JOINT INSTITUTE FOR NUCLEAR RESEARCH
Veksler and Baldin laboratory of High Energy Physics

FINAL REPORT ON THE SUMMER STUDENT PROGRAMM

Perfomance improvement of Ef software with OpenMP and MPI

Supervisor:
Alexey Boytsov

Student:
Rogalev Kirill, Russia
Saint-Petersburg State University

Participation period:
June 30 - August 16

Dubna — 2019

Contents

1 Abstract

Using OpenMP and MPI technlogies and CUSP numerical library we enhanced perfomance of the Ef software package - freeware software that is used for simulation of charged particle dynamics. Instead Jacobi method for solving discrete Poisson equation conjugated gradient method was implemented. It allows to perform complex parcticle-in-cell (PIC) calculations for EBIS modeling applications in acceptable amount of time.

2 Introduction

Electron beam ion sources (EBIS) are commonly used as sources of high charge ions that are injects into a accelerator facility. EBIS sources are complicated enough, therefore computer simulations of particle motion in them during work is necessary for their desiging. The direct calculation of binary Coulomb interaction is almost impossible for significant number of interacting particles, but PIC method solved this problem [1]. Shortly, instead of direct computation of every binary interaction to evaluate result electric force on a particle PIC method propose to substitute discrete set of charges $\{Q_i\}_{i=1}^{N}$ by continuous charge density distribution $\rho(r)$ and solve Poisson (in electrostatic approximation) equation $\Delta\phi = -4\pi\rho$ in SGS units. In practice continuous charge density and Poisson equation are used in their disctretitized form as:

$$\frac{\phi_{i-1,j,k} - 2\phi_{i,j,k} + \phi_{i+1,j,k}}{\Delta x^2} + \frac{\phi_{i,j-1,k} - 2\phi_{i,j,k} + \phi_{i,j+1,k}}{\Delta y^2} + \frac{\phi_{i,j,k-1} - 2\phi_{i,j,k} + \phi_{i,j,k+1}}{\Delta z^2} = -4\pi\rho_{i,j,k} \tag{1}$$

where i, j, k indices correspond to x, y, z coordinate axes. One have to solve system of M linear equations instead of system of $N(N-1)/2$ equations in Coulumb case. Here M is the number of points where potential is computed and N is total number of particles, which usually much larger than M. The electric field in node points is calculated as a finite gradient of potential; electric field between nodes is calculated as superposition of electric field values in adjacent nodes with weight coefficients.

Computer programs using PIC are generally consist of two modules - field solver and pusher. Field solver does previously discribed work. Pusher calculates values of electric and magnetic fields at particle positions, defines resultant forces and changes particles positions and velocities. In theory both of those modules are highly paralellizable. Our work related to such computer program — Ef software [2], [3]. Ef is freeware software based on PIC method and that is being developed by A. Yu. Boytsov and A. A. Bulychev. This software is required for EBIS development.

The first part of this work was dedicated to realization CG method for field solver module of Ef and comparing of efficiency of CG and Jacobi solvers. The second part was much smaller and in fact related mainly to restore the code of parallelized pusher which was implemented and deleted later. Also this paper presents the results of the simulations for different numbers of OMP-threads and MPI-processes.

2.1 OpenMP and MPI

OpenMP and MPI technolgies are widely used in parallel computing applications.

OpenMP is an API that supports multi-platform shared memory multiprocessing in programming C, C++ and Fortran. It allows to run application in several parallel threads that have same memory space by default. User can make variables private to protect the code from data race. In our programm OpenMP is used in both field solver and pusher. OpenMP can engage all cores within one CPU but there is no way to use multiprocessor systems as clusters with maximum efficiency. MPI helps to overcome this problem.

MPI is the standart of message passing between processes. Unlike OpenMP-threads every MPI-process has their own memory space and data exchange is made through sending data from one process to another. MPI-processes can occupy both different cores within one CPU and different CPUs inside one computer system. Within every process OpenMP-parallelization is permitted. In this way OpenMP-MPI bundle provides opportunity to use available resources of supercomputer or computer cluster more rationally. In our programm MPI is used for pusher realization.

2.2 CUSP library

CUSP is a library for sparse linear algebra based on Thrust — lower-level library allows to speedup computations using CUDA on GPUs or OpenMP on CPUs (due to several architecture features we decided to use CPU mode). CUSP contains several methods to solve linear systems including Krylov methods (such as CG method and conjugate resudals method), relaxation methods (Jacobi, Zeidel), e.t.c. All ones need is to construct the $\mathbf{A}$ matrix, allocate $\mathbf{x}$ unknown vector, build $\mathbf{b}$ right-hand vector, set tolerance and choose a solving method.

