

# Parallelization of Sparse grid Particle-In-Cell methods, application to plasma physics

M. Chung-To-Sang<sup>‡</sup>, F. Deluzet<sup>†</sup>, G. Fubiani<sup>‡</sup>, L. Garrigues<sup>‡</sup>, C. Guillet<sup>†‡</sup>, J. Narski<sup>†</sup>

<sup>†</sup> Institut de Mathématiques de Toulouse (IMT)

<sup>‡</sup> Laboratoire Plasma et Conversion d'énergie (LAPLACE)

October 11, 2022

**JCAD 2022: Journée Calcul Données**

# Outline

- 1 Introduction to plasmas
- 2 Particle-In-Cell methods
  - Standard Particle-In-Cell
  - Sparse grid Particle-In-Cell
- 3 Numerical applications
  - Performance
  - Parallelization for shared memory architectures
  - Parallelization for single node GPU

# Plasma

- Overall neutral mixture of charged particles, ions and electrons
- One of the main applications is **thermonuclear fusion**: reaction in which atomic nuclei are combined to form different atomic nuclei and subatomic particles (neutrons or protons). The difference in mass between the reactants and products is manifested as either release or absorption of energy.
- ITER project: "Fusion, the nuclear reaction that powers the Sun and the stars, is a potential source of safe, non-carbon emitting and virtually limitless energy. Harnessing fusion's power is the goal of ITER, which has been designed as the key experimental step between today's fusion research machines and tomorrow's fusion power plants" (<http://www.iter.org>)

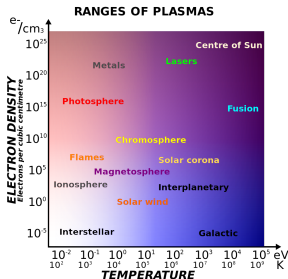
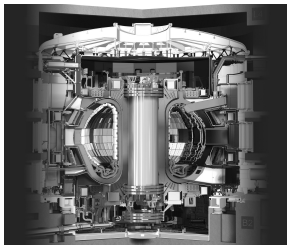


Figure 1: ITER Tokamak (left). Different plasmas (right).

- Vlasov-Poisson kinetic model in electrostatic regime: evolution of charged particles in an electromagnetic field which is either generated by the particles or externally applied or both.

$$\left\{ \begin{array}{l} \frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} f_s + \frac{q_s}{m_s} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \nabla_{\mathbf{v}} f_s = 0, \\ \nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}, \quad \mathbf{E} = -\nabla \Phi, \end{array} \right. \quad (1)$$

- $f_s$  : Phase-space distribution function of species  $s$ ,  $\rho$  : charge density.

$$\rho(\mathbf{x}, t) = \sum_s \rho_s(\mathbf{x}, t) = \sum_s q_s \int f_s(\mathbf{x}, \mathbf{v}, t) d\mathbf{v}. \quad (2)$$

- $\mathbf{E}$ ,  $\mathbf{B}_{\text{ext}}$ ,  $\Phi$  : electric field, external magnetic field and electric potential.
- $q_s$ ,  $m_s$  : charge and mass of a particle of species  $s$ ,  $\epsilon_0$  : vacuum permittivity.
- Difficulties:
  - Vlasov equation posed in 6D phase space (particle positions 3d and velocities 3v) + time.
  - Non linear coupling between Vlasov and Poisson: The evolution of particles depends on the electric field  $\mathbf{E}$ , computed from the charge density  $\rho$  depending on the particle distribution.

- Vlasov-Poisson kinetic model in electrostatic regime: evolution of charged particles in an electromagnetic field which is either generated by the particles or externally applied or both.

$$\left\{ \begin{array}{l} \frac{\partial f_s}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} f_s + \frac{q_s}{m_s} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \nabla_{\mathbf{v}} f_s = 0, \\ \nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}, \quad \mathbf{E} = -\nabla \Phi, \end{array} \right. \quad (1)$$

- $f_s$  : Phase-space distribution function of species  $s$ ,  $\rho$  : charge density.

