PicSim: 5D/6D Particle-In-Cell Plasma Simulation

Code Porting and Validation

Patrick Guio

Department of Physics and Astronomy, University College London

November 2012

1 Introduction

The plasma state —ionised gas— is ubiquitous in the Universe. In particular the ionosphere is the plasma embedded in the high-altitude atmosphere of a planet. Ionospheres (and magnetospheres) play a key role in mediating the interaction between space and planetary environments through complex dynamic processes that involve interactions between electromagnetic waves and charged particles.

One way to investigate the physics of such plasma is computer modelling simulation, where PDE's of different approximations, full kinetic to fluid descriptions are solved numerically. Alternatively Lagrangian particle methods aim at time-integrating the dynamics of 'macro-particles'. One particle method widely used in plasma simulation is the Particle-In-Cell (PIC) method (Dawson, 1983; Birdsall and Langdon, 1985). PicSim is a collection of 2D and 3D parallel plasma PIC C++ codes where the internal electrostatic field is updated on a fixed Eulerian mesh, and external uniform electromagnetic, electric and gravitation fields can be enabled. Such models where particles interact through average fields are called Particle-Mesh (PM). PicSim has been used to simulate the effect of a temperature gradient on low-frequency ion modes (Guio et al., 2001; Guio and Pécseli, 2010), beam particle interaction and organised structures (Daldorff et al., 2001; Guio et al., 2003), waves and wake pattern caused by charged dust (Guio and Pécseli, 2003; Guio et al., 2008) and waves, wake pattern and organised structures caused by finite size obstacle (Guio and Pécseli, 2004, 2005).

Fig. 1 sketches the basic principle of the PIC algorithm. A time iteration consists of four steps. The operations performed at each step are: (i) particle weighting, where particle charge is deposited onto the density grid – using a first order linear weighting scheme, the Cloud-In-Cell (CIC) scheme; (ii) field solver that compute electrostatic potential and electric field on the mesh – using multigrid method; (Wesseling, 1991); (iii) field weighting (reciprocal of particle weighting), where forces on the grid are interpolated to particles' position; and (iv) particle mover that integrate Newton 2$^{\text{nd}}$ law equation.

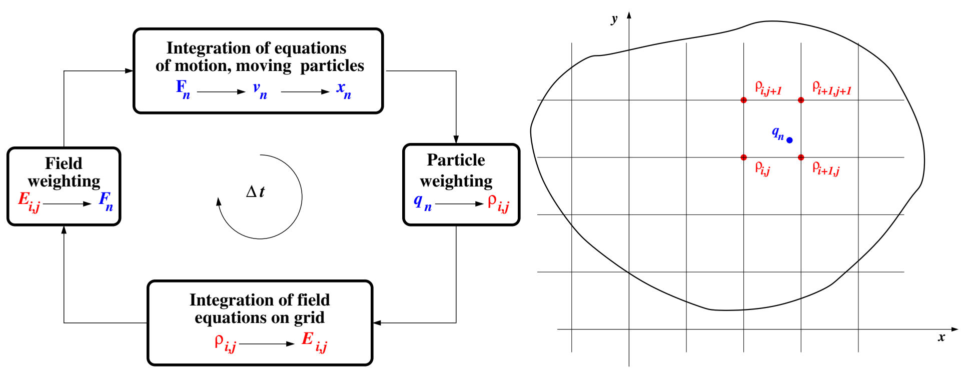
Integration of equations of motion, moving particles
$F_n \longrightarrow v_n \longrightarrow x_n$
Particle weighting
$q_n \longrightarrow \rho_{i,j}$
Integration of field equations on grid
$\rho_{i,j} \longrightarrow E_{i,j}$
Field weighting
$E_{i,j} \longrightarrow F_n$
Δt
y
x
$\rho_{i,j+1}$
$\rho_{i+1,j+1}$
q_n
$\rho_{i,j}$
$\rho_{i+1,j}$

Figure 1: Left: basic cycle of one time iteration. Right: sketch of the relation between particle charge and density mesh.

2 Porting and Profiling

Dr. Ferini gave me a thorough introduction to the IBM SP6 system. The first step was to compile both sequential and parallel PicSim without optimisation and test the validity of the results.

I was given an introduction on how to make a parallelism profiling with Scalasca to allow identification of potential performance bottlenecks —-in particular those concerning communication and synchronisation— and offers guidance in exploring their causes.

We profiled both non-optimised and optimised versions of PicSim using 4 CPUs and 16 CPUs. We found that at each time iteration about 63 % of the computing time is used to interpolate the electric field from the grid to particle positions (field weighting) and then move particles, while 31 % of the computing time is used to compute the source (particle weighting).