3 CG implementation

Previously Jacobi method was used to solve discrete Poisson equation. The simplicity of that method is it does not requires to built explicit linear system to solve but slow convergance rate is price of it. The conjugate gradient method (CG) is free from this drawback but require to built linear system $\mathbf{Ax} = \mathbf{b}$ corresponds to equations (1), where $\mathbf{A}$ is Poisson matrix, $\mathbf{x}$ is vector of potentials and $\mathbf{b}$ is right hand charge density vector. Luckily CUSP library has realized CG method for linear systems, so direct implementation of CG is not needed. But the constructing of matrix $\mathbf{A}$ for any more complicated solution area geometry than right parallelogram is non-trivial task, and we spent significant amount of time to implement matrix building algorithm.

3.1 Jacobi and CG comparing

To compare the efficiency of both methods we use model example where disk-shaped emitter in free space (without external fields) ejects electrons with same momenta directed along the disk axis. The simulation time was equal $T = 3 \cdot 10^{-9}\, s$, time step was $\Delta t = 3 \cdot 10^{-11}\, s$, and every 10 steps data was written to HDF5 file. The spatial distribution of electrons looked as shown at figure 1.

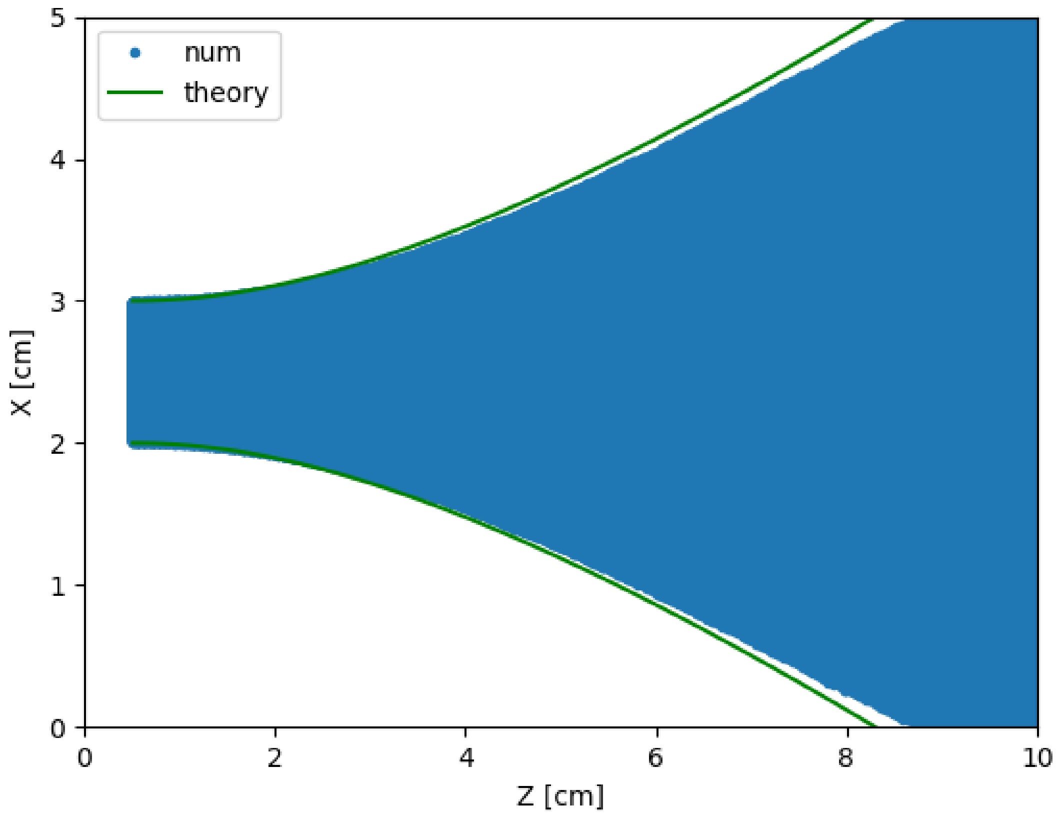

Figure 1: Spatial distribution of electrons in model example

The grid size is $50 \times 50 \times 100 = 250000$ nodes. Initial number of particles is 5000 and it increase by 5000 every time step. The convergence criteria of the solver defined as:

$$r \leqslant \frac{||\mathbf{A}\mathbf{x}^k - \mathbf{b}||}{||\mathbf{b}||} \tag{2}$$

where $\mathbf{x}^k$ is the potential vector generated at k iteration, r is relative tolerance.

We conducted several simuations for both methods with different relative tolerance for solver without parallelization. Their result are shown in table 1.

Relative	Computation time, s	
tolerance	Jacobi method	CG method
10^{-1}	50 ± 2	21.3 ± 0.3
10^{-2}	290 ± 10	42 ± 1
10^{-3}	1080 ± 40	71.5 ± 0.5

Table 1: Comparation of programm execution time for Jacobi and CG methods

This simulations was performed on Intel Core i3-4005U 1.7 GHz machine.

Here we can see that conjugate gradient method works much faster than Jacobi at a relatively high accuracy. It significantly reduce the amount of time which spent field solver module of Ef during simulation. Now let's look how good OpenMP helps to reduce it even more.

