School of Industrial and Information Engineering
Master of Science in Mathematical Engineering

POLITECNICO
MILANO 1863

Massively parallel Particle-In-Cell simulations of laser-plasma interaction: a performance analysis

Supervisor: Prof. Matteo Passoni
Co-supervisor: Dr. Arianna Formenti

Dissertation of:
Matteo Lo Verso
Personal ID 899678

Academic Year 2019–2020

Abstract

This thesis reports on a numerical investigation on the performances of massively parallel Particle-In-Cell simulations of laser-plasma interaction in regimes that are relevant in the context of laser-driven particle acceleration. Particle-In-Cell simulations are by far the most established numerical tool for the approximated solution of the Vlasov-Maxwell system, which governs the dynamics of relativistic, collisionless plasmas under a kinetic description. Two physical scenarios are simulated and studied separately in a two-dimensional geometry: a high-power laser pulse interacting with a thin solid foil or a with a low-density thick plasma slab. These two configurations represent two cases-studies of relativistic laser-plasma interaction in the over-critical and near-critical regimes respectively. Two Particle-In-Cell codes – Smilei and WarpX – implementing different parallelization strategies are exploited on the CPU-based cluster Galileo, hosted at Cineca.
The computational load required by the simulations is examined, exploring the role of different code parameters and parallel configurations. Special attention is given to the parallelization approaches implemented by the codes, with a particular focus on the domain decomposition strategies, and to the consequential effects on the simulation times of the main routines, which are broken down according to the main building-blocks of a Particle-In-Cell algorithm. Moreover, one of the codes – WarpX – has been tested on the GPU-based cluster Marconi100, hosted at Cineca as well, so that a hybrid GPU-CPU parallelization framework has also been considered. These results indicate that the benefits of different parallelization strategies, both code and cluster dependent, can be different depending on the simulated physical scenarios.

Sommario

Questa tesi riporta un'indagine numerica sulle performance di simulazioni massivamente parallelizzate Particle-In-Cell di interazione laser-plasma in regimi che sono rilevanti nel contesto di accelerazione di particelle tramite laser. Le simulazioni Particle-In-Cell costituiscono lo scenario numerico più largamente adottato per la risoluzione del sistema di Vlasov-Maxwell, ovvero il modello che descrive, con una formulazione di tipo cinetico, la dinamica di un plasma non collisionale relativistico.
L'oggetto di studio è costituito da due scenari fisici, simulati in un'ambientazione bidimensionale: un impulso laser superintenso interagisce con un foglio solido sottile o con una porzione spessa di plasma a bassa densità. Queste due configurazioni sono rappresentativi di due casi tipicamente studiati nell'ambito dell'interazione laser-plasma relativistica, ovvero il regime sovra-denso e quello a densità quasi-critica rispettivamente. Tali simulazioni vengono affrontate tramite l'utilizzo di due codici Particle-In-Cel – Smilei e WarpX – caratterizzati da diverse strategie di parallelizzazione. A tal fine si ricorre all'architettura CPU di Galileo, un supercomputer appartenente al centro di calcolo Cineca.
Nello specifico, l'aspetto esaminato è il carico computazionale richiesto dalle simulazioni in questione, esplorando il ruolo ricoperto da diversi parametri dei due codici e dalle configurazioni di parallelizzazione. Un'attenzione particolare è riservata allo studio degli approcci di parallelizzazione implementati dai due codici: in particolare, uno studio apposito è dedicato alle strategie di decomposizione del dominio e ai conseguenti effetti sui tempi di simulazioni delle principali routine, che sono analizzate separatamente, in accordo con lo schema dell'algoritmo Particle-In-Cell. Inoltre, uno dei due codici – WarpX – è stato testato anche sul supercomputer ad architettura GPU Marconi100, anch'esso appartenete al Cineca, in modo da considerare anche una parallelizzazione ibrida GPU-CPU. Tali risultati indicano che i benefici derivanti da differenti strategie di parallelizzazione, dipendenti sia dal codice sia dal cluster in questione, possono essere diversi a seconda degli scenari fisici simulati.

Estratto

Dalla sua invenzione nel 1960 fino ad oggi, lo studio e lo sviluppo di una tecnologia laser avanzata ha reso possibile innumerevoli applicazioni tecnologiche e scientifiche. In particolare, negli ultimi decenni il perfezionamento di alcune tecniche, sempre più raffinate, ha consentito di realizzare impulsi laser superintensi: questi vantano una potenza elevatissima (~ 1 PW), una durata ultrabreve (qualche decina di fs) e sono focalizzati su aree minuscole (qualche μm^2), permettendo di raggiungere enormi intensità ($\gtrsim 10^{21}$ W/cm^2).
L'interazione laser-materia ha da sempre costituito un interessante campo di studio, per via dell'enorme potenziale che esso riserva. Infatti, come risaputo, laser sufficientemente intensi, interagendo con opportuni bersagli, possono indurre una separazione della carica, inducendo così la ionizzazione del materiale irraggiato, fino ad arrivare alla creazione di plasmi.
Questo fenomeno è stato ampiamente sfruttato, negli ultimi anni, dalla comunità scientifica, che ha sfruttato l'interazione laser-materia per diversi scopi di ricerca, tra cui l'accelerazione di particelle. Nello specifico, laser dall'intensità superiore a 10^{18} W/cm^2, negli ultimi anni, costituiscono un oggetto di studio per l'enorme potenziale derivante dalla loro interazione con la materia che, seppur inizialmente solida, viene rapidamente trasformata in plasma.
In futuro, acceleratori di particelle basati sull'interazione laser-plasma potranno quindi risultare di notevole utilità, potenzialmente divenendo persino una valida alternativa agli acceleratori convenzionali, grazie alle dimensioni e ai costi notevolmente inferiori.
E' in atto, in particolare, una fase esplorativa che si pone come obiettivo lo sfruttamento degli effetti derivanti dall'interazione laser-plasma finalizzato alla realizzazione di sorgenti compatte di ioni accelerati. Uno degli schemi più usati, a tal proposito, consiste nell'utilizzo di laser superintensi per irraggiare bersagli solidi sottili. Gli elettroni che si trovano sulla superficie colpita, eccitati dall'impulso, acquisiscono un'energia cinetica tale da passare attraverso il bersaglio fino a formare una nube elettronica sul retro. Questa separazione di carica, quindi, induce un campo elettrico longitudinale talmente potente da accelerare gli ioni sul retro fino a velocità relativistiche. Utilizzando laser con impulsi della durata di qualche fs, si possono ottenere fasci composti da $\sim 10^{10}$ ioni accelerati, caratterizzati da energie massime fino a ~ 10 MeV, con un largo spettro esponenziale. In particolare, una

strategia promettente, sempre meglio studiata, prevede un aumento delle prestazioni offerte da questo schema attraverso un ampliamento degli effetti derivanti dall'accoppiamento fra l'impulso laser incidente e gli elettroni del bersaglio. Infatti, ciò significherebbe aumentare la temperatura degli elettroni eccitati, e quindi la loro energia cinetica, il che si traduce in un'intensificazione del campo elettrico accelerante e quindi in una più efficiente accelerazione di ioni. In questo frangente, indagini numeriche sempre più corpose risultano di fondamentale importanza, al fine di riuscire a studiare sempre più nel dettaglio il potenziale derivante dalla fisica di questo fenomeno. Più precisamente, le indagini degli ultimi anni sono state mirate allo studio di una struttura del bersaglio irradiato che possa ottimizzare le prestazioni, sia nel numero degli ioni accelerati sia nell'energia cinetica da essi posseduta. A tal proposito, le prestazioni fornite da diversi tipi di bersagli, più o meno elaborati, sono state confrontate. Da ciò è emerso il fatto che un bersaglio doppio-strato, composto da una schiuma nanostrutturata a bassa densità (qualche mg/cm^3) depositata su un foglio solido sottile può aumentare notevolmente la resa finale. Infatti, tale strato protettivo è stato concepito nell'ottica di sfruttare al meglio l'accoppiamento che si instaura fra un laser ad alta intensità e un plasma a densità critica (o quasi), il che si traduce in un guadagno nell'assorbimento di energia dell'impulso da parte delle particelle, intensificando il processo di accelerazione.

La fisica che descrive l'interazione fra laser superintensi e materia è assai complessa e articolata e può essere caratterizzata da molti sofisticati meccanismi. Ad ogni modo, data l'alta intensità degli impulsi in questione, la materia viene fortemente ionizzata, raggiungendo ben presto lo stato di plasma. Ciò significa che può essere considerata come un insieme di particelle cariche, che si muovono, interagendo, sotto l'effetto di campi elettromagnetici autoconsistenti generati dalla loro stessa dinamica.
Nell'ambito dell'accelerazione di ioni da laser, è possibile considerare le interazioni a lungo raggio dominanti rispetto a quelle a corto raggio e, quindi, tali plasmi mostrano un comportamento prevalentemente collettivo. Diversi modelli matematici esistono per descrivere questo stato della materia: in particolare, quello che risulta più adatto in questo ambito consiste in un'estensione della teoria cinetica dei gas a questo particolare sistema. Nello specifico, il modello matematico più usato deriva dall'accoppiamento fra l'equazione di Vlasov relativistica e le equazioni di Maxwell. Studi analitici di questo sistema di equazioni sono possibili solo tramite semplificazioni derivanti da forti assunzioni a priori, le quali risultano inappropriate per descrivere contesti realistici, quale l'accelerazione di ioni. Di conseguenza, l'unica strada percorribile prevede l'adozione di un approccio numerico e, tra gli schemi numerici esistenti, il più utilizzato è senz'altro il metodo Particle-In-Cell (PIC), dato che risulta essere quello che meglio si presta a simulazioni cinetiche di interazione laser-plasma in regime relativistico e non collisionale. Tuttavia, la simulazione di scenari così articolati, chiaramente, implica una complessità numerica non indifferente, che richiede

elevate risorse computazionali. Il numero di particelle coinvolte, infatti, è davvero elevato e, per riuscire a catturare fenomeni che avvengono a scale caratteristiche del plasma (come la *lunghezza di pelle*), è essenziale adottare una risoluzione numerica raffinata. Risulta, dunque, proibitivo simulare scenari con parametri realistici: per esempio, una significativa riduzione nella densità dei plasmi simulati, rispetto ai valori osservati in laboratorio, deve essere introdotta al fine di alleggerire gli sforzi computazionali. Nonostante ciò, al momento attuale, risulta spesso impossibile realizzare simulazioni tridimensionali, che richiederebbero tempi eccessivi, e quindi il loro utilizzo viene riservato solo a casi selezionati. Ciò significa che, nella maggior parte dei casi, deve essere intrapreso un approccio bidimensionale, il quale, però, è utile prevalentemente per raccogliere informazioni di tipo qualitativo. Tuttavia, dato il grado di complessità sempre crescente degli scenari fisici che sono oggetto di studio negli ultimi anni, l'adozione di un approccio eccessivamente semplificato, sia nei parametri fisici sia nella dimensionalità nelle simulazioni, si rivela senz'altro limitante ed emerge sempre più la necessità di simulazioni quanto più realistiche.
Urge, dunque, una campagna investigativa finalizzata allo studio dettagliato dei codici PIC maggiormente utilizzati. L'obiettivo di questo lavoro di tesi consiste in un'indagine delle performance raggiungibili con codici PIC, al fine di riuscire a sfruttare al più le potenzialità di tali codici.

Il lavoro di tesi qui presentato riporta un'indagine numerica delle performance associate a simulazioni bidimensionali selezionate di interazione laser-plasma, affrontate col metodo numerico PIC. A tal fine, sono stati selezionati due codici open-source – Smilei e WarpX – caratterizzati da diverse strategie di parallelizzazione. Inoltre, tale parallelizzaione, dotata di logica altamente massiva, è stata esplorata su due diversi supercomputer, al fine di testare sia un ambiente CPU sia un'architettura ibrida GPU-CPU. Due scenari fisici costituiscono l'oggetto di studio. Il primo è rappresentato dall'interazione fra un laser superintenso e un bersaglio solido sottile, mentre nel secondo scenario tale tipo di laser viene fatto interagire con una schiuma spessa a bassa densità. Queste simulazioni numeriche sono state eseguite mediante Galileo, un supercomputer ad architettura CPU che risiede nel centro di calcolo Cineca, a Bologna. Inoltre, le performance raggiunte da WarpX sono state studiate anche su Marconi100, un supercomputer recentemente avviato dal medesimo centro di calcolo equipaggiato con un'architettura ibrida GPU-CPU.
Quindi, tali indagini, che si inseriscono all'interno di una campagna in fase di avvio, potranno, in futuro, dare adito a una migliore comprensione delle strategie di parallelizzazione più convenienti, anche in scenari fisici altamente sofisticati. Oltre a ciò, grazie all'estrema efficienza insita nella natura multi-core delle GPU, simulazioni più realistiche (che al momento risultano troppo onerose) potranno essere intraprese, magari senza ricorrere (almeno per il caso bidimensionale) a parallelizzazioni massive, ma utilizzando laptop comuni. Tali risultati, quindi, costituiscono il primo passo verso una conoscenza più completa di come

algoritmi PIC possano meglio adattarsi ad ambienti altamente parallelizzati, gettando le fondamenta per un'ulteriore ottimizzazione dell'uso di tale metodo numerico.

Contents

Introduction

Since the laser technology was invented in 1960, the interaction between a laser pulse and matter has been the focus of a great number of investigations. In the past decades, the effects of several properties of both the laser pulses and the irradiated matter on the interaction regimes have been studied in order to achieve a better understanding of these processes, with the ultimate goal of controlling the physical mechanisms at play. The development of laser technology has let the scientific community achieve higher and higher laser focused intensities: in this context, a great conquest is represented by the realization of the *chirped pulse amplification* technology, thanks to which very intense laser pules can now be realized. Thus, the availability of ultra-intense and ultra-short pulses, called *superintense* pulses, opened the way to the exploration of new laser-matter interaction regimes, which can also be exploited for laser-driven particle acceleration. Such acceleration can be activated by suitably exploiting the complex physical mechanism that arise during the interaction between a superintense laser and matter. In particular, given the high intensity of such pulses, the irradiated matter is rapidly ionized, turning into the state of *plasma*. Under this condition the irradiated matter can be considered as a collection of charged particles, in electromagnetic interaction with each other.

Ongoing researches are exploring these complex phenomena in order to realize compact sources of accelerated particles. In general, laser-plasma based accelerators, in fact, would be preferable to conventional accelerators because of their significantly lower demands in terms of size and cost. In particular, the acceleration of ions, in the most typical configuration, can be obtained through the interaction of superintense pulses and thin solid foils. In this context, a great issue consists in finding smart strategies to optimize the conventional ion acceleration scheme. One option, studied in the last years, is to exploit *double-layer targets*, which basically consist in a nanostructured, low-density material, called *foam*, attached to a thin solid foil.

Intensive numerical campaigns are essential to achieve a deeper comprehension of these complex mechanisms. The most widely exploited numerical tools are based on the Particle-In-Cell (PIC) approach, which allows one to perform numerical simulations involving laser-

plasma interaction including kinetic and relativistic effects. However, as it often happens, simulations performed with high accuracy come with considerable computational efforts. Actually, realistic physical parameters are often impossible to reproduce so that reduced values are chosen. Moreover, most of the times simulations in three-dimensional geometry result prohibitive, especially when wide parametric scans are desired. Hence, many times a two-dimensional framework becomes essential, even if this approach is limiting, because of the inherent three-dimensional nature of the actual physics. For this reason, a numerical investigation focused on understanding how to make an optimal use of PIC codes while assessing their performances becomes of great interest.

The scope of the present thesis is the exploration of the numerical strategies which can improve the performances of PIC codes when used in massively parallel environments, with a special attention to the effects due to different parallelization strategies. The general goal is to gain insights on optimal strategies which may reduce the computational times required for typical laser-plasma interaction simulations. Two very different regimes of interaction are selected: the interaction of a superintense laser pulse with a low-density plasma, representative of a foam material, or with a thin solid foil. Two codes are tested which are the open source, massively parallel, PIC codes *Smilei* and *WarpX*. The performances have been tested on the CPU-based cluster Galileo and on the hybrid GPU/CPU-based supercomputer Marconi100, both of them hosted at CINECA, Bologna, Italy.

The thesis is organized as follows.

- Chapter 1 - *Super-intense laser-plasma interaction* - provides a background from the physical and mathematical point of view. After an introduction of the concept of high-intensity lasers, the plasma state is presented, with its mathematical description given by the relativistic Vlasov-Maxwell system. Then, some general features of the physics characteristic of laser-plasma interactions are discussed. Lastly, the basic elements specific of laser based ion acceleration are exposed, with particular attention to advanced double-layer targets consisting in nanostructured materials deposited on solid foils.
- Chapter 2 - *Numerical approaches to the kinetic description of laser-plasma interaction* - frames the two main classes of methods used for the numerical resolution of the Vlasov-Maxwell system: grid-based methods and particle methods. In particular, an in-depth description of the widely used PIC method, in all its constituent algorithms, is given. The main differences between the two approaches are highlighted. Motivations and goals of the present thesis are detailed at the end of this chapter.
- Chapter 3 - *Performance analyses of laser-plasma interaction simulation on a CPU-based cluster* - provides an extended discussion of the numerical campaign to test the

performances of the codes. Firstly, the selected codes are presented, with an in-depth treatment of the implemented parallelization strategies. Then, two physical scenarios are selected to be simulated. The computational times of several simulations involving different numerical configurations are examined. Moreover, the single computational contributions related to the algorithm routines and the communication between different processes are investigated.

- Chapter 4 - Performance analyses of laser-plasma interaction simulations on a hybrid CPU-GPU system cluster - is devoted to an investigation of the benefits deriving using a hybrid GPU-CPU parallelization framework.
- Chapter 5 - Chapter 5 – *Conclusions and perspectives* – discusses the over-all results of the work and highlights possible further developments along with ideas for potential future studies.

Chapter 1

Super-intense laser-plasma interaction

Super-intense lasers and their interaction with matter is an interesting open field of research: the possibility of exploiting lasers to manipulate matter has been a source of great interest in the last decades, since the first realization of a laser pulse in 1960.
Currently laser facilities can achieve very high intensities and can produce concentrated both in space and time pulses (focused on a very small area and with a very small duration).
In particular, in the last years an interesting field under investigation consists in the ionization of matter and the generation of plasmas through super-intense lasers: in fact, the interaction between high-power lasers and plasmas finds a large number of promising applications, such as the particle acceleration.
So, ultra-intense and ultras-short pulses can ionize matter and it's possible to exploit the laser-plasma interactions to generate accelerated beams of particles.
The availability of laser-plasma interactions to achieve particle accelerations was suggested for the first time by Veksler in 1957 ([1]): unlike in conventional particle accelerators, where the motion of each particle is completely determined by an external source and is independent of the other particles, this method implies the concept of coherent acceleration, i.e. the magnitude of the accelerating field acting on each individual particle is proportional to the number of particles being accelerated.
The laser-induced particle acceleration, thanks to the compactness of this scheme, could be an attractive alternative to conventional accelerators, addressing some of their limitations, such as high-costs, not compact size, radioprotection issues and non-tunable energy.

1.1 High-power lasers

A laser, acronym for "Light Amplification by Stimulated Emission of Radiation", is a device that generates or amplifies coherent radiation at frequencies in the infrared, visible, or ultraviolet regions of the electromagnetic spectrum.
The theoretical foundations for laser were established for the first time in 1917 by Albert Einstein in the paper *"Quantum Theory of Radiation"* [2], in which theorized the concept of stimulated emission, according to which electrons could be stimulated to emit light of a particular wavelength.
But it took nearly 40 years before scientists have been able to amplify those emissions and the first laser was realized only in 1960 by Thedore H. Maiman at Hughes Research Laboratories, in California ([3]).
A distinctive trait of this device is the spatial and temporal coherence of the emitted light: spatial coherence allows a laser to be focused to a tight spot and to stay narrow over great distances; temporal coherence, instead, allows to emit light with a very narrow spectrum, in order to produce monochromatic waves.
Alternatively, temporal coherence can be used to produce ultrashort pulses, that are pulses of light with a broad spectrum and time duration of the order of a picosecond (10^{-12} second) or less.
Among the main parameters of a laser pulse there are the wavelength λ, the time duration τ and the spot size σ.
The energy of the pulse can be estimated as the energy contained in the corresponding electromagnetic field, expressed in the Gaussian system of units:

$$\varepsilon \approx \frac{|E|^2}{4\pi}\sigma c\tau \tag{1.1}$$

where c is the speed of light and $\sigma c\tau$ is the volume occupied by the pulse.
Moreover, it's possible to define the intensity of the pulse as the energy per unit of area per unit of time:

$$I \approx \frac{|E|^2}{4\pi}c \tag{1.2}$$

The electromagnetic laser pulses mostly can be modeled, in the most simple way, as gaussian beams, which are beam of monochromatic electromagnetic radiation whose intensity envelope in the transverse plane follows a Gaussian distribution:

$$I\left(x,y,z,t\right) = I_0 \exp\left(-2\frac{\left(x^2+y^2\right)}{w\left(z\right)^2}\right)\exp\left(-\frac{\left(t-\frac{z}{c}\right)^2}{{w_t}^2}\right) \tag{1.3}$$

where z is the propagation direction, I_0 is the peak intensity of the pulse, $w\left(z\right)$ is the waist – the radius at which the intensity fall to $1/e^2$ and w_t is described by the relation:

$$w_t = \frac{FWHM}{\sqrt{\ln 2}},$$

where FWHM is the full-width-half-maximum of the intensity envelope in time.

Today, many laser technologies exist, which generate laser pulses with different characteristics (see [4]). Concerning high power lasers, three main technologies exist: Titanium:Sapphire lasers, CO_2 lasers and Nd:YAG (neodymium-doped yttrium aluminum garnet, Nd:$Y_3Al_{15}O_{12}$) lasers.
Ti:Sapphire lasers can produce pulses with duration down to tens of fs (10^{-15} s), whereas durations in the ps (10^{-12} s) and hundreds of fs ranges can be obtained with CO_2 and Nd:YAG lasers respectively. Moreover, the pulse generated by Ti:Sapphire technology has a wavelength of approximately 0.8 μm, while CO_2 and Nd:YAG generate pulses close to 1 μm and 10 μm respectively.
Today, laser-driven ion acceleration experiments typically employs solid state lasers based on the Titanium:Sapphire technology, which can generate pulses with durations about tens of fs transporting tens of J, focused to a few wavelengths spot sizes. This implies powers over of 1 PW (10^{15} W) and intensities up to 10^{22} W/cm^2.
These ultra-intense laser system are based on the *Chirped Pulse Amplification* (CPA), a technique introduced in the mid-1980s by Donna Strickland and Gérard Mourou, work for which they received the Nobel Prize in Physics in 2018 (see [5]). Essentially, the CPA is a technique which allow to amplify a short, low-energy laser pulse. A conceptual scheme of its principle is described in figure 1.1. Initially, the laser pulse undergoes a stretching and

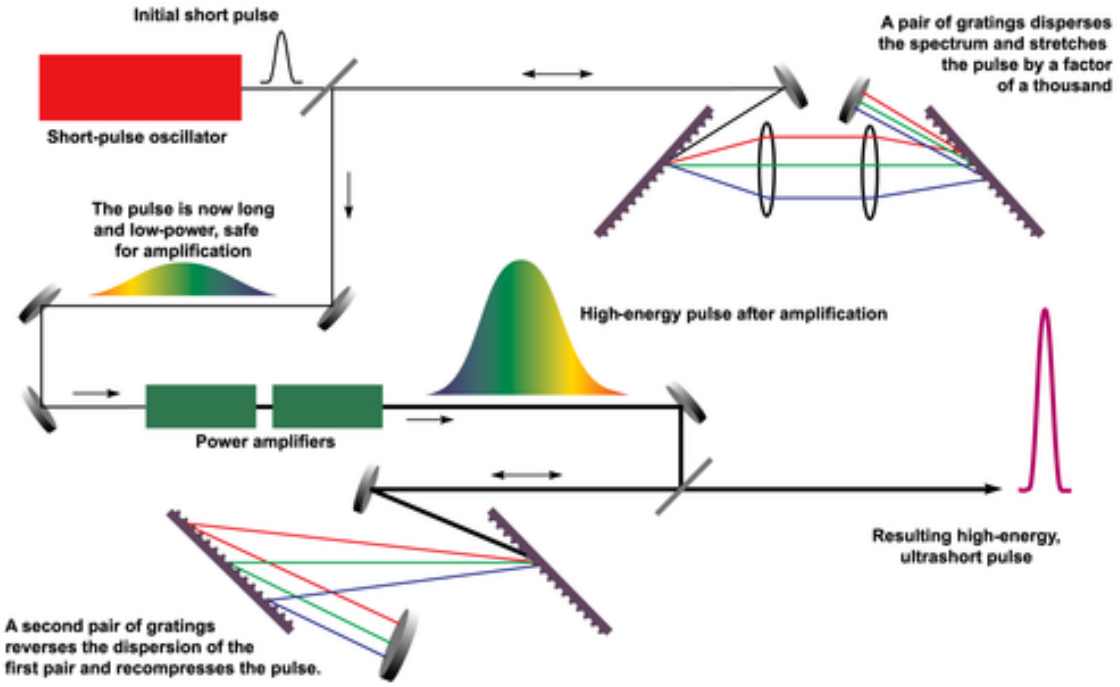

Figure 1.1: Conceptual scheme of the Chirped Pulse Amplification technique

its duration is increased by a dispersion in time of the spectral components.

Then, the resulting stretched pulse has lower power and it is easier and safer to amplify: so, the pulse is subjected to undergoes different different amplification stages, which increment its energy by several orders of magnitude.
Finally, the amplified laser pulse is recompressed back to the original width, achieving a final power which laser systems could generate before the invention of CPA.

1.2 Plasma generation

When a laser pulse interacts with matter it can produce ionization effects: in fact, if an intense pulse impinges on an atom, it can distort the Coulomb potential and release the electrons. So, if the electric field carried by the laser pulse is comparable with the atomic binding field of electrons, matter can be strongly ionized with the consequent plasma generation (the description of the plasma state is presented in next section).
Hence, if the laser is intense enough, matter ionization can occur before the arrival of the intensity peak, and therefore the laser peak interacts always with plasma.
Basically, when an oscillating electromagnetic wave interacts with a plasma, electrons react over faster timescales with respect to the ions, because of the higher charge-to-mass ratio (so ions are restrained by the greater inertia).
In general, when an electron is affected by an incident plane wave, that can be represented by its potential vector

$$A(\mathbf{r},t) = \mathrm{Re}\{A_0 e^{i\varphi}\} = A_0 \cos\phi$$

it begins oscillating around its equilibrium position due to Lorentz force with a quiver velocity

$$v_q = \frac{eA_0}{m_e c}. \tag{1.4}$$

In addition, for high values for the laser intensity, relativistic effects must be taken in consideration in the dynamics of the electrons: in this case it is essential to introduce another parameter, the normalized vector potential

$$a_0 = \frac{eA_0}{m_e c^2} = \frac{v_q}{c} = \sqrt{\frac{e^2\lambda^2 I_0}{2\pi m_e c^3}} = 0.85\sqrt{\frac{\lambda_{\mu m}^2 I_0}{10^{18} W cm^{-2}}}. \tag{1.5}$$

Hence, it is easy to see that this quantity explicates the relevance of relativistic effects in the electron dynamics: in fact $a_0 \cong 1$ means that the quiver velocity is near to c, so their kinetic energy is near to $m_e c^2 = 0.511$ MeV, which is the electrons rest energy, in a single laser cycle or, equivalently, the laser *irradiance* $\lambda^2 I_0 \cong 10^{18} W cm^{-2} \mu m^2$.
Summarizing, if an ultra-intense laser interacts with matter, it can produce plasma with relativistic effects, at least for what concerns electron dynamics.

1.3 The plasma state

"Except near the electrodes, where there are sheaths containing very few electrons, the ionized gas contains ions and electrons in about equal numbers so that the resultant space charge is very small. We shall use the name plasma to describe this region containing balanced charges of ions and electrons."

The American Nobel Physicist Award Irving Langmuir introduced the *plasma* word in 1928 in the paper "*Oscillations in Ionized Gases*" ([6]), borrowing it from the Ancient Greek term $\pi\lambda\alpha\sigma\mu\alpha$, meaning "moldable substance", to describe a *"region containing balanced charges of ions and electrons*".
The plasma constitutes the 99.9% of the visible matter in the universe: the interior of the stars, interstellar space, solar wind, boreal aurora, ionosphere and lightning are made of plasma. In addition to these natural forms, plasma are reproduced in laboratory for specific scopes, such as neon tubes, plasma balls, electric arches, radiofrequency discharges for industrial applications, up to high temperature plasmas for controlled thermonuclear fusion research and laser generated plasmas.
It is different from the classical three states of matter (solid, liquid and gas) and, so, it is considerable as the fourth state of matter.
So, the plasma state requires a special definition all its own: as a first simplified approximation, it could be definable as a ionized gas, constituted by charged particles and globally neutral, meaning that the total electrical charge is null.
The concept of charge separation in the plasma state had already emerged in the previous section: namely, it has been explained that a high-power laser pulse can induce matter ionization until reaching plasma generation: indeed, in a plasma different species of charges coexist, the so called *populations*, that arise when electrons split from the rest of the nucleus (creating ions).
Although these particles are unbound, they are not "free" in the sense of not experiencing forces: the motion of the charges generates electromagnetic fields, and any motion of a charged particle affects and is affected by the fields created by the other charged particles. So a plasma, basically, is a many-body system of particles whose physical behavior is strongly affected by the presence of the charges and it requires a description which takes in account both the electromagnetic nature and the particle dynamics.

1.3.1 Fundamental parameters

There is a set of parameters to describe a plasma: a particularity of this state of matter is that its fundamental parameters can vary in a very large range (as it is easy to see in the table in figure 1.2).
So, plasmas can differ a lot from each other but it's possible to describe all of them with

the same theoretical framework.
Two fundamental parameters are the density n_a and the temperature T_a of each species: in particular, the latter represents an information on the thermal kinetic energy of the particles and so it is commonly measured in electronvolts [eV]. Concerning spatial parameters,

Plasma Type	n cm^{-3}	T eV	ω_{pe} sec^{-1}	λ_D cm	$n\lambda_D{}^3$	ν_{ei} sec^{-1}
Interstellar gas	1	1	6×10^4	7×10^2	4×10^8	7×10^{-5}
Gaseous nebula	10^3	1	2×10^6	20	8×10^6	6×10^{-2}
Solar Corona	10^9	10^2	2×10^9	2×10^{-1}	8×10^6	60
Diffuse hot plasma	10^{12}	10^2	6×10^{10}	7×10^{-3}	4×10^5	40
Solar atmosphere, gas discharge	10^{14}	1	6×10^{11}	7×10^{-5}	40	2×10^9
Warm plasma	10^{14}	10	6×10^{11}	2×10^{-4}	8×10^2	10^7
Hot plasma	10^{14}	10^2	6×10^{11}	7×10^{-4}	4×10^4	4×10^6
Thermonuclear plasma	10^{15}	10^4	2×10^{12}	2×10^{-3}	8×10^6	5×10^4
Theta pinch	10^{16}	10^2	6×10^{12}	7×10^{-5}	4×10^3	3×10^8
Dense hot plasma	10^{18}	10^2	6×10^{13}	7×10^{-6}	4×10^2	2×10^{10}
Laser Plasma	10^{20}	10^2	6×10^{14}	7×10^{-7}	40	2×10^{12}

Figure 1.2: Approximate magnitudes in some typical plasmas. Reprinted from the NRL Plasma Formulary [7].

instead, an important quantity is the *Debye length*, so defined

$$\lambda_D = \left(4\pi e^2 \sum_{a=1}^{N} \frac{n_{a,0} Z_a^2}{T_a} \right)^{-\frac{1}{2}}, \tag{1.6}$$

where $n_{a,0}, Z_a$ and T_a are respectively the density in the unperturbed configuration, the atomic number and the temperature of the *a-th* population. This quantity is important because is related to the collective behavior of the charges: in fact λ_D is the minimal spatial scale at which the plasma can be considered neutral and at which motion of each particle can be seen as independent from the other charges.
So a particle in a plasma resent of the Coulomb-interactions only with the particles whose distance is less than the Debye length, and all the others, out from the so called *Debye spehere* are negligible.
Hence, it's possible to see the entire *Debye spehere* as a unique body, called *quasi-particle*, that interacts with the other bodies of the system as a classical particle in a gas: the

number of particles in this sphere is clearly

$$N_D = \frac{4}{3}\pi\lambda_D^3 n = 1.72 \cdot 10^9 \frac{T^{\frac{3}{2}}}{n_0^{\frac{1}{2}}} \tag{1.7}$$

This number is useful to relate other quantities, which are the kinetic and potential energy of a particle in a plasma:

$$\varepsilon_k \simeq \frac{3}{2}T, \quad \varepsilon_{pot} \simeq \frac{e^2}{n_0^{-\frac{1}{3}}}$$

For convention, it's better to consider the inverse of this quantity, the *plasma parametere*

$$g = \frac{1}{N_D} \tag{1.8}$$

which is a measure of the ideality of the plasma: in fact the smaller is g, the shorter range interactions are negligible ($N_D \gg 1$) and the plasma is hot and rarefied.
At the limit $g = 0$, the entire system can be seen as an ideal gas, i.e. a gas composed by particles moving only due to their thermal energy and without inter-particle interactions.
Finally, another important parameter is the *plasma frequency*, here definied, which measures the harmonic oscillations of electrons

$$\omega_{p,e} = \sqrt{\frac{4\pi n_0 e^2}{m_e}}, \tag{1.9}$$

and it is a good approximation of the plasma frequency due to all the particles oscillations (being the electrons very light compared to protons or ions).

1.3.2 Mathematical modeling of the plasma

There is not a unique mathematical model to describe the plasma state, but it's possible to choose from several existing alternatives depending on the nature of problem considered and the level of details desired.
One of these is the *multifluid* description in which the identity of the individual particle is neglected, and only the average motion is considered, just like in a fluid. In this case, the plasma is seen as a mixture of fluids, one for each population, endued with electric currents.
This mathematical description, basically, consists of the *continuity equation* and the *Navier-Stokes-like* momentum law:

$$\begin{cases} \dfrac{dn_a}{dt} + n_a \nabla \cdot \mathbf{u}_a = 0 \\ m_a n_a \dfrac{d\mathbf{u}_a}{dt} = -\nabla \cdot \ P_a + q_a n_a \left(\mathbf{E}_a + \dfrac{\mathbf{u}_a}{c} \times \mathbf{B} + R_a \right) \end{cases}$$

where, n_a and u_a are the density and the velocity of the a-th fluid, P_a is the pressure related to the random motion of the particles in the fluid and R_a is a term representing the momentum exchanges caused by collisions with particles of other populations.

Moreover, when the length scales of the problem under consideration are larger than the Debye length and the time scales are larger than the inverse of plasma frequency, it is possible to neglect the charge separation. In this case it's useful to consider a further approximation, the so called *Magnetohydrodinamic model* which combines the equations of the multifluid model of the various populations in order to obtain new laws governing the plasma considered as a unique fluid.
However this model cannot be chosen to describe a plasma involving high-frequency phenomena, which may involve charge separation.

The two approaches presented, however, are very approximate and do not ensure a detailed description of the plasma state. However, when an high level of detail is required, a third approach must be considered, which is inherited from the kinetic theory and represents the most complete, solvable description.
Essentially, this approach consists in readjusting the kinetic theory of gases in order to incorporate the self-consistent electromagnetic interactions: so, starting from the microscopic information of the single particles, it is possible to use averaged quantities to describe the system with a *kinetic model*.