$$\rho(\mathbf{x}, t) = \sum_s \rho_s(\mathbf{x}, t) = \sum_s q_s \int f_s(\mathbf{x}, \mathbf{v}, t) d\mathbf{v}. \quad (2)$$

- $\mathbf{E}$ ,  $\mathbf{B}_{\text{ext}}$ ,  $\Phi$  : electric field, external magnetic field and electric potential.
- $q_s$ ,  $m_s$  : charge and mass of a particle of species  $s$ ,  $\epsilon_0$  : vacuum permittivity.
- Difficulties:
  - Vlasov equation posed in 6D phase space (particle positions 3d and velocities 3v) + time.
  - Non linear coupling between Vlasov and Poisson: The evolution of particles depends on the electric field  $\mathbf{E}$ , computed from the charge density  $\rho$  depending on the particle distribution.

# Particle-In-Cell (PIC)

- Most used numerical method for simulation of plasma (robust, easy to implement and to parallelize), used for years:
  - C. K Birdsall, D. Fuss. Clouds-in-clouds (1969)
  - J.M. Dawson. Particle simulation of plasmas (1983)
  - L. Garrigues, et al. Application of sparse grid combination techniques to low temperature plasmas particle-in-cell simulations (2021)
  - etc.
- Based on Vlasov-Poisson system of equations.
- Mixed discretization (**set of particles** and **mesh/grid**): Vlasov equation solved by integrating particle trajectories and Poisson equation solved with mesh-based methods.
- Distribution of particles  $f_s$  represented by a collection of numerical particles causing a statistical noise ( $\mathcal{E}_s$ ) **depending on the mean nb. of particles per cell.**

$$\mathbb{V}(\mathcal{E}_s)^{\frac{1}{2}} \approx \left( \frac{1}{Nh_n^d} \right)^{\frac{1}{2}} \quad (3)$$

where  $h_n = 2^{-n}$  is the grid discretization,  $d$  the dimension of the problem and  $N$  is the total number of particles.

# Particle-In-Cell (PIC)

- Most used numerical method for simulation of plasma (robust, easy to implement and to parallelize), used for years:
  - C. K Birdsall, D. Fuss. Clouds-in-clouds (1969)
  - J.M. Dawson. Particle simulation of plasmas (1983)
  - L. Garrigues, et al. Application of sparse grid combination techniques to low temperature plasmas particle-in-cell simulations (2021)
  - etc.
- Based on Vlasov-Poisson system of equations.
- Mixed discretization (**set of particles** and **mesh/grid**): Vlasov equation solved by integrating particle trajectories and Poisson equation solved with mesh-based methods.
- Distribution of particles  $f_s$  represented by a collection of numerical particles causing a statistical noise ( $\mathcal{E}_s$ ) **depending on the mean nb. of particles per cell**.

$$\mathbb{V}(\mathcal{E}_s)^{\frac{1}{2}} \approx \left( \frac{1}{Nh_n^d} \right)^{\frac{1}{2}} \quad (3)$$

where  $h_n = 2^{-n}$  is the grid discretization,  $d$  the dimension of the problem and  $N$  is the total number of particles.

# Particle-In-Cell (PIC)

- Steps of the method (one time iteration):
  - Accumulation of particle properties (charge density) onto the grid.
  - Resolution of Poisson equation on the grid to get the electric field (Finite difference, FEM, FFT).
  - Interpolation of field properties (electric field) onto the particles.
  - Evolution of particles (position, velocity) from their trajectory (Leap frog).

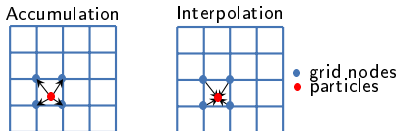


Figure 2: steps ① and ③ with linear shape functions.

- Error in PIC schemes is **dominated by statistical noise**  $\mathcal{E}_s \Rightarrow$  Requires substantial number of particles. *E.g* for 3D3V simulation with grid discretization  $2^{-10}$  in each direction and 1000 particles per cell:
  - $10^{12}$  particles with 6 coordinates (3 for positions, 3 for velocity),  $10^9$  grid nodes for potential, density, electric field.
  - 51TB of particle data, 42GB of grid data.