An overload of the master task was spotted as only this task solves the field equation after gathering of source terms (All-to-One MPI reduce). This imbalance is not important though in the tested configuration as solving the field equation represents only about 2 % of the computing time of one time iteration for the non-optimised version. Nevertheless we decided to modify the code from the All-to-One MPI reduce to an All-to-All MPI reduce in order to balance the particle weighting on each task and solve the electric field equation on each CPU.

Before performing the benchmark we tried different options set covering a wide range of optimisation. All together five different combinations of optimisation options for the C++ IBM compiler `xlC` (Version: 11.01.0000.0006) were tested.

The five setup are summarised in tables 1 and 2. Table 1 shows the time it takes to compile the complete PicSim code suite while Table 2 show short tests run times.

Our compiling and run times experiments show that the best choice was to disable inlining (`-qinline`) and inter procedural analysis optimisation (`-qipa`).

Table 1: Combinations of compiler optimisation options tested for `xlC`

Setup	Optimisation options	Compilation time
1	`-O3 -qstrict -qstrict_induction -qinline -qansialias -qhot -qunroll=yes -qarch=pwr6 -qtune=pwr6 -qipa`	10240 s
2	`-O3 -qstrict -qstrict_induction -qinline -qansialias -qhot -qunroll=yes -qarch=pwr6 -qtune=pwr6`	7710 s
3	`-O3 -qstrict -qstrict_induction -qansialias -qhot -qunroll=yes -qarch=pwr6 -qtune=pwr6`	7412 s
4	`-O3 -qstrict -qstrict_induction -qansialias -qunroll=yes -qarch=pwr6 -qtune=pwr6`	5423 s
5	`-O3 -qstrict -qarch=pwr6 -qtune=pwr6`	5656 s

Table 2: Summary of run times for 1 CPU and 2 CPU's short tests.

Setup	1 CPU run times (mm:ss)		2 CPU's run times (mm:ss)	
	min	Max	min	Max
1	14:58	16:51	09:08	09:08
2	13:11	16:58	08:50	08:52
3	13:01	15:12	08:42	08:42
4	13:40	17:45	09:25	09:25
5	17:36	17:58	08:32	09:00

3 Benchmarks and Scaling

We performed a series of benchmarks for different problem size and a wide range of number of CPUs p.

All the runs we performed are for a configuration with one plasma species interacting through electrostatic forces and where the particle number is conserved and the boundary is free space. The dimensions of the simulation box is $100{\times}100\,\lambda_0^2$ and $21{\times}25{\times}21\,\lambda_0^3$.

The run times $T_1, \ldots, T_p$ were fitted to

$$t(p) = t_p p^{-1} + t_s + t_1 p + t_2 p^2. \tag{1}$$

t_p and t_s are parallel and sequential computing; ideally $t_p{\sim}T_1$. t_1 and t_2 are linear (*Point-to-Point* and *One-to-All*) and quadratic (*All-to-All*) MPI communications. Eq. 1 is an extension of Amdahl's law $t(p){=}t_p/p{+}t_s$ (Eijkhout, 2011) where only parallelisable and sequential times are accounted.

Note that parallelism measures such as speedup $S(p) = t(p)/T_1$ and parallel efficiency $E(p) = S(p)/p$ can be derived from run times.

Figs. 2 and 3 show the results of our benchmarks and the scalability for the 2D version of PicSim while Figs. 4 and 5 show the results of our benchmarks and the

