A survey of the performance of classical potential solvers for charge distributions

Joseba Alberdi Rodriguez (UPV/EHU), Pablo García Risueño (BIFI)

Fast Methods for Long-Range Interactions in Complex Systems - 2011

Abstract

We perform a survey of different methods to calculate the classical electrostatic Hartree potential, created by charge distribution $(\rho(\vec{r}))$ in a mesh. Our goal is to provide the reader with a estimation on the performance -both in numerical complexity and in accuracy- of popular solvers, and to give her an intuitive idea about the way these solvers operate. Highly parallelizable routines (using PFFT (2) and FMM (3) libraries) have been implemented to be used in our tests, so that reliable conclusions about the capability of methods to tackle large systems in cluster computing can be obtained from our work.

Theory

Methods

Since the limiting factor of many quantum simulations (specially those involving large numbers of atoms) is the execution time, techniques have to be developed and implemented to reduce it. If our charge distribution is represented as a set of values corresponding to a grid in real space, the direct evaluation of [1] in every point would take $O(N^2)$ operations, being *N* the number of points of our grid, what is prohibitive for most interesting systems.

FFT

Fast Fourier Transform is a way to perform quickly (in $O(N\log_2 N)$ operations) the discrete Fourier transform X_k of a set of *N* values x_n

$$X_k := \sum_{n=0}^{N-1} x_n \exp\left(-i\frac{2\pi kn}{N}\right)$$

which otherwise should take $O(N^2)$ floating point operations (flops). Fast Fourier transform is very useful to evaluate the potential created by a density expressed as a discrete set of values. Since the FFT's require parallelepiped grids, if our original set of data is not parallelepiped it will have to be fulfilled using zeroes in order to become parallelepiped.

FMM

The fast multipole method (FMM) is a mathematical technique that was developed to speed the calculation up of long-ranged forces in the n-body problem. It does this by expanding the system Green's function using a multipole expansion, which allows one to group sources that lie close together and treat them as if they are a single source.*

It requires a very high order multipole expansion, which makes the calculation prefactor very high. However, recent research make possible to developed a competitive versions of FMM, being very accurate and efficient. As a result, this FMM algorithm is also of *O(N)*.

Multigrid

Multigrid methods (5) are suitable to solve partial differential equations in an iterative fashion, so they can be used to solve [2]. Variables are defined in a grid and partial derivatives are converted into finite differences. The grid is divided into a hierarchy of grids of different spacing (multigrid levels), where the equations are solved. The solution in broader grids (larger spacing, less points) is used to correct the iterative solution in the finer grids. By operating in this way, accurate solutions for the Poisson equation can be obtained in O(N) flops.

Conjugate gradients

Conjugate gradients (4) methods can be used to solve discretized [2] in an iterative manner. They are efficient to solve linear systems LV = ρ where V is the unknowns vector, ρ is the inhomogeneous term and L is a sparse coordinate matrix.

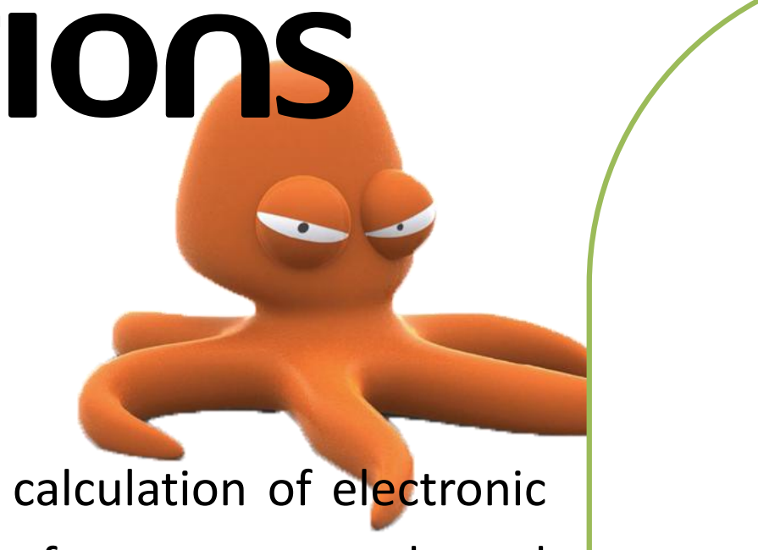
In our tests, we use a corrected version of the conjugate gradients method to solve for the Hartree potential, which uses an analytically built auxiliary density.