Relativistic collisionless kinetic description

Consider a non-relativistic plasma composed by N populations each one with mass m_a, charge q_a and N_a particles. The dynamics of a single particle i affected by the microscopic fields $\mathbf{E}_{micr}$ and $\mathbf{B}_{micr}$, is dominated by the equation of motion:

$$\begin{cases} \dfrac{d\mathbf{r}_{i,a}(t)}{dt} = \mathbf{v}_{i,a} \\ m_a \dfrac{d\mathbf{v}_{i,a}(t)}{dt} = q_a \left(\mathbf{E}_{micr}(\mathbf{r}_{i,a}, t) + \dfrac{\mathbf{v}_{i,a}}{c} \times \mathbf{B}_{micr}(\mathbf{r}_{i,a}, t) \right) \end{cases} \tag{1.10}$$

Moreover, the space-time evolution of the microscopic fields, expressed in the Gaussian system of units, is governed by the *Maxwell equations*

$$\begin{cases} \nabla \cdot \mathbf{E}_{micr} = 4\pi \rho_{micr} \\ \nabla \times \mathbf{E}_{micr} = -\dfrac{1}{c} \dfrac{\partial \mathbf{B}_{micr}}{\partial t} \\ \nabla \cdot \mathbf{B}_{micr} = 0 \\ \nabla \times \mathbf{B}_{micr} = \dfrac{4\pi}{c} \mathbf{J}_{micr} + \dfrac{1}{c} \dfrac{\partial \mathbf{E}_{micr}}{\partial t} \end{cases} \tag{1.11}$$

and, in order to close the system, it's convenient to explicit the contribuits of charge and current densities of external particles

$$\rho_{micr}\left(\mathbf{r},t\right)=\rho_{ext}+\sum_{a=1}^{N}q_a\sum_{i=1}^{N_a}\delta\left(\mathbf{r}-\mathbf{r}_{i,a}\right) \tag{1.12}$$

$$\mathbf{J}_{micr}\left(\mathbf{r},t\right)=J_{ext}+\sum_{a=1}^{N}q_a\sum_{i=1}^{N_a}\mathbf{v}_{i,a}\delta\left(\mathbf{r}-\mathbf{r}_{i,a}\right) \tag{1.13}$$

Hence, system (1.10) represents a Lagrangian description of the trajectory of a particle in time, while the second describes fields evolution in an Eulerian point of view.
It is important to underline that, while in the Lagrangian point of view the position is a function of time, in the Eulerian one space coordinates and the time are independent, since it's possible to fixate t and evaluate the fields at different points in space.
Hence, even though the two descriptions are physically equivalent, a full-Eulerian description would be more suitable. For this purpose, it is convenient to transform particle dynamic equations in an Eulerian form: to do so, the system needs to be described in the *phase-space*, the six-dimensional space $(\mathbf{r},\mathbf{v})$.
Now, consider the *microscopic distribution function* of the density of particles for the a-th species in the space-phase

$$f_{micr,a}\left(\mathbf{r},\mathbf{v},t\right)=\sum_{i=1}^{N_a}\delta\left(\mathbf{r}-\mathbf{r}_{i,a}(t)\right)\delta\left(\mathbf{v}-\mathbf{v}_{i,a}(t)\right) \tag{1.14}$$

Expressing the external sources in Maxwell equations (1.11) in terms of $f_{micr,a}$, as

$$\rho_{micr}=\rho_{ext}+\sum_{i=1}^{N_a}q_a\int f_{micr,a}(\mathbf{r},\mathbf{v},t)\,d\mathbf{v} \tag{1.15}$$

$$\mathbf{J}_{micr}=\mathbf{J}_{ext}+\sum_{i=1}^{N_a}q_a\int \mathbf{v}f_{micr,a}(\mathbf{r},\mathbf{v},t)\,dv \tag{1.16}$$

and, using the equation of motion, it's possible to derive the dynamic equation of the distribution function

$$\frac{\partial f_{micr,a}}{\partial t}+\nabla_{\mathbf{r}}\cdot\left(f_{micr,a}\mathbf{v}\right)+\nabla_{\mathbf{v}}\cdot\left(f_{micr,a}\frac{q_a}{m_a}\mathbf{E}_{micr}+\frac{\mathbf{v}\times\mathbf{B}_{micr}}{c}\right)=0 \tag{1.17}$$

This is the *Klimontovich equation*: it is physically equivalent to the motion equation.
At this point, it is convenient to look for a less detailed description. Let the distribution function f_a be the ensemble average of the microscopic distribution function and in the same fashion define the mean electromagnetic field $\mathbf{E}$, $\mathbf{B}$

$$\begin{cases} f_{micr,a}=\langle f_{micr,a}\rangle+\tilde{f}_a=f_a+\tilde{f}_a \\ \mathbf{E}_{micr}=\langle\mathbf{E}_{micr}\rangle+\tilde{\mathbf{E}}=\mathbf{E}+\tilde{\mathbf{E}} \\ \mathbf{B}_{micr}=\langle\mathbf{B}_{micr}\rangle+\tilde{\mathbf{B}}=\mathbf{B}+\tilde{\mathbf{B}} \end{cases} \tag{1.18}$$

This allows to decompose the quantity in a sum of an average (over all possible microscopic configurations that correspond to the same macroscopic state) and a fluctuating component ($\tilde{\mathbf{E}}$,$\tilde{\mathbf{B}}$,$\tilde{f}$).
This statistical operation represent a sort of filter for the spatial scales which keep only those characteristic of the problem.
Applying the same operation to the Maxwell equations (1.11) yields:

$$\begin{cases} \nabla \cdot \mathbf{E} = 4\pi\rho \\ \nabla \times \mathbf{E} = -\dfrac{1}{c}\dfrac{\partial \mathbf{B}}{\partial t} \\ \nabla \cdot \mathbf{B} = 0 \\ \nabla \times \mathbf{B} = \dfrac{4\pi}{c}\mathbf{J} + \dfrac{1}{c}\dfrac{\partial \mathbf{E}}{\partial t} \end{cases} \tag{1.19}$$

where

$$\rho = \rho_{ext} + \sum_{i=1}^{N_a} q_a \int f_a(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v} \tag{1.20}$$

$$\mathbf{J} = \mathbf{J}_{ext} + \sum_{i=1}^{N_a} q_a \int \mathbf{v} f_a(\mathbf{r}, \mathbf{v}, t)\, dv \tag{1.21}$$

Now, the application of the average operation to the Klimontovich equation leads to the so-called *Boltzmann equation*

$$\frac{\partial f_a}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{r}} f_a + \frac{q_a}{m_a}\left(\mathbf{E} + \frac{\mathbf{v} \times \mathbf{B}}{c}\right) \cdot \nabla_{\mathbf{v}} f_a = -\langle \frac{q_a}{m_a}\left(\tilde{\mathbf{E}} + \frac{\mathbf{v} \times \tilde{\mathbf{B}}}{c}\right) \cdot \nabla_{\mathbf{v}} f_a \rangle \tag{1.22}$$

The left hand side of (1.22) is related to the collective motion of relatively large volumes of particles through the mean self consistent field and expresses the macroscopic, averaged and long-range interactions.
Instead, the right hand side, commonly indicated as C_a, manifests the microscopic, chaotic and short-range collisions between particles: so it represents the creation and elimination of particles in the phase space. However, C_a, which is a priori unknown, needs to be addressed with a suitable mathematical model in order to close the system.
Nevertheless, when the time scales considered are shorter than the relaxation time of the plasma, which is the scale time over which the system returns to the equilibrium configuration after a perturbation, it's possible to assume that the collision effects do not influence the average behavior of the plasma and so C_a can be neglected.
This simplification leads to a closed system and introduces the *Vlasov equation*, holding for a non-relativistic collisionless plasma:

$$\frac{\partial f_a}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{r}} f_a + \frac{q_a}{m_a}\left(\mathbf{E} + \frac{\mathbf{v} \times \mathbf{B}}{c}\right) \cdot \nabla_{\mathbf{v}} f_a = 0 \tag{1.23}$$

Finally, it's possible to write a closed solvable system, known as the *Vlasov-Maxwell system*, whose unknown are the distribution function and the electric and magnetic fields:

$$\begin{cases} \dfrac{\partial f_a}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{r}} f_a + \dfrac{q_a}{m_a} \left(\mathbf{E} + \frac{\mathbf{v} \times \mathbf{B}}{c} \right) \cdot \nabla_{\mathbf{v}} f_a = 0 \\ \nabla \cdot \mathbf{E} = 4\pi\rho \\ \nabla \times \mathbf{E} = -\dfrac{1}{c} \dfrac{\partial \mathbf{B}}{\partial t} \\ \nabla \cdot \mathbf{B} = 0 \\ \nabla \times \mathbf{B} = \dfrac{4\pi}{c} \mathbf{J} + \dfrac{1}{c} \dfrac{\partial \mathbf{E}}{\partial t} \end{cases} \tag{1.24}$$

Collisionless relativistic kinetic description

All of the previous considerations, as premised at the beginning, hold if the particles do not show a relativistic behavior: nevertheless, in a laser-generated plasma, for example, electrons can reach relativist velocities. For these kind of situations, so, the kinetic description must be modified in order to consider relativist effects.
Obviously, only the component related to the particles dynamics must be corrected, because Maxwell equations hold even in relativistic regimes.
Hence, the motion equations become:

$$\begin{cases} \dfrac{d\mathbf{r}_{i,a}}{dt} = \mathbf{v}_{i,a} = \dfrac{\mathbf{p}_{i,a}}{m_a \gamma_{i,a}} \\ \dfrac{d\mathbf{p}_{i,a}}{dt} = q_a \left(\mathbf{E}_{micr} + \dfrac{\mathbf{p}_{i,a}}{m_a \gamma_{i,a} c} \times \mathbf{B}_{micr} \right) \end{cases} \tag{1.25}$$

where, $\mathbf{p}_{i,a}$ and $\gamma_{i,a}$ are the linear momentum and the Lorentz factor respectively, so defined:

$$\begin{cases} \mathbf{p}_{i,a} = m_a \gamma_{i,a} \mathbf{v}_{i,a} \\ \gamma_{i,a} = \sqrt{1 + \dfrac{p_{i,a}^2}{(m_a c)^2}} \end{cases}$$

Proceeding in a similar way as the non relativistic case, the relativistic Vlasov equation is obtained:

$$\frac{\partial f_a}{\partial t} + \frac{\mathbf{p}_a}{m_a \gamma_a} \cdot \nabla_{\mathbf{r}} f_a + \frac{q_a}{m_a} \left(\mathbf{E} + \frac{\mathbf{p}}{m_a \gamma_a c} \times \mathbf{B} \right) \cdot \nabla_{\mathbf{p}_a} f_a = 0 \tag{1.26}$$

where the linear momentum $\mathbf{p}_a$ and the Lorentz factor γ_a associated to the a-th species are defined by:

$$\begin{cases} \mathbf{p}_a = m_a \gamma_a \mathbf{v}_a \\ \gamma_a = \dfrac{1}{\sqrt{1 - \dfrac{v_a^2}{c^2}}} \end{cases} \tag{1.27}$$

Now, it is possible to write the relativistic correction of the Vlasov-Maxwell system:

$$\begin{cases} \dfrac{\partial f_a}{\partial t} + \dfrac{\mathbf{p}_a}{m_a\gamma_a} \cdot \nabla_{\mathbf{r}} f_a + \dfrac{q_a}{m_a}\left(\mathbf{E} + \dfrac{\mathbf{p}}{m_a\gamma_a c} \times \mathbf{B}\right) \cdot \nabla_{\mathbf{p}_a} f_a = 0 \\ \nabla \cdot \mathbf{E} = 4\pi\rho \\ \nabla \times \mathbf{E} = -\dfrac{1}{c}\dfrac{\partial \mathbf{B}}{\partial t} \\ \nabla \cdot \mathbf{B} = 0 \\ \nabla \times \mathbf{B} = \dfrac{4\pi}{c}\mathbf{J} + \dfrac{1}{c}\dfrac{\partial \mathbf{E}}{\partial t} \end{cases} \tag{1.28}$$

1.4 Laser-plasma interaction

A fundamental issue in the physics of laser-plasma interaction is whether pulse can propagate through a given plasma or not: this section explains the basic elements of laser-plasma interaction, showing the different regimes of interaction with can occur.
As a first approach, we will start studying a very simplified case, far from the realistic phenomenon, of a single particle hit by a laser pulse.

1.4.1 Single particle approach

Consider a single non-relativistic electron interacting with a laser pulse with frequency ω propagating along the direction $\mathbf{k}$, represented by the electromagnetic field

$$\mathbf{E}(\mathbf{x}, t) = \mathbf{E}_0(\mathbf{x}, t)\cos(\mathbf{k} \cdot \mathbf{x} - \omega t) \tag{1.29}$$

So, the Newton equation describing the motion of the particle will be:

$$\ddot{\mathbf{x}} = \frac{q}{m}(\mathbf{E} + \frac{\dot{\mathbf{x}}}{c}\mathbf{v} \times \mathbf{B}) \tag{1.30}$$

It can be meaningful to look for an approximate solution through a linearization of the equation: using a perturbative approach the solution is split in a sum of two contributions

$$\mathbf{x} = \mathbf{x}^{(0)} + \mathbf{x}^{(1)}$$

where $\mathbf{x}^{(0)}$ is the solution at the 0-order of the linearized problem and $\mathbf{x}^{(1)}$ correspond to the perturbation of the system.
Now, expanding the electric field around the initial position of the particle $\mathbf{x}_c$

$$\mathbf{E} \approx \mathbf{E}(\mathbf{x}_c) + \left[(\mathbf{x} - \mathbf{x}_c) \cdot \nabla\right] \mathbf{E}|_{\mathbf{x}=\mathbf{x}_c}$$

and linearization allows to arrive at

$$\mathbf{x}^{(0)} = \mathbf{x}_c - \frac{q\mathbf{E}_0(\mathbf{x}_c)}{m\omega^2}\cos(\mathbf{k} \cdot \mathbf{x}_c - \omega t) = \mathbf{x}_c - \frac{\mathbf{v}^{(0)}}{\omega}\cos(\phi) \tag{1.31}$$

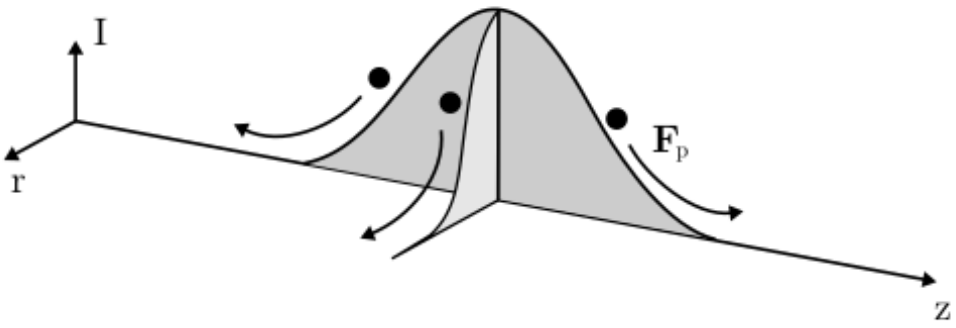

Figure 1.3: Conceptual scheme of the ponderomotive force effects. Reprinted from [8]

where $\mathbf{v}^{(0)}$ is the quiver velocity (1.4).
Thus, it's immediate to derive the perturbative the force acting on the particle

$$\ddot{\mathbf{x}}_{(1)} = \left(\frac{q}{m\omega}\right)^2 [(\mathbf{E}\cdot\nabla)\mathbf{E} + \mathbf{E}_0 \times (\nabla \times \mathbf{E}_0)\sin^2\phi + (\dots)\sin\phi\cos\phi] \tag{1.32}$$

As a result, this perturbative result is written as a sum of several contributions: the first term is a function only of the spatial dishomogeneities while the last two terms depend also on time, oscillating with frequency 2ω.
At this point, an average over the laser period $T = \frac{2\pi}{\omega}$ allows to get rid ot the two last terms

$$m\frac{d^2\langle \mathbf{x}^{(1)}\rangle}{dt^2} = -\frac{q^2}{4m\omega^2}\nabla|\mathbf{E}_0|^2 = \mathbf{F}_p \tag{1.33}$$

This is the so called *Ponderomotive force*: nonlinear force which, because of the non-uniformities in the spatial profile of the field, causes the particle to move towards the area of the smaller field amplitude (see figure 1.3).
It's meaningful to notice that this force is inversely proportional to the mass of the charge: this means that only the electrons feel significantly its effects while protons and ions displacement, in comparison, is negligible. Equivalently, it's possible to say that the ponderomotive force move the electrons towards regions with smaller *ponderomotive potential*

$$U_p = \frac{q^2}{4m\omega^2}|\mathbf{E}_0|^2 \tag{1.34}$$

However, this formulations doesn't take in account relativistic effects, which come into play when high-intensities lasers are considered: in this case the relativistic correction of the ponderomotive force yelds to

$$F_p = -mc^2\nabla\sqrt{1 + \langle \mathbf{a}^2\rangle} \tag{1.35}$$

where $\mathbf{a} = \dfrac{q\mathbf{A}}{mc^2}$ is the normalized vector potential.
As premised in the beginning, this single particle approach is a very simplified and not

realistic physical configuration: however it is useful to clarify the importance and the role of the ponderomotive force. In fact, despite all the simplifications, it allows to build qualitative arguments to understand why it is possible to accelerate particles through laser-plasma interactions: the interpretation of the ponderomotive effect explains how the laser profile can affect the density of a plasma, pushing and piling up the electrons according to its intensity profile.

1.4.2 Electromagnetic radiation-plasma interaction

The previous section was devoted to describe the effects felt by a single particle hit by a laser pulse. Otherwise, a plasma is composed by a lot of particles, so it is required a generalization of the previous concept to a full plasma.
In particular, a laser pulse shot into a dispersive medium, such as plasma, can lead to several kinds of perturbations inside the medium. So, it is important to understand which kind of relations (between lasers and plasmas) allow to obtain propagating effects.
In general a plane monochromatic electromagnetic wave propagating within a plasma must satisfy Maxwell laws together with Kramers-Kronig relations (see Jackson [9]): under suitable hypothesis, it's possible to arrive to simple conditions relating the wave frequency ω the wave vector $\mathbf{k}$.
First of all, neglect spatial dispersion: it could be a good approach if the spatial variations of the wave are greater than the characteristic lengths of the plasma, i.e the Debye length:

$$\lambda_D \ll \frac{1}{k} \Leftrightarrow v_t \ll \frac{\omega}{|\mathbf{k}|}$$

where $v_t = \sqrt{\frac{T}{m}}$ is the thermal velocity. So neglecting spatial dispersions is equivalent to neglecting *thermal effects*: this is the so called *cold plasma approximation.*
Thus,consider a cold, homogeneous, collisionless, non-relativistic, non-magnetized plasma: thinking the incident electromagnetic wave as a small perturbation and assuming linear response to small perturbations, it is possible to achieve the dispersion relations, i.e. the conditions under which the wave is free to propagate inside the plasma.
In particular, in the case of transverse wave this relation is given by:

$$\omega^2 = \omega_p^2 + |\mathbf{k}|^2 c^2 \tag{1.36}$$

while, if the wave is longitudinal, the condition has a more simple form:

$$\omega = \omega_p \tag{1.37}$$

So, this is a minimum-value-condition for the propagation: if $\omega \geq \omega_p$ wave propagations occurr.
If, instead, $\omega < \omega_p$ the wave is partially refecleted and partially absorbed by the plasma. The penetrating component travels through the surface of the plasma for a very small

distance, called *skin depth length*, before being exponentially dumped.
This distance is related both to ω and ω_p by the formula

$$L = \frac{c}{\sqrt{\omega_p^2 - \omega^2}} \tag{1.38}$$

but, if $\omega \ll \omega_p$, $L \simeq \frac{c}{\omega_p}$ becomes a property of the plasma itself.

Regimes of interaction

It's a common procedure rewriting the dispersion relations in terms of the *critical density*, which is the density at which $\omega = \omega_p$

$$n_c = \frac{m_e \omega^2}{4\pi e^2} = 1.1 \times 10^{21} cm^{-3} \left(\frac{\lambda}{1\mu m}\right)^{-2}$$

where $\lambda = \frac{2\pi c}{\omega}$ is the wavelength. Thus, the above considerations to the frequencies can be transferred into considerations on the densities. Given a monochromatic wave and a plasma, three different regimes can be distinguished:

- $n_e < n_c(\omega)$ *undercritical plasma*: the wave can propagate
- $n_e > n_c(\omega)$ *overcritical plasma*: there is no propagation
- $n_e \approx n_c(\omega)$ *near-critical plasma*: intermediate complex situation where a strong wave-plasma coupling occurs

However, these results have been obtained under the hypothesis if small perturbations; however, if this assumption does not hold, non-linear effects may arise. So, when relativistic effects are included, a non-linear treatment has to be adopted: this happens when high-power lasers are involved, which, interacting with plasmas, induce relativistic effects.
In this case, the underlying physics become more complex and a relativistic treatment is required: so, once introduced the relativistic factor

$$\gamma = \sqrt{1 + \frac{a_0^2}{2}}$$

substituting m_e with γm_e the relativistic correction appears:

$$n_c^{rel} = \gamma n_c = \gamma \frac{m_e \omega^2}{4\pi e^2} \tag{1.39}$$

Equation (1.39) shows that, if $\gamma > 1$, the maximum threshold and so propagation can occur: this is the so called *relativistic self-induced transparency*, thanks to which the range $n_c < n_e < \gamma n_c$ is still valid for waves propagation.

1.4.3 Laser-driven particle acceleration

Laser-plasma interaction can be exploited in order to obtain accelerated particles: exploiting ultra-intense and ultra-short pulses (with intensity greater than $10^{18} W/cm^2$), it is possible to rapidly ionize a target and to achieve a plasma state.
At this point, the laser-plasma interaction can excite the particles contained in the plasma, inducing an acceleration. There exist scenarios of particle accelerations, depending, basically, on the regime of interaction under consideration.
Firstly, consider the undercritical regime: in this case the pulse is able to propagate through the considered plasma for long distances. Traveling across matter, the pulse acts on the electrons via the ponderomotive force, described by (1.33).
So, the electrons are pushed away from the region where the pulse is located towards regions where the ponderomotive potential, defined in (1.34), is lower. Therefore, electrons that have not met the pulse yet are moved forward along the laser propagation direction. Meanwhile, the pulse, which moves further at the speed of light, can overtake those electrons. So, the electrons are pushed back and forward by the laser.
Hence, this particular kind of interaction can produce a collective oscillation of the electrons, i.e. a plasma wave, consisting in a wake of oscillating electrons, generated behind the propagating pulse. Moreover, for proper laser wavelengths, the oscillation may be resonant. This phenomena is called *wakefield generation* and is crucial for laser-based electron acceleration (for more details on this topic see [10]). Basically, if resonance occurs, huge electric fields arise within the plasma increasing up to a maximum value at which the so-called *wavebreaking* occurs. In this scenario, part of the oscillating electrons are trapped in the regions close to the maximum of the field and are effectively accelerated. A conceptual scheme of this phenomena is reported in figure 1.4.

A different scenario appears when the overcritical regime is involved. Consider a thin

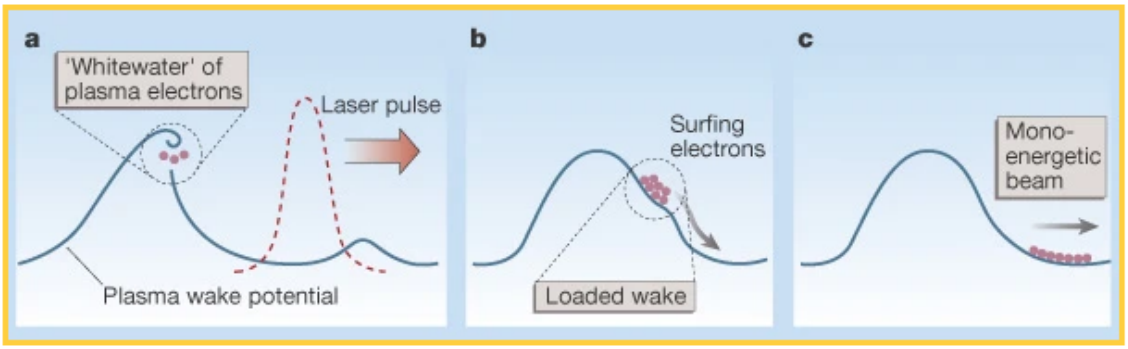

Figure 1.4: Wakefield acceleration. Reprinted from [11]

solid material impinged by laser pulse: in this case, as described previously, this target behaves like a sort of mirror reflecting backward the laser pulse.
So, the interaction between laser and plasma occurs only at the superficial level: the elec-

trons populating the front side of the plasma can gain energy form the laser pulse through many different mechanisms.
Then, the resulting laser-plasma coupling induces a strong charge separation which, in turn, generates intense electric fields in the longitudinal direction of the pulse; finally, these longitudinal electric fields are responsible for the ion acceleration process.
In the last twenty years, several laser-based acceleration mechanisms have been studied, focusing on enhancing both the number of accelerated particles and the achieved velocities ([12], [13],[14],[15],[16]): one of the most studied and understood is the *Target Normal Sheath Acceleration* (TNSA), first proposed by Wilks *et al.* [17].
The basic idea is the following. A high-intensity laser impinges on a μm-thick solid target: the pulse interacts with the electrons located in the front of the target, which are excited by the energy of the laser and expand at relativistic energies, turning into *fast* (or *hot*) electrons.
However, only electrons displacement occurs because today's laser facilities are not powerful enough to win also ions' inertia, which is much greater inertia due to their mass. So, fast electrons gain enough kinetic energy to cross the bulk and to reach the rear of the target, forming a cloud of relativistic electrons, which extends out of the target for a few Debye lengths.
Thereupon, the huge amount of relativistic electrons displaced generates an extremely intense electric field (of the order of $\mathrm{MV}/\mu m$), called *sheath field*, propagating along the normal direction and rapidly decaying outside the target after few μs. The generation of the sheath field depends both on the form of the electron distribution and on the density profile of the surface.
This entire process causes ions acceleration, which occurs perpendicularly to the surface. Clearly, this acceleration is most effective on protons (up to tens MeV), coming from the ionization of the impurities located on the rear surface, thanks to their small charge-to-mass ratio. Anyway, other kinds of heavier ions can be accelerated to similar values, on longer time scales, if protons are not numerous enough to balance the charge of the escaping hot electrons, and especially if impurity protons have been removed before the interaction (for example, by preheating the target).

It's possible to exhibit rough estimations of the intensity of the sheath field and the peak energy achieved by accelerated ions using simply dimensional considerations: consider as characteristic parameters to describe the problem the length of the hot electrons cloud L_c and the hot electrons density n_h and temperature T_h.
First of all, T_h can be estimated, starting from the ponderomotive effects felt by the hot electron population, as the kinetic energy contained by the electric field of a laser in vac-

uum, averaged over one oscillation

$$T_h \sim \mu_h m_e c^2(\gamma - 1) = \mu_h m_e c^2 \left(\sqrt{1 + \frac{a_0^2}{2}} - 1 \right)$$

where μ_h is the fraction of laser energy absorbed by hot electrons.
Now, as mentioned before, it's empirically observed that the hot electrons travel for a few Debye length beyond the rear of the target, so the size of the electron cloud is estimated to be of the order of λ_D

$$L_c \sim \lambda_D = \sqrt{\frac{T_h}{4\pi e^2 n_h}}$$

Thereupon, the sheath field can be approximated with the ratio

$$E_s \sim \frac{T_h}{eL_c}$$

and the peak energy of a ion immersed in this field is estimated as

$$\varepsilon \sim ZeE_sL_c \sim ZT_h$$

So, consider the following typical experimental values:

- $\mu_h = 0.1$
- $n_h \sim 10^{23} cm^{-3}$
- Laser irradiance $I_0\lambda^2 \sim 10^{20} W/cm^2\mu m^2 (i.e. a \sim 8.5)$

the following values are obtained

- $L_c \sim \mu m$
- $T_h \sim$ MeV
- $E_s \sim 1$ MV/ μm
- $\varepsilon \sim MeV, \quad \varepsilon \sim \sqrt{I}$

In addition, other theoretical models exist for the TNSA mathematical description and they lead to more sophisticated power laws of ε.

By the way, there are several strategies to enhance the acceleration scheme: one option consists in enhancing the number and the cut-off energy of accelerated ions increasing the rate of laser absorption.
In this regard, a large body of research have been carried out on the design of the irradiated target, in order to find the structure which maximize the acceleration. In particular, many studies have noticed that the laser-plasma coupling is largely improved in the near-critical

regime, i.e. when the plasma electron density is close to the critical density ([18], [19], [20]): considered the typical experimental values $\lambda \simeq 0.8 \div 1$, this values is approximately

$$n_e \sim n_c = \frac{m_e \omega^2}{4\pi e^2} = 1.1 \times 10^{21} cm^{-3} \left(\frac{\lambda}{1\mu m}\right)^{-2} \simeq 1.1 \div 1.5625 \times 10^{21} cm^{-3}$$

with a corresponding mass density of few mg/cm^3.
Experimentally, producing these kind of plasmas is already an issue and, so, there are very few concrete options. One possible choice is represented by *nanostructured* low-density targets, i.e. materials with a complex density profile characterized by non-homogeneity on the nm-μm scale. So, thanks to their inherent non-homogeneity, nanostructured plasmas allow a deep propagation of the pulse, resulting in a better conversion of laser energy into plasma kinetic energy.
Hence, today an important issue in laser-plasma field is the research of an accurate mathematical modeling of both nanostructured plasmas and their interaction with lasers. However, some numerical investigations have been carried out in order to study the behavior of nanostrucured targets impinged by ultra-intense pulses and they have confirmed the enhancement in energy conversion with respect to homogeneous materials ([21]),[22], [23]). Moreover, a a promising strategy emerging consists the exploitation the laser-plasma coupling within the TNSA mechanism using a Double-Layer Targets (DLT), which can further increase both the number of accelerated ions and their energy, achieving an enhanced TNSA regime. A visual illustration of this scheme is given by figure 1.5, consisting in an over-critical solid target with a near-critical layer, attached on the irradiated side. Thus,

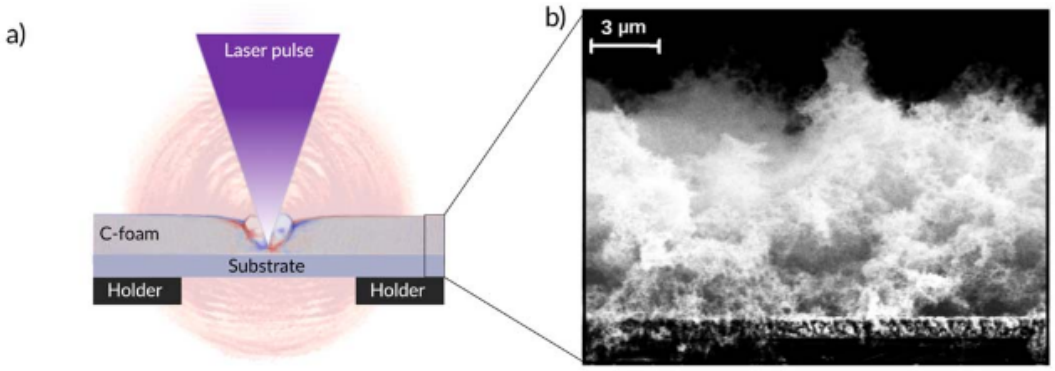

Figure 1.5: a) Double Layer Target conceptual scheme. b)DLT Scanning Electron Microscopy cross section view. Taken from [24]

advances in laser technology and the development of optimized DLTs targets can lead to a compact, high-repetition rate laser-driven acceleration. Several investigations have been conducted in order to verify these expectations and the results confirm that a very greater yield can be achieved through the DLT technology (see figure 1.6). At this regard, material science offers opportunities for manufacturing targets with low and controlled density, such

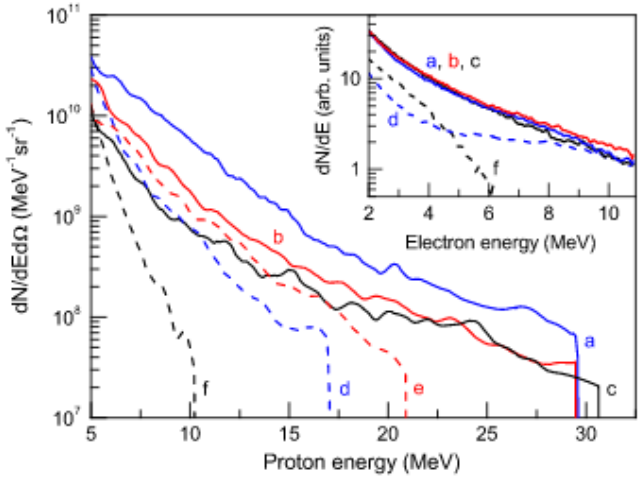

Figure 1.6: Energy spectra of the protons obtained with a laser intensity of 4.1, 3.5, 3.7 × 10 20 W/cm 2 for S, P and C polarization respectively. The targets are double-layer targets with 8 μm- thick foam (spectra a, b, c respectively) and single targets (spectra d, e and f). The spectra are collected along the target normal direction. The inset shows the electron energy spectra when using double-layer targets with 12 μm-thick foam with S, P and C polarization, (spectra a, b, c respectively) and using single targets for S and C polarization (spectra d and f). Reprinted from [23].

as foam targets.
Consequently, the resulting challenge relies on the possibility to assemble a thin solid foil with a nanostructured near-critical density layer, attached on the irradiated side.

The ongoing H2020 ERC project ENSURE ("Exploring the New Science and engineering unveiled by Ultraintense ultrashort Radiation interaction with mattEr") at the Micro and Nanostructured Materials Lab, Department of Energy, Politecnico di Milano, focuses on the experimental and numerical investigations of laser-driven ion acceleration with the nanostructured, foam-attached targets described above. The present thesis work has been carried out within the framework of the same project.

Chapter 2

Numerical approaches to the kinetic description of laser-plasma interaction

In chapter 1 we have seen how the Vlasov-Maxwell system can describe the kinetic effects of relativistic collisionless plasma. However, this model is way too complex to be solved analytically, thus numerical schemes for their approximation must be considered: this chapter is focused in summing up the numerical approaches, which can be adopted for the resolution of this system.

Now, a brief review of the system of equations describing a plasma is developed, making use of the kinetic approach, and of the corrections necessary in order to be equipped with a relativistic description.

Consider a relativistic collisionless plasma composed by N populations of different charges: as talk at length in the previous chapter, an Eulerian description for both the particle dynamics and the electromagnetic fields can be achieved, expressed by the *relativistic Vlasov-Maxwell system*:

$$\begin{cases} \dfrac{\partial f_a}{\partial t} + \dfrac{\mathbf{p}_a}{m_a \gamma_a} \cdot \nabla_{\mathbf{r}} f_a + \dfrac{q_a}{m_a} \left(\mathbf{E} + \dfrac{\mathbf{p}}{m_a \gamma_a c} \times \mathbf{B} \right) \cdot \nabla_{\mathbf{p}_a} f_a = 0 \\ \nabla \cdot \mathbf{E} = 4\pi\rho \\ \nabla \times \mathbf{E} = -\dfrac{1}{c} \dfrac{\partial \mathbf{B}}{\partial t} \\ \nabla \cdot \mathbf{B} = 0 \\ \nabla \times \mathbf{B} = \dfrac{4\pi}{c} \mathbf{J} + \dfrac{1}{c} \dfrac{\partial \mathbf{E}}{\partial t} \end{cases} \tag{2.1}$$

where ρ and $\mathbf{J}$ are given by

$$\rho = \rho_{ext} + \sum_{a=1}^{N} q_a \int f_a(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v} \tag{2.2}$$

$$\mathbf{J} = \mathbf{J}_{ext} + \sum_{a=1}^{N} q_a \int \mathbf{v} f_a(\mathbf{r}, \mathbf{v}, t)\, dv \tag{2.3}$$

2.1 Phase-space grid methods

In this section we will show the basic concepts of the phase-space grid methods used for the numerical resolution of Vlasov equation and we will discuss their advantages and limitations. Codes implementing this kind of methods are often called *Vlasov codes*: namely they find a numerical solution only for the Vlasov equation, while Maxwell equations need to be solved separately using other algorithms. The main idea of phase-space grid methods is, as their name suggests, to discretize the distribution function $f(\mathbf{r}, \mathbf{v}, t)$ in a grid of phase-space: this is a six-dimensional space and, so, these methods are extremely expensive in terms of computations in $3D3V$ simulations (as will be discussed later) and, indeed, their applications are often limitated to low-dimensional problems simulations.
In the following, some concrete aspects of phase-space grid methods will be briefly examined.

