. Efficient data layouts for a three-dimensional electrostatic Particle-in-Cell code. Journal of Computational Science, 27:345–356, July 2018.

[7] A. Beck, J. Derouillat, M. Lobet, A. Farjallah, F. Massimo, I. Zemzemi, F. Perez, T. Vinci, and M. Grech. Adaptive SIMD optimizations in particle-in-cell codes with fine-grain particle sorting. Computer Physics Communications, 244:246–263, November 2019.

[8] Charles K Birdsall and Dieter Fuss. Clouds-in-clouds, clouds-in-cells physics for many-body plasma simulation. Journal of Computational Physics, 3(4):494–511, April 1969.

[9] C.K. Birdsall and A.B Langdon. Plasma Physics via Computer Simulation. CRC Press, 0 edition, October 1985.

[10] H.-J. Bungartz, M. Griebel, D. Röschke, and C. Zenger. Pointwise Convergence Of The Combination Technique For Laplace's Equation. East-West J. Numer. Math, 2:21–45, 1994.

[11] Hans-Joachim Bungartz. Finite elements of higher order on sparse grids. Berichte aus der Informatik. Shaker, Aachen, als ms. gedr edition, 1998.

[12] Hans-Joachim Bungartz and Stefan Dirnstorfer. Higher Order Quadrature on Sparse Grids. In Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Marian Bubak, Geert Dick van Albada, Peter M. A. Sloot, and Jack Dongarra, editors, Computational Science - ICCS 2004, volume 3039, pages 394–401. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. Series Title: Lecture Notes in Computer Science.

[13] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. Acta Numerica, 13:147–269, May 2004.

[14] G.-H. Cottet and P.-A. Raviart. Particle methods for the one-dimensional vlasov-poisson equations. SIAM J. Numer. Anal., 21(1):52–76, February 1984.

[15] G. H. Cottet and P. A. Raviart. On particle-in-cell methods for the vlasov-poisson equations. Transport Theory and Statistical Physics, 15(1-2):1–31, February 1986.

[16] Pierre Degond, Fabrice Deluzet, and David Doyen. Asymptotic-preserving Particle-In-Cell methods for the Vlasov-Maxwell system near quasi-neutrality. arXiv:1509.04235 [physics], September 2015. arXiv: 1509.04235.

[17] F. Deluzet, G. Fubiani, L. Garrigues, C. Guillet, and J. Narski. Sparse grid reconstructions for particle-in-cell methods. Submitted, 2022.

[18] R.E. Denton and M. Kotschenreuther. {delta}f Algorithm. Technical Report DOE/ET/53088–629, IFSR–629, 10105190, nov 1993.

[19] J. Derouillat, A. Beck, F. Pérez, T. Vinci, M. Chiaramello, A. Grassi, M. Flé, G. Bouchard, I. Plotnikov, N. Aunai, J. Dargent, C. Riconda, and M. Grech. Smilei : A collaborative, open-source, multi-purpose particle-in-cell code for plasma simulation. Computer Physics Communications, 222:351–373, January 2018.

[20] Jochen Garcke. Sparse Grids in a Nutshell. In Jochen Garcke and Michael Griebel, editors, Sparse Grids and Applications, volume 88, pages 57–80. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. Series Title: Lecture Notes in Computational Science and Engineering.

[21] L. Garrigues, G. Fubiani, and J.P. Boeuf. Negative ion extraction via particle simulation for fusion: critical assessment of recent contributions. Nucl. Fusion, 57(1):014003, January 2017.

[22] L. Garrigues, B. Tezenas du Montcel, G. Fubiani, F. Bertomeu, F. Deluzet, and J. Narski. Application of sparse grid combination techniques to low temperature plasmas particle-in-cell simulations. I. Capacitively coupled radio frequency discharges. Journal of Applied Physics, 129(15):153303, April 2021.

[23] L. Garrigues, B. Tezenas du Montcel, G. Fubiani, and B. C. G. Reman. Application of sparse grid combination techniques to low temperature plasmas Particle-In-Cell simulations. II. Electron drift instability in a Hall thruster. Journal of Applied Physics, 129(15):153304, April 2021.

[24] Salimou Gassama, Eric Sonnendrücker, Kai Schneider, Marie Farge, and Margarete O. Domingues. Wavelet denoising for postprocessing of a 2D Particle - In - Cell code. ESAIM: Proc., 16:195–210, 2007.

[25] Michael Griebel. The combination technique for the sparse grid solution of pde's on multiprocessor machines. Parallel Process. Lett., 02(01):61–70, March 1992. Publisher: World Scientific Publishing Co.