Equations

[1] $$V(\vec{r}) = \int_{\Omega} \frac{d\vec{r}\,'}{4\pi\varepsilon_0} \frac{\rho(\vec{r}\,')}{|\vec{r} - \vec{r}\,'|}$$

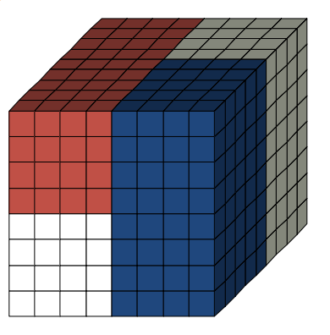
[2] $$\nabla^2 V(\vec{r}) + \frac{\rho(\vec{r})}{\varepsilon_0} = 0$$

Implementations

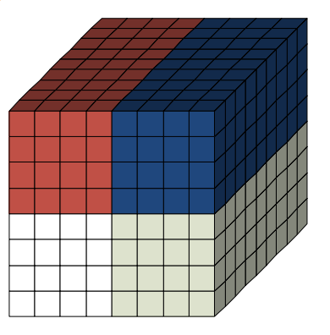


Octopus

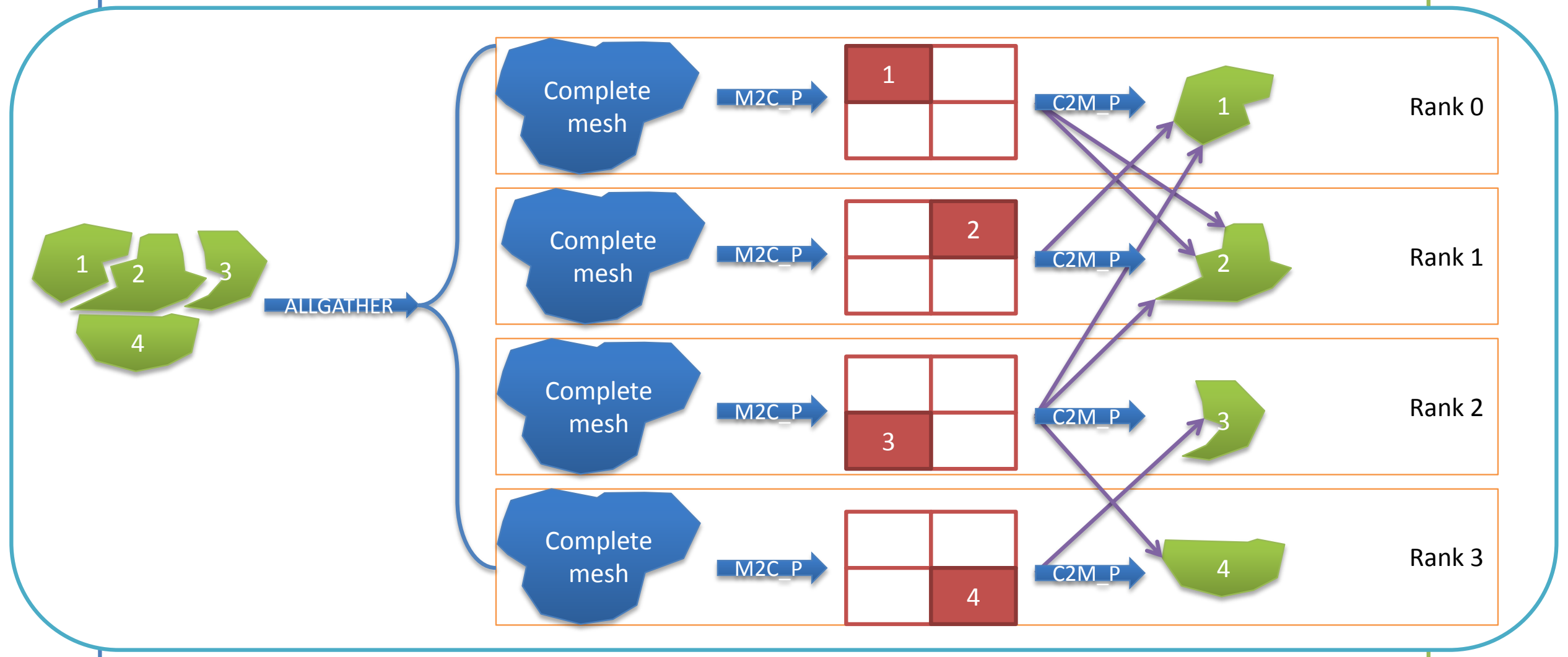
- Octopus is a complete software package for the calculation of electronic properties of the matter. It has a GPL license, so is free to use, study and modify. It is actively developed at the UPV/EHU in collaboration with other universities. Octopus has two main calculations:
    - Ground State (GS)
    - Time Dependent (TD)
- We will focus on the ***optimization of the Poisson solver*** because it is the main bottleneck of the speed-up in TD. Poisson solver is used in both main calculations (GS and TD). Time dependent (TD) calculation is an iterative process, that is repeated thousand of times to reach a physically meaningful information. In every iteration the Poisson solver is used once or twice, depending on the input.
- Octopus works with a mesh that is represented in 3D. Zoltan or Metis external libraries are used to partition the mesh, in a 3D decomposition