Time splitting

An original technique, proposed for the first time by Cheng and Knorr for the electrostatic case [25] and later recovered by Cheng for the magnetized case [26], consists in splitting the Vlasov equations at each time step into two advection equations.
The distribution function is evolved from time t to time $t + \Delta t$ in four steps

- Solve for half a time step
$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{r}} f = 0 \tag{2.4}$$
with $\mathbf{v}$ fixed.
- Solve the Maxwell equation 1.19
- Integrate for a whole time step
$$\frac{\partial f}{\partial t} + \frac{\mathbf{F}(\mathbf{r}, \mathbf{v}, t)}{m} \cdot \nabla_{\mathbf{v}} f = 0 \tag{2.5}$$
with fixed $\mathbf{r}$.
- solve again 2.4 for half a time step.

Then, the advection field of 2.4 is independent of the advection variable, so it's possible to solve explicitly it. However, the advection field of the 2.5 is not independent of the

advection variable in general, but under suitable hypothesis, such as the electrostatic case,it becomes independent and it gets explicit to solve.

Semi-Lagrangian methods

Semi-Lagrangian methods exploit the conservation of the distribution function

$$\frac{df(\mathbf{r}(t),\mathbf{v}(t),t)}{dt} = 0 \tag{2.6}$$

along characteristics of the Vlasov equation, namely the solution of the system

$$\frac{d\mathbf{r}}{dt} = \mathbf{v}(t)\frac{d\mathbf{v}}{dt} = \frac{\mathbf{F}}{m}(\mathbf{r}(t),t) \tag{2.7}$$

Firstly, build a grid ($\mathbf{r}_i$,$\mathbf{v}_i$) on the phase-space and a partition of time $\{t^k\}$.
For the sake of simplicity, from now on, a compact notation is used: $\mathbf{x} = (\mathbf{r},\mathbf{v})$ summarizes the position and velocity Eulerian coordinates, whereas while $\mathbf{a}(\mathbf{x},t) = \left(\mathbf{v}(t), \frac{\mathbf{F}}{m}(\mathbf{r}(t),t)\right)$ represents the total advection field.
Hence, the characteristic system, can be rewritten as

$$\frac{d\mathbf{X}}{dt} = \mathbf{a}(\mathbf{X}(t),t) \tag{2.8}$$

The unique solution of 2.8, with initial condition $\mathbf{X}(s) = \mathbf{x}$ is indicated with $\mathbf{X}(t;\mathbf{x},s)$.
Now, it is possible to choose if computing f forward or backwards in time([27], [28],[29],[30],[31]).
The first method can be thought as a particle method where the distribution function is reconstructed on the phase-space grid at each moment, following forward in time the path of the characteristic, exploiting the conservation of the distribution function:

$$f(\mathbf{x}_i,t^{n+1}) = f(\mathbf{X}(t^{n+1};\mathbf{x}_i,t^n),t^n)$$

. In the backward method, instead, the characteristics are followed backwards in time: indeed, for each point, the origins of the characteristics ending at it are found and then f is computed

$$f(\mathbf{x}_i,t^{n+1}) = f(\mathbf{X}\left(t^n;\mathbf{x}_i,t^{n+1}\right),t^n)$$

In the general case, it is necessary to use a fixed-point or Newton method to follow the characteristics; moreover, since in general, $\mathbf{X}(t^{n+1};\mathbf{x}_i,t^n)$ and $\mathbf{X}\left(t^n;\mathbf{x}_i,t^{n+1}\right)$ does not coincide with any specific point of the grid, the distribution function is evaluated interpolating the values from the grid: so, to avoid too much dissipation, high order interpolation is needed: typically, cubic splines ([25]) or cubic Hermite with derivative transport ([32]) interpolation schemes are used.
In addition, a third option exists, known as the *conservative method*: the basic idea is to exploit the conservation of the average of f over one cell along characteristics

$$\frac{1}{V}\int_V f(\mathbf{r},\mathbf{v})\,d\mathbf{r}d\mathbf{v}$$

where V is the volume of the cell.
So, starting from (2.6), time splitting application yields to 6 one-dimensional advection equations, which can be written in conservative form:

$$\frac{\partial f}{\partial t} + \frac{\partial a(x,t) f}{\partial x} = 0$$

. Now, firstly, indicating with f_i^n the cell average of f at time step n, construct on every cell a high order polynomial function whose cell average is equal to f_i^n (for example using an interpolation method). Then, compute the origins of the characteristics ending at the grid points, i.e. backtrack each cell, solving the equation of the characteristics.
Finally, a projection step occurs consiting in computing cell average f_i^{n+1}) at the new time step $n+1$ using the conservation property:

$$\int_{x_{i-1/2}}^{x_{i+1/2}} f^{n+1}(x)\,dx = \int_{X(t^n;x_{i-1/2},t^{n+1}}^{X(t^n;x_{i+1/2},t^{n+1}} f^n(x)\,dx$$

where $f^n(x)$ is the high order reconstruction found in the first step. So, after a time-splitting scheme, this method allows to solve 1D advections successively for which the volumes are merely intervals which are fully determined by their end points (see [27] for details on this method).

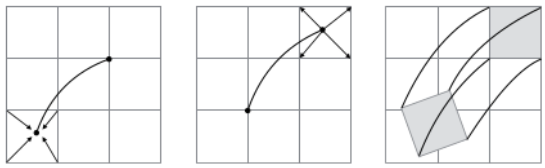

Figure 2.1: Backward semi-Lagrangian method (left), forward semi-Lagrangian method (middle) and finite volume (right). Taken from [27]

.

Spectral methods

Different spectral methods exist, depending on the kind of transform chosen for the physical and the velocity space. A possible strategy could be Fourier transforming both the physical and the velocity space, as proposed in [33] and [34] for the electrostatic 1D1V case.
Grant and Feix in [35], instead, proposed a different strategy consisting in apply a Fourier transform for the physical space and a Hermite transform for the velocity one.
However, some difficulties arise in this kind of methods, such as code parallelization, boundary conditions definition and prevention of the filamentation, so they not represent the best choice.

In this section a collection of the most famous phase-space grid methods was presented.
The great advantage of these methods is that they are not affected by numerical noise and guarantee a fine and uniform resolution of the phase-space, even in those regions where the distribution function takes very small values.
However, all of them show additional difficulties, due, in particular, to the filamentation phenomena in the phase-space which often occurs in non-linear Vlasov simulations. So, long time scale become impossible to simulate using a phase-space grid, because at some point the grid will not be able to solve the filamentations arising during the evolution of the distribution function: this implies numerical instabilities that can lead to non-positive values of the distribution function, which has no physical meaning.
For this reason, many times phase-space grid methods do not represent the best choice.

2.2 Particle-In-Cell

Particle-In-Cell method is the most widely exploited numerical method in finding a numerical approximation to the solution of the Vlasov-Maxwell system.
This method began to gain popularity since the late 1950s when John Dawson and Oscar Buneman illustrated this algorithm in some papers regarding plasma physics numerical simulations ([36], [37], [38], [39]).
After that, the PIC method continued to be deeply studied in the decades and a large literature was build, enlighting both the mathematical ([40]) and the physical ([41]) aspects.

2.2.1 Particles and grid

The philosophy behind the PIC method consists in a combined Lagrangian-Eulerian approach able to deal with both the particle motion and the electromagnetic field. As seen in chapter 1, indeed, particles and fields are two concepts indissoluble: namely, the motion of each particle under the effects of the electromagnetic field (felt by the particle) can be summarized in a unique equation, which is the Klimontovich equation (1.17), through the introduction of the distribution function concept. So, the basic idea consists in following the charged particles of the plasma along their trajectories in a Lagrangian frame, whereas the electromagnetic fields are computed on a stationary Eulerian mesh grid.
The original innovation which makes feasible this mixed approach resides in the concept of *macro-particles* (also called $super - particles$): namely, the distribution function of each population is approximated using virtual numerical particles, each of which correspond to a large number of real and physical particles inside the plasma. Hence, since each macro-particle represents a bunch of particles, finite extension is assigned to them in the physical space, while the point-like nature is kept in the velocity space. This finds a concrete realization substituting the Dirac distribution of the particle positions with the more stretched

Shape function: basically, for a considered population represented by N_p super-particles, the discrete approximation of the distribution function results in

$$f(\mathbf{r},\mathbf{p},t) \approx f_h(\mathbf{r},\mathbf{p},t) = \sum_{p=1}^{N_p} w_p S(\mathbf{r}-\mathbf{r}_p(t))\delta(\mathbf{p}-\mathbf{p}_p(t))$$

where w_p is a macro-particle weight depending on the density associated to the cell it originates from

$$w_p = \frac{n_p(\mathbf{r}_p(t=0))}{N_p}$$

and $S(\mathbf{r}-\mathbf{r}_p(t))$ is the mentioned shape function.
So, as its name suggests, this function describes the shape of macro-particles: so it has to be an even function localized in a neighborhood of zero, and it must preserve the normalization to unity of the Dirac distribution

$$\int_{\mathbb{R}^3} S(\mathbf{r}-\mathbf{r}_p(t))\, d\mathbf{r} = 1$$

There exists several shape functions, consisting in piece-wise polynomials of different

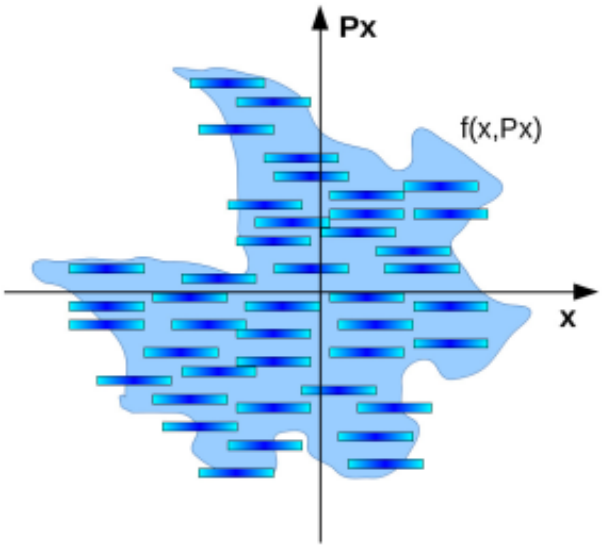

Figure 2.2: Sampling of the distribution function with macro-particles. Each macro-particle has a definite momentum, but is extended in space

.

orders. Typically the second order is chosen to approximate the macro-particles shape, since the first order shape function has a too sharp profile, while the third order has too long tails and are no indicated to approximate the distribution of the particles represented by a single macro-particle. If a 1D uniform grid with spacing h and points $i = 1, ..., N_x$ is

considered, the second order shape function can be written as

$$S^2(x_i - x) = S^2(u) = \frac{1}{h} \begin{cases} \frac{3}{4} - \left(\frac{|u|}{h}\right)^2 & |u| \leq \frac{1}{2}h \\ \frac{1}{2}\left(\frac{3}{2} - \frac{|u|}{h}\right)^2 & \frac{1}{2}h \leq |u| \leq \frac{3}{2}h \\ 0 & else \end{cases}$$

However, the PIC method allows to find a solution to the Vlasov-Maxwell system, without actually solving the Vlasov equation: indeed, this method bypasses the difficulties arising from Vlasov equation, which is a partial differential equation complex to solve, transforming the problem in a system of equations of motion for the macro-particles, which are ordinary differential equations. Thus, the equivalent continuous problem would be the following:

- Consider a plasma provided by N populations, each of them composed by $N_{p,a}$ macro-particles
- Let $\Omega \subset \mathbb{R}^3$ be the physical domain, let $V \subset \Omega$ the volume in which the plasma is contained
- Choose a proper shape function $S : \Omega \to \mathbb{R}$
- Each particle occupies an initial position $\mathbf{r}_{p,a,0}$ with a velocity $\mathbf{v}_{p,a,0}$
- The electric and magnetic fields at the beginning are given by $\mathbf{E}_0, \mathbf{B}_0 : \Omega \to \mathbb{R}^3$

Hence, find:

- $\mathbf{r}_{p,a}$ and $\mathbf{v}_{p,a} : (0,T) \to \mathbb{R}^3$ for each super-particle
- $\mathbf{E}$,$\mathbf{B} : \Omega \times (0,T) \to \mathbb{R}^3 \quad \forall \mathbf{r} \in \Omega \quad \forall t \in (0,T)$

such that:

$$\begin{cases} \frac{d\mathbf{r}_{p,a}(t)}{dt} = \mathbf{v}_{p,a}(t) \quad \text{in } (0,T), \forall p = 1 : N_{p,a}, \forall a = 1 : N \\ \frac{d\mathbf{p}_{p,a}(t)}{dt} = \tilde{\mathbf{F}}_{p,a}(t) \quad \text{in } (0,T), \forall p = 1 : N_{p,a}, \forall a = 1 : N \\ \nabla \cdot \mathbf{E}(\mathbf{r},t) = 4\pi\rho(\mathbf{r},t) \quad \text{in } \Omega \times (0,T) \\ \nabla \times \mathbf{E}(\mathbf{r},t) = -\frac{1}{c}\frac{\partial \mathbf{B}(\mathbf{r},t)}{\partial t} \quad \text{in } \Omega \times (0,T) \\ \nabla \cdot \mathbf{B}(\mathbf{r},t) = 0 \quad \text{in } \Omega \times (0,T) \\ \nabla \times \mathbf{B}(\mathbf{r},t) = \frac{4\pi}{c}\mathbf{J}(\mathbf{r},t) + \frac{1}{c}\frac{\partial \mathbf{E}(\mathbf{r},t)}{\partial t} \quad \text{in } \Omega \times (0,T) \\ \mathbf{r}_{p,a}(0) = \mathbf{r}_{p,a,0} \quad \forall p = 1 : N_{p,a}, \forall a = 1 : N \\ \mathbf{v}_{p,a}(0) = \mathbf{v}_{p,a,0} \quad \forall p = 1 : N_{p,a}, \forall a = 1 : N \\ \mathbf{E}(\mathbf{r},0) = \mathbf{E}_0(\mathbf{r}) \quad \text{in } \Omega \\ \mathbf{B}(\mathbf{r},0) = \mathbf{B}_0(\mathbf{r}) \quad \text{in } \Omega \\ + \quad \textit{boundary conditions on } \partial\Omega \end{cases}$$

where

$$\mathbf{p}_{p,a}(t) = m\gamma_{p,a}(t)\mathbf{v}_{p,a}(t), \qquad \gamma_{p,a}(t) = \sqrt{1 + \frac{|\mathbf{p}_{p,a}(t)|^2}{(m_a c)^2}}$$

$$\tilde{\mathbf{F}}_{p,a}(t) = q_a \int_\Omega S(\mathbf{r} - \mathbf{r}_{p,a}(t)) \Big[\mathbf{E}(\mathbf{r},t) + \frac{\mathbf{v}_{p,a}(t) \times \mathbf{B}(\mathbf{r},t)}{c}\Big] d\mathbf{r}$$

$$\mathbf{J}(\mathbf{r},t) = \frac{1}{V}\sum_{a=1}^{N} q_a \sum_{p=1}^{N_{p,a}} w_{p,a} S(\mathbf{r} - \mathbf{r}_{p,a}(t))\mathbf{v}_{p,a}(t), \qquad \rho(\mathbf{r},t) = \frac{1}{V}\sum_{a=1}^{N} q_a \sum_{p=1}^{N_{p,a}} w_{p,a} S(\mathbf{r} - \mathbf{r}_{p,a}(t))$$

Each macro-particle has the same mass and the same charge of the real particles in the plasma; moreover, the total number of macro-particles $N_p = \sum_{a=1}^{N} N_{p,a}$ is chosen a priori: in general is several orders of magnitude lower than the real number of physical particles in the plasma, but, at the same time, a very small number cannot be chosen without leading to undesired numerical noises.

Actually, the system requires boundary conditions in order satisfy well-posedness. The most easy to implement are the *periodic boundary conditions* for both particles and fields, which means that when a particle exits from a side of the domain is assumed to reappear in the opposite side, with the same momentum, whereas fields are assumed to have the same values along the opposite sides. In addition, sometimes other kinds of conditions are chosen, according to the system simulated, also in order to avoid re-circulation effects which may arise if the simulation box is not big enough: in any case these alternatives imply more difficulties in the implementation, so often the best choice consists in enlarging the box domain.

2.2.2 The standard PIC loop

In the following, the underlying logic of this numerical method will be illustrated: basically, this procedure can be decomposed in a loop scheme. The current density field associated to the motion of the macro-particles is scattered on a mesh on the configuration space, on which the electric and magnetic fields are then computed. From the grid values of the electromagnetic field, the Lorentz force acting on every particle is calculated by interpolation, hence particles motion can be updated. The procedure can be iterated for as many times as desired.
Figure 2.3 summarizes the basic steps of the PIC loop.
In the following, we will focus on each single step of this scheme.

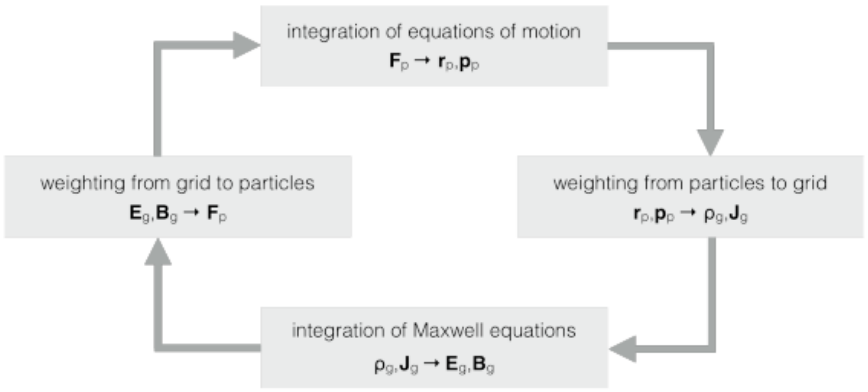

Figure 2.3: Sampling of the distribution function with macro-particles. Each macro-particle has a definite momentum, but is extended in space. Taken from [40]

.

First of all, introduce a Cartesian computational domain $\Omega = X \times Y \times Z$ with $N_x \times N_y \times N_z$ nodes forming cells with volume $\Delta x \times \Delta y \times \Delta z$; then, discretize the time interval $(0, T)$ in N_t timesteps with spacing Δt.

Maxwell solver

The numerical solutions of the Maxwell equations are often obtained with the second order Finite Difference Time Domain approach: the electromagnetic fields are discretized onto a staggered grid, named the *Yee-grid*, first introduced by Yee in [42]. The current density and the electric field are calculated in the middle point of the each edge of the cells and the magnetic one in the center of each face (see figure 2.4)
Time and space derivatives are discretized using centered finite differences; in addition, a *leap-frog scheme* is adopted for the evaluation of time derivatives: this means that $\mathbf{E}$ and $\mathbf{B}$ are not evaluated in the same timestep, but the second is shifted by $\Delta t/2$ with respect to the first.

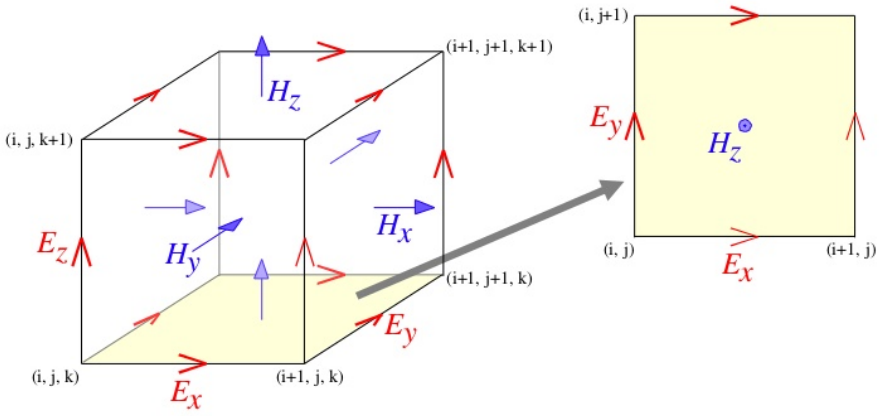

Figure 2.4: Yee-lattice

So it's clear that the Yee-lattice is a staggered grid both in space and in time: this strategy turns out to be very useful due to the relation between spatial and time derivatives in Maxwell equations. Thus, the magnetic field at timestep $n+\frac{1}{2}$ is obtained solving explicitely the discretized Ampère-Maxwell equation

$$\mathbf{B}^{n+\frac{1}{2}} = \mathbf{B}^{n-\frac{1}{2}} - (c\Delta t \nabla \times \mathbf{E}^n)$$

while, the electric one is evaluated at n from the Faraday equation

$$\mathbf{E}^{n+1} = \mathbf{E}^n + \Delta t(c\nabla \times \mathbf{B}^{n+\frac{1}{2}} - 4\pi \mathbf{J}^{n+\frac{1}{2}})$$

and the curls computation is expressed by

$$\nabla \times E^n = \begin{pmatrix} \frac{E^n_{z_{i,j+1,k+\frac{1}{2}}} - E^n_{z_{i,j,k+\frac{1}{2}}}}{\Delta y} - \frac{E^n_{y_{i,j+\frac{1}{2},k+1}} - E^n_{y_{i,j+\frac{1}{2},k}}}{\Delta z} \\ -\frac{E^n_{z_{i+1,j,k+\frac{1}{2}}} - E^n_{z_{i,j,k+\frac{1}{2}}}}{\Delta x} + \frac{E^n_{y_{i,j+\frac{1}{2},k+1}} - E^n_{y_{i,j+\frac{1}{2},k}}}{\Delta z} \\ \frac{E^n_{y_{i+1,j+\frac{1}{2},k}} - E^n_{y_{i,j+\frac{1}{2},k}}}{\Delta x} - \frac{E^n_{x_{i+\frac{1}{2},j+1,k}} - E^n_{x_{i+\frac{1}{2},j,k}}}{\Delta y} \end{pmatrix}$$

$$\nabla \times \mathbf{B}^{n+\frac{1}{2}} = \begin{pmatrix} \frac{B^{n+\frac{1}{2}}_{z_{i+\frac{1}{2},j+\frac{1}{2},k}} - B^{n+\frac{1}{2}}_{z_{i+\frac{1}{2},j-\frac{1}{2},k}}}{\Delta y} - \frac{B^{n+\frac{1}{2}}_{y_{i+\frac{1}{2},j,k+\frac{1}{2}}} - B^{n+\frac{1}{2}}_{y_{i+\frac{1}{2},j,k-\frac{1}{2}}}}{\Delta z} \\ -\frac{B^{n+\frac{1}{2}}_{z_{i+\frac{1}{2},j+\frac{1}{2},k}} - B^{n+\frac{1}{2}}_{z_{i-\frac{1}{2},j+\frac{1}{2},k}}}{\Delta x} + \frac{B^{n+\frac{1}{2}}_{y_{i+\frac{1}{2},j,k+\frac{1}{2}}} - B^{n+\frac{1}{2}}_{y_{i+\frac{1}{2},j,k-\frac{1}{2}}}}{\Delta z} \\ \frac{B^{n+\frac{1}{2}}_{y_{i+\frac{1}{2},j,k+\frac{1}{2}}} - B^{n+\frac{1}{2}}_{y_{i-\frac{1}{2},j,k+\frac{1}{2}}}}{\Delta x} - \frac{B^{n+\frac{1}{2}}_{x_{i,j+\frac{1}{2},k+\frac{1}{2}}} - B^{n+\frac{1}{2}}_{x_{i,j-\frac{1}{2},k+\frac{1}{2}}}}{\Delta y} \end{pmatrix}$$

Finally, in order to guarantee numerical stability, the spatial and time steps must satisfy the *Courant-Friedrichs-Lewy* condition

$$c\Delta t \sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}} = C < 1$$

where c is the light speed (i.e. the velocity of the laser pulse) and C is the so called *Courant number*

Grid-to-particles field interpolation

In the PIC algorithm, the electromagnetic field is computed on the grid, whereas all the quantities related to the macro-particles (ρ and $\mathbf{J}$) are evaluated at the particle positions. Nevertheless, in order to calculate the motion of each macro-particle, it's mandatory to know the Lorentz force on the particle position. For this reason the method needs to deal with the values of the electric and magnetic fields in the particle position and so, in order to fix the issue, these quantities are estimated with the interpolation technique.
Thus, since the electric field is already known at time n, it needs only to be interpolated at the particle position $\mathbf{r}_p$

$$\mathbf{E}^n(\mathbf{r}_p) = \frac{1}{\Delta x \Delta y \Delta z} \int_V \mathbf{E}^n S(\mathbf{r} - \mathbf{r}_p^n)\, d\mathbf{r}$$

where V is the volume of a cell, while the magnetic field is known only at half-integer time steps, so it requires to be advanced in two half-integer timesteps

$$\mathbf{B}^n = \frac{B^{n+\frac{1}{2}} + B^{n-\frac{1}{2}}}{2}$$
$$\mathbf{B}^n(\mathbf{r}_p) = \frac{1}{\Delta x \Delta y \Delta z} \int_V \mathbf{B}^n S(\mathbf{r} - \mathbf{r}_p^n)\, d\mathbf{r}$$

For cartesian meshes a standard choice is to perform interpolation choosing B-splines as shape functions, i.e piecewise polynomial functions with compact support.

Particle pusher

Once the electromagnetic fields are evaluated, it's time to calculate their effects on the macro-particles motion: also in this case, a leap-frog algorithm is used to evaluate the position and the momenta of the macro-particles on a staggered time grid. So, the discretization of the equations of motions yelds to

$$\begin{cases} p^{n+\frac{1}{2}} = p^{n-\frac{1}{2}} + \Delta t \frac{q}{m} [\mathbf{E}^n + \mathbf{v}^n \times \mathbf{B}^n] \\ \\ \mathbf{r}^{n+1} = \mathbf{r}^n + \Delta t \mathbf{v}^{n+\frac{1}{2}} \end{cases}$$

where $\mathbf{E}^n$ and $\mathbf{B}^n$ are the values of the fields interpolated at the center of the macro-particle at position $\mathbf{r}^n$ at time $t = n\Delta t$. However, the velocity, and so the momentum, of the macro-particles are known only at half-integer time, so their expression at n has to be approximate in this way:

$$\begin{cases} p^n = \frac{p^{n+\frac{1}{2}} + p^{n-\frac{1}{2}}}{2} \\ \mathbf{v}^n = \frac{p^n}{\gamma^n} \end{cases}$$

So, substituting $p^{n+\frac{1}{2}}$ in the previous equation the following implicit equation for p^n appears:

$$p^n = p^{n-\frac{1}{2}} + \frac{\Delta t}{2}\frac{q}{m}\left[\mathbf{E}^n + \frac{\mathbf{p}^n}{\gamma^n} \times \mathbf{B}^n\right]$$

At this point, this equation can be converted in an explicit equation using the *Boris pusher algorithm*. First, introduce a normalized magnetic field vector

$$\mathbf{b} = \frac{q}{m}\frac{\Delta t}{2}\frac{\mathbf{B}^n}{\gamma^n}$$

and define

$$\tilde{\mathbf{p}} = \mathbf{p}^{n-\frac{1}{2}} + \frac{q}{m}\frac{\Delta t}{2}\mathbf{E}^n$$

As a consequence, γ^n can be approximated in this way

$$\gamma^n = \sqrt{1 + \mathbf{p}^n \cdot \mathbf{p}^n} = \sqrt{1 + \tilde{\mathbf{p}} \cdot \tilde{\mathbf{p}} + O(\Delta t^2)} \approx \sqrt{1 + \tilde{\mathbf{p}} \cdot \tilde{\mathbf{p}}}$$

Hence, from

$$(\mathbf{p}^n - \mathbf{p}^n \times \mathbf{b}) \times \mathbf{b} = \tilde{\mathbf{p}} \times \mathbf{b}$$

$$\mathbf{p}^n = \frac{1}{1 + b^2}(\tilde{\mathbf{p}} + \tilde{\mathbf{p}} \times \mathbf{b} + (\tilde{\mathbf{p}} \cdot \mathbf{b})\mathbf{b})$$

to the explicit form turns out

$$\mathbf{p}^{n+\frac{1}{2}} = 2\mathbf{p}^n - \mathbf{p}^{n-\frac{1}{2}}$$

Particles-to-grid current deposition

Current deposition, i.e. charge and current density projection onto the grid, is then performed using the charge-conserving algorithm proposed by Esirkepov in [43]. The basic idea is to compute the local current density from the discrete continuity equation written for the single macro-particle and then, thanks to the linearity of the continuity equation, to sum the contributions of all macro-particles. So, the single contribution of the $p-th$ macro-particle to charge density evaluated at the grid point $\mathbf{R}$ is

$$\rho_p(\mathbf{R}) = q_p S(\mathbf{R} - \mathbf{r}_p) \tag{2.9}$$

At this point, the algorithm requires the definition of the *density decomposition vector* $\mathbf{W}$

$$W_{i,j,k} = -\frac{\Delta t}{q_p}\nabla^+ \cdot \boldsymbol{J}_{i,j,k} \tag{2.10}$$

where $\boldsymbol{J}$ is the current related to the single macro-particle and

$$\nabla^+ \cdot f_{i,j,k} := \left(\frac{f_{i+1,j,k} - f_{i,j,k}}{\Delta x}, \frac{f_{i,j+1,k} - f_{i,j,k}}{\Delta y}, \frac{f_{i,j,k+1} - f_{i,j,k}}{\Delta z}\right)$$

Now, using (2.9) the discretized continuity equation becomes

$$\frac{\rho^{n+1}_{i,j,k} - \rho^n_{i,j,k}}{\Delta t} + \frac{(J_x)^{n+\frac{1}{2}}_{i+\frac{1}{2},j,k} - (J_x)^{n+\frac{1}{2}}_{i-\frac{1}{2},j,k}}{\Delta x} + \frac{(J_y)^{n+\frac{1}{2}}_{i,j+\frac{1}{2},k} - (J_y)^{n+\frac{1}{2}}_{i,j-\frac{1}{2},k}}{\Delta y} + \frac{(J_z)^{n+\frac{1}{2}}_{i,j,k+\frac{1}{2}} - (J_z)^{n+\frac{1}{2}}_{i,j,k-\frac{1}{2}}}{\Delta z}$$

and this can be expressed with the relation

$$W^x_{i,j,k} + W^y_{i,j,k} + W^z_{i,j,k} = S(x + \Delta x, y + \Delta y, z + \Delta z) - S(x, y, z)$$

where the vector $(\Delta x, \Delta y, \Delta z)$ stands for the 3D-shift of the macro-particle position due to motion. Shifting the macro-particles generates eight functions:

$$\begin{aligned}
&S(x, y, z), S(x + \Delta x, y, z), S(x, y + \Delta y, z), S(x, y, z + \Delta z),\\
&S(x + \Delta x, y + \Delta y, z), S(x + \Delta x, y, z + \Delta z), S(x, y + \Delta y, z + \Delta z),\\
&S(x + \Delta x, y + \Delta y, z + \Delta z)
\end{aligned}$$

Thus, thanks to the assumption that **W** and the corresponding current density linearly depend on these functions, it's possible to decompose any three-dimensional shift as a linear combination of one-dimensional shifts. In particular, it ca be shown (for more details see [43]), that the decomposition vectors can be rewritten through the following linear combinations

$$\begin{aligned}
W^x_{i,j,k} = &\frac{1}{3}\Big(S(x + \Delta x, y + \Delta y, z + \Delta z) - S(x, y + \Delta y, z + \Delta z) + S(x + \Delta x, y, z) - S(x, y, z)\Big)\\
&+\frac{1}{6}\Big(S(x + \Delta x, y, z + \Delta z) - S(x, y, z + \Delta z) + S(x + \Delta x, y + \Delta y, z) - S(x, y + \Delta y, z)\Big)\\
W^y_{i,j,k} = &\frac{1}{3}\Big(S(x + \Delta x, y + \Delta y, z + \Delta z) - S(x + \Delta x, y, z + \Delta z) + S(x, y + \Delta y, z) - S(x, y, z)\Big)\\
&+\frac{1}{6}\Big(S(x, y + \Delta y, z + \Delta z) - S(x, y, z + \Delta z) + S(x + \Delta x, y + \Delta y, z) - S(x + \Delta x, y, z)\Big)\\
W^z_{i,j,k} = &\frac{1}{3}\Big(S(x + \Delta x, y + \Delta y, z + \Delta z) - S(x + \Delta x, y + \Delta y, z) + S(x, y, z + \Delta z) - S(x, y, z)\Big)\\
&+\frac{1}{6}\Big(S(x, y + \Delta y, z + \Delta z) - S(x, y + \Delta y, z) + S(x + \Delta x, y, z + \Delta z) - S(x + \Delta x, y, z)\Big)
\end{aligned}$$

Hence, it's now possible to solve (2.10) to obtain the current density associated with the motion of a single macro-particle. Finally, the total **J** turns out summing up all contributions on each grid point.

2.3 Potential and open challenges of PIC-codes

2.3.1 Capabilities of the PIC method

Both phase-space grid and particle-in-cell methods represent a very widely used tool in collisionless plasma physics simulations: according to the problem under consideration and the degree of precision desired, it's possible to choose which of them to adopt.
In fact, Vlasov codes guarantee a fine and uniform resolution in the phase space and, so, detailed information about the distribution of the particles.
Thus, in some particular situations, in which a high degree of details is required Vlasov

codes may represent a wise choice. Although, the high degree of precision increases the numerical complexity and, as a result, requires large computational efforts.
For example, consider a three-dimensional plasma simulation in which 1000 points for every direction are used: the resulting discretized phase-space would be a six-dimensional space and it would contain 10^{18} nodes. So, representing the distribution function with a double (8 bytes) would require 8 ExaBytes of RAM, far beyond present-day supercomputer capabilities.
For this reason, Vlasov codes in many cases do not represent the best tool for the prohibitive computational effort and very often they are limited to low-dimensional simulations.
So, the strong suit of particle-in-cell scheme basically is the lower numerical complexity:

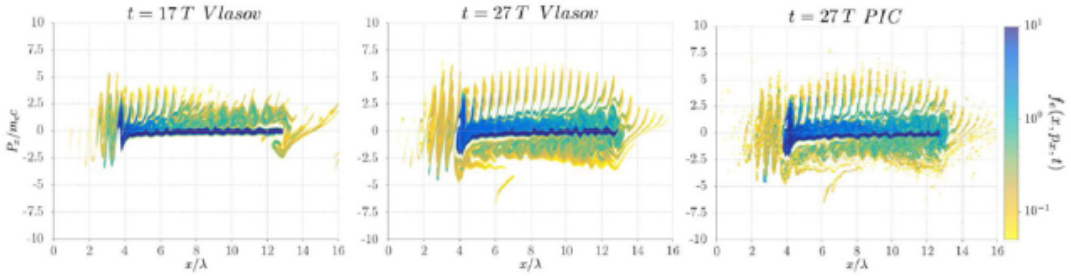

Figure 2.5: Comparison between the electron distribution functions in the phase space $x - P_x$ obtained with a Vlasov and a PIC code for a electron–proton plasma simulation irradiated by a laser pulse. The electron distribution functions are observed at times t=17T and t=27T for the Vlasov simulation, and at times t=27T for the PIC simulation (where $T = \lambda/c$ is the laser pulse period): in a suitable limit the second solution converges to the first. Taken from [44].

however, many times the level of numerical precision in solutions offered by this method is quite enough satisfactory and the recourse to Vlasov codes would take to useless very higher costs (see figure 2.5).
Moreover, current PIC codes provide the possibility to consider additional physics, such as collisions, ionization, quantum electrodynamics emission (QED), multiphoton Breit-Wheeler pair creation, high-energy photon emission and radiation reaction: so they allow to enlarge the range of scenarios to simulate.
They, also, provide a scheme relatively simple to implement and adapt very well to parallelization logics. For these reasons, PIC codes in many situations are the best choice: in fact they are the most widely used numerical scheme in plasma physics community, especially for laser-plasma simulations.