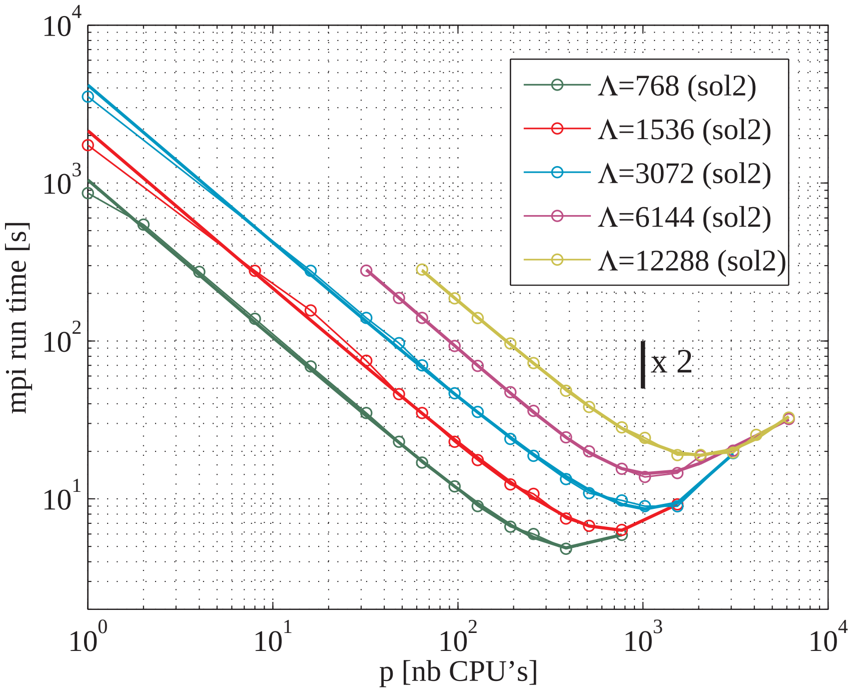

Figure 2: Run times T_p and fitted $t(p)$ for different problem size (as Λ is increased by a factor of 2) for 2D PicSim.

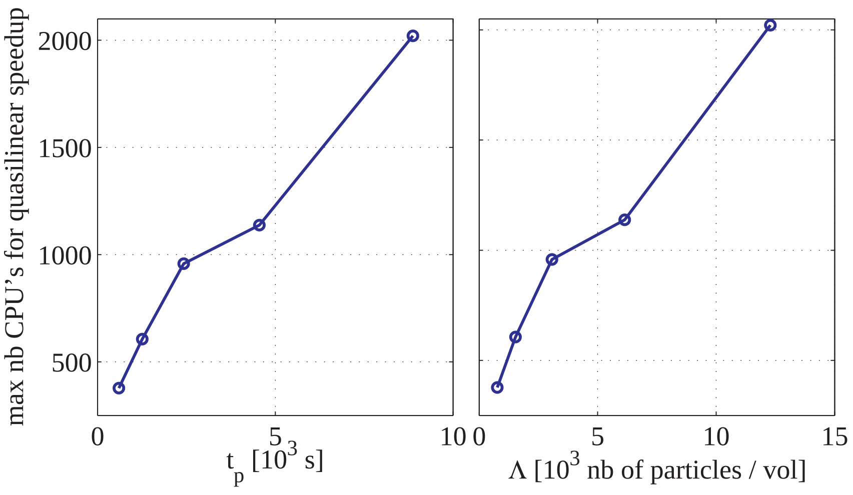

Figure 3: Largest 'optimal' p with linear speedup as function of the size problem for 2D PicSim.

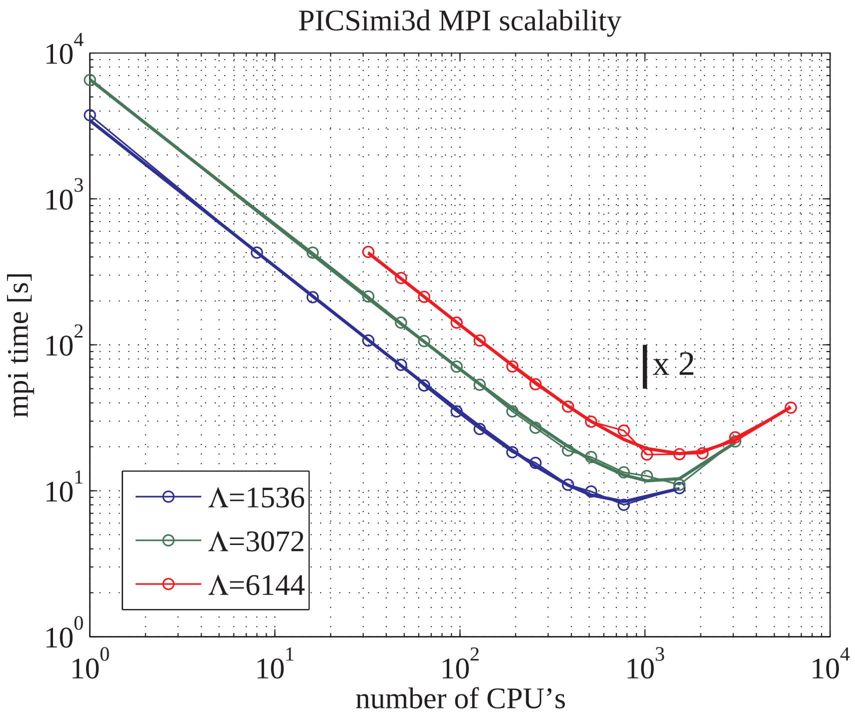

Figure 4: Run times T_p and fitted $t(p)$ for different problem size (as Λ is increased by a factor of 2) for 3D PicSim.