Octopus
mesh with a
3D decomposition



PFFT
mesh with a
2D decomposition



PFFT

For an efficient FFT implementation we used the PFFT library (2).

The main problem that we have to solve is that the data representation is in both codes. So an efficient translation between Octopus and PFFT library has to be done.

FMM

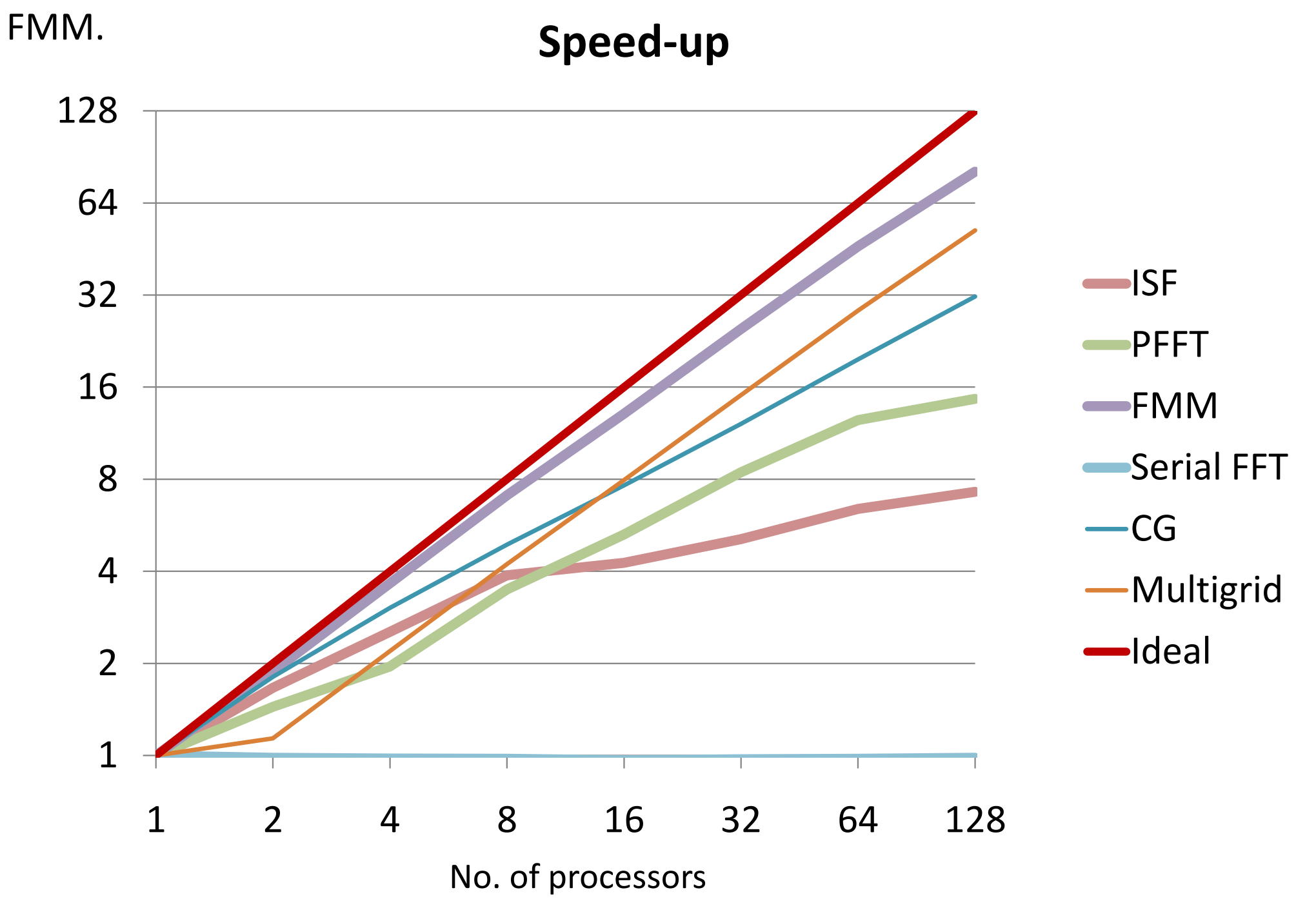
For an implementation of FMM we used the LIBFM library (3).