2.3.2 Criticalities of the simulations

Accuracy of the simulated physics

As explained previously, PIC codes guarantees a lower degree of fidelity for the numerical solutions because of an unavoidable numerical noise deriving just from the concept of macro-particles.
In fact, the distribution function is sampled through a limited number of super-particles, which is several orders of magnitude lower than the real number of particles, and this can lead to an excessive level of approximation, with the consequence of numerical results very far from realistic phenomena.
The limited number of macro-particles, also, implies a filter in space scale, under which the density cannot be accurately resolved: this represents a big problem if the scenario under consideration involves small perturbations or low density regions where the macro-particle population is too small.
Furthermore, in laser-plasma physics a central role is played by phenomena which occurs at lengths near to the skin depth and the numerical experiments must take it in consideration when the space resolution is chosen. Otherwise, if the mesh is not fine enough, the aspect under investigation will be numerically inexistent and the entire numerical campaign will be totally useless. But, recalling the CFL condition

$$c\Delta t\sqrt{\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}} = C < 1 \tag{2.11}$$

increasing the numerical resolution results in a reduction of the time step, making feasible only short simulation time.
Moreover, the numerical noise can be reducted with a big number of particles per cell (PPC): in general, denoting with n_e the plasma electron density, as a rule of thumb

$$PPC \geq \min\left\{1, \frac{n_e}{n_c}\right\}$$

. However, the price to pay for the reduction of numerical noise is the increase of computational costs; moreover, simulating solid-density plasmas, whose electron density n_e wander around $200n_c$ results unaffordable if an high number of PPC is chosen: to address this issue numerical simulations use lower, non-realistic electron densities ($\sim 40-80n_c$ for solid-density plasmas), considering a lower threshold to have overdensity (i.e. laser reflection) with respect to the physical values. For all these reasons, a superficial approach it's forbidden when performing laser-plasma simulations, but rather all these considerations must be taken in account with attention.

Computational issues

As mentioned above, one should pay attention to selecting suitable values for the spatial resolution and number of particles per cell in order to achieve a high enough level of

accuracy. On the other hand, too accurate values for these numerical parameters may have a significant impact on computational costs. More precisely, given a mesh size Δ and a particles per cell number PPC, the computational time required to simulate a n-dimensional simulation with a standard PIC code can be estimated as

$$T \sim \alpha \left(\frac{1}{\Delta}\right)^{n+1} + \beta PPC \left(\frac{1}{\Delta}\right)^{n+1} \tag{2.12}$$

where α and β are two suitable constants.
Usually, $PPC \gg 1$ and so most of the computational load derives from the particle contribution; moreover, from (2.12) it is easy to notice that a 3D simulation would require at least $\frac{1}{\Delta}$ the times required by the same simulation in a two-dimensional environment and it is a huge additional load. Hence, two-dimensional simulations may represent a good compromise in order not to spend immense amounts of computational hours but, anyway, many times a three-dimensional approach might be essential to capture sensitive aspects, such as 3D electrostatic effects or real morphologies of structured plasmas.
To give an idea, typical 2D simulations of laser-driven ion acceleration cost $\sim 10^2 - 10^3$ core hours, while 3D ones require $\sim 10^3 - 10^4$. So, computational loads required by laser-plasma simulations represent a complicated matter to handle and some optimization strategies must be considered.

2.3.3 Parallelization and High Performance Computing Clusters

As discussed above, PIC codes must find a solution to optimize the run-time behavior, in order to be competitive and to guarantee the realization of interesting numerical simulations. For this purpose current PIC codes implement some parallization strategies, using the well popular MPI and OpenMP protocols.
The Message Passing Interface (MPI) is a communication interface for parallel programming, which allows to split code routines in several *processes* (or *tasks*) , assigned to as many different computing units (or *cores*), which can operate simultaneously. These processes, however, do not share the same memory regions, and that's why this strategy provides both point-to-point communications, in which two specific cores exchange information, and collective communication, which allow to share information among all tasks.
Today, PIC codes avail of the MPI protocol. In computer science, several strategies of parallelization are possible, focused on optimizing some distinctive issues of the algorithms concerned.
The main point in parallelizing a PIC code lies in the coupling the grid and particle aspects of the code. Usually, most of the computational load is attributable to computations related to the macro-particles.
So, a purely particle-based decomposition might seem to make sense: an equal distribution of the macro-particles between the cores would guarantee an almost perfect load balance for the entire simulation. Moreover, particles are independent from each other and they

only interact with fields, so no particle communications are required.
For this reason, typically PIC codes exploit a grid-based parallelization technique, consisting in *domain decomposition*.
The box of the simulation is split in several parts, called *MPI regions*, each of which is assigned to a specific process: in other words, every unit executes only the calculations associated to the particles and the fields belonging to the core itself.
As a result, the only communications required are those between neighbor domains and are not a fundamental threat to performance or scalability.
Nonetheless, the randomness of macro-particle positions, which travel throughout the entire domain, is a big issue: they require communications when local domain boundaries are crossed and random access to the grid at every interpolation and projection phases is required.
Nevertheless, this approach implies that all the processes have access to a shared global memory of grid quantities (fields and currents), and all the accesses should be synchronized: this requires frequent global communication, which prevent any form of scalability when computations are split between a lot of cores. Most of the time, this issue is addressed by operating a proper sorting of the particles: several algorithms exist in order to allocate contiguous particles in contiguous spaces of memory.
On the other hand, the Open Multi-Processing (OpenMP) system consists of a set of compiler directives, library routines, and environment variables which allow to guide a set of *threads*, i.e. series of instructions executed consecutively, which globally share the same memory.
So, OpenMP is artificer of *multi-threading* implementation: a *master* thread forks a specified number of *slave* threads, that run concurrently providing better performances in terms of computational times.
Thus, many PIC codes take benefits from these two protocols, sometimes combining them in a *hybrid* MPI-OpenMP version: this stratagem permits to make accessible some numerical simulations which would be impossible to perform otherwise.
Nevertheless, in laser-plasma simulations even the parallelization strategies aren't able to remove the problem of simulations duration. Indeed, the most of numerical simulations performed by laser-plasma community, as mentioned before, require a degree of accuracy, both in spatial resolution and in the number of macro-particles, which can't be sustained by current laptop, even by the most powerful on the market.
Therefore, only a big number of computing cores, handled with parallelization strategies, represent a considerable option in this kind of numerical simulations: this task is covered by *supercomputers*, i.e. very powerful computers organized in *High Performace Computing Clusters*.
The term *"super computing"* was utilized for the first time in 1920 by the daily newspaper *New York World* to describe some IBM tabulators allocated at the Columbia University.

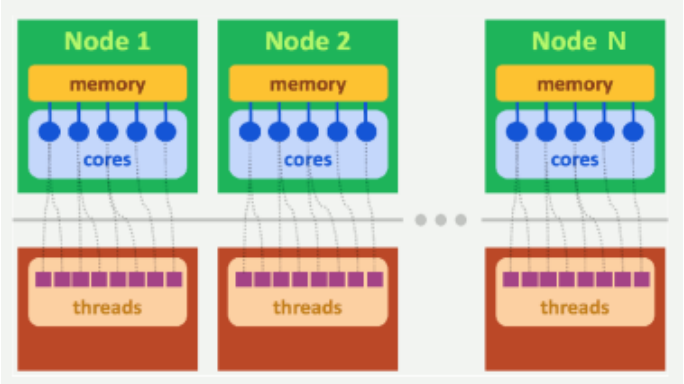

Figure 2.6: Simplified scheme of a supercomputer composed by N nodes

However, the creation of supercomputers, as today are intended, took some extra decades as long as, in 1954, the realization of what is considered, by the most, the first supercomputer occurred: it was the *Naval Ordnance Research Calculator* (NORC), built by IBM for the United States Navy's, which, at the presentation ceremony, was employed to calculate π to 3089 digits, which was a record at the time.
Basically, these cluster allow to split the same computations on a very bigger number of units, tearing down computational times, thanks to their architecture.
A large number of processing units are grouped in *nodes*, interconnected in proximity to each other, forming a massively parallel system. Each node has a proper memory which allows the cores contained, sharing the same information, to operate on the same data, at the same time, without needing to involve communications.
So, all the cores, contained by all the nodes, can work together in a parallel way, through the MPI protocol
In addition, supercomputers exploit the OpenMP interface in order to implement a multi-threading logic (for each node), which allows to split the process administration in several threads.
Figure 2.6 provides a simple schematization of supercomputers architecture.

Today, a lot of powerful supercomputers exist, whose competitiveness is tracked by the *TOP500* project, which ranks and details the most powerful 500 systems twice a year: today the first place is occupied by *Fugaku*, a supercomputer placed at the RIKEN Center for Computational Science in Kobe, Japan.
So, thanks to their incredible power, supercomputers are exploited in computational science, with computationally intensive tasks in various fields, including laser-plasma physics. Finally, thanks to the massively parallel system sustained by modern supercomputers, more and more laser-plasma simulations find realization, making feasible some scenarios which otherwise couldn't be studied.

2.3.4 Exploiting Graphics Processing Units in PIC-simulations

In the last years, the literature about PIC codes has began to consider the idea to perform numerical simulations using *Graphics Processing Units*.
"A graphics processing unit (GPU) is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device."
GPUs, unlike the classical *Central Processing Unit* (CPU), in which at most a few dozen cores are incorporated, usually contains hundreds or even thousands of cores, able to work to a frequency of ~ 1 GHz (see figure 2.7).
GPUs are usually incorporated in embedded systems, mobile phones, game consoles and,

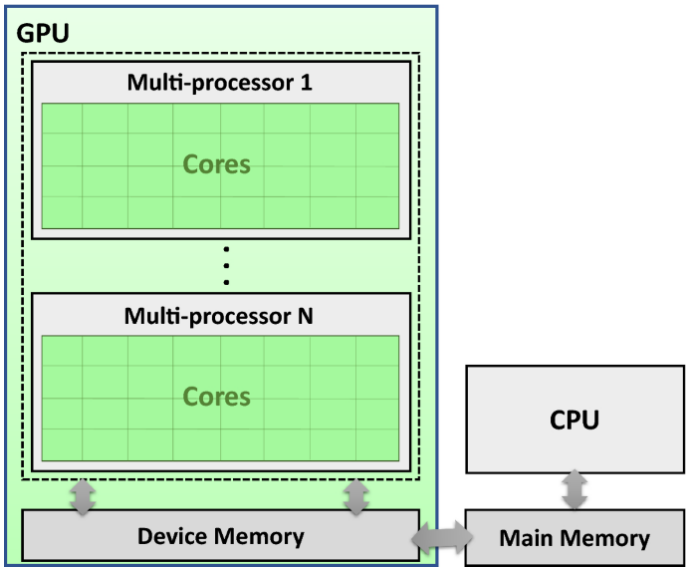

Figure 2.7: Comparison between CPU and GPU architectures. Taken from [45]

.

obviously, in personal computers: in the last case can be present on a video card o directly embedded in the motherboard.
Thanks to its highly parallel structure, a GPU can do computations on algorithms that process large blocks of data in parallel, very faster than CPUs, even if the latter work to higher frequencies ($\sim 2 \div 5$ GHz). So, CPUs performances are not comparable to those of GPUs, if parallelization is invoked, but the former have a higher clock speed (i.e. the rate at which a processor can complete a processing cycle) which allow them to provide better performance in serial computations.
That's why, in the last years, the *General-purpose computing on graphics processing units* (GPGPU) is expanding significantly, covering a lot of scientific fields (physics, mathematics, astronomy, chemistry, biology, ecc). Basically, this system uses GPGPU pipelines, which are parallel systems between one or more GPUs and CPUs, that migrate data into a form which GPUs can scan and analyze. The sequential part of the workload runs on

the CPU, which is optimized for single-threaded performance, while the compute intensive portion of the application runs on all the GPU cores in parallel.
Today, there exist some software layers designed for give direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of compute kernels, which are the routines compiled for GPUs (separate from but used by a main program). The most widely used id the *Compute Unified Device Architecture* (CUDA) , which is a parallel computing platform and application programming interface (API) model developed by NVIDIA, designed to work with programming languages such as C, C++, and Fortran. Essentially the CUDA processing flow occurs in the following way: first of all data are copied from main memory to GPU memory, then, after that CPU has initiated the GPU compute kernel, GPU's CUDA cores execute the kernel in parallel, copying the resulting data from GPU memory to main memory. For a visual representation of this scheme see figure 2.8. Clearly, even some Particle-In-Cell codes have implemented a GPU

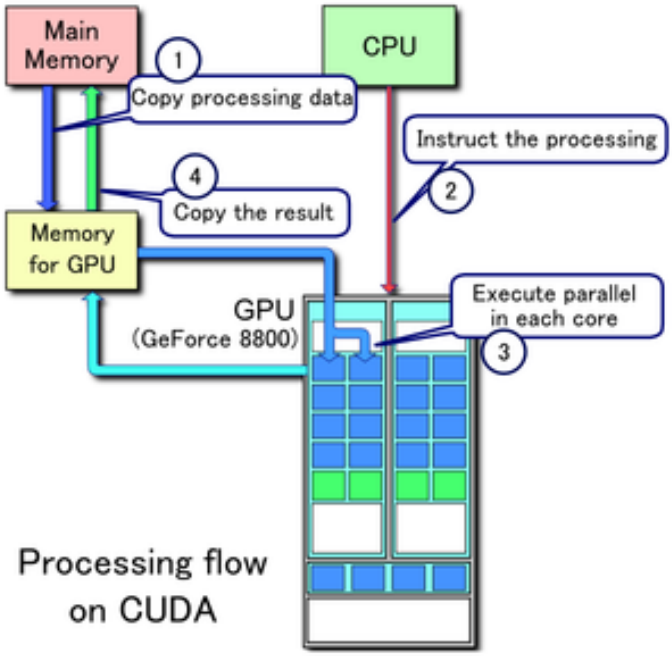

Figure 2.8: Example of CUDA processing flow. Reprinted from Wikipedia

implementation, in order to reduce further computational costs. At the state of art, this strategy is still set to an exploration phase in which some basic simulations are performed, with the aim to quantify the real benefits with respect to classical CPU-codes performances ([46],[47],[48],[49],[50], [51], [52], [53], [54]): in all these cases a significant improvement in performances is noticed. Moreover, in chapter 4 of this thesis work, some numerical tests with a hybrid CPU-GPU scheme will be carried out, accompanied by some considerations and perspectives.

2.4 Motivations and goals of this thesis work

At this point, all the basic concepts necessary to explain the motivations and goals of this thesis work have been introduced. Essentially, in chapter 1 the plasma state has been described, focusing on both the physical aspects and the mathematical modeling, with a special attention for the relativistic collisionless Vlasov-Maxwell system. In addition, a particular attention has be devoted to the discussion of the main physical aspects underlying the laser-plasma interaction, its connection with ions acceleration and its enhancement through nanostructured double-layer targets.
After that, chapter 2, so far, was devoted to introduce the main numerical methods used to find an approximation of the relativistic collisionless Vlasov-Maxwell system, distinguishing phase-space grid from Particle-In-Cell methods: these last have been deepened, with a detailed description of the several numerical ingredients incorporated.
Then, the chapter continues describing the issues arising from the numerical complexity of this methods and the parallelization strategies exploited, in order to reduce the computational efforts, with an overview on massively parallelized systems, such as High Performance Computing Clusters. Finally, we have introduced the concept of General-purpose computing on Graphics Processing Units, which allows to exploit the highly parallelized structure of the GPUs, in order to further reduce computational times in PIC simulations.

The motivations of this thesis derive from the necessity to enlight the computational load required for the advanced PIC simulations, under complex scenarios of actual physical interest. PIC codes represent a wide used numerical tool since the 1960s, in order to exploit simulations involving plasmas.
However, as discussed in chapter 1, in the last years the laser-plasma community is studying more and more sophisticated physical scenarios (such as the nanostructured double-layer targets described in chapter 1). Hence, this field of reasearch needs intensive numerical campaigns in order to understand which strategy represents a promising way to enhance ion acceleration.
So, the arising open challenge consists in exploiting PIC codes to perform simulations involving very complicated frameworks, far from the classical scenarios for which PIC methods have been designed.
Clearly, simulating such complex physical scenarios requires ever more computational efforts, which may be prohibitive even if supercomputers are engaged. For this reason, a three-dimensional approach results too expensive and, usually, it is adopted only for selected cases; however, two-dimensional simulations constitute only a qualitative investigation, but they can not report quantitative informations, which must be researched entirely through experimental sessions.
Moreover, even exploiting a 2D approach, realistic physical configurations are extremely hard to reproduce numerically and today we are enable to simulate realistic parameters,

solving all the characteristic lengths (e.g the skin depth, described in chapter 1).
Therefore, it becomes essential, when using PIC codes, looking for some strategies to achieve reasonable computational times when complex simulations are performed. Otherwise, the existing codes are quite complex and each of them has specific strategies for handling the parallelization.
Hence, need to deepen the understanding of code performances when dealing with complex physical configurations, in order to be able to squeeze as much as possible the codes capabilities: indeed, once that specific strategies will be found, more realistic simulations will become possible to perform.
Nonetheless, at the state of art there exist not a consistent literature reporting solutions which could address these issues.
Therefore, this thesis work represents a sort of investment, in terms of computational hours, to search for configurations which may increase the codes performances, reducing the computational efforts.
In particular, this work set itself the objective to analyze the behavior of different PIC codes when distinct scenarios are simulated, exploiting different clusters.
For this purpose, two open-source PIC codes, *Smilei* and *WarpX* have been selected as tool to perform a collection of numerical tests. Chapter 3 is devoted to the description of the main aspects of both these applications and to the presentation of the main results obtained. To this regard, we have availed of *Galileo*, a CPU based supercomputer belonging to the Italian computing centre CINECA, located in Bologna.
Moreover, we have conducted an investigation, described in chapter 4, on the performances reachable in simulations exploiting graphics processing units. We have reported the results, achieved through the hybrid GPU-CPU based cluster Marconi100, the latest supercomputer acquired by CINECA. All these simulations have been performed with WarpX: chapter 4 contains also a brief explanation about the adaption of the code to GPUs parallelization.

Lastly, chapter 5 highlights the conclusions of this work and its future perspectives.

Chapter 3

Performance analyses of laser-plasma interaction simulations on a CPU-based cluster

3.1 Selection of open source Particle-In-Cell codes

At the present day, it is possible to choose among several Particle-In-Cell codes, which often find application in a wide range of physics studies, among which not only plasma physics ([40], [41]), but also cosmology ([55], [56]) and solid ([57], [58]) and fluid ([59], [60])) mechanics.
In particular, thanks to the high versatility of many PIC codes, various kinds of simulations in a wide variety of environments are possible, from the laboratory experiments, such as laser-plasma interaction, to astrophysics problems. In this thesis work, two codes have been considered, *Smilei* ([61]) and *WarpX* ([62]), whose main features will be largely discussed in the following.
First of all, the first reason that justifies this selection is the fact that both the codes are open source, so they are available for use or modification from the original design, if some adaptations are required by particular frameworks.
They are object-oriented codes and are designed to run on a wide range of machines, from laptops to supercomputers: their approach guarantees a large user community and, as a consequence, support for a variety of applications. Moreover, these codes are submitted to continuous active development and upgrades, in order to improve the utilization. Furthermore, they can guarantee very good performances, in terms of computational costs, thanks to the efficient adaptability on massively parallel systems, relying on an hybrid MPI-OpenMP implementation and on other special techniques of load balancing which will be discussed more in detail in the following. Hence, Smilei and WarpX are two valu-

able representatives of PIC codes exploited in plasma physics, which work even in massively parallel logic, involving different parallelization strategies, based on the MPI and OpenMP paradigms.

3.1.1 Smilei

Smilei is a fully-relativistic electromagnetic PIC code, developed at *Ecole Polytechnique*, in Paris, by the laboratory of research in numerical simulations *Maison de la simulation*. It is designed to perform one-dimensional, two-dimensional and three-dimensional simulations, using a cartesian or cylindrical geometry.

Parallelization strategies

Smilei avails of the MPI protocol, in order to split computations between many processes, which exchange information communicating with each other: as many PIC codes (see 2.3.3), it exploits a grid decomposition strategy. More specifically, Smilei uses a brilliant strategy (introduced in [63])), founded on *patch-based* decomposition: it consists in a very fine-grain grid decomposition in group of cells, the so called *patches*, which, basically, constitute small sub-domains.
Essentially, groups of cells are collected in patches and, then, group of patches are organized in MPI regions and assigned to the different processes, in such a way that every process handles many patches (as shown in figure 3.1). The basic idea is to provide a variable number of patches to each MPI process in order to balance the computational load carried by each rank. Actually, this strategy allows a sort of particle sorting: all the particles related to a given patch are naturally stored in a compact array of memory and attached to the grid portion they can interact with. Moreover, any patch, due to the relative small number of cells contained, can fit very well into the cache.
A direct result of this fine domain decomposition is the necessity of additional communication at the patch-level. Actually the real trouble, in terms of computational costs, is represented only by the patches belonging to different processes: indeed, intra-process communications are carried out copying a relatively small amount of ghost cells (extra cells out from the region) and exchanging particles in a shared memory system.
A completely different issue arises when inter-process synchronizations are invoked, in which, in order to exchange data between patches belonging to different MPI regions, expensive routine provided by the MPI library are called.
That's why, a wise distribution policy is needed, which tends to minimize the boundaries of the MPI regions (and so the number of MPI calls), but, at the same time, flexible enough to be applied for an arbitrary number of processes and of patches per processes. An elegant solution is achieved ordering the patches along a *Hilbert space-filling* curve: this sophisticated mathematical function, a shown in figure 3.2, guarantees the minimization of contiguous portions of the regions, which are created simply following the curve (see [64]

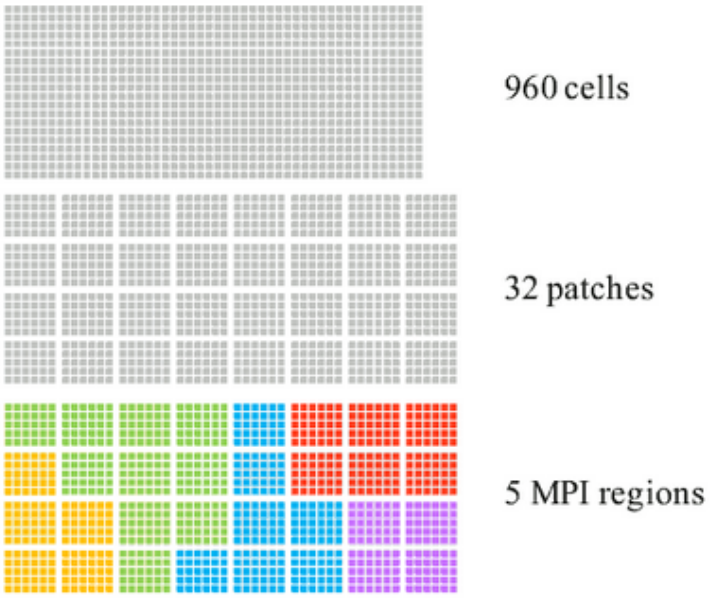

Figure 3.1: Example of a patch-based decomposition. The full domain is discretized with 960 cells. Then, the mesh is divided in 32 patches, composed by 6×5 cells. Lastly, the patches are organized in 5 MPI regions, each of them assigned to a different MPI process.

and [65]).

Furthermore, Smilei provides a strategy to minimize cache misses: in fact a further particle sorting occurs inside the patches, thanks to which, even when the size of the patches is too large, particles remain sorted without loss of performances.

This specific domain decomposition is also well adapted to the application of OpenMP threads parallelism over this collection of patches per process: basically, a further parallelization can be applied to the patches inside a MPI region, being independent subdomains. This means that the operations to apply on each patch can be split with a multi-threading logic.

For example, if a region is formed by 10 patches and 4 threads per process are imposed, the 4 threads will handle the 10 patches, working in parallel, patch by patch, until the 10-th patch is done.

One great advantage deriving from his scheme allows a *dynamic scheduling*: inside one MPI region, the faster threads, which operates on patches with few macro-particles, can continue working passing to other available patches, without waiting for the busier colleagues, thus avoiding long waiting times.

Thus, a smart use of the hybrid MPI-OpenMP scheme can bring to significant reductions of computational times and useless waists of time.

Most of the times, is convenient to choose the number of MPI processes equal to the number

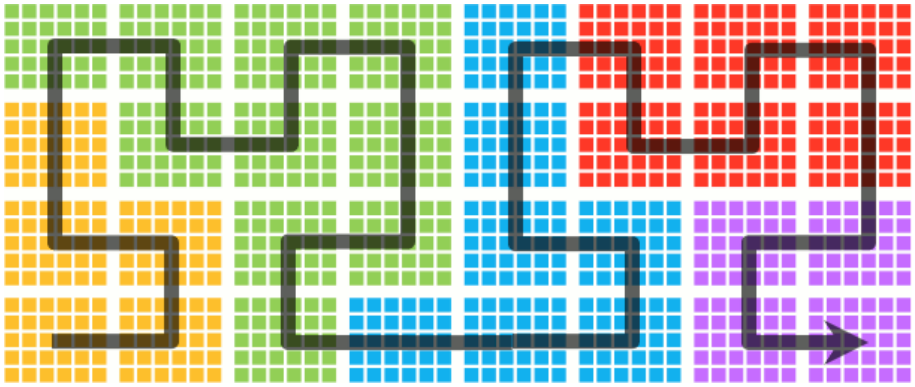

Figure 3.2: Example of a 8 × 4 patches domain decomposition, shared between 5 MPI processes. MPI domains are delimited by different colors. The Hilbert curve (black line) passes through all the patch centers, starting from the patch in the south-west corner

of nodes selected, in order to optimize the memory distribution: indeed, less MPI processes is not possible because they cannot be split among separate memory spaces, whereas more MPI processes is not recommended because they are not as efficient as OpenMP threads. The multi-threading, also, requires proper considerations and should not be taken light. In order to use all the cores, it is recommended to impose as many threads per process as cores per node (even more, but only if the architecture supports it well).
Therefore, the dynamic scheduling, handled by the threads, provides high performance if, clearly, the total number of patches is larger than the total number of threads, in such a way that the faster threads can work, sequentially, on several patches.

Load balancing

Smilei provides also the possibility to enable a global *dynamic load balancing*. This scheme plays a relevant role when the macro-particles, and consequently the operations related to them, are non equally distributed between the process: in fact, with this solution, overloaded regions are allowed to send some patches to the contiguous ones, just following the direction of the Hilbert curve (in other words, following the direction of the arrow in figure 3.2). Inversely, an underloaded process will increase its number of patches to handle. So, before the beginning of the simulation, Smilei tends to distribuite the patches in such a way that a fair load is respected. After that, every fixed number of timesteps, set by the user, the code performs a control on the loads associated to the different regions, and if it is needed, operates a new redistribution in the way described above.
Thus, in order to implement this balancing strategy, the calculation of computational load associated to a patch p, is calculated in this way

$$P(p) = N_{part} + C_{cell} \times N_{cells}$$

where N_{part} is the number of particles in the patch, N_{cells} is the number of cells contained and C_{cell} is a user-defined coefficient (default values is 1.0) representing the computational cost of cells (mostly solving Maxwell equation).Generally, the most of the load is represented by N_{part}.
Now, the computational load sustained by a rank r results, simply, as the sum the loads related to all the patches under its management

$$L(r) = \sum_{p} P(p),$$

while the total computational load of the simulation is

$$L_{tot} = \sum_{r} L(r)$$

So, given N_{MPI} ranks, a perfect load distribution implies the same amount

$$L_{opt} = \frac{L_{tot}}{N_{MPI}}$$

for every process.
Thus, the balancing algorithm proceeds to a new assignement of the pacthes, in such a way that each $L(r)$ is very close to L_{opt}. Clearly, concrete improvements in terms of performances are achieved if the load balancing occurs quite often: in fact, frequent and small corrections give superior benefits than rare and dramatic adjustments (see figure 3.3).

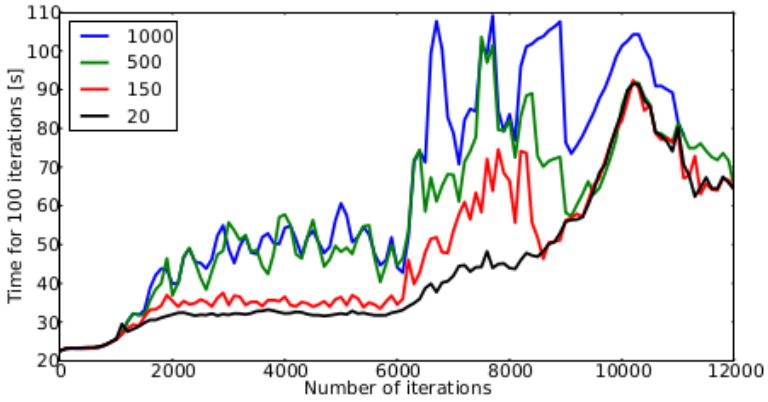

Figure 3.3: Evolution of time spent for 100 iterations, as a function of the number of completed iterations, for four different values of N_b (the number of iterations between two load-balancing events). Taken from [61].

3.1.2 WarpX

Warpx is a Particle-In-Cell code developed, as a part of the *U.S. Department of Energy's Exascale Computing Project*, by a collaboration between the *Lawrence Berkeley National*

Laboratory, *SLAC National Accelerator Laboratory* and *Lawrence Livermore National Laboratory*. This software is still in active development with the aim to explore outstanding questions in the physics of acceleration and transport of particle beams in chains of plasma channels, in order to benefit the numerical investigation of laser-plasma accelerators.
Actually, this numerical tool derives from the orchestration of three major softwares (see 3.4):

- *Warp* ([66]) is a pre-existing PIC code responsible for the modeling of plasma, beam and particle accelerators. This framework provides a Python interface with modules for code control, user steering, additional physics and diagnostic packages.
- *AMReX* ([67]) is a C++ library which exploits *Adaptive Mesh Refinement* (AMR) methodology to deal with of adaptive grids and particles, which interact with data defined on a block-structured hierarchy of meshes. In addition, it handles MPI and OpenMP parallelisms and load balancing strategies.
- The *PICSAR* package ([68]) contains FORTRAN routines for basic PIC operations, subjected to continuous optimizations for the best adaptation to massively parallelization.

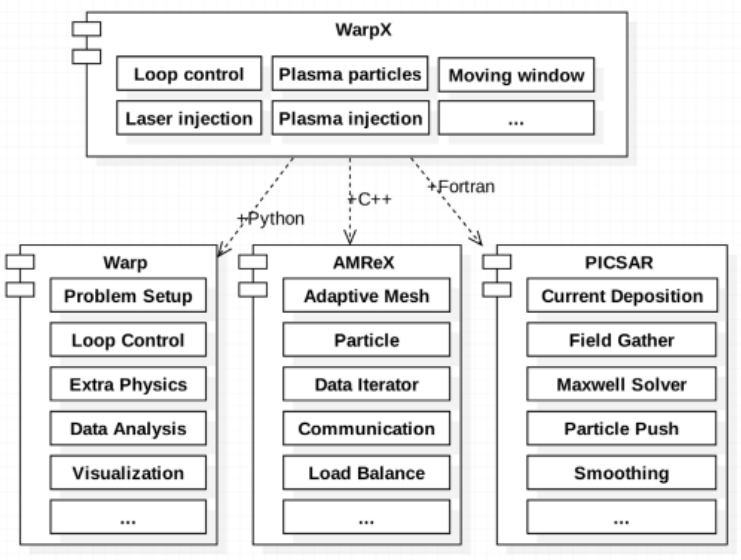

Figure 3.4: UML diagram of WarpX. In addition to WarpX's own source code ('WarpX-Source'), functionalities are provided by three packages: AMReX for the handling of AMR, communication and load balancing, PICSAR for the low level individual PIC functionalities, and Warp for extra physics packages, alternate user interface and control.

WarpX provides the possibility to perform both 2D3V and 3D3V Particle-In-Cell simulations, exploiting a cartesian or cylindrical geometry. At the beginning of the project, WarpX, resulted from the combination of the tools described above, was tested against

Warp for some simple simulations, and the results of the two codes were in excellent agreement (for more details see [62]).

Parallelization strategies

As already discussed, WarpX inherits the hybrid MPI-OpenMP parallelization strategies from Amrex: thus, for a proper documentation on that one should refer to [67].
First of all, like many other PIC codes, among which Smilei, the program opts for a domain decomposition based scheme: the entire mesh is subdivided in independent rectangular subdomains, called simply *grids*. The user does not explicit a priori the domain decomposition but the grids are automatically created on the basis of two numerical parameters imposed, i.e. the `maxgridsize` and the `blockingfactor`. The `maxgridsize` corresponds to a threshold (for each dimension) for the size of the subdomains, that the program must not exceed during the grids creation. However, an assurance for good multi-grid performance is fundamental: indeed, the lengths of the subdomains created must be divisible by the value chosen for the `blockingfactor` (it's possible to choose different values for different directions), which must be a power of 2.
So, these parameters allow to draw a limit to the freedom of to the algorithm, which operates the decomposition at run-time.
Basically, domain decomposition happens following this procedure.
At the beginning, a single grid, with size equal to the number of cells (in each direction), is defined covering the entire physical domain. After that, if the value assigned to the `maxgridsize` is lower than the number of cells, an iterative loop starts, which generates subdomains until each grid is no longer than the `maxgridsize`. Clearly, each step of this procedure must respect the blocking factor criterion, otherwise the loop is interrupted even if the constraint imposed by the `maxgridsize` is violated. Furthermore, if the algorithm produces less subdomains than MPI processes, further subdivisions are operated in order to guarantee that each process is associated to at least one grid.
So, a good practice consists in carefully choosing these two numerical parameters, keeping in mind that the best optimization is achieved, in computational terms, if at least one grid is reserved to any process, in such a way that no shared regions exist.
By the way, even if the user cannot choose an explicit decomposition a priori, it is possible to force the program to create a desired number of identical grids, setting the same value for both the parameters, with the result that subdomains with this fixed size will be created. Anyway, many times this forcing procedure, if not well motivated, does not reveal a good idea because the algorithm implemented by WarpX is already designed to optimize the computational load of each rank and, in many situations, identical fixed-size grids are not the best option.
Beside the inter-process communication scheme, provided by the MPI interface, as already mentioned, as already pre-announced, WarpX includes a intra-process parallelization

scheme, offered by OpenMP.
In the following, an explanation about how the Amrex package deals with multi-threading is addressed.
Basically, once the domain decomposition is completed, there are several grids distributed among the available processes; then, as figure 3.5 shows, each of these grids are *logically* decomposed, through the so called *tiling* process: essentially, this procedure allows to transform the loops, operated by a rank on a entire subdomain, into tiling loops that iterate over tiles and element loops that iterate over the data elements within a tile. This results in thread parallelization, in which several threads per grid cooperate, working on different tiles, improving data locality.