[26] Michael Griebel. Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. Computing, 61(2):151–179, June 1998.

[27] Michael Griebel and Jan Hamaekers. Sparse grids for the Schrödinger equation. ESAIM: M2AN, 41(2):215–247, March 2007.

[28] Mario Heene. A massively parallel combination technique for the solution of high-dimensional PDEs. 2018. Publisher: Universität Stuttgart.

[29] M. Hegland. Adaptive sparse grids. ANZIAMJ, 44:335, April 2003.

[30] R Hockney and J Eastwood. Computer Simulation Using Particles. Taylor & Francis, January 1988.

[31] Philipp Hupp. Performance of Unidirectional Hierarchization for Component Grids Virtually Maximized. Procedia Computer Science, 29:2272–2283, 2014.

[32] Philipp Hupp and Riko Jacob. A Cache-Optimal Alternative to the Unidirectional Hierarchization Algorithm. In Jochen Garcke and Dirk Pflüger, editors, Sparse Grids and Applications - Stuttgart 2014, volume 109, pages 103–132. Springer International Publishing, Cham, 2016. Series Title: Lecture Notes in Computational Science and Engineering.
[33] Riko Jacob. Efficient Regular Sparse Grid Hierarchization by a Dynamic Memory Layout. In Jochen Garcke and Dirk Pflüger, editors, Sparse Grids and Applications - Munich 2012, volume 97, pages 195–219. Springer International Publishing, Cham, 2014. Series Title: Lecture Notes in Computational Science and Engineering.
[34] Nicholas A. Krall, Alvin W. Trivelpiece, and Robert A. Gross. Principles of Plasma Physics. American Journal of Physics, 41(12):1380–1381, December 1973.
[35] P. C. Liewer, V. K. Decyk, J. M. Dawson, and G. C. Fox. A universal concurrent algorithm for plasma particle-in-cell simulation codes. In Proceedings of the third conference on Hypercube concurrent computers and applications -, volume 2, pages 1101–1107, Pasadena, California, United States, 1988. ACM Press.
[36] Sriramkrishnan Muralikrishnan, Antoine J. Cerfon, Matthias Frey, Lee F. Ricketson, and Andreas Adelmann. Sparse grid-based adaptive noise reduction strategy for particle-in-cell schemes. Journal of Computational Physics: X, 11:100094, June 2021.
[37] J. Petri. Non-linear evolution of the diocotron instability in a pulsar electrosphere: 2D PIC simulations. A&A, 503(1):1–12, August 2009. arXiv: 0905.1076.
[38] Alexander A. Philippov and Anatoly Spitkovsky. AB INITIO PULSAR MAGNETOSPHERE: THREE-DIMENSIONAL PARTICLE-IN-CELL SIMULATIONS OF AXISYMMETRIC PULSARS. ApJ, 785(2):L33, April 2014.
[39] L F Ricketson and A J Cerfon. Sparse grid techniques for particle-in-cell schemes. Plasma Phys. Control. Fusion, 59(2):024002, February 2017.
[40] Stephen Russell and Niall Madden. An Introduction to the Analysis and Implementation of Sparse Grid Finite Element Methods. Computational Methods in Applied Mathematics, 17(2):299–322, April 2017.
[41] Jie Shen and Haijun Yu. Efficient Spectral Sparse Grid Methods and Applications to High-Dimensional Elliptic Problems. SIAM J. Sci. Comput., 32(6):3228–3250, January 2010.
[42] Igor Surmin, Sergey Bastrakov, Zakhar Matveev, Evgeny Efimenko, Arkady Gonoskov, and Iosif Meyerov. Co-design of a Particle-in-Cell Plasma Simulation Code for Intel Xeon Phi: A First Look at Knights Landing. In Jesus Carretero, Javier Garcia-Blas, Victor Gergel, Vladimir Voevodin, Iosif Meyerov, Juan A. Rico-Gallego, Juan C. Díaz-Martín, Pedro Alonso, Juan Durillo, José Daniel Garcia Sánchez, Alexey L. Lastovetsky, Fabrizio Marozzo, Qin Liu, Zakirul Alam Bhuiyan, Karl Fürlinger, Josef Weidendorfer, and José Gracia, editors, Algorithms and Architectures for Parallel Processing, volume 10049, pages 319–329. Springer International Publishing, Cham, 2016. Series Title: Lecture Notes in Computer Science.
[43] William Tang, Bei Wang, Stephane Ethier, Grzegorz Kwasniewski, Torsten Hoefler, Khaled Z. Ibrahim, Kamesh Madduri, Samuel Williams, Leonid Oliker, Carlos Rosales-Fernandez, and Tim Williams. Extreme Scale Plasma Turbulence Simulations on Top Supercomputers Worldwide. In SC16: International Conference for High Performance Computing, Networking, Storage and Analysis, pages 502–513, Salt Lake City, UT, USA, November 2016. IEEE.
[44] D. Tskhakaya and R. Schneider. Optimization of PIC codes by improved memory management. Journal of Computational Physics, 225(1):829–839, July 2007.
[45] H. Vincenti, M. Lobet, R. Lehe, R. Sasanka, and J.-L. Vay. An efficient and portable SIMD algorithm for charge/current deposition in Particle-In-Cell codes. Computer Physics Communications, 210:145–154, January 2017.