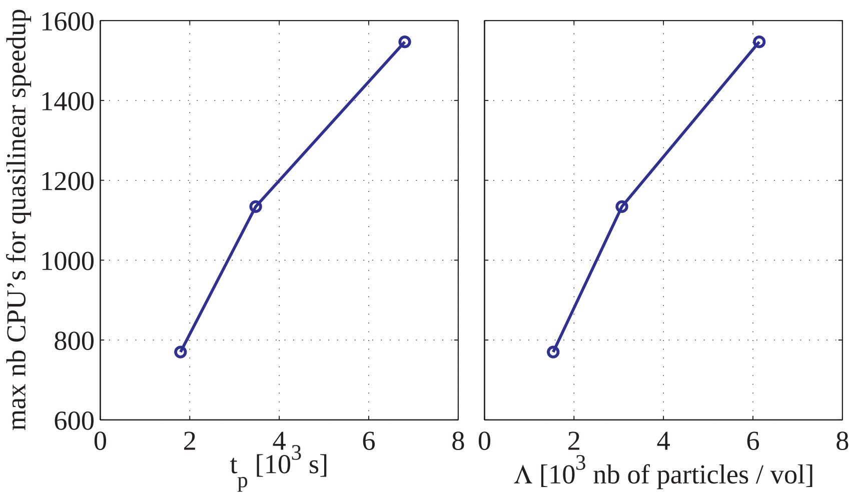

Figure 5: Largest 'optimal' p with linear speedup as function of the size problem for 3D PicSim.

scalability for the 3D version of PicSim

PicSim presents an overall good scalability but it would be interesting to understand the reason for the discontinuity in linear alignment in both plots of Fig. 3 located between 1000-1200 CPUs on the vertical axis.

4 Conclusion

Thanks to a fruitful collaboration and visits at CINECA, PicSim has been benchmarked for parallelism for the first time. Results are very encouraging and show linear speedup to several thousands CPUs, making PicSim a good candidate for PRACE, the Partnership for Advanced Computing in Europe.

Acknowledgements. The author is grateful to Dr. G. Ferini and the HPC team at CINECA for useful discussions on code optimisation and performance.

References

C. K. Birdsall and A. B. Langdon. *Plasma Physics via Computer Simulation*. McGraw-Hill, New York, 1985. ISBN 0-07-005371-5.

L. K. S. Daldorff, P. Guio, S. Børve, H. L. Pécseli, and J. Trulsen. Ion phase-space vortices in 3 spatial dimensions. *Europhys. Lett.*, 54:161–167, April 2001. doi: 10.1209/epl/i2001-00290-6.

J. M. Dawson. Particle simulation of plasmas. *Rev. Mod. Phys.*, 55:403–447, April 1983. doi: 10.1103/RevModPhys.55.403.

V. Eijkhout. Introduction to High-Performance Scientific Computing, 2011. Freely available online.

P. Guio and H. L. Pécseli. Collisionless Plasma Shocks in Striated Electron Temperatures. *Phys. Rev. Lett.*, 104(8):085002–+, February 2010. doi: 10.1103/PhysRevLett.104.085002.

P. Guio and H. L. Pécseli. Radiation of sound from a charged dust particle moving at high velocity. *Phys. Plasmas*, 10:2667–2676, July 2003. doi: 10.1063/1.1585031.

P. Guio and H. L. Pécseli. Phase space structures generated by an absorbing obstacle in a streaming plasma. *Geophys. Res. Lett.*, 31:L03806–+, February 2004. doi: 10.1029/2003GL018461.

P. Guio and H. L. Pécseli. Phase space structures generated by absorbing obstacles in streaming plasmas. *Ann. Geophysicæ*, 23:853–865, March 2005. doi: 10.5194/angeo-23-853-2005.

P. Guio, S. Børve, H. L. Pécseli, and J. Trulsen. Low frequency waves in plasmas with spatially varying electron temperature. *Ann. Geophysicæ*, 18:1613–1622, January 2001. doi: 10.1007/s00585-001-1613-1.

P. Guio, S. Børve, L. K. S. Daldorff, J. P. Lynov, P. Michelsen, H. L. Pécseli, J. J. Rasmussen, K. Saeki, and J. Trulsen. Phase space vortices in collisionless plasmas. *Nonlin. Proc. in Geophys.*, 10:75–86, 2003. doi: 10.5194/npg-10-75-2003.

P. Guio, W. Miloch, H. L. Pécseli, and J. Trulsen. Patterns of sound radiation behind pointlike charged obstacles in plasma flows. *Phys. Rev. E*, 78(1):016401–+, July 2008. doi: 10.1103/PhysRevE.78.016401.

Pieter Wesseling. *An introduction to multigrid methods*. John Wiley & Sons Ltd, Chichester, England, 1991. ISBN 0-471-93083-0.