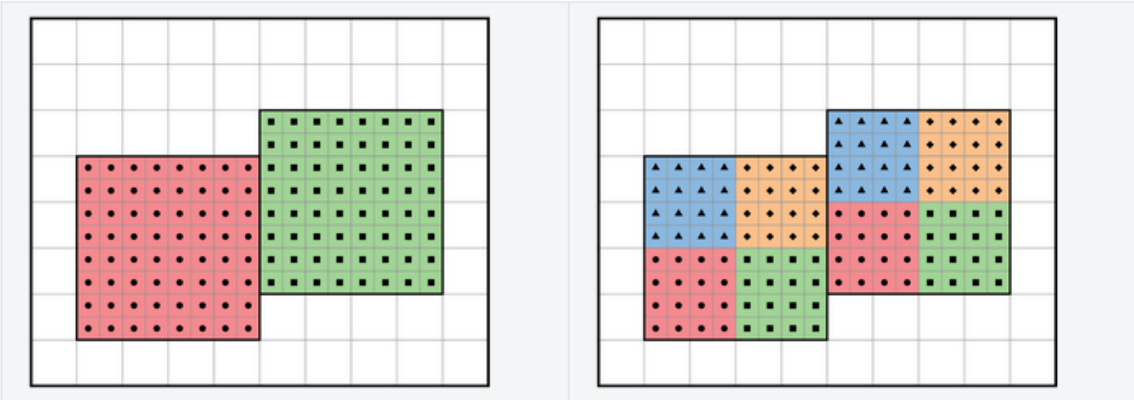

Figure 3.5: Comparison of domain decompositions without and with tiling. In the first example, there are 2 grids, each one formed by 8×8 cell, while in the second one the 2 grids are *logically* broken in 4 tiles of 4×4 cells.

Load balancing

As illustrated before, each MPI rank typically computes a few grids, assigned in the initialization from the code. This assignment is lead focusing on equalize the computational load between the processes. However, this procedure does not ensure a good balancing for the full duration of the simulation, because probably the computational load felt by each grid, and so by each process, will be subjected to alterations, due to macro-particles motion.
That's why WarpX provides the possibility to activate the *load balancing*: this procedure implies a periodical redistribution of the grids, whose shape obviously remain unchanged, among the processes in order to maintain a fair balance, and, so, to optimize performances. In other terms, thanks to this scheme the processes with a lot of work can transfer one or several grids to their neighbors: in this regard, the code provides two ways for estimating the load of each ranks.
The user can choose the *timers* mode, which consists simply in a direct estimation of the

computation time of each process, through in-code timers. Otherwise, the user can opt for a *heuristic* approach, in which load balance costs are estimated according to a measure of particles and cells assigned to every grid handled by a process. So the effort required by a grid g is computed in this way

$$L(g) = w_{particle} \times n_{particle} + w_{cell} \times n_{cell}$$

where $w_{particle}$ and w_{cell} are two coefficients set to 0.9 and 0.1 by default, unless a redefinition is made. After that, this algorithm calculates the computational load assigned to each rank r simply summing over the loads of its grids

$$L(r) = \sum_{g} L(g)$$

Now that costs estimation has been performed, the subdomains redistribution can take place and, by the way,the code provides two possible strategies.
The first option is represented by the *Space Filling Curves* (SFC) algorithm, which consists in grids distribution following the filling-space *Z-Morton curve* (illustrated in 3.6).
Thanks to the shape properties of this mathematical curve ([69],[70]), this scheme guar-

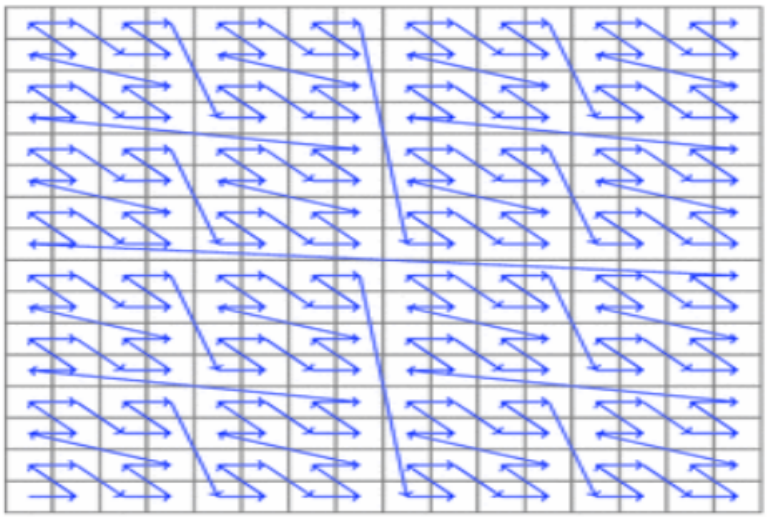

Figure 3.6: The grids are enumerated following the space-filling Z-Morton curve and, then, distribute among the ranks, following the order, in a way that balances the load

antees compactness, which limits the length of the boundaries between communicating processes, simple and fast encoding ability and flexibility, in order to ensure a good adaptability for an arbitrary number of subdomains.
Alternatively, the user can select the *knapsack algorithm*. In this case, each grid is marked with a weight, corresponding to the number of cells contained, and the same maximum capacity is fixed for every process. Hence, the subdomains ripartitions takes place, trying

to maximize the minimum load between the ranks. However, this strategy, most of the time, does not represent the best choice, because the load of each grid is estimated only considering its size, ignoring the macroparticle positions: so, when the plasma is not uniformly distributed among the domain, inefficient balances occur.
The entire load balancing scheme is reiterated periodically, diring the simulation, and the frequency can be chosen a priori: in order to ensure substantial benefits, it's recommended to activate frequent load balances.

3.2 Simulated laser-plasma interaction scenarios

As discussed in section 1.4.3, double-layer targets represent a promising road to follow in order achieve enhanced TNSA mechanisms. Therefore, more and more in-depth researches, through numerical simulations, are required to understand which aspects, both in the morphology of the DLT and in the laser parameters, can lead to a maximum yield in terms of ion acceleration.
The DLT structure has been already elucidated: a solid density thin target, whose front side, where the laser impinges, is covered with a thick near-critical foam. However, this complex structure and its interaction with laser pulses requires a sophisticated modeling and a big amount of computational efforts.
As explained in section 2.4, this thesis work aims to find a set of guidelines which can lead to an optimal usage of the two PIC codes considered, providing some indications about how much the choice of numerical parameters and parallelization configurations can impact on the computational costs. Therefore, the numerical tests presented constitute a sort of investment, in terms of computational hours, to delineate a good utilization of the two codes in the future.
More precisely, this explorative numerical investigation aims at studying separately the computational efforts required by simulating laser interaction with the two layers composing a DLT.
Hence, two distinct scenarios constitute the subjects of our simulations: a thin solid target and a thick near-critical foam. The choice of studying the two distinct layers separately is attributable to several reasons.
First of all, since this work constitutes a first approach to the research of strategies focused in reducing computational times, it is better two consider simplified situations, which may be rendered more sophisticated in a future investigation.
Moreover, the solid target and the near-critical foam present different physics mechanism, when interacting with high-power laser pulses, and, so, they represent interesting scenarios to be studied separately. Namely, very often numerical campaigns are dedicated to simulate a single layer scenario. For example, when the behavior of these configurations under additional physics is wanted to be analyzed, it may be reasonable to simulate the responses

of the different regimes separately.
Lastly, a deeper investigation of these two configurations clearly can suggest how to improve the DLT structure.
In the following, a more detailed presentation of the two scenarios simulated is given.

3.2.1 Thin overcritical plasma

The first scenario, object of this study, is represented by a laser pulse impinging on a solid target, represented by a thin overcritical plasma. In this physical scenario, the laser is mostly reflected by the plasma, the interaction occurring in the overdense regime. Figures 3.7 and 3.8 give a visual representation of the dynamics of this kind of laser-plasma interaction.
The simulated domain consists of a square of 51.6 μm per side.
An ultra-intense laser pulse with $a_0 = 10$ is shot by an antenna, located at the left side of the domain. The pulse, which has a duration of 18 fs and a waist of 3 μm, travels along the x-direction with a Gaussian space-time profile. Assuming a wavelength of 0.8 μm, the resulting peak intensity is $I_0 \simeq 2.2\times10^{20}$ W/cm^2.
Moreover, is P polarized, i.e. its electric field oscillates along the transverse y-direction, and impinges perpendicularly on a target situated at the center of the square. This target, of 1 μm thickness, with atomic number Z=13 and mass number A=27, represents a aluminium foil, composed by electrons and ions Al^{13+}. It's electron density is 80 n_c and, consequently, the corresponding ion density amounts to 80/13 n_c . Both the electron and the ion density are sampled with 80 particles per cell.
This scenario is explored, from a numerical point of view, both with Smilei and WarpX. The physical domain is discretized with a mesh of 4096×4096 cells, involving a spatial resolution of 0.0125 μm in both the directions.
The total time of the simulation is 340 fs, split in 12194 timesteps with a time resolution of 0.028 fs circa.

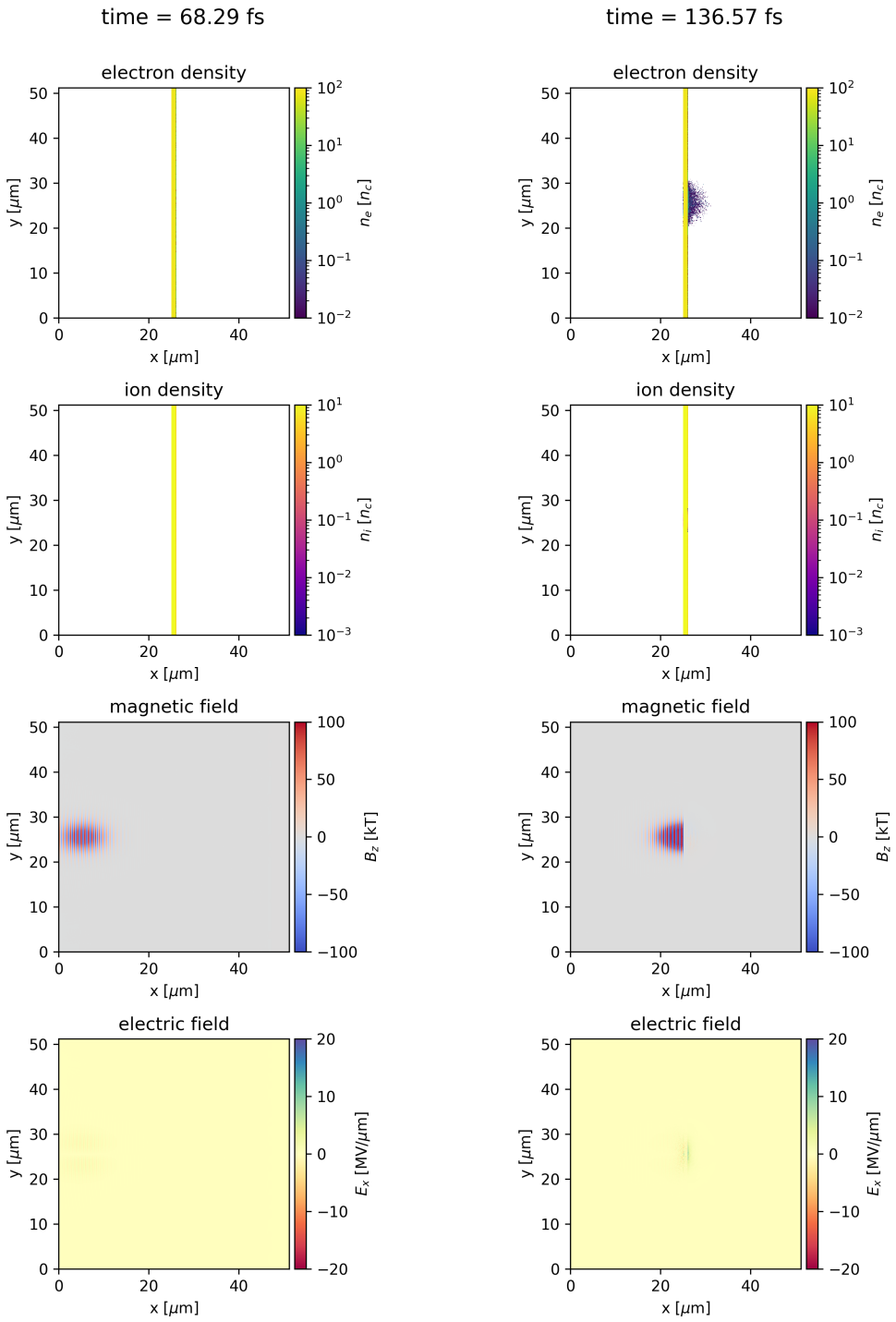

Figure 3.7: Sketches of the laser-plasma interaction simulations in the thin solid target scenario using Smilei. The figure reports the profiles of the electron density, ion density, magnetic field B_z and electric field E_z (from up to bottom) for different timesteps (from left to right: 68.29 fs, 136.57 fs).

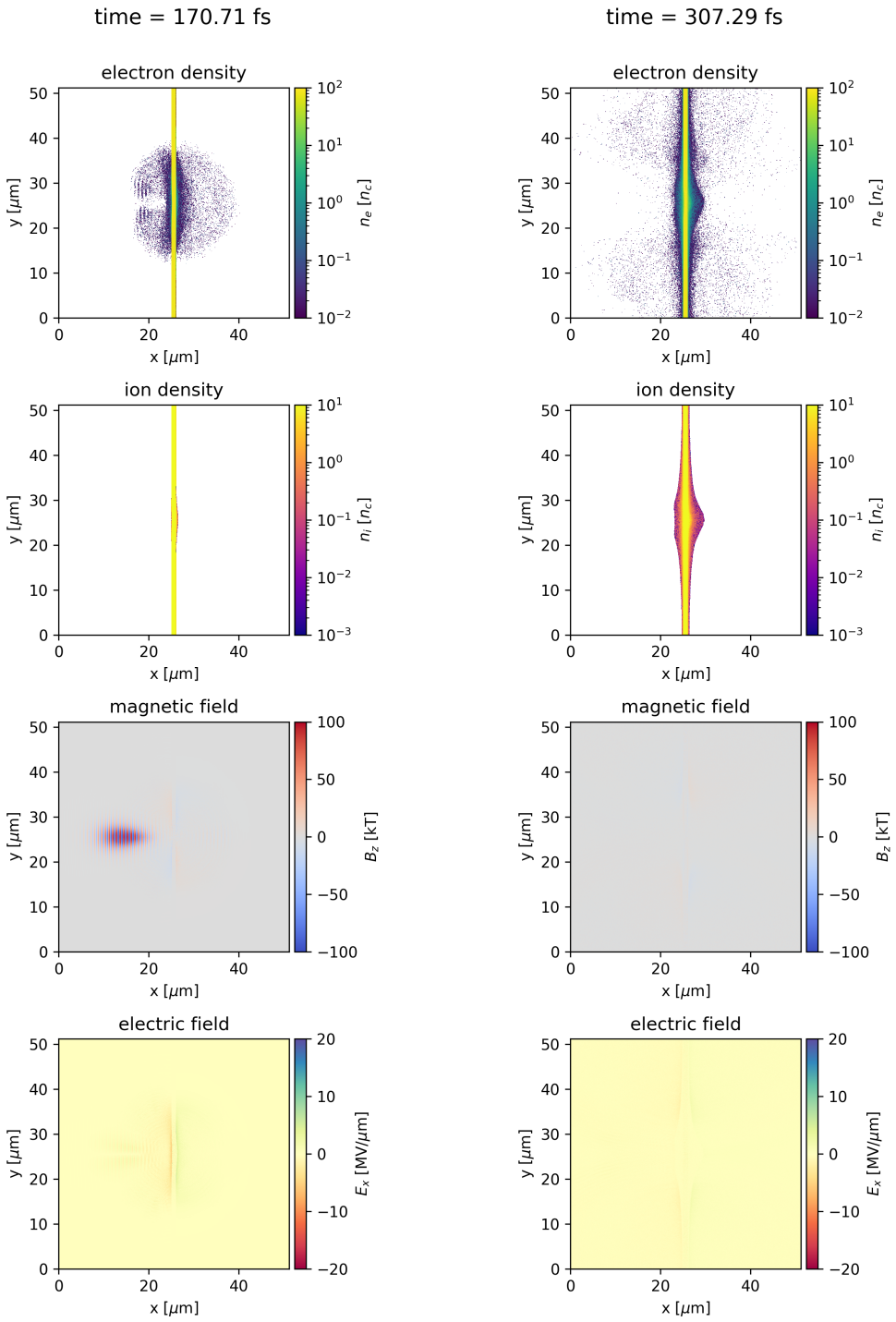

Figure 3.8: Sketches of the laser-plasma interaction simulations in the thin solid target scenario using Smilei. The figure reports the profiles of the electron density, ion density, magnetic field B_z and electric field E_z (from up to bottom) for different timesteps (from left to right: 170.71 fs, 307.29 fs).

Table 3.1 summarizes both the physical and numerical parameters.

Table 3.1: Solid target: simulations parameters

Mesh	‡
Domain	51.2 μm × 51.2 μm
Number of cells	4096×4096
Resolution	[0.0125 μm, 0.0125 μm]
	- - -
Time	‡
Simulation time	340 fs
Timesteps	12194
Δt	0.028 fs
	- - -
Laser	‡
Type	Gaussian
Incidence	0°
Normalized vector potential a_0	10
Duration w_t	18 fs
Waist $w(z)$	3 μm
Polarization	P
	- - -
Solid foil	‡
Electron density	80 n_c
Thickness	1 μm
Electrons per cell	80
Ions per cell	80

3.2.2 Thick near-critical plasma

The second scenario considered is the interaction of a laser pulse with a thick near-critical plasma, that we also refer to as foam. Figures 3.9 and 3.10 give a visual representation of the dynamics of this kind of laser-plasma interaction. It can be seen that the laser is able to penetrate and propagate along the foam. In order to reproduce the easiest case, the foam simulated has a homogeneous structure. In this configuration, the physical domain in represented by a rectangle, with length equal to 102.4 μm and height equal to 51.2 μm. In this case, an antenna, located at the left edge of the domain, shoots a high-power P-polarized Gaussian laser pulse, traveling along x-direction with a_0=20 and normal incidence, with a duration of 18 fs and a waist of 3 μm.
Moreover, assuming a wavelength of 0.8 μm, the pulse has a peak intensity $I_0 \simeq 8.7\times10^{20}$

W/cm^2.

The foam starts at 21.6 μm from the antenna and extends up to the right side of the domain, with a thickness of 80.8 μm. The foam represented is a carbon foam, composed by electrons and C^{6+} ions (the atomic number and the mass number are Z=6 and A=12). Since the foam to simulate must be a near-critical foam, the electron density is imposed to 1 n_c, and so ion density is equal to 1/6 n_c. Both the densities are sampled with 4 particle per cell. Instead, for what concerns the space discretization, the same resolution is desired in both the direction: so a mesh of 6144 $\times$ 3072 cells is created, with a spatial step of 0.0167 μm along x and y. In addition, the total time of the simulation is 340 fs, discretized with 9146 timesteps with a resulting $\Delta t \simeq 0.037$ fs.

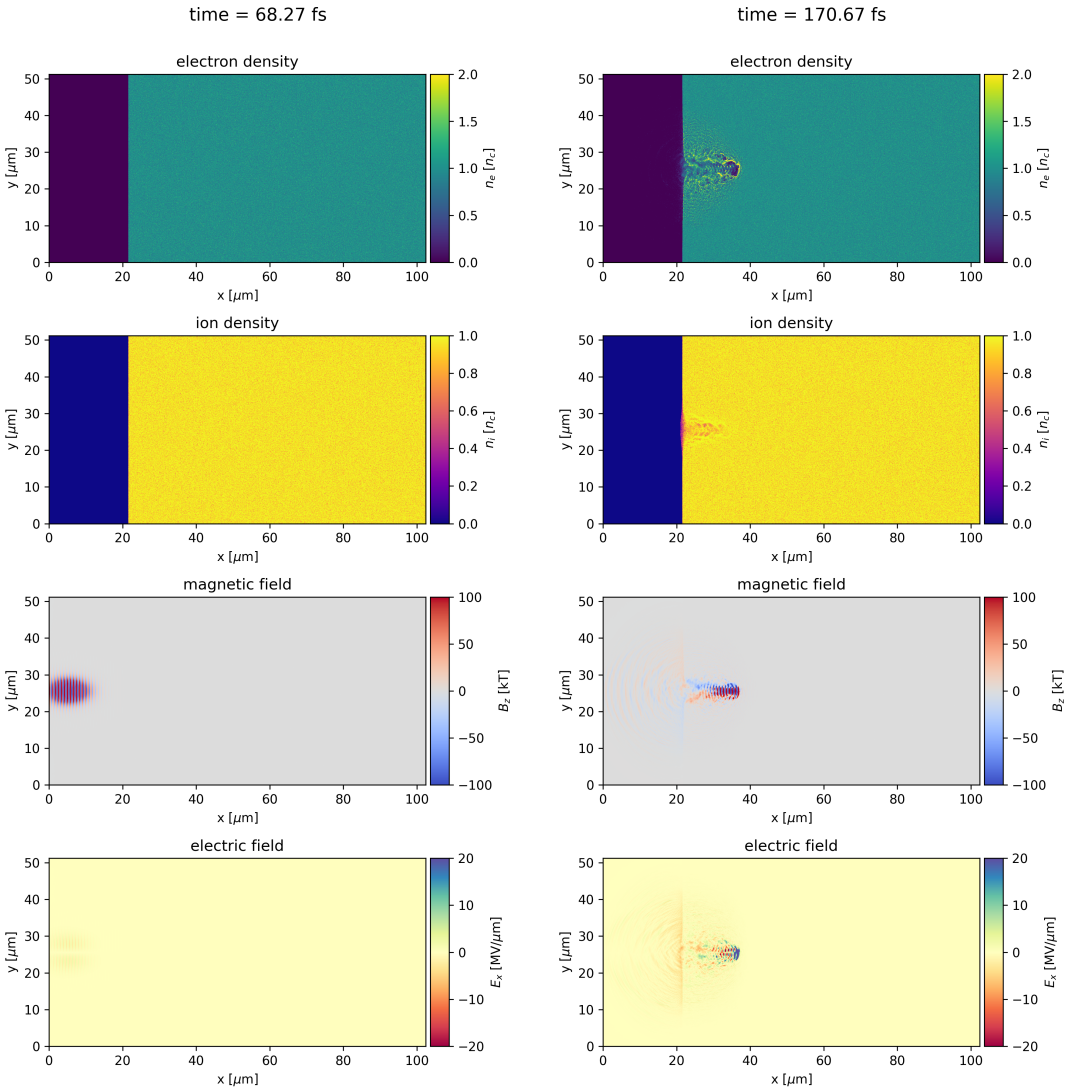

Figure 3.9: Representation of the laser-plasma interaction in the thick near-critical foam scenario using Smilei. The figure reports the profiles of the electron density, ion density, magnetic field B_z and electric field E_z (from up to bottom) for different timesteps (from left to right: 68.27 fs, 170.67 fs).

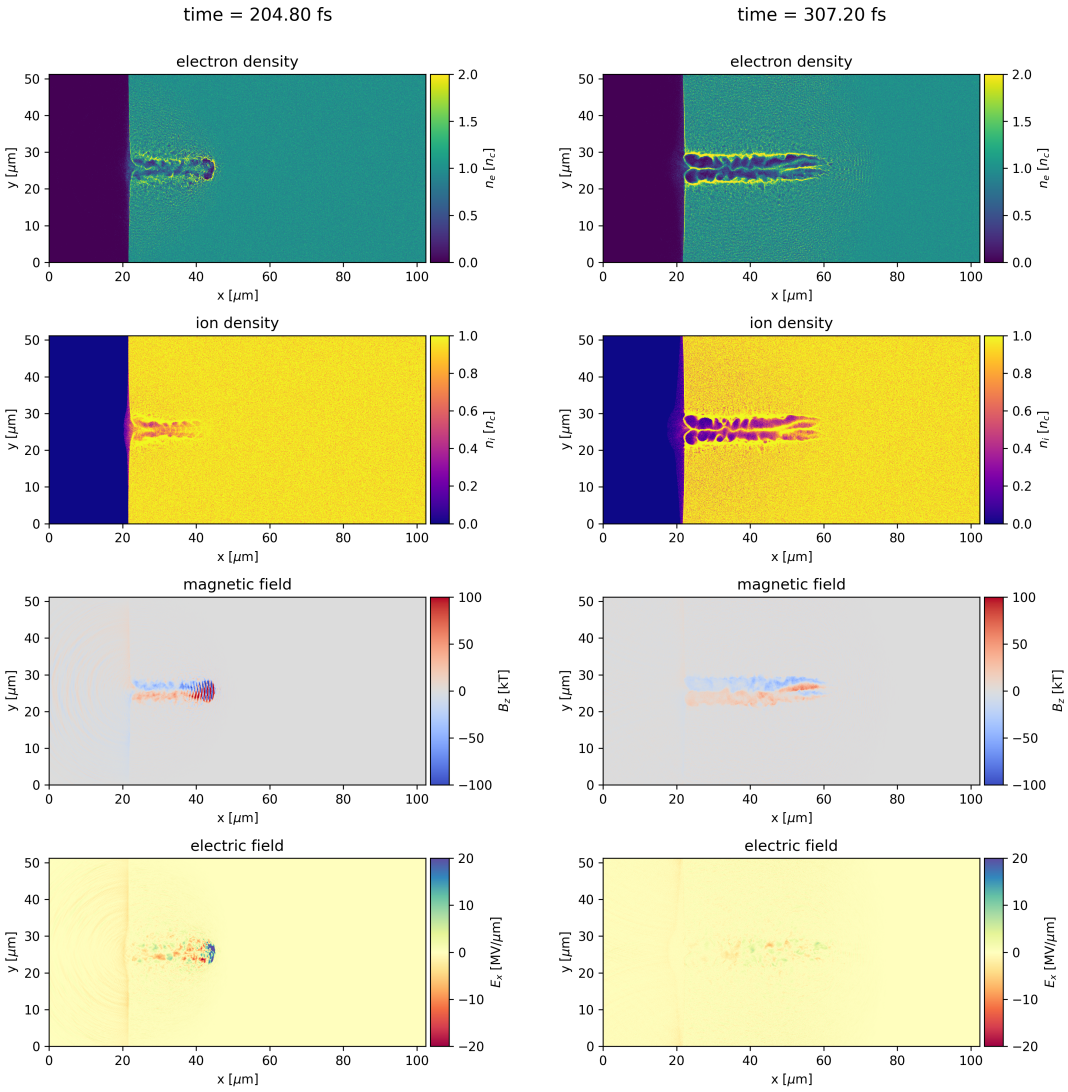

Figure 3.10: Representation of the laser-plasma interaction in the thick near-critical foam scenario using Smilei. The figure reports the profiles of the electron density, ion density, magnetic field B_z and electric field E_z (from up to bottom) for different timesteps (from left to right: 204.80 fs, 307.20 fs).

Table 3.2 summarizes both the physical and numerical parameters. Hence, the main physical and numerical frameworks of the two scenarios, common to all the numerical tests which follow, have been shown: some other parameters, characteristic of the two codes, will be explored in the following, focusing on the effects determined by their variation.

Table 3.2: Near-critical foam: simulations parameters

Mesh	‡
Domain	$102.4 \times 51.2\ \mu m$
Number of cells	6144×3072
Resolution	[0.0167 μm, 0.0167 μm]
	- - -
Time	‡
Simulation time	340 fs
Timesteps	9146
Δt	0.037 fs
	- - -
Laser	‡
Type	Gaussian
Incidence	0°
Normalized vector potential a_0	20
Duration w_t	18 fs
Waist $w(z)$	3 μm
Polarization	P
	- - -
Near-critical foam	‡
Electron density	1 n_c
Thickness	80.8 μm
Electrons per cell	4
Ions per cell	4

3.3 Numerical results

This section is devoted to the analysis of the behavior, in terms of computational impact, of several test simulations, in which different configurations will be investigated.
This numerical campaign will be conducted with the aid of the Galileo HPC, at CINECA. Table 3.3 summarizes the main technical details of Galileo's hardware.

Since the object of interest of this chapter are numerical simulations running on CPUs, only the nodes equipped with CPUs will be exploited.

3.3.1 Scaling using different multiprocessing configurations

To begin, an explorative campaign is has been lead, focused on the observation of the behavior of the simulation time when variations on the parallelization scheme and numerical parameters are applied.
First of all, some initial choices, for both the simulated scenario, in the parameters respon-

Table 3.3: Galileo: system architecture

Model	Lenovo NeXtScale
Architecture	Linux Infiniband Cluster
Nodes	1022
Processors	2 x 18-cores Intel Xeon E5-2697 v4 (Broadwell) at 2.30 GHz
Cores	36 cores/node
RAM	128 GB/node, 3.5 GB/core
Internal Network	Intel OmniPath, 100 Gb/s
Peak performance single node	1.3 TFlop/s
Peak Performance	1.5 PFlop/s
Accelerators	60 nodes with 1 nVidia K80 GPU
	+ 2 nodes with 1 nVidia V100 GPU

sible for domain decomposition are made, in both the codes.
In the following, the results achieved in the two scenarios are presented separately, with special attention to the domain decomposition.

Solid target

Let us start considering the simulations of laser interaction with a thin solid-density plasma: because of the very limited thickness of the target, with respect the full length of the domain, it is reasonable to think that a good domain decomposition may be represented by several horizontal thin stripes, i.e. rectangles with a large width and a small height. This domain decomposition, indeed, allows to divide the full target between a lot of subdomains without excessively splitting the rest of the mesh, where the plasma is absent.
However, as explained before, an explicit definition of the domain decomposition is manageable in Smilei, while in WarpX implies more difficulties, considering that the user provides only indications to the code but the effective decomposition occurs at run-time.
Specifically, in the simulations performed with Smilei, the mesh is divided in 4×128 patches, formed by 1024×32 cells while those performed with WarpX, as a first attempt, a large degree of freedom is let to the code, choosing a large `max grid size`, equal to 1024 (in both the directions), and a low `blocking factor`, equal to 4.
Moreover, concerning the OpenMP implementation, since the simulations are launched on nodes of 36 cores, 36 threads per MPI process (i.e. `OMP_NUM_THREADS` = `36`) are imposed, in order to guarantee as many threads per process as cores per node are used.

Now, we analyze the numerical results without involving load balancing algorithms. The first analysis consists in exploring different parallelization patterns, varying the number of nodes involved (1,2,3,4) and, consequently, the number of MPI processes (36,72,108,144).

The main focus of this experiment is to observe the scaling of the computational times when increasing the number of available ranks. Clearly, in order to run a simulation, it is necessary that the operating cores are reserved for the entire duration of the calculations. To be more specific, Galileo quantifies the computational cost related to a simulation by computing the product between the duration of the simulation and the number of cores demanded. So, these computational costs, also referred to as cores hours, correspond to the amount scaled from the residual budget.

Figure 3.11 shows both the computational times and costs used for all the performed

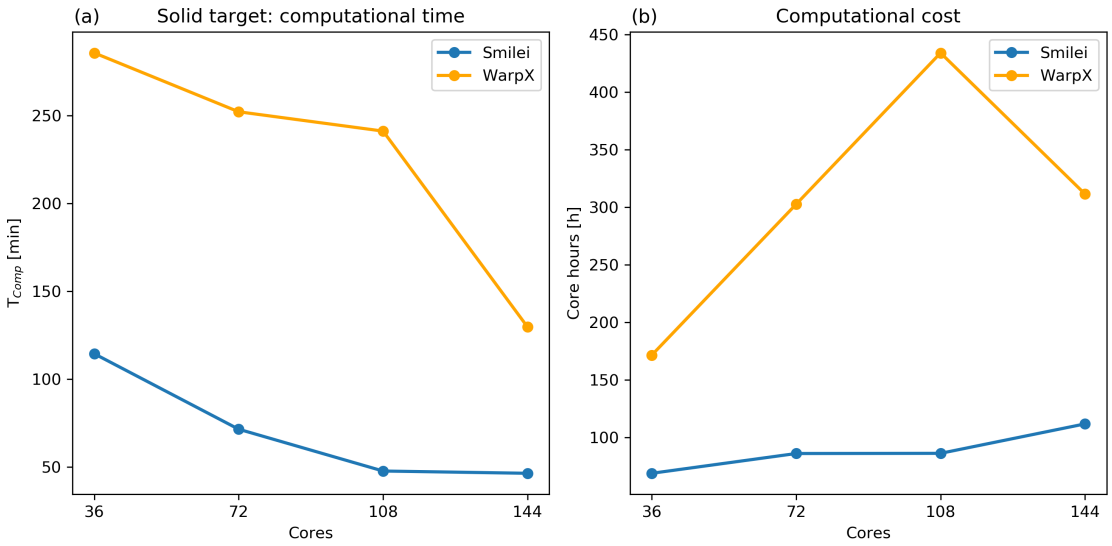

Figure 3.11: (a) Computational time required for the thin solid foil simulations using Smilei (blue line) and WarpX (orange line) as a function of the exploited number of cores. (b) Computational cost (i.e. computational time $\times$ number of cores) spent for the thin solid foil simulations using Smilei (blue line) and WarpX (orange line) as a function of the exploited number of cores.

simulations. As expected (see figure 3.11(a)), the massive parallelization offers consistent benefits, allowing to conclude the computations faster and faster when using more nodes, with both codes. On the other hand, also the core hours must be taken in consideration. From a purely budget point of view (see figure 3.11(b)), namely, it emerges that the most convenient configurations is the one making use of 1 node only. Hence, the optimal number of nodes is not straightforwardly known a priori but, depending on the context, a suitable number of cores should be established as a compromise between the desired speed and the available budget.

Moreover, running the simulations with the parameters mentioned before, the behavior of the computational cost, using WarpX, shows a peak when using 3 nodes, i.e. 108 cores, with a subsequent decline with 4 nodes.

The figure highlights that the same simulation costs more than twice if going from 1 to 3 nodes and using 3 nodes is even more expensive than using 4 nodes. This behavior is

not straightforward to expect. Actually , WarpX defines different domain decompositions when using different number of nodes. In particular, the simulation box is decomposed in 64, 128, 128 and 256 number of subdomains for 1, 2, 3 and 4 nodes respectively.
Smilei shows a trend that is easier to understand, consisting in a decreasing profile and in an increasing profile for the computational time and cost respectively. In particular, the same simulation requires less than half the time going from 1 to 4 nodes with an increment in the computational cost of a factor close to 1.6.
Hereinafter, we investigate the computational times and costs when tuning different numerical parameters other than the number of nodes, which we fix equal to 2, in order to avoid both excessive durations and wastes of budget.
Based on the previous results obtained with WarpX, we investigate the role of the specific domain decomposition adopted with both codes. This is clearly an interesting point in order to verify the previous considerations about the best domain decomposition. For these reasons, several configurations, considering different sizes for the subdomains, have been tested in both Smilei and Warpx.

Figure 3.12(a) shows the computational time for the simulations performed with Smilei,

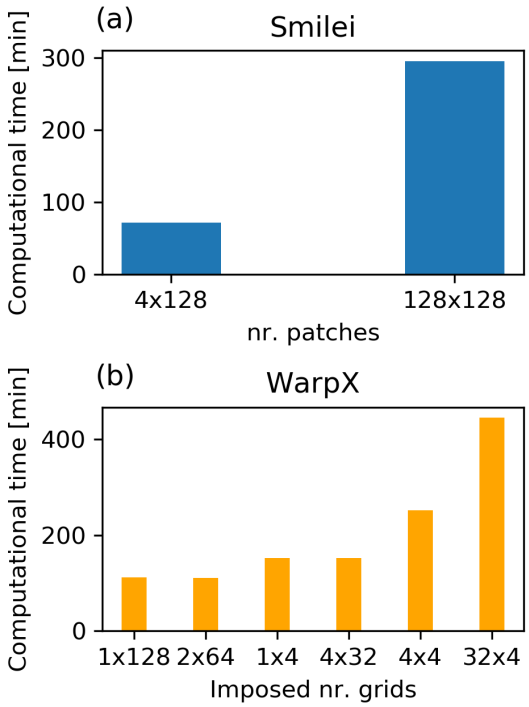

Figure 3.12: a)Computational times required for the thin solid foil simulations using Smilei with different number of subdomains. b) Computational times required for the thin solid foil simulations using WarpX with respect to different number of selected subdomains. The effective number of subdomains operated by WarpX are 1×128,2×64,4×32,4×32,8×16, 32×4 respectively.

whereas the results achieved with WarpX are shown in figure 3.12(b). These results seem

to confirm the initial assumption showing that it's very beneficial to have few subdomains along the x-direction (along the plasma occupies a small region) and several subdomains along the y direction (along which the plasma occupies the full space): in particular, it is possible to decrease the simulation time up to a factor of 3 in Smilei and 4 in WarpX when changing the subdomain decomposition with a clever set up.