# Particle-In-Cell (PIC)

- Steps of the method (one time iteration):
  - Accumulation of particle properties (charge density) onto the grid.
  - Resolution of Poisson equation on the grid to get the electric field (Finite difference, FEM, FFT).
  - Interpolation of field properties (electric field) onto the particles.
  - Evolution of particles (position, velocity) from their trajectory (Leap frog).

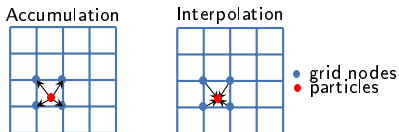


Figure 2: steps ① and ③ with linear shape functions.

- Error in PIC schemes is **dominated by statistical noise**  $\mathcal{E}_s \Rightarrow$  Requires substantial number of particles. *E.g* for 3D3V simulation with grid discretization  $2^{-10}$  in each direction and 1000 particles per cell:
  - $10^{12}$  particles with 6 coordinates (3 for positions, 3 for velocity),  $10^9$  grid nodes for potential, density, electric field.
  - 51TB of particle data, 42GB of grid data.

# Sparse grid reconstruction

- Cartesian grid is substituted by a set of component grids with coarse discretization. Cartesian grid has  $O(2^{dn})$  nodes while there are  $O(n^{d-1})$  component grids with  $O(2^n)$  nodes.
- Particle properties are accumulated onto each component grid and electric field is computed on each component grid. Electric field is reconstructed at particle positions with combination technique [6].

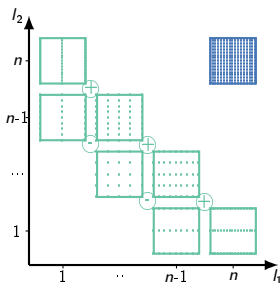


Figure 3: Component grids within the combination and Cartesian grid.

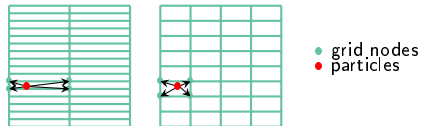
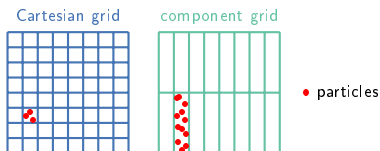


Figure 4: Accumulation step on component grids.

# Sparse grid reconstruction

- Significant reduction of statistical noise because component grids have larger cells than Cartesian grid  $\Rightarrow$  more particles per cell.



$$\mathbb{V}(\mathcal{E}_s)^{\frac{1}{2}} \underset{\sim}{\leq} \underbrace{|\log h_n|^{d-1} \left( \frac{1}{Nh_n} \right)^{\frac{1}{2}}}_{\text{sparse grid reconstruction}} \leq \underbrace{\left( \frac{1}{Nh_n^d} \right)^{\frac{1}{2}}}_{\text{standard}}, \quad (4)$$

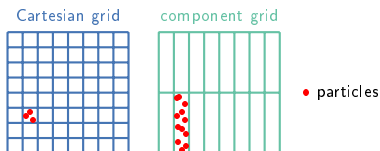
$h_n = 2^{-n}$ : grid discretization,  $d$ : dimension,  $N$ : number of particles.

- Memory requirements are much lower because less particles for equivalent statistical error. Computational cost of field solver is significantly mitigated because less total number of nodes.
- A slight additional grid error ( $\mathcal{E}_g$ ) is introduced due to the approximation with the sparse grids [1]:

$$\mathcal{E}_g \approx \underbrace{|\log h_n|^{d-1} h_n^2}_{\text{sparse grid reconstruction}} \geq \underbrace{h_n^2}_{\text{standard}} \quad (5)$$

# Sparse grid reconstruction

- Significant reduction of statistical noise because component grids have larger cells than Cartesian grid  $\Rightarrow$  more particles per cell.



$$\mathbb{V}(\mathcal{E}_s)^{\frac{1}{2}} \underset{\sim}{\leq} \underbrace{|\log h_n|^{d-1} \left( \frac{1}{Nh_n} \right)^{\frac{1}{2}}}_{\text{sparse grid reconstruction}} \leq \underbrace{\left( \frac{1}{Nh_n^d} \right)^{\frac{1}{2}}}_{\text{standard}}, \quad (4)$$

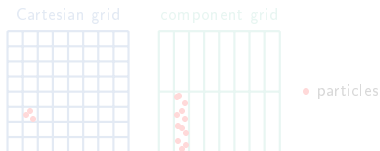
$h_n = 2^{-n}$ : grid discretization,  $d$ : dimension,  $N$ : number of particles.