Appendix A. Appendix

proof of proposition 2.2. From the hierarchical basis representation of functions, it holds:

$$\begin{aligned}
\Phi_n^C &= \sum_{\mathbf{l}\in L} c_{\mathbf{l}} I_{V_{h_{\mathbf{l}}}}^{\mathcal{H}} \Phi_{h_{\mathbf{l}}} \\
&= \sum_{\mathbf{l}\in L} c_{\mathbf{l}} \sum_{\mathbf{k}\le\mathbf{l}} \sum_{\mathbf{i}\in\mathcal{B}_{h_{\mathbf{k}}}} \alpha_{\mathbf{k},\mathbf{i}} \varphi_{h_{\mathbf{k}},\mathbf{i}} \\
&= \sum_{\mathbf{l}\in L} c_{\mathbf{l}} \sum_{|\mathbf{k}|_\infty\le n} \sum_{\mathbf{i}\in\mathcal{B}_{h_{\mathbf{k}}}} \beta_{\mathbf{k},\mathbf{i}} \varphi_{h_{\mathbf{k}},\mathbf{i}}, \qquad \text{where } \beta_{\mathbf{k},\mathbf{i}} = \begin{cases} \alpha_{\mathbf{k},\mathbf{i}} & \text{if } \mathbf{k}\le\mathbf{l}, \\ 0 & \text{else.} \end{cases} \\
&= \sum_{|\mathbf{k}|_\infty\le n} \sum_{\mathbf{i}\in\mathcal{B}_{h_{\mathbf{k}}}} \left(\sum_{\mathbf{l}\in L} c_{\mathbf{l}} \beta_{\mathbf{k},\mathbf{j}} \right) \varphi_{h_{\mathbf{k}},\mathbf{i}}.
\end{aligned}$$

□

Let us introduce the transformation from the nodal basis to the hierarchical basis in matrix formulation. Let $\boldsymbol{\alpha}_{\mathbf{l}} = (\alpha_{\mathbf{k},\mathbf{i}})_{\mathbf{i}\in\mathcal{B}_{h_{\mathbf{k}}},\mathbf{k}\le\mathbf{l}}$, $\boldsymbol{\Phi}_{\mathbf{l}} = (\Phi_{h_{\mathbf{l}}}(\mathbf{j}h_{\mathbf{l}}))_{\mathbf{i}\in I_{h_{\mathbf{l}}}} \in \bigotimes_{j=1}^{d} \mathbb{R}^{2^{l_j}+1}$ be vectors with coordinates arranged in ascending order according to their global position in each dimension, which are of the same size because of the definitions of $\mathcal{B}_{h_{\mathbf{l}}}$ and $I_{h_{\mathbf{l}}}$. The hierarchization can thus be written as follows:

$$\boldsymbol{\alpha}_{\mathbf{l}} = \boldsymbol{\mathcal{H}}_{\mathbf{l}} \boldsymbol{\Phi}_{\mathbf{l}}, \quad \boldsymbol{\mathcal{H}}_{\mathbf{l}} = \begin{pmatrix} \boldsymbol{\mathcal{H}}_{l_1} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \boldsymbol{\mathcal{H}}_{l_d} \end{pmatrix}, \quad \boldsymbol{\mathcal{H}}_l = \boldsymbol{\mathcal{H}}_l^{(l-1)} \ldots \boldsymbol{\mathcal{H}}_l^{(1)} \tag{47}$$

where $\boldsymbol{\mathcal{H}}_l^{(k)} \in \mathcal{M}_{2^l+1}(\mathbb{R})$, $k \in \{1, ..., l-1\}$ is defined by:

$$(\boldsymbol{\mathcal{H}}_l^{(k)})_{i,j} = \begin{cases} 1 & \text{if } j = i \\ \frac{1}{2} & \text{if } j = i \pm 2^{k-1} \text{ and } i \in 2^k\mathbb{Z}/\{0, 2^l\} \\ 0 & \text{else} \end{cases} . \tag{48}$$