So far, none strategy of load balancing has been considered intentionally, in order to define a clear picture of what happens when different strategies of mesh decomposition are implemented. Indeed, invoking load balancing strategies – focused on redistributing the subdomains among the processes to ensure a fair balance – could reduce the discrepancies between the performances obtained with different numerical setups.
This being said, now that we have understood the role of the domain decomposition, the analysis of the effective benefits coming with a well balanced scheme clearly acquires significance. For this purpose, also the balancing logic has been investigated in both the codes.

Figure 3.13 shows a comparison between simulations, performed by Smilei and WarpX,

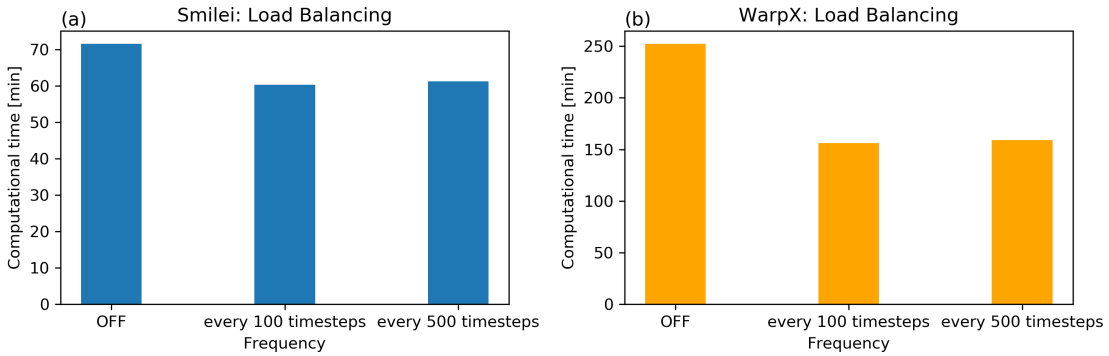

Figure 3.13: Computational time required for the thin solid foil simulations using Smilei (a) and WarpX (b) when load balancing schemes are I) not activated II) activated every 100 timesteps III) activated every 500 timesteps

with and without load balancing. These results have been obtained with the original domain decomposition: in Smilei the mesh has been decomposed in 4×128 patches, while a high degree of freedom is left to WarpX, resulting in 8×16, in order to not allow the code to handle the load balancing without previous constraints.
In this case, it's quite clear that applying load balancing strategies is successful, especially if a non-optimal scheme has been implemented for the creation of the subdomains. In fact, as shown in figure 3.13 (a), a slight improvement is registered in the results obtained with Smilei ($\sim$ 20% of time saved), while the simulations performed with WarpX, where a not wise decomposition strategy has been selected, reveal a distinct improvement ($\sim$ 40% of time saved).

Lastly, it may be interesting to study the effects resulting from a modification in the target position within the domain. In this regard, additional simulations with a non-centered solid target configuration have been performed. Namely, generally simulations involving laser interaction with a solid target do not take particular care in exploit a symmetric scenario: we want to observe if, instead, this point should be taken in consideration.

So, the same values of the previous cases have been set for the domain decomposition and

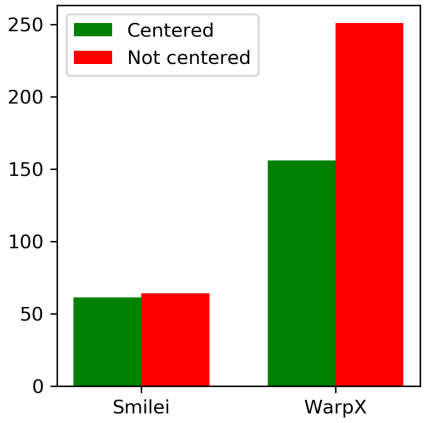

Figure 3.14: Comparison between the computational time required for the thin solid foil simulations using Smilei and WarpX in a non-centered (red column) and centered (green column) framework.

the balancing scheme is activated.

Essentially, the simulations run with Smilei do not exhibit significant alterations. On the other side, a symmetrical configuration helps WarpX – where a non ideal decomposition has been set – handling the subdomains assignment to the MPI processes and so avoiding useless wastes of time ($\sim 40\%$ of time saved).

Near-critical plasma

In this section the physical scenario under study is the laser interaction with a thick near-critical plasma. In this case, considering the homogeneous structure of the foam, which occupy long portions of the mesh in both the direction, not a particular shape of the sub-domains seems to be smart to set.

So, the mesh is, initially, subdivided in squared patches of 128×128 cells for the simulations carried out by Smilei, while grids with `max grid size` equal to 1024 are generated, with a `blocking factor` equal to 4 (in order to let the code to be free enough in operating the grids creation).

As in the previous scenario, since a number of threads per core equal to the number of cores per node are desired, each node is handled by 36 threads (i.e. `OMP_NUM_THREADS` = `36`).

In addition, no balancing effects are desired for now: namely the first object of interest is the time scaling in basic configurations, which is more easy to observe if no balancing

benefits are involved.
At this point, let us begin exploring how the computational time depends on the number of MPI process selected. The first analysis compares different parallelization patterns, with different number of nodes involved (1,2,3,4) and, consequently, different number of MPI processes (36,72,108,144). Figure 3.15 highlights both the computational times and costs used for all the performed simulations.
It is clarified (see 3.15(a))that the increasing the number of ranks (at least until the maximum number tested, corresponding to 144 cores) brings considerable reduction to the time required for the computations. On the other side, the importance of the computational cost required by each simulation is equally important, when the parallelization configuration is decided.
By the way, figure 3.15(b) shows that the results achieved using Smilei register a slight increment of the computational costs ($\sim$ 10% extra when going from 1 to 4 nodes), with respect to the number of nodes demanded. Instead, WarpX presents an almost constant profile, when 1,2 or 4 nodes are involved, while demanding 3 nodes implies a considerable additional cost ($\sim$ 40% extra). Therefore, after these considerations, for the following sim-

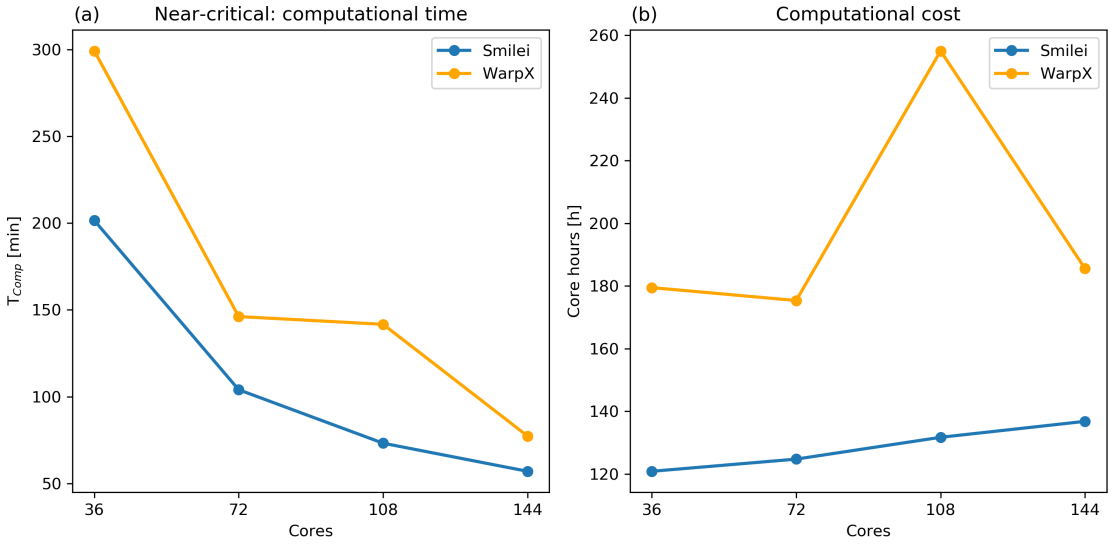

Figure 3.15: (a) Computational time required for the thick near-critical foam simulations using Smilei (blue line) and WarpX (orange line) as a function of the exploited number of cores. (b) Computational cost (i.e. computational time $\times$ number of cores) spent for the thick near-critical foam simulations using Smilei (blue line) and WarpX (orange line) as a function of the exploited number of cores.

ulations 72 computing cores, i.e. 2 nodes, are exploited, in order to spend with moderation the available budget of CPU hours and to avoid excessive long computations.
At this point, as in the previous case, let us explore other choices for the domain decomposition, with the aim to observe how the codes respond to modifications on the subdomains

creation. The plots reported in 3.16 show the behaviors when modifications on the number and shape of the subdomains are applied. More precisely, the shape of the patches seems not to be relevant, while an overly abundance of patches appears to be counterproductive: in particular, the simulation increases up to a factor of 1.4 if too many subdomains are created. Whereas, in this case WarpX shows to be able to manage the decomposition, without great differences when the imposed number of grids is modified.
Therefore, in the following the original shaping of the subdomains are preserved.

As mentioned above, so far no load balancing has been introduced. Thus, next goal

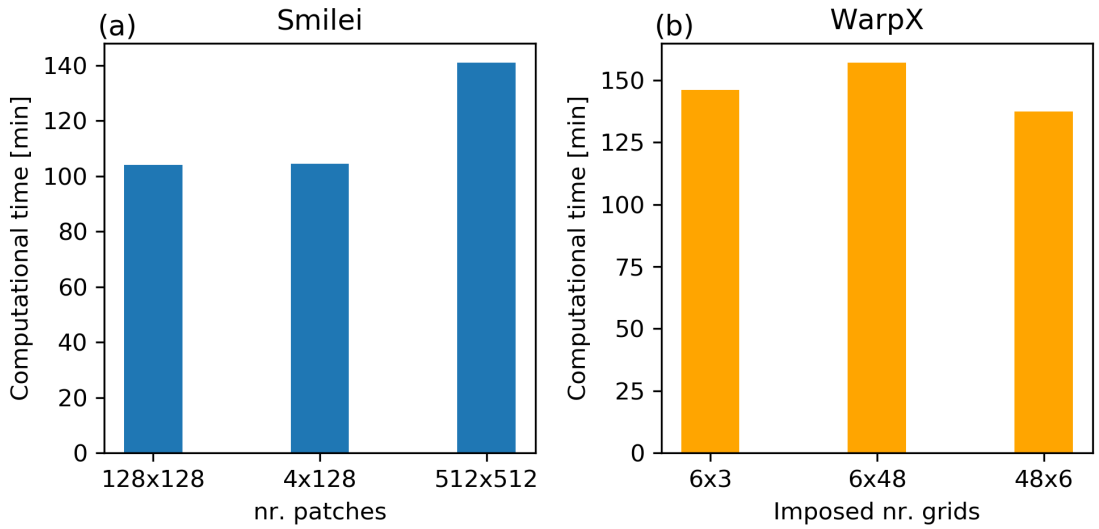

Figure 3.16: a)Computational times required for the thick near-critical simulations using Smilei with different number of subdomains. b) Computational times required for the thick near-critical simulations using WarpX with respect to different number of selected subdomains. The effective number of subdomains operated by WarpX are 12×6,6×48,48×6 respectively.

consists precisely in pointing out how much invoking a periodical redistribution of the subdomains between the processes can impact on the final results. In this context, different balancing setups have been compared in both the codes. However, not consistent enhancements have been achieved(see figure 3.17): namely, an activation on Smilei allow to save not much more of the 10% of the time, while the benefits felt by WarpX are practically null.

3.3.2 Investigation on PIC-routines and computational load

In the previous section, a numerical investigation, focused on an exploration of the reaction of the two codes to some perturbations in the numerical scenarios, has been carried out. So far, the debate revolved around only the global computational time required by the simulations, i.e. the effective time during which Galileo has reserved its nodes for performing computations. Nevertheless, in order to realize a simulation, not only routines related to

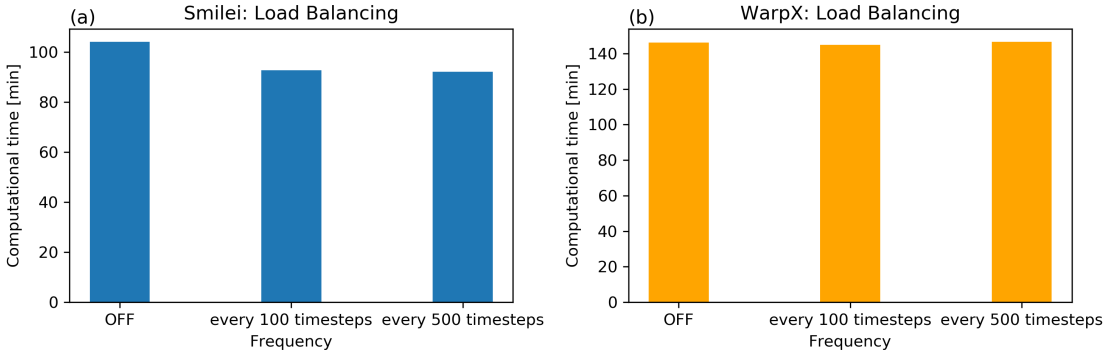

Figure 3.17: Computational time required for the thick near-critical foam simulations using Smilei (a) and WarpX (b) when load balancing schemes are I) not activated II) activated every 100 timesteps III) activated every 500 timesteps

the PIC algorithm are invoked: there are many other procedures which, clearly, require extra time. Indeed, since a massive parallelization has been exploited, a lot of inter-processes communications play a role: substantially, part of the time is spent in allowing the ranks to exchange information at each step of the simulation. Moreover, communication times are not the only operations taking time. Several other activities are also included in the global time: some of them are, for example, the periodical updates of the diagnostics files, the initialization of the mesh, the creation of the particles, the video-output of the current status (when the simulation is running) etcetera.
This section, instead, sets itself the objective of an accurate examination of the time spent in performing the computations related to the different pieces of the PIC scheme. In addition, also the time required for the MPI communications represents an important term to keep under observation. Both the codes enable an analysis of this kind, since they save these information in specific output files.
Given the high number of processes implicated, the same operations can be faster or slower, depending on the region in which they are executed. Hence, recalling the fact that different MPI regions are assigned to different ranks, the time required to a specific activity varies a lot, depending on the process involved. To be more specific, Smilei provides a file, named "profil.txt", where for each timestep, the cumulative time required for each PIC routine and for the operations which involve communications: actually, the minimum, maximum and mean time, between the processes, is registered for each timestep. WarpX, instead, provides the total time (as output at the end) required by every kind of routine necessary for accomplishing a simulation.
Hence, after an investigation in the source codes, the terms related to PIC routines and communications have been discerned. In particular, given the role covered by the particles and the grid in Particle-In-Cell methods, it seems wise to split the computational time required by PIC scheme in two components.

The first component, named T_{part}, must take into account all the operations which involve particles. Clearly, the first contribution is represented by the Boris pusher algorithm, which is responsible for the computation of the macro-particles trajectories. Moreover, this term should include the time required for the fields interpolation, since the values of the electromagnetic fields are estimated on the position occupied by the particles. Finally, the deposition operations should also be considered, in which the densities and the currents, starting from the particles positions, are projected on the grid.
On the other hand, the second component, named T_{grid}, has been defined to consider the computations performed on the grid points, without considering the macro-particle locations: so, this term basically coincides with the time required for the Maxwell solvers, i.e. the time updating of the electromagnetic fields.
For what concerns the time spent in the communications among processes, the term T_{comm} has been designed to quantify all the routines involving the MPI calls, which substantially take care of synchronizing particles and fields data.
To provide a mathematical framework, the total computational time T_{comp}^{seq}, considering a sequential environment (i.e. only one MPI process and one OpenMP thread), is defined as:

$$\begin{cases} T_{comp}^{seq} = T_{1step}^{seq} N_{step} \\ T_{1step}^{seq} = \alpha N_{cell} T_{1grid}^{seq} + \beta N_{part} T_{1part}^{seq} + T_{other} = \alpha T_{grid}^{seq} + \beta T_{part}^{seq} + T_{other} \\ N_{step} = \dfrac{T_{sim}}{\Delta t} \\ \Delta t = \dfrac{C_{cfl}}{c} \dfrac{\Delta}{n} \\ N_{cell} = \left(\dfrac{L_{sim}}{\Delta} \right)^n \\ N_{part} = PPC \times N_{cell} \end{cases} \tag{3.1}$$

where T_{1step}^{seq} is the computational time required to perform 1 step (roughly assuming that each step implies the same computational time), N_{step} is the total number of steps, $T_{1grid} = \dfrac{T_{grid}}{N_{cell}}$, $T_{1part} = \dfrac{T_{part}}{N_{cell}}$, α and β are two suitable constants, N_{cell} is the total number of grid cells, N_{part} is the total number of macro-particles, T_{sim} is the simulated time of the system evolution (i.e. $\sim 10^2 - 10^3$ of fs), Δt is the timestep duration, Δ is the grid cells length (assuming same resolution along every direction), C_{cfl} is the Courant number, c is the speed of light in vacuum, n is the dimensionality of the simulation, L_{sim} is the simulated box size in each dimension (assuming a cubic geometry), PPC is the number of macro-

particles per cell and T_{other} is the residual computational time (which is out of the scope of this investigation). Once that these contributions have been defined, it is advisable to perform some simple test simulations, in order to verify the behavior of this terms when the number of particles and cells changes.
Since we will work on a massively parallelized environment, it should proper to generalize the previous model for the computational time as:

$$T_{comp}(p) = \left(\frac{T_{comp}(1) N_{step}}{p} \right)^{\gamma} + T_{comm} \tag{3.2}$$

where p is the number of MPI processes, $\gamma > 1$ is a suitable constant and $T_{comp}(1) \equiv T_{comp}^{seq}$ defined above. In this simple description we are assuming that every process takes the same amount of time in performing the simulation, so that this can fairly represent the computational time averaged over the p processes.
To check the soundness of this formulation, we performed some simple simulations involving a circular thermal plasma, where no incident waves are injected so that the macro-particle motion is ascribable only to the kinetic energy of the populations. In these simulations we have changed the values of PPC and N_{cell}. The underlying idea consists in separately doubling both the resolution and the PPC to examine the evolution of the contributions of T_{part} and T_{grid}. Specifically, a reference configuration is performed with PPC=20 and a resolution of 40 points per μm in each direction (see the table REF for more details).

In addition, other two simulations have been executed. The first doubling PPC and keeping the same resolution, while the second with a resolution of 55 points for μm in both the directions and PPC=10. It is important to underline that, when doubling the resolution, the actual number of cells is little less than doubled (with a ratio of 1.9 circa instead of 2). Moreover, this second case deserves a further consideration: indeed, doubling the number of cells would imply also a double number of particles, since the PPC is defined on a single cell, that is why we halved PPC=10. In addition, one should no forget the time discretization is related to the the spatial resolutions by the CFL condition (2.11): therefore, this second configuration performs the calculations a greater number of steps ($\times$ 1.4), resulting in a further increase of run-time.
The results obtained with both the codes are summarized in figure 3.18 and 3.19, where the averaged time, between the processes, have been plotted.
[hbt] Figures 3.18 (a) and (c) show the behavior of T_{part} when doubling PPC, thus N_{part} for both Smilei (blue) and WarpX (orange), respectively. It can be seen that a double number of macro-particles simulated implies a doubling in the computational time related to the operations on particles (T_{part}). Besides, figures 3.18 (b) and (d) illustrate that when the number of cells increases and the total number of particles is unaltered, then T_{part} increases only because of the increment of the timesteps number. More precisely it increases by a factor close to 1.3, which is in agreement with the expectations.

Table 3.4: Check: simulations parameters

Mesh	**Reference – Doubling Particles – Doubling cells**
Domain	25.6 μm × 25.6 μm
Number of cells	1024×1024 – 1024×1024 – 1408×1408
Resolution	[0.025 μm, 0.025 μm] – [0.025 μm, 0.0125 μm] – [0.018 μm, 0.018 μm]
	- - -
Time	‡
Simulation time	85 fs
Timesteps	1524 – 1524 – 2096
Δt	0.056 fs – 0.056 fs – 0.041
	- - -
Plasma	‡
Center	[12.8 μm, 12.8 μm]
Radius	10 μm
Electron density	10 n_c
Ion density	10/6 n_c
Electrons per cell	20 – 40 –10
Ions per cell	20 – 40 – 10

On the other hand, figure 3.19 (a) and (c) show no dependence of T_{grid} on the amount of macro-particles (for both codes). However, when the resolution is modified, it is reasonable to expect that T_{grid} increases as a consequence of the modifications of both N_{cell} and N_{steps} by a factor close to $1.9\times \ \ 1.4 \simeq 2.7$. However, figures 3.19 (b) and (d) show that both the codes reveal results which slightly deviate from this estimate. This can be explained by the following argument: when the resolution has been changed, the numerical parameters responsible for the domain decomposition (the number of patches and the `max grid size` in Smilei and WarpX respectively) have been kept unchanged. So, basically, these mutations of the number of cells contained in the patches (in Smilei) and of the shape of the grids (in WarpX) imply that the previous estimate cannot fully capture the actual behavior of the codes, which may handle better or worse the subdomains.

These achieved results are interesting for a twofold reason. Firstly, they constitute a consistency check which confirms that the particles and the grid computations have been isolated in a reasonable way. Secondly, it is fair to assume a linear dependence of T_{part} and T_{grid} on N_{part} and N_{cell} respectively. Hence, the latter point allows one to consider normalized times and to compare even different configurations (in terms of PPC and resolution) in a meaningful way.

At this point, it is appropriate to proceed with a deeper analysis of the results. In the following, the two physical scenarios will be examined separately.

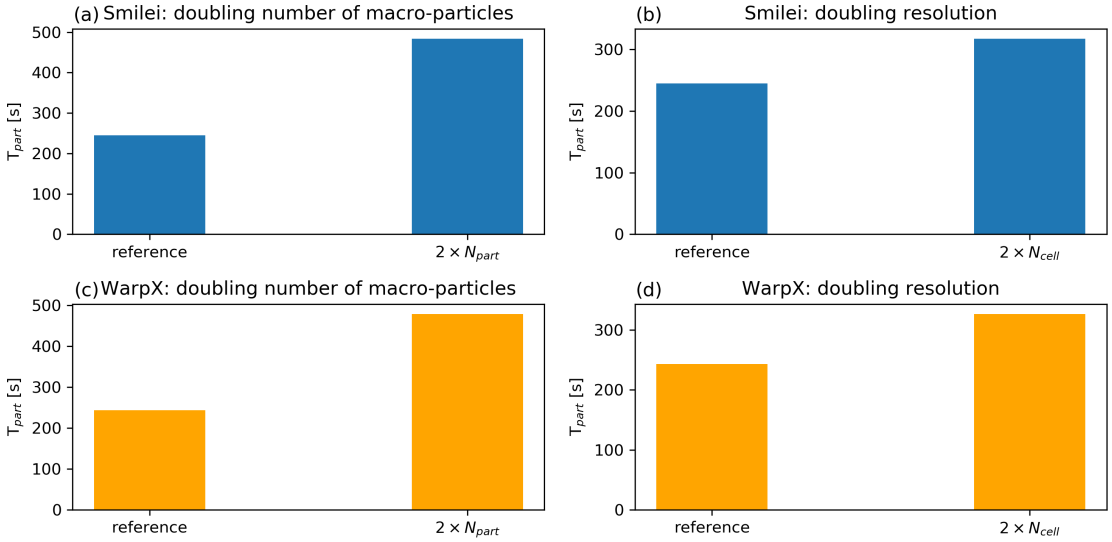

Figure 3.18: Behavior of T_{grid} when the number of macro-particles (left column) and the number of cells (right column) are doubled, using Smilei (first row) and WarpX (second row). These results are achieved in a pure MPI environment exploiting 2 nodes (i.e. 72 cores).

Solid target

In order to maintain the same structure of the previous section, the first scenario presented is the solid target environment. Thus, thanks to the previous investigation on the particle-time and grid-time components, it is possible to differentiate the contribution in the solid target simulations.
So, a more detailed investigation has been carried out considering the results given by the basic configuration, when load balancing algorithms have not been activated: the object of interest is represented by the behavior of the several components when the number of MPI processes varies. The same examination, obviously, has been conducted for the results provided both by Smilei and WarpX.

Figure 3.20 shows the profiles of T_{part}/N_{part}, T_{grid}/N_{cell}, T_{comm} and the sum $T_{part}+T_{grid}+T_{comm}$ in each column from left to right respectively. The top row shows the results for Smilei and the second row for WarpX. The calculations related both to the particles and to the grid become faster and faster, on average, as the number of cores increases. Namely, the MPI regions, as the number of processes increases, become smaller and, consequently, also the portion of data administered by each process decreases. This means that the number of macro-particles moving in a MPI domain is being reduced if a stronger parallelization is exploited, resulting in a particle-time reduction. An equivalent consideration can be applied to the grid-time behavior, since also the number of cells managed by a single rank becomes smaller. The price to pay resides in the efforts required by communication routines. As

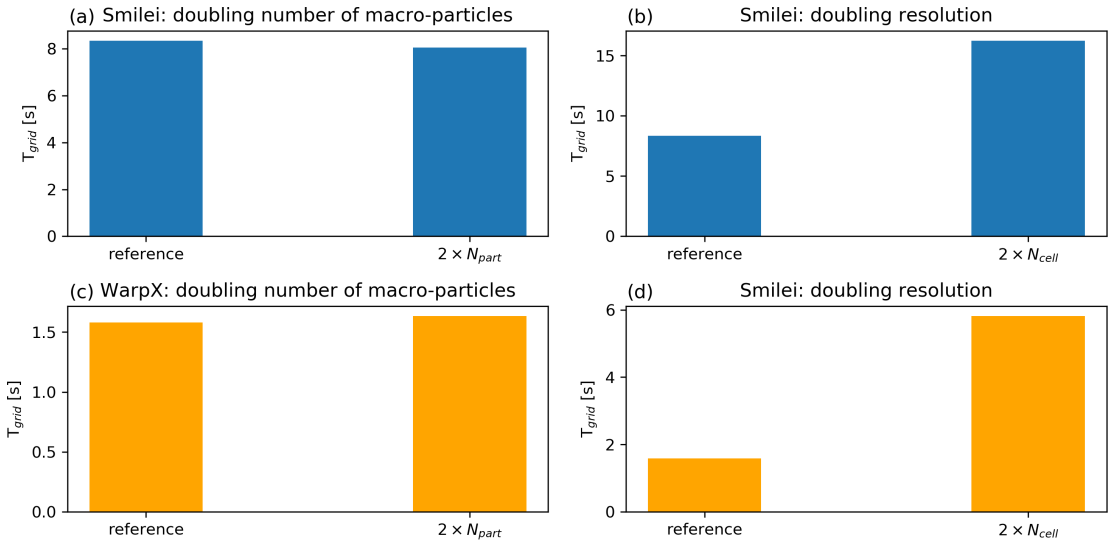

Figure 3.19: Behavior of T_{part} when the number of macro-particles (left column) and the number of cells (right column) are doubled, using Smilei (first row) and WarpX (second row). These results are achieved in a pure MPI environment exploiting 2 nodes (i.e. 72 cores).

mentioned before, both these PIC codes involve communications between processes, hence overheads, when the parallelization is activated in order to exchange information along the boundaries of the MPI domains or to synchronize different processes. Otherwise, if the number of nodes changes, also the MPI regions change in size and shape, and so the boundaries depend on the specific configurations considered. So, what is interesting to study is not the effective duration required for communicating, but rather the computational costs (intended, as always, as $time \times cores$). For this reason we also show the computational costs, associated to each type of routine, in figure 3.21. This figure shows a reasonable ascending trend, for the computational costs required by communications in Smilei. Instead, with WarpX a less obvious profile appears: indeed, the cost spent for MPI communications grows if the number of nodes increases up to 3, as one could expect, but it abruptly drops down when 4 nodes are requested. Hence, this case requires a deeper investigation. For one thing, it should not be underestimated the fact that, as explained in 3.1.2, WarpX does not receive an actual instruction of how to perform the box decomposition, as opposed to Smilei; rather, the grids creations is operated at run-time, according to the constraints imposed by the user. Therefore, the effective shape of the grids created in the four simulations has been studied more in detail. As expected, not the same decomposition has been applied: table 3.5 summarizes which choices have been done at run-time.

Thus, it may be interesting to examine how the situation changes if the same decomposition is imposed in the four parallel configurations. As a first approach, WarpX has been

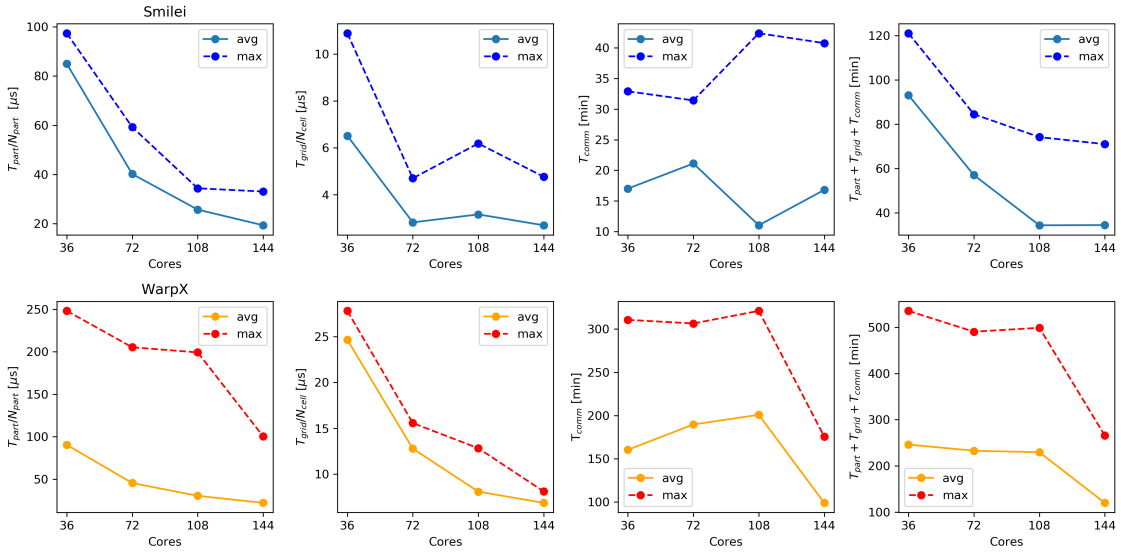

Figure 3.20: Profiles of T_{part}/N_{part}, T_{grid}/N_{cell}, T_{comm} and the sum $T_{part} + T_{grid} + T_{comm}$ in each column from left to right respectively required for the thin solid foil simulations. The maximum (dashed lines) and the average (solid lines) values between the MPI processes are plotted. These profiles are shown for Smilei (top row) and WarpX (second row).

Number of cores	**Grids size [number of cells]**	**Number of grids**
36	512×512	8×8
72	512×256	8×16
108	512×256	8×16
144	256×256	16×16

Table 3.5: Grid decomposition performed by WarpX in the four configurations with 1,2,3,4 nodes (i.e. 36,72,108,144 cores)

forced to generate grids of size 512×256 (number of cells), since this decomposition was adopted by the code in half the cases. The obtained results are plotted in figure 3.22: it is easy to observe that the situation almost remains unaltered, showing the same drop in communication costs when 144 cores are running.

At this point, recalling that, as tested in the previous section, an efficient decomposition strategy consists in creating few grids along x and many more grids along y, the following strategy has been checked. The domain has been forced to split in grids of 2048×64 cells, i.e the configuration boosting the best performances among those experimented in the previous campaign. In this case, as reported in 3.23, the average computational cost exhibits a behavior similar to the previous cases (i.e. with less efficient grid decompositions), with a peak when 108 cores are used with a subsequent fall.

But more interesting is the profile of the maximum cost (purple line), which monotonically increases with the number of cores. Therefore, probably most of the exchanged

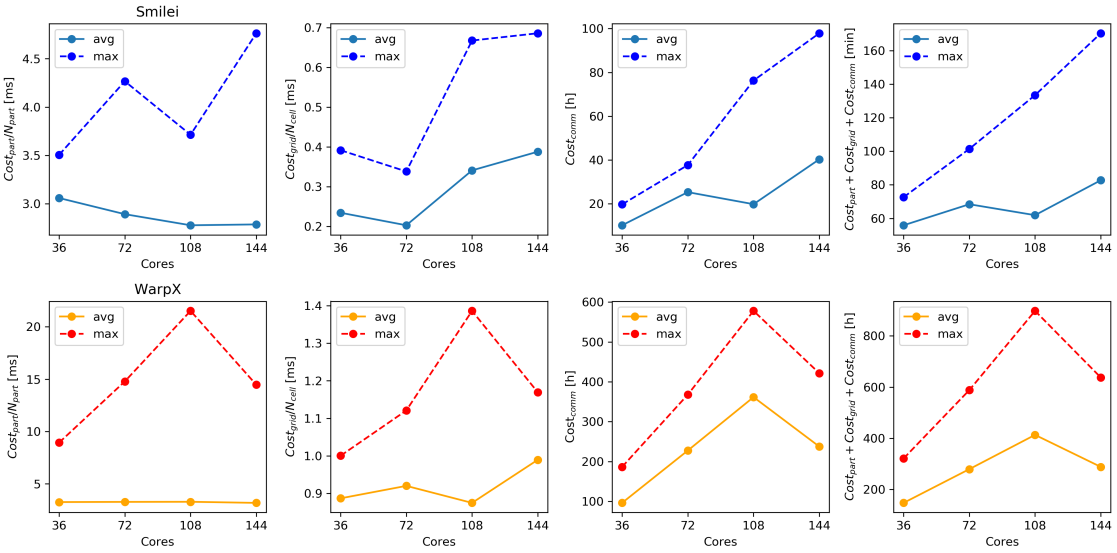

Figure 3.21: Profiles of C_{part}/N_{part}, C_{grid}/N_{cell}, C_{comm} and the sum $C_{part}+C_{grid}+C_{comm}$ in each column from left to right respectively required for the thin solid foil simulations. The maximum (dashed lines) and the average (solid lines) values between the MPI processes are plotted. These profiles are shown for Smilei (top row) and WarpX (second row).

information occurs between the central processes, which handle regions having a high density of macro-particles. So, in this case, the behavior of the maximum cost – associated to the overloaded regions – may be more meaningful than the average one. Indeed, the high number of subdomains which do not contain any plasma and do not require particles synchronization contribute to lowering the mean cost.
Overall, from the results obtained with different griddings (automatic and the two forced), we can conclude that the optimal configuration is able to minimize also the communication contributions to the costs. Indeed, looking at figure 3.23 is easy to that the average (green line) and maximum (purple line) communication profiles required by the optimal forced environment are very lower than the ones involved in a non-optimal environment. So, the optimal decomposition manage to minimize the communications in each parallelization configuration (1,2,3 and 4 nodes): so the resulting profile for the maximum cost will be clearly monotonically increasing (because the number of MPI processes increases), while a non-optimal environment is not able to minimize these costs in each parallel configuration (producing bizarre communications cost profiles).