- Memory requirements are much lower because less particles for equivalent statistical error. Computational cost of field solver is significantly mitigated because less total number of nodes.
- A slight additional grid error ( $\mathcal{E}_g$ ) is introduced due to the approximation with the sparse grids [1]:

$$\mathcal{E}_g \approx \underbrace{|\log h_n|^{d-1} h_n^2}_{\text{sparse grid reconstruction}} \geq \underbrace{h_n^2}_{\text{standard}} \quad (5)$$

# Sparse grid reconstruction

- Significant reduction of statistical noise because component grids have larger cells than Cartesian grid  $\Rightarrow$  more particles per cell.



$$\mathbb{V}(\mathcal{E}_s)^{\frac{1}{2}} \underset{\sim}{\leq} \underbrace{|\log h_n|^{d-1} \left( \frac{1}{Nh_n} \right)^{\frac{1}{2}}}_{\text{sparse grid reconstruction}} \leq \underbrace{\left( \frac{1}{Nh_n^d} \right)^{\frac{1}{2}}}_{\text{standard}}, \quad (4)$$

$h_n = 2^{-n}$ : grid discretization,  $d$ : dimension,  $N$ : number of particles.

- Memory requirements are much lower because less particles for equivalent statistical error. Computational cost of field solver is significantly mitigated because less total number of nodes.
- A slight additional grid error ( $\mathcal{E}_g$ ) is introduced due to the approximation with the sparse grids [1]:

$$\mathcal{E}_g \approx \underbrace{|\log h_n|^{d-1} h_n^2}_{\text{sparse grid reconstruction}} \geq \underbrace{h_n^2}_{\text{standard}} \quad (5)$$

# Charge accumulation in PIC methods

- Irregular memory accesses in grid array. Data must be loaded from the main memory.
- Usually bypassed with particle sorting (particles stored next to neighbors according to the grid).

---

### Algorithm 6 Charge accumulation on grid

---

**Require:** Array particle[:]

**Ensure:** Array grid[:] containing the charge density

**for all particles do**

**Read** positions of particles in the particle array

**Determine** the nodes of the cell containing the particle

**for all**  $n_i$  nodes of the cell containing the particle **do**

**Compute** the charge contribution  $\rho_i$  of the particle at the node  $n_i$

**Add** the contribution  $\rho_i$  in the grid array at position  $n_i$ :  $\text{grid}[n_i] \leftarrow \rho_i$

**end for**

**end for**

---

Contiguous accesses  
to particle array

Non-contiguous accesses  
to grid array

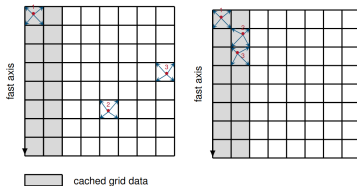


Figure 5: Particles are accessed in order 1-2-3. Without sorting (middle), with sorting (right).

# Computational time of steps

- PIC methods are usually memory-bounded. Too many memory accesses (particle data) in comparison to computations  $\Rightarrow$  Particle advance in time most-time consuming step because a lot of memory accesses (particle positions, velocity, contribution of electric field) and few computations.
- Sparse grid method has less memory accesses (less particles) and more (independent) computations (charge accumulation and field resolution on each grid)  $\Rightarrow$  charge accumulation is by far most time-consuming step: **95%**.

