Optimizing EUTERPE for Petascaling

Xavier Sáez*, Taner Akgün, Edilberto Sánchez

*Corresponding author. E-mail address: xavier.saez@bsc.es

Available online at www.prace-ri.eu
Partnership for Advanced Computing in Europe

Abstract

In this paper we report the work done in Task 7.2 of PRACE-1IP project on the code EUTERPE. We report on the progress made on the hybridization of the code to MPI and OpenMP; the status of the porting to GPUs; the outline of the analysis of parameters; and the study on the possibility of incorporating I/O forwarding to improve performance. Our initial findings indicate that particle-in-cell algorithms such as EUTERPE are suitable candidates for the new computing paradigms involving heterogeneous architectures.

Application Code: EUTERPE

1. Introduction

EUTERPE is a gyrokinetic particle-in-cell (PIC) code for the global linear and non-linear simulations of fusion plasma instabilities in three-dimensional geometries, in particular in tokamaks and stellarators. It was created at Centre de Recherches en Physique des Plasmas (CRPP) in Switzerland, and has subsequently been further developed at Max-Planck-Institut für Plasmaphysik (IPP) in Germany.

The code solves the non-linear gyrokinetic equations by discretizing the distribution function using markers. Each marker contributes a part of the distribution function whose evolution is given by the gyrokinetic Vlasov equation. The delta-f approximation is employed in the calculation of the distribution function in order to reduce the discretization noise and the computational resources needed. Both the magnetic and the electric fields are taken into account in the equations of motion of the markers. The electric potential at each point of the spatial grid is obtained by solving the quasi-neutrality equation, while the electric charge in the grid points is given by the markers in their vicinity. The electric field is obtained from the potential values at the grid points. Once having the fields, the markers are advanced according to the gyrokinetic equations and their new positions are computed. The magnetic field structure used in the simulation is computed using the Variational Moments Equilibrium Code (VMEC), which is the standard code for calculating three-dimensional equilibria. EUTERPE uses a Fourier transform library (either one of the free FFTW, IBM’s FFT_ESSL, or Intel’s FFT_MKL) and a sparse matrix library (either the free PETSc or IBM’s WSMP library).

EUTERPE is at the forefront of plasma simulations and requires a huge amount of computational resources. It provides good results both in linear and non-linear simulations of ion temperature gradient (ITG) instabilities carried out in screw-pinched geometry [1]. Simulations of plasma micro-in-stabilities are crucial for future stellarator and tokamak experiments such as Wendelstein 7-X and ITER. The full kinetic treatment of the electrons and the inclusion of magnetic field perturbations extend the scope of applicability to the magnetohydrodynamic (MHD) regime and will make EUTERPE the first gyrokinetic code worldwide that is able to simulate global electromagnetic instabilities in three dimensions. The Fusion Theory unit from Centro de
Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT) and Barcelona Supercomputing Center (BSC) in Spain collaborate with IPP on the development and exploitation of this code.

The parallelization strategy consists of a domain cloning concept, which is a combination of particle decomposition and domain decomposition. The simulation domain is cloned (multiple copies of the same domain are made) and the processing elements (PEs) are divided into as many groups as clones. Each group of PEs is assigned to one of the domain clones. Each domain clone is loaded with its own set of particles and a one-dimensional decomposition is performed in the toroidal direction, such that each PE has a corresponding subdomain clone. This parallelization strategy has been implemented using the MPI library and it allows the number of PEs to be chosen independently from the number of equilibrium points.

The code has been heavily optimized and its scaling has been studied to 2048 IBM Power6 (PEs) in the framework of the DEISA project. In the framework of the PRACE project, it has shown excellent performance up to several thousands of processors, for example, up to 2560 PEs on the Huygens supercomputer (SARA). For larger number of PEs, the scaling has been quite good after increasing the number of markers (up to 5 billion markers). In particular, it has been investigated for up to 61440 PEs on the JUGENE supercomputer (FZJ) with an average efficiency of 71% [1,2].

The amount of computational resources that a global three-dimensional PIC code requires for typical simulations is huge. A single simulation of turbulence transport of several hundreds of microseconds in a cylindrical domain could take 100,000 CPU-hrs. An extension of the simulation time and domain could reach easily two orders of magnitude higher than this time. Petaflop machines are a critical and necessary infrastructure in this scenario.