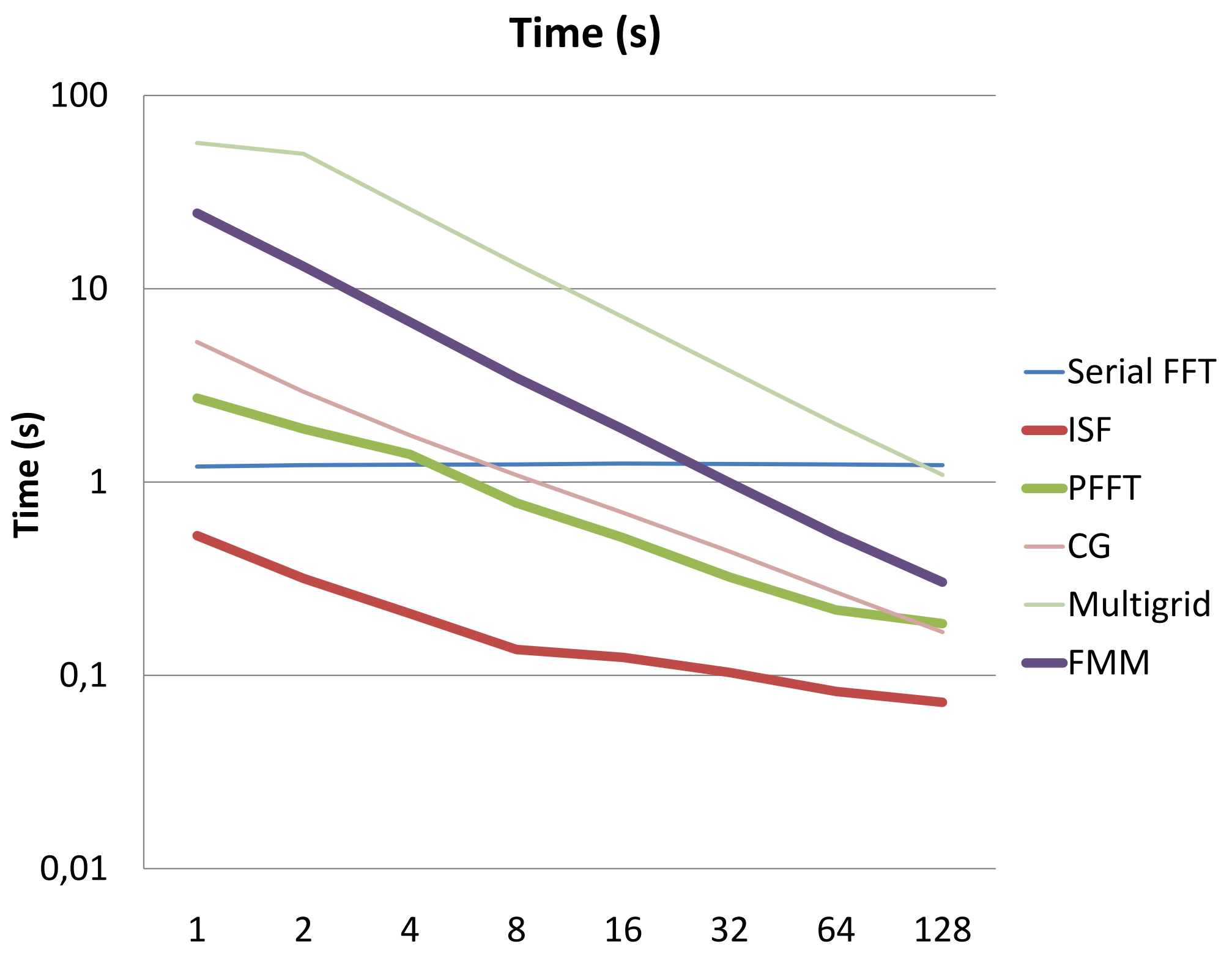
The version of FMM explained so far is suitable to calculate the potential created by pointlike charged particles. In our case, yet, we have a collection of points in a grid in space, each with an associated value of the charge density, so we adapted the potential calculated by this FMM (=VFMM) in three ways:

