Parallel Computing 52 (2016) 42–64

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Two particle-in-grid realisations on spacetrees

T. Weinzierl[a,*], B. Verleye[b,c], P. Henri[d], D. Roose[b]

[a] School of Engineering and Computing Sciences, Durham University Stockton Road, Durham DH1 3LE, Great Britain, UK
[b] Department of Computer Science, KU Leuven Celestijnenlaan 200A, 3001 Leuven, Belgium
[c] Vrije Universiteit Brussel, Pleinlaan 2, 1050 Elsene, Belgium
[d] Laboratoire de Physique et Chimie de l'Environnement et de l'Espace (LPC2E) CNRS, Université d'Orléans, 45071 Orléans Cedex 2, France

ARTICLE INFO

Article history:
Received 22 January 2015
Revised 22 December 2015
Accepted 28 December 2015
Available online 4 January 2016

Keywords:
Particle-in-cell
Spacetree
Particle sorting
AMR
Lagrangian–Eulerian methods
Communication-avoiding

ABSTRACT

The present paper studies two particle management strategies for dynamically adaptive Cartesian grids at hands of a particle-in-cell code. One holds the particles within the grid cells, the other within the grid vertices. The fundamental challenge for the algorithmic strategies results from the fact that particles may run through the grid without velocity constraints. To facilitate this, we rely on multiscale grid representations. They allow us to lift and drop particles between different spatial resolutions. We call this cell-based strategy particle in tree (PIT). Our second approach assigns particles to vertices describing a dual grid (PIDT) and augments the lifts and drops with multiscale linked cells.

Our experiments validate the two schemes at hands of an electrostatic particle-in-cell code by retrieving the dispersion relation of Langmuir waves in a thermal plasma. They reveal that different particle and grid characteristics favour different realisations. The possibility that particles can tunnel through an arbitrary number of grid cells implies that most data is exchanged between neighbouring ranks, while very few data is transferred non-locally. This constraints the scalability as the code potentially has to realise global communication. We show that the merger of an analysed tree grammar with PIDT allows us to predict particle movements among several levels and to skip parts of this global communication a priori. It is capable to outperform several established implementations based upon trees and/or space-filling curves.

1. Introduction

Lagrangian–Eulerian descriptions of physical phenomena are used by a wide range of applications. They combine the short-range aptitude of particle-based with the long-range capabilities of grid-based approaches. Besides their popularity driven by application needs, particle-grid methods are also popular in supercomputing. They are among the best scaling algorithms today (cf. for example [8,20,21]). This scaling mainly relies on two ingredients. On the one hand, particle-particle interactions often are computationally expensive in terms of floating point operations with moderate memory footprint. On the other hand, the particle-grid interaction requires a mapping of particles to the grid. This is a spatial sorting problem. It

* Corresponding author. Tel.: +441913342519.
E-mail addresses: tobias.weinzierl@durham.ac.uk, tobias.weinzierl@mytum.de (T. Weinzierl), bverleye@vub.ac.be (B. Verleye), pierre.henri@cnrs-orleans.fr (P. Henri), dirk.roose@cs.kuleuven.be (D. Roose).
URL: http://www.dur.ac.uk/tobias.weinzierl (T. Weinzierl)

http://dx.doi.org/10.1016/j.parco.2015.12.007
0167-8191/© 2016 Elsevier B.V. All rights reserved.

either is performed infrequently, is computationally cheap as the particles move at most one grid cell at a time, or the grid can be constructed efficiently starting from the particles [21].

The present paper is driven by a plain electrostatic PIC simulation [9,27] of an unmagnetised plasma. Its computational profile differs from the previous characteristics as it solves a partial differential equation (PDE) on an adaptive Cartesian grid which has to be stored persistently in-between two time steps. At the same time, the particles do not interact with each other – they induce very low computational workload – but may move at very high speed through the grid. Our work focuses on well-suited data structures and algorithms required for such a code within a dynamic adaptive mesh (AMR) environment where the simulation is ran on a distributed memory machine.