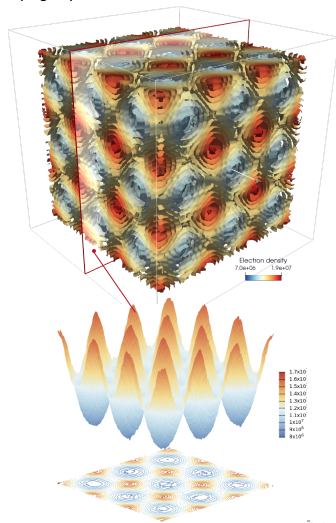
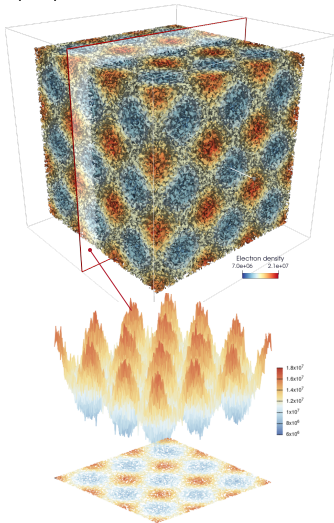
Steps	Standard (1)	Standard (2)	Sparse grid
Charge accumulation	4%	4%	<b>95%</b>
Poisson solver	4%	5%	0.7%
Field interpolation	33%	7%	4%
Advance particles	59%	84%	0.3%

Table 6: (1) without sorting of particles, (2) with pre-sorting of particles.

- Necessity of optimized implementation and efficient parallelization of charge accumulation.

# Sequential execution results

- Landau Damping. Electrons are immersed in a background of ions. Perturbation of maxwellian electron distribution considered. Electron density: standard PIC (left)  $2.09 \times 10^8$  particles, sparse grid PIC (right)  $2.6 \times 10^6$  particles.





# Sequential execution results

- Benefits of sparse grid over standard methods deepen the more the grid is refined. Demanding problems are more achievable...

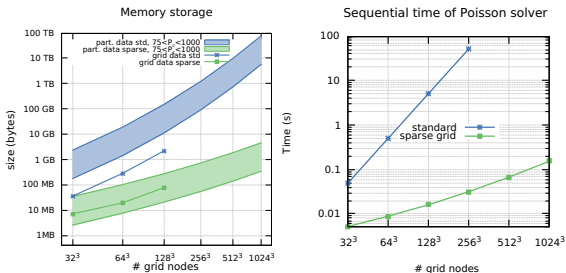


Figure 7: Memory storage, time execution of Poisson solver.

- 3D-3V Landau damping simulation with  $2^{-9}$  grid discretization in each direction and 1000 particles per cell on a laptop (Intel<sup>®</sup> Core<sup>™</sup> i9-10885H CPU 8 cores @2.40 GHz with 30GB of RAM memory). Equivalent simulation in standard PIC would require  $10^{12}$  particles and 60TB of memory.

# Sequential execution results

- Benefits of sparse grid over standard methods deepen the more the grid is refined. Demanding problems are more achievable...

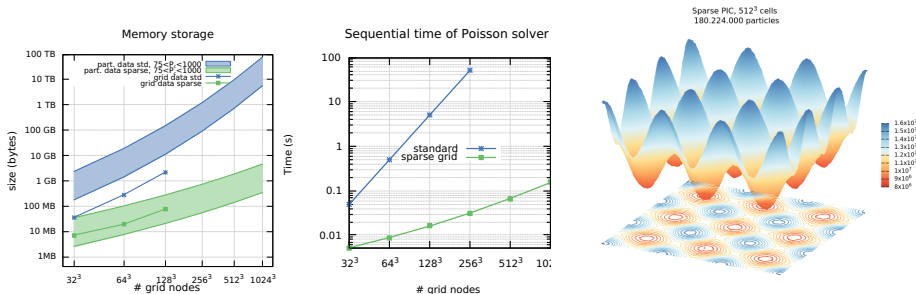


Figure 7: Memory storage, time execution of Poisson solver and electron density (2d x-y profile).

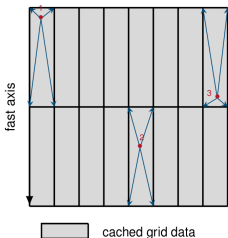
- 3D-3V Landau damping simulation with  $2^{-9}$  grid discretization in each direction and 1000 particles per cell on a laptop (Intel<sup>®</sup> Core™ i9-10885H CPU 8 cores @2.40 GHz with 30GB of RAM memory). Equivalent simulation in standard PIC would require  $10^{12}$  particles and 60TB of memory.