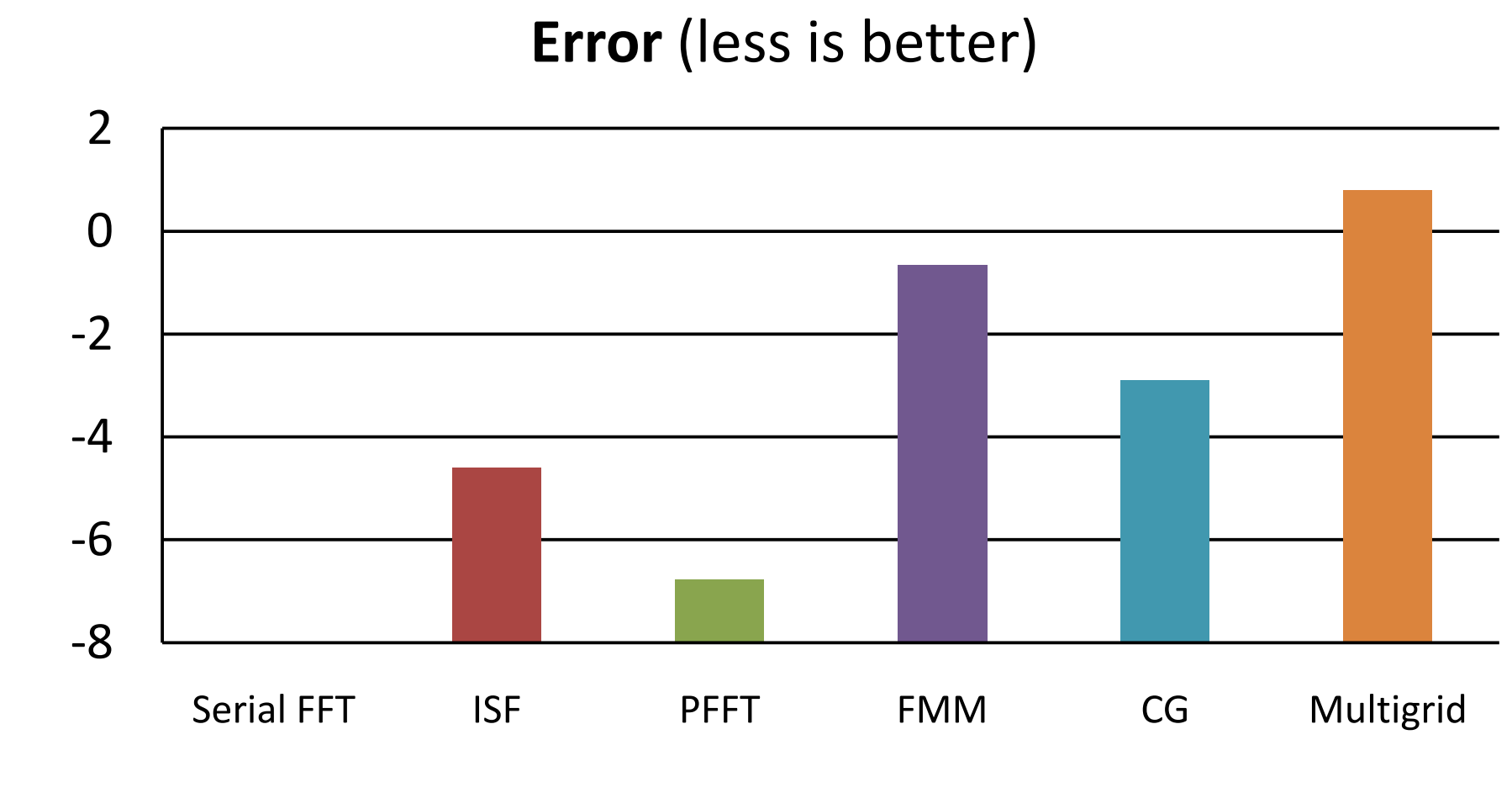
1. Since in every point we have a density of charge ρ rather than a charge, we had to multiply the density in each point by the volume of the associated grid cell W_{cell} (which is commonly a constant value).
2. At every grid point, we had to calculate a potential created by the density within the grid cell centered in that point.
3. In order to reduce the error without losing efficiency, we built a correction term for the potential in every point. It calculates the density in 6 points neighboring.