We simulated the same example on the Intel Xeon E5-2680 v3 2.50 GHz machine with 6 cores and got following results:

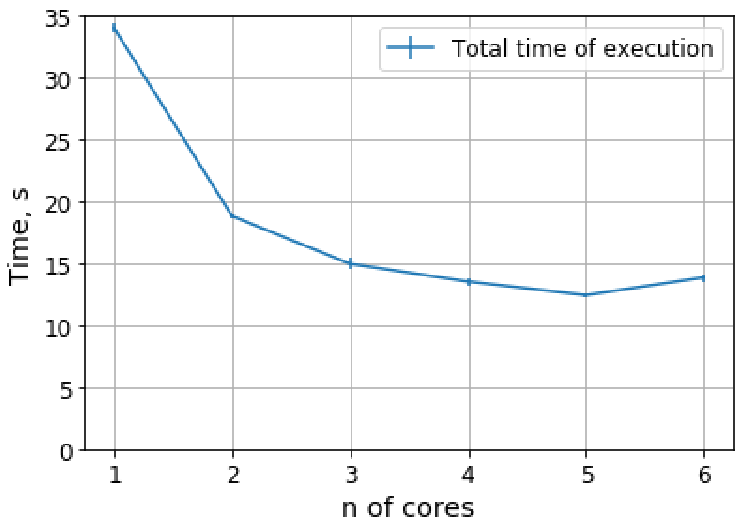

Figure 2: Time of computations depending on the number of cores

4 MPI realization and testing

MPI is well suited for the pusher because most of operations on particles in PIC method are well-parallizable and they are not require synchronization and interprocess data transmission in contradicton to solver. Every MPI-process storages their own portion of particles (more precisely, data related to those particles), pushes this particles using information about total electromagnetic field, computes charge density distribution related with only this bunch of particles. A single synchronization point is a summing of the charge densities to compute the total field by the field solver at the next time step.

To test Open-MPI version of Ef we took modificated version of the example used in section 3.1. The charge, mass and initial momentum of all particles was reduced by 10 times and intensity of particle injection was increased by 10 times. It was made for to enlarge the share of time that spends on the pusher work. We ran simulation with various numbers of OpenMP-threads and MPI-processes and got following results.

The simulations was ran on the Hibrilyt claster at two nodes with two Intel Xeon Phi E5-2695 v2 processor with 12 cores each.

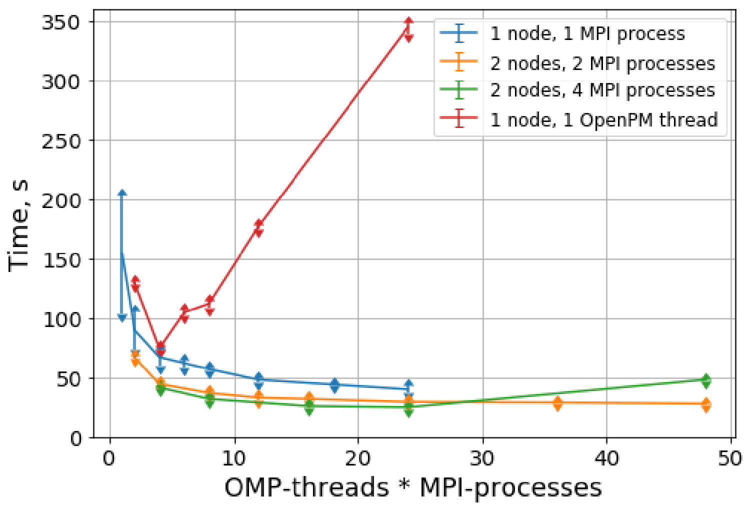

Figure 3: Time of computations depending on the number of cores

Looking at this figure we can say that acceleration of program strongly depends on how parallel branches of code are distributed between OpenMP-threads and MPI-processes. Moreover, in case of single OpenMP-thread the programm begin to work slowly after 4 MPI-processes. At the moment, we did not find an explanation for this.

5 Conclusions

Replacing the Jacobi field solver by CG field solver and returning MPI realization of pusher we achivied the perceptable enhance of perfomance - now practical simulations may take several days or weeks instead of months or years. However, now we can not explain the results presented on the figures. Understending of them may help to increase perfomance more.

6 Acknowledgments

I am very grateful to the JINR ESIS group for their help and advice regarding this project. Especially I would like to thank my supervisor Alexey Boytsov for consultations, transfer of invaluable knowledge and personal excursion to the Nuclotron. Several computations were held on the basis of the heterogeneous computing cluster HybriLIT (LIT, JINR). I would also like to thank the organizers of the student summer program at JINR and AYSS for performing of the wonderful educational program.

References

[1] Y.N. Grigoryev, V.A Vshivkov, M.P Fedoruk, *Numerical "Particle-in-Cell" Methods: Theory and Applications.* (VSP, 2002)

[2] A.Yu. Boytsov, A.A. Bulychev, *Ef: Software for Nonrelativistic Beam Simulation by Particle-in-Cell Algorithm.* EPJ Web of Conferences **177**, 07002 (2018)

[3] *Ef* (2017). Available at: https://github.com/epicf (accessed 15 August 2019).