# Parallelization for shared memory architectures

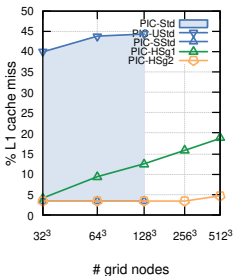
- Difficulties of parallelization:
  - Standard PIC is not suited for shared memory parallelization (memory-bounded).
  - Overload of memory bandwidth because of irregular memory accesses.
  - Particle sorting too expensive with sparse grids (one different sorting for grids).
- Key elements of efficient parallelization:
  - Particle population is divided into samples of particles and distributed into the NUMA domains.
  - Accumulations of density onto the component grids are independent. Component grids are distributed to the cores inside NUMA domains.
  - Each component grid (all nodes) fits in the L1-cache private to each core.
  - Each sample of particles is divided into clusters of particles in order to achieve perfect load balance (number of grids not necessarily equal to the number of cores inside a NUMA domain)

# Parallelization for shared memory architectures

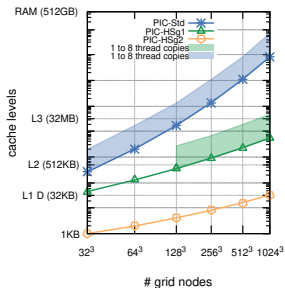
- Difficulties of parallelization:
  - Standard PIC is not suited for shared memory parallelization (memory-bounded).
  - Overload of memory bandwidth because of irregular memory accesses.
  - Particle sorting too expensive with sparse grids (one different sorting for grids).
- Key elements of efficient parallelization:
  - Particle population is divided into samples of particles and distributed into the NUMA domains.
  - Accumulations of density onto the component grids are **independent**. Component grids are distributed to the cores inside NUMA domains.
  - Each component grid (all nodes) fits in the L1-cache private to each core.
  - Each sample of particles is divided into clusters of particles in order to achieve perfect load balance (number of grids not necessarily equal to the number of cores inside a NUMA domain)



b) L1 data cache miss



c) Storage size of the density arrays



# Parallelization for shared memory architectures

- NUMA architecture: two AMD EPYC™ 7713 *Milan* CPUs with a total of **128 cores** and RAM memory of 512GB. 8 NUMA nodes. Maximum memory bandwidth of 190.73GB/s. 32MB L3-cache, private 512 KB L2-cache and 32KB L1-cache.

Method	grid discretization	particle data	nb. procs	Time	Speedup
Standard (sorted)	$2^{-7}$	75GB	1	306.4 s	1
Sparse grid	$2^{-7}$	122MB	1	1.2 s	1
Sparse grid	$2^{-7}$	122MB	128	0.012 s	101

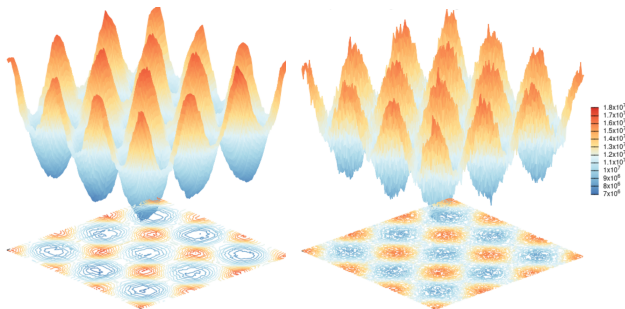


Figure 8: Sparse grid

Figure 9: Standard (sorted)

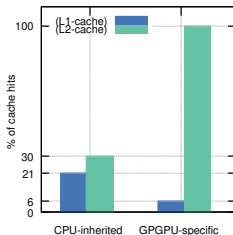
# Parallelization for GPU

- Difficulties of parallelization: Standard and CPU-inherited implementation
  - Large amount of data to transfer between host and device (Standard)
  - A lot of cores sharing the same L1-cache.
  - Reduction operations limit the number of instruction streams.
- Key elements of efficient parallelization: GPGPU-specific implementation
  - All the data on the GPU, no memory transfers between host and device except at initialization.
  - Particle population divided into clusters and distributed to Streaming Multiprocessors (SM).
  - The property of a particle is accumulated onto all the component grids by the cores of a SM in Single Instruction Multiple Thread (SIMT).
  - All the component grid nodes fit in the L2-cache of GPU compensating irregular and non-coalesced memory accesses genuine to PIC methods.
  - Atomic operation with as much as possible instruction streams in order to mask memory latency.