Performance

The result of our implementation of the Poisson solver is better than the solvers implemented before. The serial FFT and the improvement of it (the ISF library) have poorer performance than the new PFFT and FMM. Although old solvers need less time to execute with few nodes, increasing the number of nodes the performance of the new solvers is better. Actually, the crossbar is in 4 nodes for PFFT and in 32 nodes for FMM.







Accuracy: Log_{10}(100*(Numerical_error – Analitical_value)/Analitical_value

The FMM library shows an extraordinary performance, but the accuracy is not yet good. We are working in a correction that will be available in the near future.

The PFFT library, instead is very accurate, but is not so efficient yet. We have to reduce global communications, and use point-to-point communication instead.

References:

(1) Castro, A.; Appel, H.; Oliveira, M.; Rozzi, C. A.; Andrade, X.; Lorenzen, F.; Marques, M.L.; Gross, E. K. U.; Rubio, A. Phys. Stat. Sol 2006, 243, 2465.

(2) Pippig, M., An Efficient and Flexible Parallel FFT Implementation Based on FFTW Proceedings of the HPC Status Conference of the Gauß-Allianz e.V., Schwetzingen Castle June 22-24, 2010 (accepted)

(3) Kabadshow, Ivo (2006). The Fast Multipole Method - Alternative Gradient Algorithm and Parallelization Jül-4215, 2006, 78 Seiten

(4) Hestenes, M. R.; Stiefel, E. J. of research of the national bureau of standards 1952, 49, 409–436.

(5) (a) Wesseling, P. An introduction to multigrid methods; John Wiley and Sons, 1992; (b) Zhang, J. J. Comp. Phys. 1998, 149, 449–461; (c) Briggs, W. L. A multigrid tutorial; Wiley, New York, 1987; (d) Brandt, A. Math. Comput 1977, 31, 333–390; (e) Beck, T. T. Rev. Mod. Phys. 2000, 72, 1041.

* From Wikipedia

Contact:
Joseba Alberdi joseba.alberdi@ehu.es
Joxe Mari Korta eraikina, Tolosa hiribidea 72
20018 – Donostia (Gipuzkoa)

Special thanks to:

Ivo Kabadshow	Agustin Arruabarrena
Michael Pippig	Xavier Andrade
Angel Rubio	Rechenzentrum Garching
Javier Muguerza	Forschungszentrum Jülich