Incidentally, another important aspect highlights from these plots. Namely, the maximum computational particle-time profile of the forced 2048×64 scheme is very close to the average one, while the automatic configuration shows a maximum profile with values very greater. On the other hand the grid-time behavior is more or less the same in both the cases. This suggests one important point: the load represented by particle routines is better distributed in the forced mesh, whereas, when an automatic decomposition occurs, there

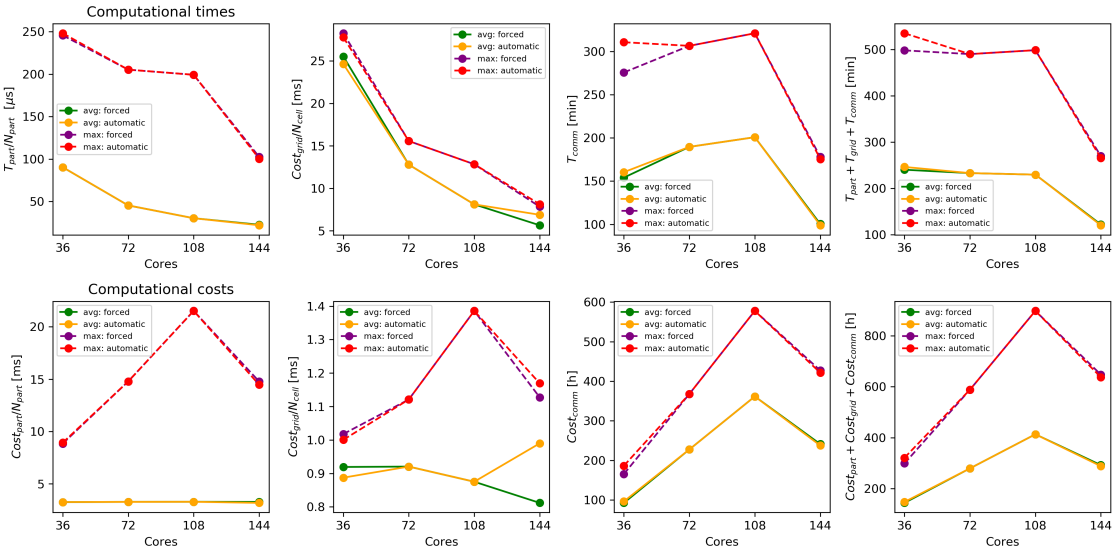

Figure 3.22: Computational times and computational costs required for an automatic and a forced (grids of 512×256 cells) domain decomposition in WarpX.

are few processes with are much more slower in performing computations related to particles. Moreover, the communications required in the original pattern require much greater costs. In fact, it appears plausible that in this specific scenario, where a high-density plasma covers a minuscule portion of the physical domain, the most of the communication costs are due to particle motion between the MPI regions.
Therefore, it seems appropriate, for multiple reasons to let to the code the least possible degree of autonomy in the grids generation.

Near-critical foam

The same approach adopted for the solid target scenario has been expanded for a deeper investigation of the results provided by near-critical simulations exploration. So, also in this case, the basic configuration has been considered, without involving load balancing between nodes. Also in this case, so, the computational times spent for particles-routines T_{part}, grid-routines T_{grid} and inter-nodes communications T_{comm} have been separated and, obviously, the proper normalization has been performed.

Figure 3.24 shows the profiles of T_{part}/N_{part}, T_{grid}/N_{cell}, T_{comm} and the sum $T_{part}+T_{grid}+T_{comm}$ in each column from left to right respectively. The top row shows the results for Smilei and the second row for WarpX. An observation of figure 3.24 allows to introduce some considerations. Figure 3.25 shows the corresponding computational costs. First of all, T_{part} and T_{grid} , also in this case, seem to benefit a lot from a massive parallelization.

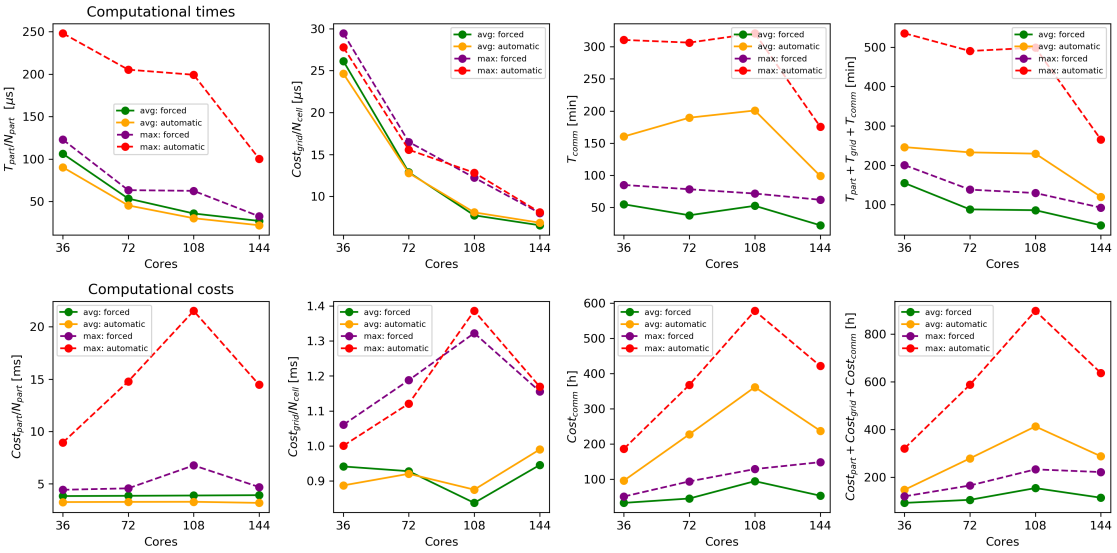

Figure 3.23: Computational times and computational costs required for an automatic and the optimal forced (grids of 2048×64 cells) domain decomposition in WarpX.

However, the behavior manifested by the communication routines less reflects the expectations, in the results achieved with both the PIC codes.

Let us start considering Smilei: the average computational cost for processes synchronization remains approximately constant with respect to the number of processes working, which represents a curious issue. Moreover, the sum of the average costs spent by the three components, even decreases when the number of process, communicating each other, increases. The sum of the maximum costs shows a counter-intuitive fall when 144 CPUs are employed, assuming a value close to the maximum cost spent by the simulations run on 36 cores.

This could be explained by the hybrid MPI-OpenMP asset: in fact all the discussed simulations exploited also a multi-threading scheme, in which several threads work on a single task.

In order to clarify this situation, further simulations, deprived of a multi-threading framework, have been studied exploiting both 36 and 144 cores (corresponding to 1 and 4 nodes respectively). Then, these new results have been compared with the original ones.

These tests, exploiting a pure MPI framework, are in a good agreement with the previous hypothesis (see figure 3.26): namely, a considerable increment in the maximum sum-cost have been measured when one move from the 36 cores to the 144 cores framework. However, the mean cost expended in communications is not subjected to substantial variations also in this case.

Therefore, it is plausible deducing that, when a thick homogeneous plasma is simulated, the filling-space Hilbert algorithm, implemented by Smilei when the patches are assigned to the ranks, constitutes a successful strategy, which manage to reduce the boundaries

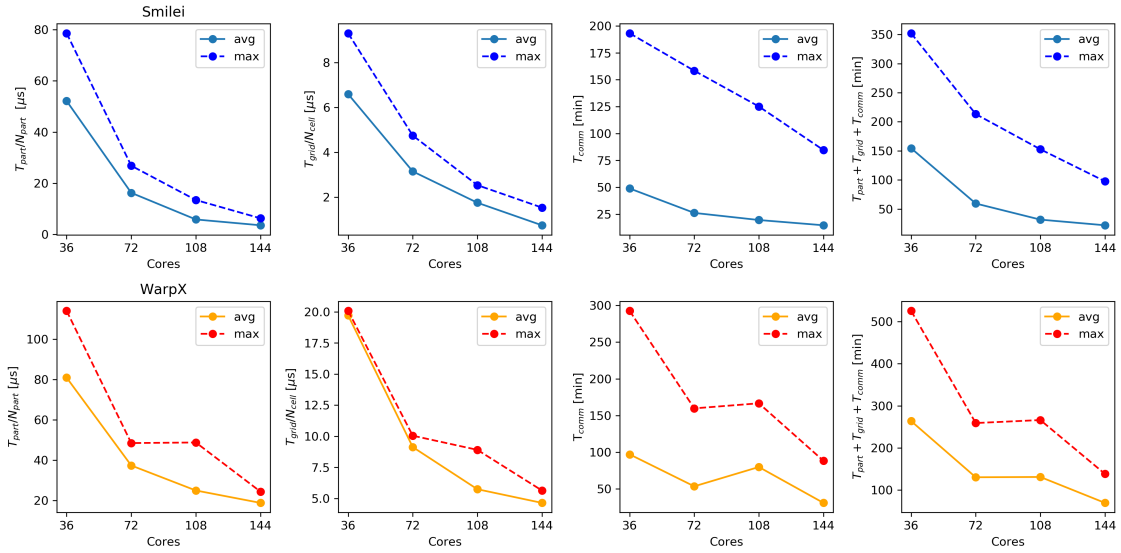

Figure 3.24: Profiles of T_{part}/N_{part}, T_{grid}/N_{cell}, T_{comm} and the sum $T_{part} + T_{grid} + T_{comm}$ in each column from left to right respectively required for the thick near-critical foam simulations. The maximum (dashed lines) and the average (solid lines) values between the MPI processes are plotted. These profiles are shown for Smilei (top row) and WarpX (second row).

lengths enough to preserve communication efforts. In addition, when multi-threading is involved, the combination of the strategies related to patches disposition and to threads administration brings admirable benefits.

Now, let us go back to consider figure 3.25 and let us focus on the results provided by WarpX (second row). In this case, all the costs assumes a bizarre profile, with a sharp peak in correspondence of the 3 nodes framework.

Hence, after the considerations arose from the analysis of the results provided by Smilei, the first approach has constituted in trying to switch off the multi-threading, repeating the simulations without exploiting OpenMP. Then, the results obtained with a hybrid MPI-OpenMP combination and with MPI scheme have been compared. Nonetheless, as clearly emerges from 3.27, the general frame, related both to PIC routines and to communication activities, practically does not change. So, this brings out the fact that WarpX , in this scenario, do not seem to benefit very much from a multi-threading logic application.

Thus, the next step has been to have a view on the specific shapes of the grids generated at run-time, in the different simulations. The different decompositions performed by WarpX are summarized in table 3.6: it is notable that the critical case, in which 108 cores are parallelized, is the unique with a number of grids greater than the number of CPUs exploited. In particular, 144 grids have been generated, meaning that there are a few processes which handle more than 1 grids, whereas all the others are associated to a single grid. So, the reason of the pathological behavior shown in the critical configuration is probably caused by a non optimal decomposition scheme adopted.

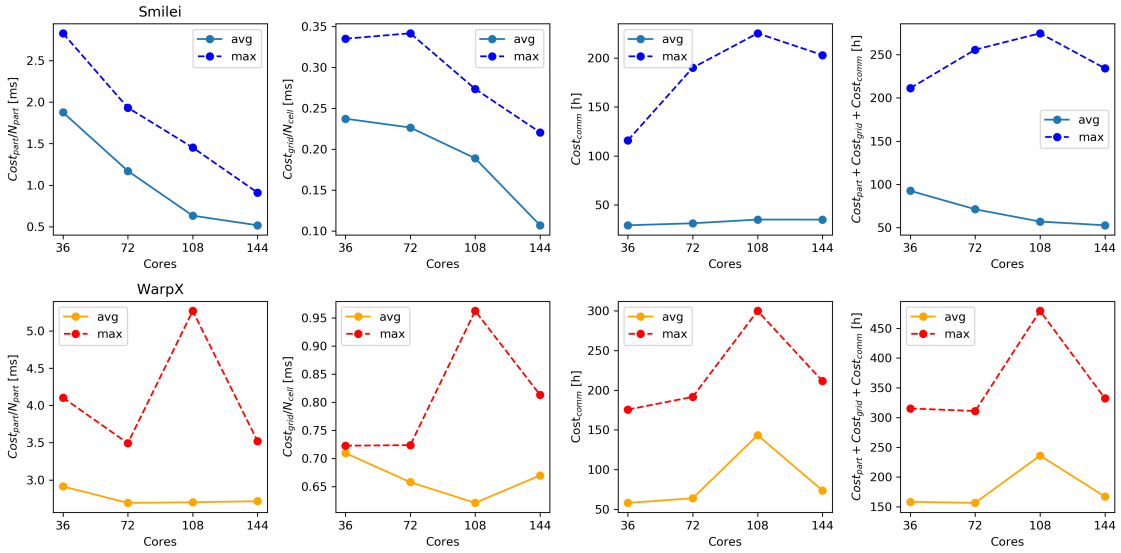

Figure 3.25: Profiles of C_{part}/N_{part}, C_{grid}/N_{cell}, C_{comm} and the sum $C_{part} + C_{grid} + C_{comm}$ in each column from left to right respectively required for the thick near-critical foam simulations. The maximum (dashed lines) and the average (solid lines) values between the MPI processes are plotted. These profiles are shown for Smilei (top row) and WarpX (second row).

Hybrid MPI-OpenMP		**Pure MPI**	
Number of cores	**Grids size [cells]**	**Number of cores**	**Grids size [cells]**
36	36 grids: 1024×512	36	36 grids: 1024×512
72	72 grids: 512×512	†	†
108	144 grids: 512×256	108	144 grids: 512×256
144	144 grids: 512×256	108	144 grids: 512×256

Table 3.6: Grid decomposition performed by WarpX in the hybrid MPI-OpenMP and in the pure MPI environments

3.4 Discussion

In this section, we will discuss the results presented above, trying to arrive to more general conclusions. The two simulated physical frameworks, i.e. the solid target and the near-critical foam scenarios, have been studied separately. These simulated scenarios are quite simplified and far from the realistic frameworks involved in experiments. Thus, it would be very useful to observe the obtained results with a broader perspective, trying to compare the two cases in order to detect possible similarities or differences.

First of all, both the scenarios exhibit a good time scaling with respect to the number of parallel computing units: running a simulation on a large quantity of CPUs, allows one to save a good portion of computational time, regardless of the physical scenario and the code exploited. So, one could conclude that the best choice consists in exploiting as many

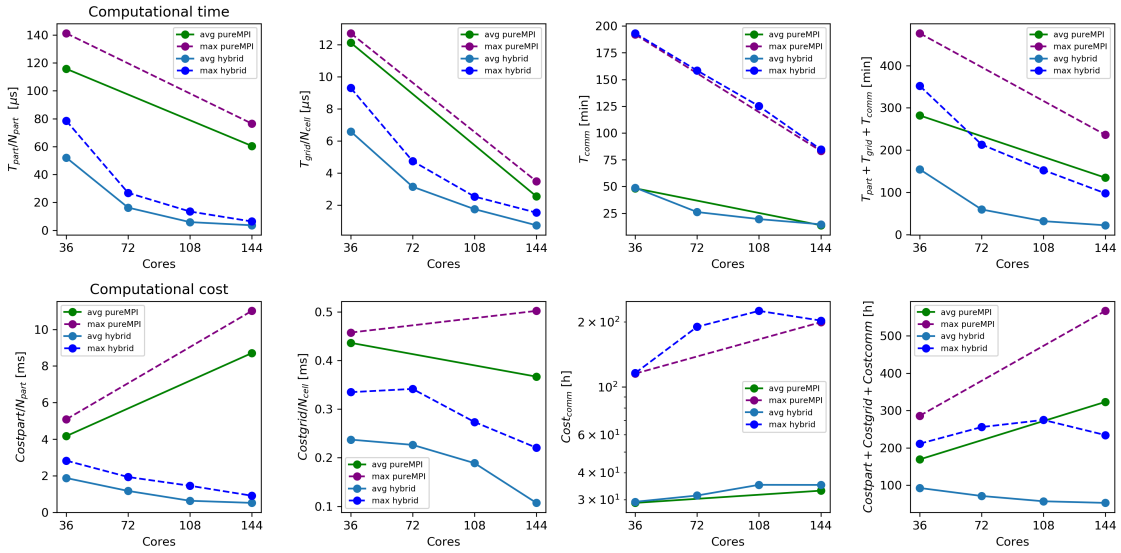

Figure 3.26: Computational times (first line) and computational costs (second line) required for a pure MPI (with 36 and 144 cores) and a hybrid MPI-OpenMP (with 36,72,108 and 144 cores) configuration in Smilei.

nodes as possible. However, it must be taken in account that the total cost required by a single simulation could considerably increase with the number of processes, at least if the simulation setup has not been cleverly adjusted, since the number of communications may increase too much. Therefore, a winning strategy is based on trying to smartly configure the numerical parallel environment, in order to minimize the overheads among processes. This challenge, in particular, consists in trying to adopt the best mesh decomposition scheme, depending on the considered scenario. As a remark, consider that only Smilei provides a simple interface to define the effective domain decomposition, while WarpX has a much greater degree of freedom in this sense. Hence, in some cases it may be better not to leave a high degree of freedom to the code.

However, the previous results suggest that, if the plasma occupies a large enough region of the simulation box and its density is low enough, ascribable to the near-critical foam case, the performances are less dependent on the specific domain decomposition. But the situation changes dramatically when dense plasmas are confined in a limited region, like thin solid foils. In this case, it is very important to consider beforehand how to distribute the subdomains in order to split the computational load as evenly as possible among the MPI regions. In such cases, it is convenient to perform many domain subdvisions within the plasma (e.g. in our case along the y direction). In addition, it is advisable to take in consideration the fact that this strategy can exploit its full potential if the plasma is located in the center of the box: so, it is worth configuring a symmetrical scenario.

In addition, activating a periodical load balancing strategy, provided by both the codes, can bring additional benefits.

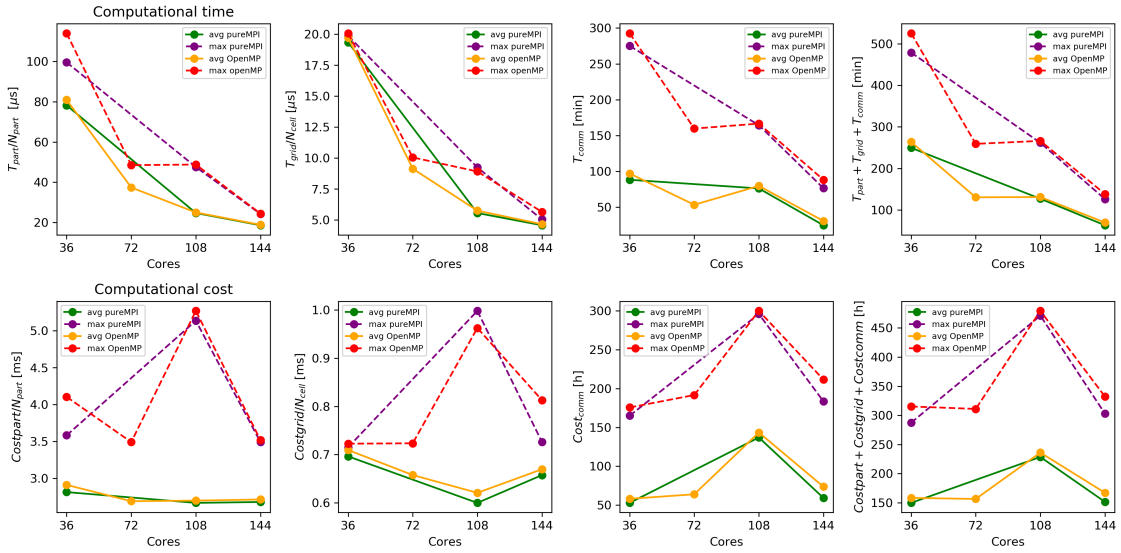

Figure 3.27: Computational times (first line) and computational costs (second line) required for a pure MPI (with 36, 108 and 144 cores) and a hybrid MPI-OpenMP (with 36,72,108,144 cores) configuration in WarpX.

By the way, one should considerate that, if the starting configuration is already well balanced, the effective profits will be limited. So, when a large plasma is considered, like the near-critical plasma, there are not consistent disparities between the MPI domains, in terms of computational load. On the other hand, if the plasma thickness decreases, the situation could change: but in this case a wise decomposition can fix this issue.
Moreover, the analysis of the specific time contributions related to the PIC algorithm and the communication routine results as a confirm of the previous considerations. Namely, when a thin plasma is considered, the domain decomposition chosen may have disastrous effects on the communication times. Instead, a smart decomposition minimize the communication efforts and, consequently, the global simulation time.
S These results show that it is not straightforward how to make an optimal use of these complex Particle-In-Cell codes.

Chapter 4

Performance analyses of laser-plasma interaction simulations on a hybrid CPU-GPU cluster

In the previous chapter we presented a numerical investigation on the performances achieved by some simulations, launched on the Galileo supercomputer. All these simulations have been performed in a parallelized context, in which only CPU were involved. This chapter, instead, wants to analyze how the performances change, when a hybrid CPU-GPU cluster is exploited.
Hence, several numerical tests have been carried out, in which the computations have been performed by set of units, composed by both CPUs and GPUs. For this purpose, we have exploit the Marconi100 cluster, the latest supercomputer acquired by the Cineca computing center. This device is equipped a with powerful architecture, thanks to which it occupies the ninth place in the TOP500 list of 2020.
Table 4.1 summarizes the main technical details of Marconi100 architecture.
All the numerical simulations presented in this chapter have been performed with WarpX: next section explains how WarpX is implemented for running on GPU.

4.1 WarpX on GPUs

One of the reasons which have directed in the selection of WarpX as Particle-In-Cell code is the fact that it avails of a GPU support, thanks to the fact that the Amrex tool, exploited by WarpX to handle parallelization, provides an adaptation for graphic processing units. This chapter briefly presents the strategy provided by WarpX to perform parallelization on GPUs (for a more detailed explanation see [67]).
Essentially, AMReX's GPU strategy focuses on providing performant GPU support with

Table 4.1: Marconi100: system architecture

Model	IBM Power AC922 (Whiterspoon)
Racks	55 total (49 compute)
Nodes	980
Processors	2x16 cores IBM POWER9 AC922 at 2.6(3.1) GHz
Accelerators	4 x NVIDIA Volta V100 GPUs/node, Nvlink 2.0, 16GB
Cores	32 cores/node, Hyperthreading x4
RAM	256 GB/node (242 usable)
Peak Performance	about 32 Pflop/s, 32 TFlops per node
Internal Network	Mellanox IB EDR DragonFly++
Disk Space	8PB raw GPFS storage

minimal changes, with respect to the CPU strategy, and maximum flexibility, allowing application teams to get running on GPUs quickly.
Generally AMReX avails of the hybrid MPI-CUDA interface for GPUs but, actually, it can be further combined with other parallel GPU languages, like OpenACC and OpenMP, to control the offloading of subroutines to the graphics processing unit.
Basically, at the beginning of each simulation, the MPI tool is exploited to create as many processes as the GPU involved: in other terms, each GPU correspond to a single rank.
After that, the domain decomposition occurs, and several grids are created, which are useful for the multi-threading scheme. As described in chapter 3, in a CPU based system the parallelization strategy consists in a combination of the MPI and OpenMP protocols, making use of the tiling strategy. However, the tiling strategy is ineffective on GPUs and the efficient use of the GPU's potential is the primary aim, considering the great benefits which a highly parallel structure, such as the one of GPUs, can provide, if well administered.
Indeed, improving resource efficiency allows a larger percentage of GPU threads to work simultaneously, increasing effective parallelism and, consequently, decreasing the computational efforts. More specifically, the calculations that can be offloaded efficiently to GPUs use CUDA threads to parallelize over a grid at a time: this is done, as illustrated in figure 4.1, exploiting a lot of CUDA threads that only work on a few cells each. AMReX utilizes CUDA managed memory to automatically handle memory movement for mesh and particle data. Moreover, a further parallelization occurs, by using CUDA streams, which are sequences of data processed by paradigms that allow some applications to more easily, simply by by restricting the parallel computations that can be performed. More in detail, a series of operations, called *kernel functions* is applied to each element in the stream, exploiting a limited form of parallel processing: CUDA guarantees execution order of kernels within the same stream, while allowing different streams to run simultaneously. AMReX places each iteration of the various loops on separate streams, allowing each independent

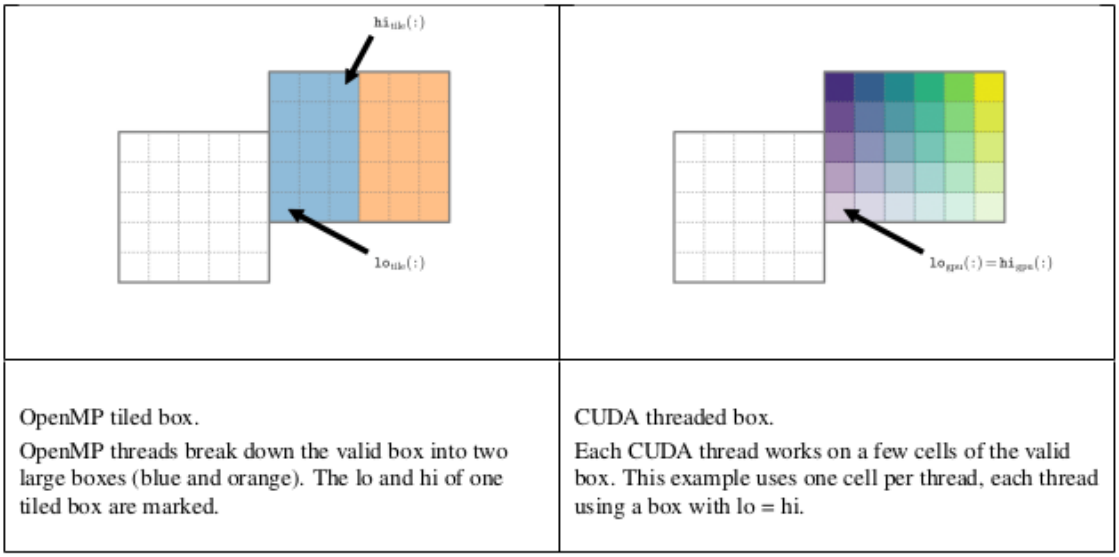

Figure 4.1: Comparison of OpenMP (left) and CUDA (right) work distribution. OpenMP threads break down the grid into two large boxes (blue and orange). The lower and higher corners of one tiled grid are marked. On the other hand, each CUDA thread works on a few cells of the grid: in this example one cell per thread is used. Reprinted from [67].

iterations to be run simultaneously (see figure 4.2).

4.2 Numerical results

In this section the main numerical results obtained are presented. As a first approach, like the previous investigation carried out with Galileo, the main focus has consisted in a preliminary examination of the global computational times, in various configurations, spent to conclude the simulations. Then, following the previous procedure, the focus has been relocated on the individual times, related to the different PIC routines and the communications. Also in this case, the two physical scenarios have been studied independently.

4.2.1 Analyses of the performances using different multiprocessing hybrid configurations

Solid target

This paragraph focuses on the analysis of the time employed to simulate the laser interaction with a thin solid target. As explained in the previous section, the MPI protocol splits the computations between as many ranks as GPUs employed, producing a one-to-one correspondence between processes and graphics processing units.
To begin, some simulations have been launched, varying the number of CPUs and GPUs running. However, in each simulation the same number of CPUs and GPUs has been ex-

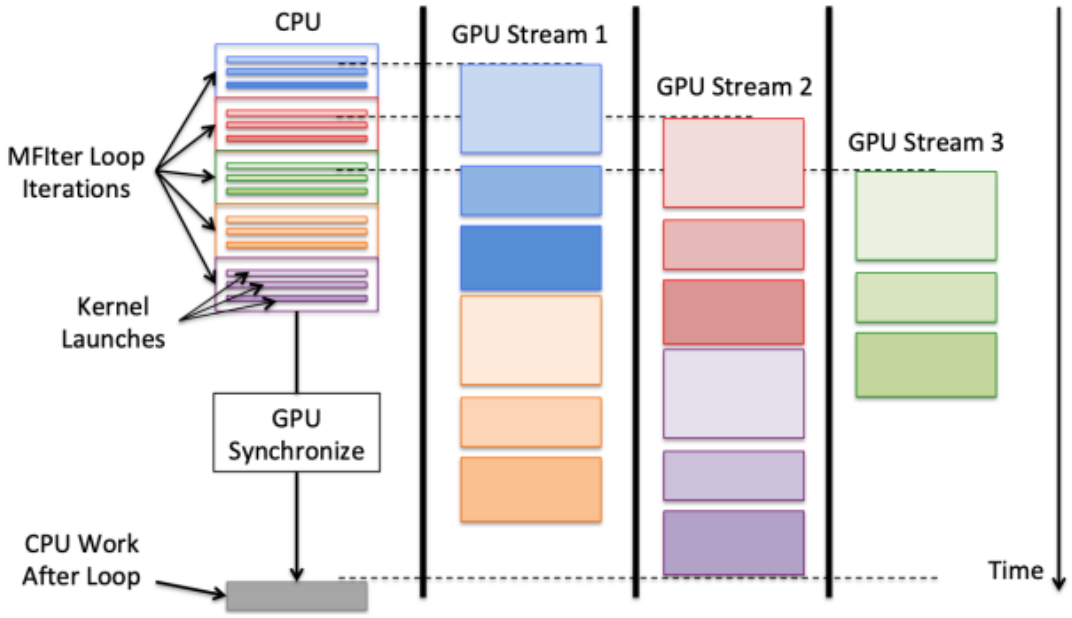

Figure 4.2: Timeline illustration of GPU streams. Illustrates the case of a loop of five iterations with three GPU kernels each being ran with three GPU streams. The CPU runs the first iteration of the loop (blue), which contains three GPU kernels. The kernels begin immediately in GPU stream 1 and run in the same order they were added. The second (red) and third (green) iterations are similarly launched in streams 2 and 3. The fourth (orange) and fifth (purple) iterations require more GPU resources than remain, so they have to wait until resources are freed before beginning. Meanwhile, after all the loop iterations are launched, the CPU reaches a synchronization and waits for all GPU launches to complete before continuing.

ploited, since it should guarantee better performances. This set of simulations has allowed to study the time scaling when the number of cores increases.

Figure 4.3 shows both the computational times (a) and costs (b) used for all the performed simulations. From this figure emerges that a good time scaling has been achieved. In particular, these results illustrates that a very slight increment of units used allows to save a consistent fraction of computational time. Moreover, also in this case the computational costs related to each configuration have been calculated. Namely, when a single simulation is launched on Marconi100, the supercomputer scales an amount of hours from the available budget of the user, depending on the number of units demanded. Otherwise, the CPU and the GPU have not the same weight in the quantification of the computational cost, but each GPU counts as 8 CPU. Hence, the explicit computation is the following:

$$Cost = 8 \times N_{GPU} \times time$$

Thus, the costs required by these tests have been compared (see figure 4.3(b)): clearly, demanding for different number of CPUs and GPUs implies variations of few hours. However, the costs do not seem to increase significantly, when the number of units grows: so,

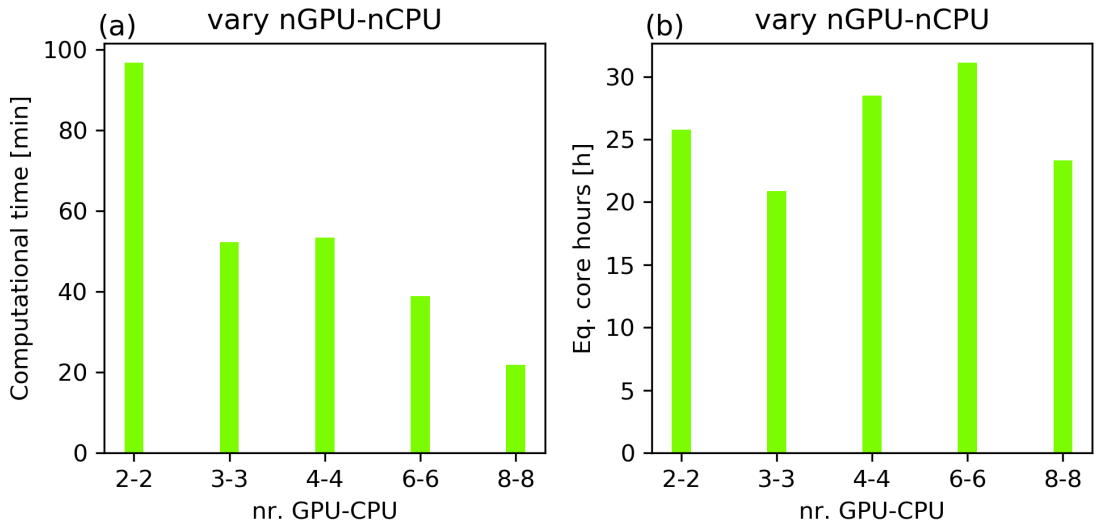

Figure 4.3: (a) Computational time required for the thin solid foil simulations using WarpX as a function of the exploited number of GPU-CPU. (b) Computational cost (i.e. computational time $\times$ (8 $\times$ number of GPUs) spent for the thin solid foil simulations WarpX as a function of the exploited number of GPU-CPU.

opting for a high number of units, could be a good choice.

After that, we have analyzed the effects resulting from the domain decomposition performed by WarpX. For this purpose, different values for the `max grid size` have been imposed. Hence, has been noticed that high values for this parameter, which allow WarpX to create big grids, lead to better performances(see figure 4.4). So, a high number of grids, with a compact size, does not represent a good choice. At the same time, also the dependence of the performances on the blocking factor has been studied. In fact, as explained in chapter 3, WarpX performs the grid creation at run time, on the basis of the values imposed by the user for both the `max grid size` and the `blocking factor`. In this case, several options for this parameters have been explored. Figure 4.5 reports the computational times involved when different values of the `blockingfactor` are considered. This figure shows that this parameter doesn't impact so much in the resulting simulation time. In fact, checking the effective decompositions operated in these several tests, it emerges that WarpX tends to create grids as large as possible, with a size equal to the maximum imposed (i.e. the `max grid size`).

All the numerical tests shown so far, in this paragraph, have no exploited load balancing procedures: so, this strategy has been tested in order to see if further benefits can be achieved. For this purpose, an extra numerical simulation have been launched, imposing a max grid size equal to 1024, since, as just shown, this configuration had presented the best performances.

Figure 4.6 shows a comparison between simulations with and without load balancing. In

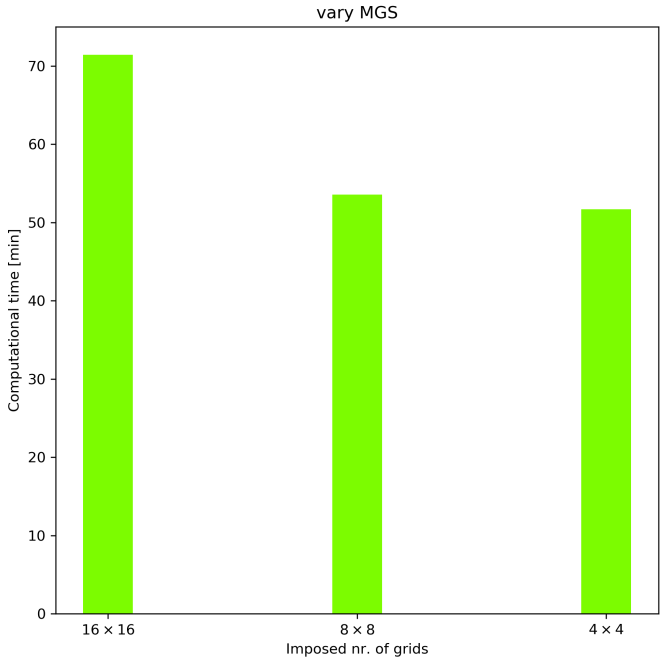

Figure 4.4: Computational times required for the thin solid foil simulations using WarpX with different number of selected subdomains.

this framework, the load balancing schemes have not led to particular improvements, but the reduction of time achieved is almost negligible.

Nevertheless, it can be not overlooked the fact that the last simulation, involving balancing among the processes, have been designed with an efficient setting for domain decomposition, so the limited benefits can be justified by this efficient configuration.