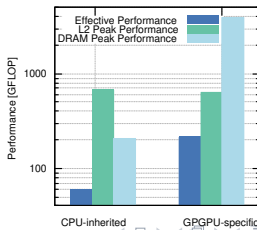
# Parallelization for GPU

- Difficulties of parallelization: Standard and CPU-inherited implementation
  - Large amount of data to transfer between host and device (Standard)
  - A lot of cores sharing the same L1-cache.
  - Reduction operations limit the number of instruction streams.
- Key elements of efficient parallelization: GPGPU-specific implementation
  - All the data on the GPU, no memory transfers between host and device except at initialization.
  - Particle population divided into clusters and distributed to Streaming Multiprocessors (SM).
  - The property of a particle is accumulated onto all the component grids by the cores of a SM in Single Instruction Multiple Thread (SIMT).
  - All the component grid nodes fit in the L2-cache of GPU compensating irregular and non-coalesced memory accesses genuine to PIC methods.
  - Atomic operation with as much as possible instruction streams in order to mask memory latency.

Ratio of L1-cache hits and L2-cache hits



Theoretical peak performance and effective performance



# Parallelization for GPU

Figure 10: Configurations with equivalent statistical error.

Method	Grid discretization	Memory footprint	Time (1 iter.)
Ref: Standard (CPU AMD <i>Milan</i> )	$2^{-7}$	151GB	585 s
Sparse-PIC (CPU Intel® <i>Sylake</i> )	$2^{-7}$	248MB	5.4 s ( $\div 108$ )
Sparse-PIC (CPU AMD <i>Milan</i> )	$2^{-7}$	248MB	2.6 s ( $\div 225$ )
Sparse-PIC (GPU Tesla V100)	$2^{-7}$	248MB	0.05 s ( $\div 11,700$ )

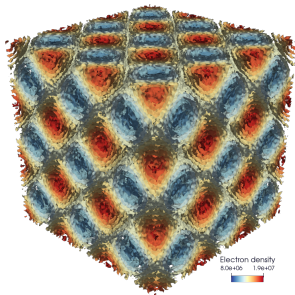


Figure 11: Standard.

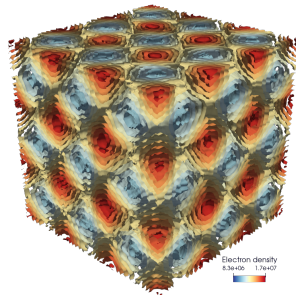








Figure 12: Sparse grid.



# References

-  F. Deluzet, G. Fubiani, L. Garrigues, C. Guillet, J.Narski, Sparse Grid reconstructions for Particle-In-Cell, ESAIM: M2AN, 56 5 (2022) 1809-1841.
-  F. Deluzet, G. Fubiani, L. Garrigues, C. Guillet, and J. Narski. Efficient parallelization for 3d-3v sparse grid particle-in-cell: shared memory architectures. Submitted, 2022
-  F. Deluzet, G. Fubiani, L. Garrigues, C. Guillet, and J. Narski. Efficient parallelization for 3d-3v sparse grid particle-in-cell: Single GPU architectures. Submitted, 2022
-  L. F. Ricketson, A. J. Cerfon, Sparse grid techniques for particle-in-cell schemes, Plasma Physics and Controlled Fusion 59 (2) (2016) 024002.
-  HPC resources from CALMIP Grant 2022-1125.
-  M.Griebel, The combination technique for the sparse grid solution of pde's on multiprocessor machines Parallel Process (1992). Lett. 2 61-70.