The setup raises an algorithmic challenge. While dynamic AMR for PDEs as well as algorithms based upon particle-particle interactions are exhaustively studied, our algorithm requires a fast mapping of particle effects onto the grid and the other way round per time step. This assignment of particles to the grid changes incrementally. Yet, some particles might *tunnel* several cells per time step: no particle is constrained to move at most into a neighbouring cell. In general, the time step size is chosen such that the majority of particles travel at most one cell per time step. This avoids the finite grid instability [2], i.e. numerical, non-physical, heating of the experiment. However, in our application as well as most non-relativistic plasma and gravitation applications, suprathermal particles do exist. Their velocity is not bounded.

The present paper proposes the parallel, locally refined, dynamically adaptive grid to result from a spacetree [29,32] yielding a Cartesian tessellation. Particles are embedded into the finest tree level. The latter is the adaptive grid hosting the PDE. A multiscale grid traversal with particle-grid updates then can be realised via a simple recursive code mirroring a depth-first search [30].

Two particle realisation variants are studied: we either store the particles within the spacetree leaves (particle in tree; PIT) or within the dual tree (particle in dual tree; PIDT) induced by the spacetree vertices. The latter is similar to a linked-list approach with links on each resolution level. This multiscale linked-list can be deduced on-the-fly, i.e. is not stored explicitly but encoded within the tree's adjacency information. Both particle storage variants render the evaluation of classical compact particle-grid operators straightforward as each particle is assigned to its spatially nearest grid entity anytime. Tunnelling is enabled as particles are allowed to move up and down in the whole spacetree. PIDT furthermore can move particles between neighbours. Besides neighbours and parent-child relations, no global adjacency data structure is required. While PIDT induces a runtime overhead and induces a more complex code compared to PIT, the multiscale linked-list nature of PIDT reduces the particle movements up (lift) and down (drop) in the tree, and particles moving along the links can be exchanged asynchronously in-between grid traversals. If we combine PIDT with a simple analysed tree grammar [11] for the particle velocities, we can predict lifts and anticipate drops in whole grid regions. This helps to overcome a fundamental problem. Since tunnelling is always possible, we need all-to-all communication in every time step: each rank has to check whether data is to be received from any other rank. This synchronisation introduces inverse weak scaling. The more particles the higher the number of tunnels. The more ranks the more tunnel checks. With a lift prediction, we can locally avoid the all-to-all that we map onto a reduction (reduction-avoiding PIDT; raPIDT). We weaken the rank synchronisation. Rank and process are used here as synonyms. For reasonably big parallel architectures, PIDT and its extension raPIDT thus outperform PIT. Though all three flavours of particle storage support tunneling, their performance is not significantly slower than a classic linked-list approach not allowing particles to tunnel. The multiscale nature of both PIT and PIDT as well as its variant raPIDT further makes the communication pattern, i.e. the sequence and choice of particles communicated via MPI, comprise spatial information. Ranks receiving particles that have to be sorted into a local grid anticipate this presorting and thus can outperform other classic approaches [4,23,25].

The remainder is organised as follows. We refer to related work before we introduce the algorithmic steps of electrostatic PIC motivating our algorithmic research (Section 3). In Section 4, we describe our spacetree grid paradigm. Two particle realisation variants storing particles either within the grid cells or within the vertices form the present work's focus (Section 5). Our experimental evaluation starts from a review of the particle movement characteristics before we study the runtime behaviour of the particle storage and sorting schemes as well as the scaling. Finally, we compare our results to three other approaches. This comparison (Section 7) highlights our contribution with respect to the particular application from Section 3 as well as the inevitable cost introduced by the tunnelling particles. A summary and outlook in Section 8 close the discussion. The appendix comprises a real-world validation run of our code as well as additional experimental data.

2. Related and used work

Requiring tunnelling in combination with a persistent grid holding a PDE solution renders many established grid data structures inappropriate or unsuited. We distinguish three classes of alternatives: on-the-fly (re-)construction, sorting and linked-list. Reconstructing the assignment for the prescribed grid from scratch per time step [21] is not an option, as the grid is given, holds data and is distributed. Local sorting followed by scattering is not an option (see e.g. [23,25] and references therein) as the decomposition, i.e. the adaptive mesh, is not known on each individual rank. The two possible modifications are to rely on replicated grids per rank and then to map these grids onto the real PDE grid – this approach is not followed-up here – or to sort locally within the grid and then to scatter those particles that cannot be sorted locally. Such a two-stage approach enriches local meta data information with additional decomposition data (ranks have to know how to scatter the particles) held on each rank, but may run into network congestion (cf. results from [4]) even if we avoid all-to-all