Near-critical foam

In this paragraph an equivalent exploration, performed on the near-critical foam framework, will be presented. The same procedure has been adopted also in this case: first of all, several simulations have been launched, each of those on a different number of GPUs, and, so, different number of MPI process have been produced. As a remark, here the tests are performed exploiting a different resolution with respect to what was done in chapter 3: a mesh of 8192×4096 cells is generated, with a spatial step of 0.0125 μm along x and y, 12194 steps and a resulting $\Delta t \simeq 0.028$ fs. Figure 4.7 shows both the computational times (a) and costs (b) used for all the performed simulations. Like in the previous scenario, also this time a good scaling has been achieved in computational times: increasing the number of graphics units, so, can enhance consistently the performances. Moreover, also in this case the fall in computational time is efficient enough to not imply big increments in the computational costs required. So, opting for a relatively number of GPUs represents a valid option.

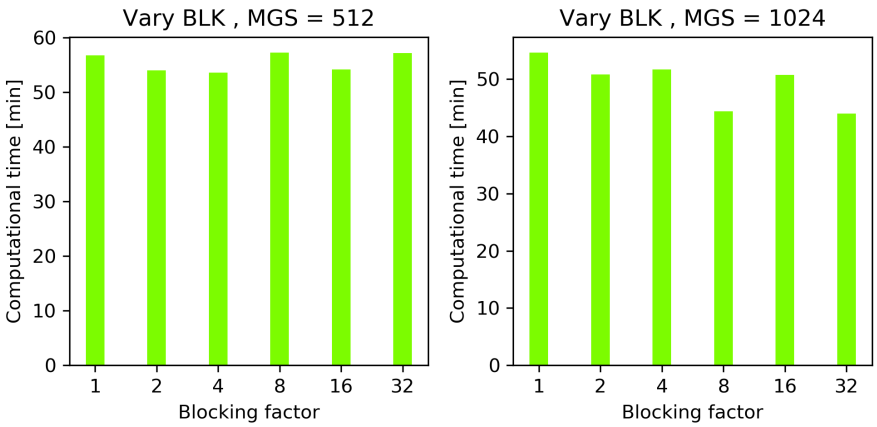

Figure 4.5: Computational times required for the thin solid foil simulations using WarpX with different values for the `blockingfactor` parameter with two different choices for the `max grid size` (left:512, right:1024).

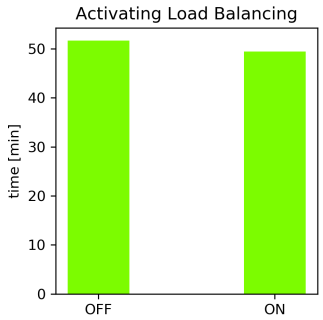

Figure 4.6: Computational time required for the thin solid foil simulations using WarpX when load balancing schemes are I) not activated II) activated every 100 timesteps.

Moreover, also in this case the load balancing has been tried, in order to analyze if the further reduction of time can be ensured. Figure 4.8 shows a comparison between simulations with and without load balancing. It appears clear that, in this case, a simulation exploiting balancing algorithms can save a discrete amount of computational time.

4.2.2 PIC-routines and computational load

The results presented so far have been studied and compared in terms of the effective simulation time required by the cluster to conclude the full simulation. Now, we are interested in observe the computational time spent in doing computations related to specific PIC routines. In this purpose, the same procedure followed in chapter 3 has been followed.
So, the computational time associated to computations involving particles (T_{part}) has been defined: in particular it has been designed to take in account Boris pusher, current depo-

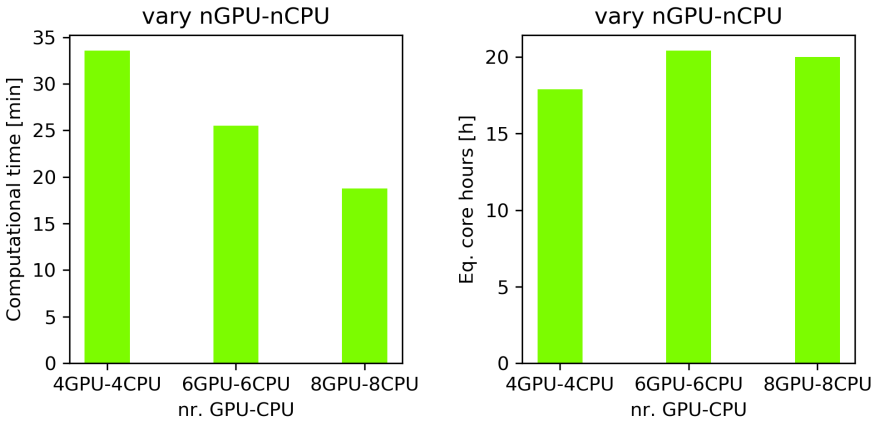

Figure 4.7: (a) Computational time required for the thick near-critical simulations using WarpX as a function of the exploited number of GPU-CPU. Both the maximum and the average value (between the MPI processes) have been plotted. (b) Computational cost (i.e. computational time $\times$ (8 $\times$ number of GPUs) spent for the thick near-critical simulations WarpX as a function of the exploited number of GPU-CPU. Both the maximum and the average value (between the MPI processes) have been plotted.

sition and field interpolations routines.
After that, since the computations performed on the grid points coincide with the Maxwell solver, a second element has been considered, consisting in the time spent for updating the electromagnetic fields (T_{grid}).
A deep analysis of the definition of these two times, which has been described in chapter 3, had allow to assume that T_{part} and T_{grid} linearly depend on the number of macro-particles and cells respectively. So, also in this case, an observation of their normalized values makes sense.
In addition, also the communications required for synchronize the MPI processes have been included in a third component.

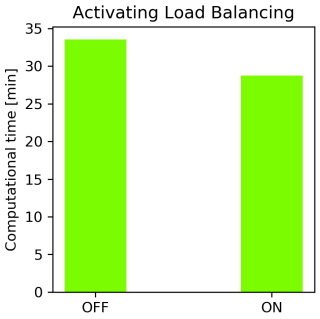

Figure 4.8: Computational time required for the thick near-critical simulations using WarpX when load balancing schemes are I) not activated II) activated every 100 timesteps.

Here the profiles of these distinct components have been reported, for both the simulated physical scenarios, i.e. the solid target and the near-critical foam framework.
Figure 4.9 reports the plots resulting from the solid target scenario: it shows the profiles of T_{part}/N_{part}, T_{grid}/N_{cell}, T_{comm} and the sum $T_{part} + T_{grid} + T_{comm}$ in each column from left to right (first row) respectively and the same for the computational costs (second row). Essentially, both the particle-time and the grid-time (average and maximum) scale with the number of processes applied, as it was reasonable to expect. Moreover, the corresponding costs seem to keep an almost constant profile.
On the other hand, the communication times tend to assume a similar decreasing profile,

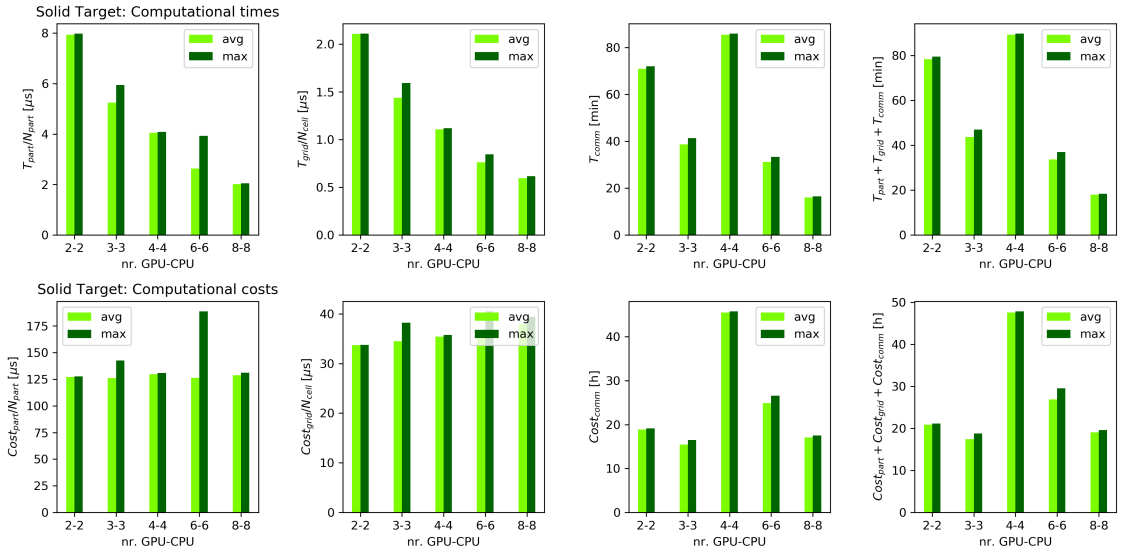

Figure 4.9: Profiles of T_{part}/N_{part}, T_{grid}/N_{cell}, T_{comm} and the sum $T_{part} + T_{grid} + T_{comm}$ in each column from left to right respectively required for the thin solid foil simulations(first row). The corresponding computational costs are reported in the second row. The maximum (dark green) and the average (light green) values between the MPI processes are plotted. These results are obtained with WarpX.

except for the configurations presenting 4 GPUs, which exhibits a value quite higher than the times required from the other configurations.
But, as can be easily checked observing the sum of the three times, the most of the time has been spent in order to keep the GPUs synchronized: in fact the communication time is close the sum of all the components. This is probably due to the particular geometry of the scenario simulated, in which the charged regions occupy a very short portion of the entire domain. However, in almost all the parallel configurations exploited, the average and the maximum time are very close: this means that an efficient domain decomposition occurred, resulting in a fair balance of the global load between the processes.
At this point, the same analysis have been done on the results obtained by the near-critical foam simulations. The different computational times and the corresponding computational

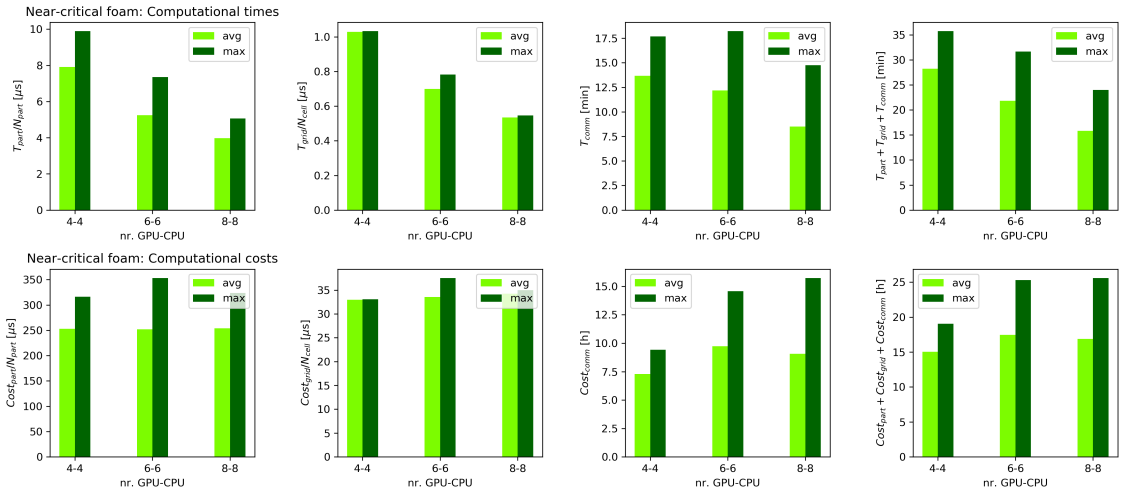

Figure 4.10: Profiles of T_{part}/N_{part}, T_{grid}/N_{cell}, T_{comm} and the sum $T_{part} + T_{grid} + T_{comm}$ in each column from left to right respectively required for the thick near-critical foam simulations(first row). The corresponding computational costs are reported in the second row. The maximum (dark green) and the average (light green) values between the MPI processes are plotted. These results are obtained with WarpX.

costs are illustrated in figure 4.10.

Also in this case, the mean and the maximum times related to the PIC routines, i.e. the particle-time and the grid-time, decreases with respect to the increasing of the number of GPUs running. Moreover, as in the solid target scenario, the corresponding costs seem to remain almost the same.

Finally, the cost associated to the mean communication time not feel big increments, while the maximum cost increases together with the number of processes. Otherwise, the maximum one is more indicative, because it is attributable to the slowest processes, which affect the effective simulation time, since each timestep doesn't finish until all the processes have done their job.

4.3 Discussion

In the previous section, we have illustrated the results obtained by the exploratory numerical investigation carried out on Marconi100 supercomputer. Now, we are interested in developing a general discussion of the results. Moreover, a main concern consists the comparison between the performances achieved with the CPU and the hybrid GPU-CPU based cluster. In practice, we are interested in the observation of the benefits which can be achieved exploiting graphics processing units parallelization.

The first evidence is the fact that a good time scaling can be achieved increasing the number of GPU parallelized. Moreover, contrary to that observed with CPU parallelization, where the computational costs, in general, feel sensible increments when the number of

computing units is increased, demanding for more GPUs do not impact significantly on the resulting costs. This happens thanks to the inherent high parallelized architecture of GPU: indeed, adding a single GPU to a parallelized system implies only some additional communications but can take consistent advantages.
Instead, when a pure CPU parallelization is exploited, like in the simulations of chapter 3 performed with Galileo, full nodes have to been added in order to see consistent variations in terms of simulation times. Otherwise, each node contains a lot of CPUs, 36 in the case of Galileo: this means that adding a single node is equivalent to add a lot of MPI processes, and, consequently, a lot of MPI calls.
Moreover, the effects arising from the inclusion of periodical load balancing schemes have been analyzed in both the scenarios. As a general consideration, the load balancing benefits could be irrelevant if a good domain decomposition occurred before, exactly like in the simulations running on CPUs systems. Otherwise, it is better to activate always a frequent redistribution of the work load between GPUs, especially if a good domain decomposition strategy is not known a priori.
Now let us focus on the behavior shown by the particle-time T_{part} and the grid-time T_{grid}. In both the physical scenarios simulated they show a very good scaling with respect to the number of GPUs implicated, with the effect of almost constant computational costs. So, adding GPUs allow to reduce these times so much that the computational costs related to do computations on particles and grids are not affected.
In addition, from a comparison of the two scenarios it results that the simulations involving a thin solid plasma spend a bigger fraction of time in performing communications, with respect to the case in which a thick near-critical foam is considered. This is attributable to the fact that the solid target scenario presents a less homogeneous environment, in which a thin surface, loaded of a lot of macro-particles, is surrounded by empty space. For this reason, given the fact that the particles begin to move when the target is impinged by the laser, a lot of particles continuously crosses the boundaries of the MPI regions, requiring frequent communications.
Moreover, it is useful to examine how much an hybrid GPU-CPU parallelization framework can benefit with respect to a CPU-based environment.
Figure 4.11 compares the performances of two simulations, involving a near-critical foam scenario, performed on Galileo and on Marconi100. Note that we are comparing the most efficient case, in terms of computational time, for what concerns Galileo and the worst one for Marconi100. The figure reports (from left to right) the profiles of the global simulation time, the normalized T_{part} and T_{grid} and the communication times. It easy to observe that the simulations run with Marconi100 show much better performances. The term T_{part}, which represents the most heavy routine of the PIC scheme, is reduced up to a factor of ~ 9.6, while T_{grid} is reduced up to a factor of ~ 5.8. Moreover, also the communication time T_{comm} takes considerable advantages from the GPU parallel environment: in fact, the

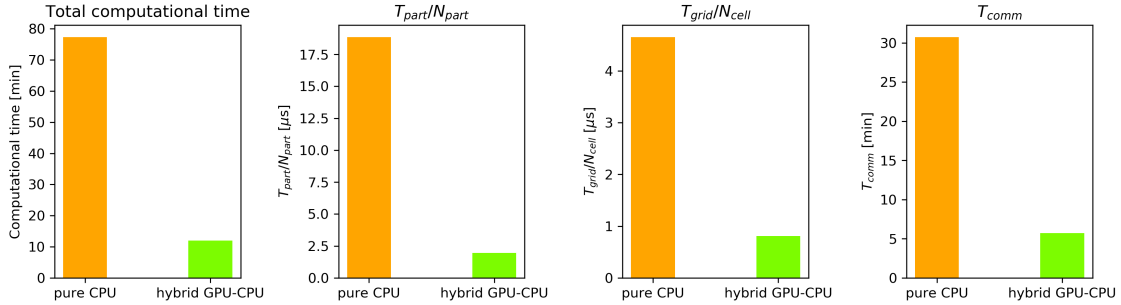

Figure 4.11: Comparison between the performances of two thick near-critical foam simulations scenario, performed on Galileo (orange) and on Marconi100 (green). The figure reports (from left to right) the profiles of the global simulation time, the normalized average T_{part} and T_{grid} and the average communication times. These results have been achieved involving 4 nodes (i.e. 144 CPUs) on Galileo and 1 GPU-CPU on Marconi100.

ratio between the communications times of these two compared simulations is ~ 5.4 All these benefits clearly result in a consistent reduction of the global simulation time, which decreases by a factor close to 6.4.
However, today there are not so many Particle-In-Cell codes compatible with a GPU parallelization. Anyway, these results show that exploiting an hybrid GPU-CPU system may considerably improve the simulations performances, with a consistent reduction of the computational efforts.

Chapter 5

Conclusions and perspectives

In this thesis we have carried out a numerical campaign to investigate the performances of selected two-dimensional kinetic Particle-In-Cell (PIC) simulations of laser-plasma interaction exploiting a massively parallel computing environment (see chapter 3 and 4). Two different open source codes – Smilei and Warpx – were adopted which implement different parallelization strategies. Moreover, two different supercomputer were used in order to explore the different CPU and hybrid GPU-CPU architectures. Also, two distinct physical scenarios have been considered: the first is represented by a high-intensity laser pulse interacting with a thin solid target (in the first case) and with a thick near-critical foam (in the second case). The numerical simulations were performed on the CPU based High Performance Computing cluster Galileo, hosted at CINECA, Bologna, Italy. On the other hand, WarpX performances have been tested on both Galileo and on the hybrid GPU-CPU based supercomputer Marconi100, the latest supercomputer acquired by CINECA.
As a first approach, a preliminary exploratory campaign has been carried out, focused on the observation of the codes behavior, for different parallel configurations. This investigation has been conducted on the two physical scenarios separately tp study how the computational times scale with the number of computing units, i.e. with different numbers of CPUs on Galileo and GPU-CPU systems on Marconi100. Not only the effective simulations times have been considered but also the computational costs, consisting in the total equivalent core hours (depending both on the effective times and on the number of CPUs or GPUs requested).
The parallelization schemes, especially the domain decomposition and the assignment of the subdomains to the various MPI regions, implemented in the two codes are quite different one from the other: this has an impact on the final computational performances. Indeed, while Smilei has shown a good ability to adapt to the increasing number of MPI processes, instead WarpX has not exhibited an equally smooth time scaling, but it has shown that there are some computing configurations which dramatically increase the computational

costs. In addition, the load balancing strategies for periodical work redistribution among the processes, provided by both the codes, resulted very useful, especially if a not smart domain decomposition has been implemented. Moreover, the geometrical distribution of the plasma inside the simulation box also plays a role in determining the final results, in terms of computational time. Next, a deeper analysis on the code routines involved in a parallel PIC scheme has been taken forward: in particular the time related to computations involving particles, grid points and MPI communications/synchronizations have been studied. Breaking down the routines into the fundamental bricks of a parallel PIC algorithm has let us isolate the portion of the computations that are responsible for spoiling the scaling of the computational time with the number of processing units. From this, we obtain that the communications can counterbalance the benefits (i.e. the cost reduction associated to the particles and grid computations) coming from stronger parallel configurations.
The results discussed in chapter 4 have suggested that an hybrid GPU-CPU parallel system can significantly improve the code performances, especially if the plasma simulated covers a large portion of the mesh, such as in the near-critical foam scenario. Indeed, a comparison between the results on Galileo and on Marconi100, has revealed that a hybrid GPU-CPU approach can tear down the computational costs. In addition, it results that the parallel strategy implemented by WarpX seems to better behave on Marconi100: this is an indication of the fact that the GPU architecture may mitigate the non-optimal parallelization strategy. Indeed it is plausible to think that, thanks to the much higher computing power of the GPUs, a given simulation can be performed with much less MPI processes, thus tearing down the impact due to communications. For this reason the GPUs can also positively counterbalance an unbalanced and non optimized simulation as for what concerns its geometry. So, considering the scenarios simulated in this thesis, the GPU architecture can give a special benefit in the overcritical regime of interaction, in terms of the performances obtained with different parallel frameworks.

These results represent a first step of a numerical investigation which is becoming more and more fundamental, in order to address the computational issues characteristic of PIC algorithms. The obtained results constitute a useful basis for the future, in the context of laser-plasma simulations involving solid targets or near-critical foams. Moreover, this work could allow to reduce the computational efforts required for simulating complex structures, such as double-layer targets, especially if exploring different sizes and structures. In particular, the considerations derived for the solid target scenario can be extended, up to a certain limit, to simulations involving a rather thin and homogeneous foam, attached to a solid foil. On the other hand, if the thickness of the simulated foam has a relevant size, it may be reasonable to exploit the considerations deduced for the foam scenario. Then, in the future, the numerical research on the performances must proceed in order to achieve strategies tailored to more complex physical scenarios. An important objective

is represented by understanding how these strategies can be adjusted if a nanostructured near-critical foam is considered. In fact, in this case the foam would be constituted by small and dense aggregates, so that particular attention must be paid to the overall numerics. A further step could consist in observing how the situation changes if the simulations are carried out in a three-dimensional geometry, focusing in understanding how the considerations developed for two-dimensional scenarios may be adapted. Also it could be of interest to explore the impact on the performances due to additional physical effects, such as photon emissions, collisions and ionization. Finally, all these tests should be carried out also exploiting other codes which implement other parallelizations, focusing on a quantification of the effective benefits even in complex situations, especially when using GPUs. Therefore, such investigations would likely allow to get a more satisfactory understanding of the most convenient parallel strategies in many kinds of situations. In this way, one may be able to squeeze the code performances and as a consequence simulate more realistic and accurate physical scenarios. Besides, thanks to the extreme efficiency of the GPU architecture it may become possible to perform reduced two-dimensional simulations on ordinary laptops. These results constitute a first step towards a more comprehensive understanding of how the PIC algorithm can perform in a massively parallel computing environment, setting the basis for a even more optimized use of this method.

List of Figures

List of Tables

Bibliography

[1] VI Veksler. The principle of coherent acceleration of charged particles. *The Soviet Journal of Atomic Energy*, 2(5):525–528, 1957.

[2] A Einstein. On the quantum theory of. *Concepts of Quantum Optics*, page 93, 2013.

[3] Theodore H Maiman. Stimulated optical radiation in ruby. *nature*, 187(4736):493–494, 1960.

[4] Orazio Svelto and David C Hanna. *Principles of lasers*, volume 1. Springer, 2010.

[5] Donna Strickland and Gerard Mourou. Compression of amplified chirped optical pulses. *Optics communications*, 56(3):219–221, 1985.

[6] Irving Langmuir. Oscillations in ionized gases. *Proceedings of the National Academy of Sciences of the United States of America*, 14(8):627, 1928.

[7] David Book and JD Huba. Nrl plasma formulary. Technical report, NAVAL RESEARCH LAB WASHINGTON DC PLASMA PHYSICS DIV, 2002.

[8] Arianna Formenti. Intense laser interaction with nanostructured plasmas: a kinetic numerical investigation. 2016.

[9] John David Jackson. Classical electrodynamics, 1999.

[10] Toshiki Tajima and John M Dawson. Laser electron accelerator. *Physical Review Letters*, 43(4):267, 1979.

[11] Thomas Katsouleas. Electrons hang ten on laser wake. *Nature*, 431(7008):515–516, 2004.

[12] Andrea Macchi, Marco Borghesi, and Matteo Passoni. Ion acceleration by superintense laser-plasma interaction. *Reviews of Modern Physics*, 85(2):751, 2013.

[13] EL Clark, K Krushelnick, JR Davies, Matthew Zepf, M Tatarakis, FN Beg, A Machacek, PA Norreys, MIK Santala, I Watts, et al. Measurements of energetic proton transport through magnetized plasma from intense laser interactions with solids. *Physical Review Letters*, 84(4):670, 2000.

[14] Anatoly Maksimchuk, Shaoting Gu, Kirk Flippo, Donald Umstadter, and V Yu Bychenkov. Forward ion acceleration in thin films driven by a high-intensity laser. *Physical Review Letters*, 84(18):4108, 2000.

[15] RA Snavely, MH Key, SP Hatchett, TE Cowan, Markus Roth, TW Phillips, MA Stoyer, EA Henry, TC Sangster, MS Singh, et al. Intense high-energy proton beams from petawatt-laser irradiation of solids. *Physical review letters*, 85(14):2945, 2000.

[16] Hiroyuki Daido, Mamiko Nishiuchi, and Alexander S Pirozhkov. Review of laser-driven ion sources and their applications. *Reports on progress in physics*, 75(5):056401, 2012.

[17] SC Wilks, AB Langdon, TE Cowan, M Roth, M Singh, S Hatchett, MH Key, D Pennington, A MacKinnon, and RA Snavely. Energetic proton generation in ultra-intense laser–solid interactions. *Physics of plasmas*, 8(2):542–549, 2001.

[18] IRENE Prencipe, A Sgattoni, DAVID Dellasega, LUCA Fedeli, LORENZO Cialfi, Il Woo Choi, I Jong Kim, Karol Adam Janulewicz, KF Kakolee, Hwang Woon Lee, et al. Development of foam-based layered targets for laser-driven ion beam production. *Plasma Physics and Controlled Fusion*, 58(3):034019, 2016.

[19] LORENZO Cialfi, LUCA Fedeli, and MATTEO Passoni. Electron heating in subpicosecond laser interaction with overdense and near-critical plasmas. *Physical Review E*, 94(5):053201, 2016.

[20] L Fedeli, A Formenti, L Cialfi, A Sgattoni, G Cantono, and M Passoni. Structured targets for advanced laser-driven sources. *Plasma Physics and Controlled Fusion*, 60(1):014013, 2017.

[21] Luca Fedeli, Arianna Formenti, Lorenzo Cialfi, Andrea Pazzaglia, and Matteo Passoni. Ultra-intense laser interaction with nanostructured near-critical plasmas. *Scientific reports*, 8(1):1–10, 2018.

[22] Luca Fedeli, Arianna Formenti, Carlo Enrico Bottani, and Matteo Passoni. Parametric investigation of laser interaction with uniform and nanostructured near-critical plasmas. *The European Physical Journal D*, 71(8):202, 2017.

[23] MATTEO Passoni, A Sgattoni, IRENE Prencipe, LUCA Fedeli, DAVID Dellasega, LORENZO Cialfi, Il Woo Choi, I Jong Kim, Karol Adam Janulewicz, Hwang Woon Lee, et al. Toward high-energy laser-driven ion beams: Nanostructured double-layer targets. *Physical Review Accelerators and Beams*, 19(6):061301, 2016.

[24] Matteo Passoni, FM Arioli, L Cialfi, D Dellasega, L Fedeli, A Formenti, AC Giovannelli, A Maffini, F Mirani, A Pazzaglia, et al. Advanced laser-driven ion sources and their applications in materials and nuclear science. *Plasma Physics and Controlled Fusion*, 62(1):014022, 2019.

[25] Chio-Zong Cheng and Georg Knorr. The integration of the vlasov equation in configuration space. *Journal of Computational Physics*, 22(3):330–351, 1976.

[26] CZ Cheng. The integration of the vlasov equation for a magnetized plasma. *Journal of Computational Physics*, 24(4):348–360, 1977.

[27] Francis Filbet and Eric Sonnendrücker. Numerical methods for the vlasov equation. In *Numerical mathematics and advanced applications*, pages 459–468. Springer, 2003.

[28] Jacques Denavit. Numerical simulation of plasmas with periodic smoothing in phase space. *Journal of Computational Physics*, 9(1):75–98, 1972.

[29] Eric Sonnendrücker, Jean Roche, Pierre Bertrand, and Alain Ghizzo. The semi-lagrangian method for the numerical resolution of the vlasov equation. *Journal of computational physics*, 149(2):201–220, 1999.

[30] Eric Sonnendrucker, John J Barnard, Alex Friedman, David P Grote, and Steve M Lund. Simulation of heavy ion beams with a semi-lagrangian vlasov solver. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 464(1-3):470–476, 2001.

[31] Andrew Staniforth and Jean Côté. Semi-lagrangian integration schemes for atmospheric models—a review. *Monthly weather review*, 119(9):2206–2223, 1991.

[32] Takashi Nakamura and Takashi Yabe. Cubic interpolated propagation scheme for solving the hyper-dimensional vlasov—poisson equation in phase space. *Computer Physics Communications*, 120(2-3):122–154, 1999.

[33] Alexander J Klimas. A method for overcoming the velocity space filamentation problem in collisionless plasma model solutions. *Journal of computational physics*, 68(1):202–226, 1987.

[34] Alexander J Klimas and William M Farrell. A splitting algorithm for vlasov simulation with filamentation filtration. *Journal of computational physics*, 110(1):150–163, 1994.

[35] Frederick C Grant and Marc R Feix. Fourier-hermite solutions of the vlasov equations in the linearized limit. *The Physics of Fluids*, 10(4):696–702, 1967.

[36] Oscar Buneman. Dissipation of currents in ionized media. *Physical Review*, 115(3):503, 1959.

[37] John Dawson. One-dimensional plasma model. *The Physics of Fluids*, 5(4):445–459, 1962.

[38] O Buneman and G Kooyers. Computer simulation of the electron mixing mechanism in ion propulsion. *AIAA Journal*, 1(11):2525–2528, 1963.

[39] O Buneman and DA Dunn. Computer experiments in plasma physics. 1965.

[40] Charles K Birdsall and A Bruce Langdon. *Plasma physics via computer simulation*. CRC press, 2004.

[41] Roger W Hockney and James W Eastwood. *Computer simulation using particles*. crc Press, 1988.

[42] Kane Yee. Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Transactions on antennas and propagation*, 14(3):302–307, 1966.

[43] T Zh Esirkepov. Exact charge conservation scheme for particle-in-cell simulation with an arbitrary form-factor. *Computer Physics Communications*, 135(2):144–153, 2001.

[44] Anna Grassi, Luca Fedeli, Andrea Sgattoni, and Andrea Macchi. Vlasov simulation of laser-driven shock acceleration and ion turbulence. *Plasma Physics and Controlled Fusion*, 58(3):034021, 2016.

[45] Amir Hossein Sojoodi, Majid Salimi Beni, and Farshad Khunjush. Ignite-gpu: a gpu-enabled in-memory computing architecture on clusters. *The Journal of Supercomputing*, pages 1–28, 2020.

[46] Ivan Kadochnikov. Accelerating the particle-in-cell method of plasma and particle beam simulation using cuda tools. 2019.

[47] Viktor K Decyk and Tajendra V Singh. Adaptable particle-in-cell algorithms for graphical processing units. *Computer Physics Communications*, 182(3):641–648, 2011.

[48] Guangye Chen, Luis Chacón, and Daniel C Barnes. An efficient mixed-precision, hybrid cpu–gpu implementation of a nonlinearly implicit one-dimensional particle-in-cell algorithm. *Journal of Computational Physics*, 231(16):5374–5388, 2012.

[49] Can-qun Yang, Qiang Wu, Hui-li Hu, Zhi-cai Shi, Juan Chen, and Tao Tang. Fast weighting method for plasma pic simulation on gpu-accelerated heterogeneous systems. *Journal of Central South University*, 20(6):1527–1535, 2013.

[50] Viktor K Decyk and Tajendra V Singh. Particle-in-cell algorithms for emerging computer architectures. *Computer Physics Communications*, 185(3):708–719, 2014.

[51] A Snytnikov and A Romanenko. The advantage of the gpu-based supercomputer simulation of plasma phenomena. *Bulletin of the Novosibirsk Computing Center. Series: Numerical Analysis*, 16:81–92, 2013.

[52] Alexey V Snytnikov and Alexey A Romanenko. High performance pic plasma simulation with modern gpus. In *Journal of Physics: Conference Series*, volume 1103, page 012013. IOP Publishing Ltd., 2018.

[53] George Stantchev, William Dorland, and Nail Gumerov. Fast parallel particle-to-grid interpolation for plasma pic simulations on the gpu. *Journal of Parallel and Distributed Computing*, 68(10):1339–1349, 2008.

[54] Hon-Cheng Wong, Un-Hong Wong, Xueshang Feng, and Zesheng Tang. Efficient magnetohydrodynamic simulations on graphics processing units with cuda. *Computer Physics Communications*, 182(10):2132–2160, 2011.

[55] Jan Veltmaat and Jens C Niemeyer. Cosmological particle-in-cell simulations with ultralight axion dark matter. *Physical Review D*, 94(12):123523, 2016.

[56] Yutaka Ohira, Brian Reville, John G Kirk, and Fumio Takahara. Two-dimensional particle-in-cell simulations of the nonresonant, cosmic-ray-driven instability in supernova remnant shocks. *The Astrophysical Journal*, 698(1):445, 2009.

[57] Deborah Sulsky, Shi-Jian Zhou, and Howard L Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer physics communications*, 87(1-2):236–252, 1995.

[58] PC Wallstedt and JE Guilkey. A weighted least squares particle-in-cell method for solid mechanics. *International journal for numerical methods in engineering*, 85(13):1687–1704, 2011.

[59] FH Harlow. A machine calculation for hydrodynamic problems. Technical report, Technical report, Los Alamos Scientific Laboratory report LAMS-1956, 1956.

[60] Francis H Harlow. The particle-in-cell method for numerical solution of problems in fluid dynamics. Technical report, Los Alamos Scientific Lab., N. Mex., 1962.

[61] Julien Derouillat, Arnaud Beck, F Pérez, Tommaso Vinci, M Chiaramello, A Grassi, M Flé, G Bouchard, I Plotnikov, N Aunai, et al. Smilei: A collaborative, open-source, multi-purpose particle-in-cell code for plasma simulation. *Computer Physics Communications*, 222:351–373, 2018.

[62] J-L Vay, A Almgren, J Bell, L Ge, DP Grote, M Hogan, O Kononenko, R Lehe, A Myers, C Ng, et al. Warp-x: A new exascale computing platform for beam–plasma simulations. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 909:476–479, 2018.

[63] Kai Germaschewski, William Fox, Stephen Abbott, Narges Ahmadi, Kristofor Maynard, Liang Wang, Hartmut Ruhl, and Amitava Bhattacharjee. The plasma simulation code: A modern particle-in-cell code with patch-based load-balancing. *Journal of Computational Physics*, 318:305–326, 2016.

[64] David Hilbert. Über die stetige abbildung einer linie auf ein flächenstück. In *Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes*, pages 1–2. Springer, 1935.

[65] Bongki Moon, Hosagrahar V Jagadish, Christos Faloutsos, and Joel H. Saltz. Analysis of the clustering properties of the hilbert space-filling curve. *IEEE Transactions on knowledge and data engineering*, 13(1):124–141, 2001.

[66] David P Grote, Alex Friedman, Jean-Luc Vay, and Irving Haber. The warp code: modeling high intensity ion beams. In *AIP Conference Proceedings*, volume 749, pages 55–58. American Institute of Physics, 2005.

[67] Weiqun Zhang, Ann Almgren, Vince Beckner, John Bell, Johannes Blaschke, Cy Chan, Marcus Day, Brian Friesen, Kevin Gott, Daniel Graves, et al. Amrex: a framework for block-structured adaptive mesh refinement. *Journal of Open Source Software*, 4(37):1370–1370, 2019.

[68] Mikhail A Belyaev. Picsar: A 2.5 d axisymmetric, relativistic, electromagnetic, particle in cell code with a radiation absorbing boundary. *New Astronomy*, 36:37–49, 2015.

[69] Guy M Morton. A computer oriented geodetic data base and a new technique in file sequencing. 1966.

[70] Carsten Burstedde, Johannes Holke, and Tobin Isaac. On the number of face-connected components of morton-type space-filling curves. *Foundations of Computational Mathematics*, 19(4):843–868, 2019.