2. Work done

2.1. Improving the new solver

The parallelization of the code had previously been extended by means of introducing OpenMP in the most time-consuming routines and developing a hybrid solver (mixing MPI and OpenMP) to solve the quasi-neutrality equation. This solver uses the Jacobi Preconditioned Conjugate Gradient (JPCG) method [3]. We have focused our investigation on the reduction of the number of iterations to find the solution. Therefore, we developed a more complex version of the hybrid solver. In this version, the preconditioner is not only the diagonal of the matrix, but also a set of diagonal blocks. Clearly, there is more workload but the convergence is much faster. This improved method is called the Block Jacobi Preconditioned Conjugate Gradient (BJPCG). It is also a hybrid code and takes advantage of threads. We performed a test to evaluate the performance of BJPCG when a task is assigned to more than one processor (for example, when more memory is needed to fit the problem data into a task). Different executions were done using the same test case and changing the number of processors and clones. The measured times were the times of one completed step iteration. The results are shown in Fig.1 and suggest that the speedup is good. The work for introducing OpenMP is thus completed.

![Fig. 1. Step time of EUTERPE using different solvers. For the BJPCG solver, a block of 128x128 was considered.](image-url)
2.2. Porting the code to GPU architecture

The first task was to identify candidate routines to port them to GPUs and add a C interface to call them from the Fortran code. We decided to start with the solver for the quasi-neutrality equation. The first routines we considered were: prod (computing the dot product), mult_Ab_asim (computing the matrix-vector product) and mult_jacobian_matrix (computing the block Jacobian matrix-vector product).

These routines were coded using the following basic guidelines: (i) copying data from host to GPU, (ii) executing a kernel on GPU, and (iii) copying results from GPU to host. This allowed us to understand and test the CUDA framework. A more detailed analysis using the Scalasca tool [4] on JUGENE suggested that particle push and current deposition are the parts of the code that would be needed to be ported to GPUs in order to accelerate the execution of EUTERPE (Tab.1).

Table 1. Time spent at each routine.

<table>
<thead>
<tr>
<th>Percentage time</th>
<th>Routine / OMP loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.83</td>
<td>_fields_NMOD_getfield</td>
</tr>
<tr>
<td>16.19</td>
<td>!omp do @part_omp.F90:297</td>
</tr>
<tr>
<td>10.31</td>
<td>!omp do @part_omp.F90:1589</td>
</tr>
<tr>
<td>10.26</td>
<td>_equil_NMOD_equil_grad</td>
</tr>
<tr>
<td>8.06</td>
<td>!omp do @bsc_cg.F90:1722</td>
</tr>
<tr>
<td>7.50</td>
<td>_equil_NMOD_equil_xy2sc_ra</td>
</tr>
<tr>
<td>6.55</td>
<td>!omp do @bsc_cg.F90:1697</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

The chosen way to proceed was working on the particle push, because there were not memory access conflicts as in the current deposition. A brief outline of the implementation carried out in this routine is as follows: at the beginning of the routine, the required data is copied to the GPU (particle information); then the work is done in the GPU, and when the particle push is finished, the data is copied back to the host and the execution continues normally. The result is a straightforward implementation working with global memory. It is possible that some unnecessary memory transfers are being done between the GPU and host, so this part requires further tuning.

2.3. Evaluating the potential of I/O forwarding to improve I/O perfomance of EUTERPE

This work was done in collaboration with Task 7.6 of the PRACE-1IP project with the participation of Jan Christian Meyer and Jørn Amundsen (NTNU, Norway) [5].

On HPC systems, the gap between computation and I/O capability has increased by a factor of 10 in recent years, so it is interesting from the performance point of view to evaluate the insertion of I/O forwarding software layers between applications and the file system layer. An analysis of EUTERPE has revealed that its I/O behavior can be replicated by a performance model. Therefore, a model function parametric was built and its purpose was to estimate write-bandwidth without running EUTERPE to evaluate potential performance with I/O forwarding.

The execution has showed four main sources of I/O activity: an initialization phase, periodic updates of a histogram file, periodically storing restart states, and a finalization which included storing a restart state. The study was focused on the periodic behavior, because its cost grows in proportion to application run time, while initialization and finalization are constant. The I/O mode of EUTERPE is to access files per MPI process. The histogram files were found to have a regular size, while the restart files reflect the state of ions in the simulated plasma, so their size displayed inherent variability.

Although the borderlines in the test case were too small to verify it empirically, addressing the I/O requirements of EUTERPE was most relevant at large scale with checkpoints. This means that the defining characteristic of the relevant I/O load is a large number of comparatively small files.

This study concluded that while a file creation bottleneck is observed, which can be reduced by user-space I/O forwarding, using the IOFSL forwarding software layer comes at prohibitive costs. This conclusion was
based on the cost balance observed using nodes with small number of powerful processing cores, and may not be appropriate for massively parallel, simplified core architectures.

2.4. Analysis of parameters

The quality of the simulations carried out with EUTERPE can be measured in terms of two parameters: the signal to noise ratio (S/N) and the degree to which energy is conserved. The signal to noise ratio here is defined as the ratio of the spectral power kept inside the (noise) filter to the spectral power filtered out. In tests carried out with varying step sizes, number of markers, and grid sizes it turns out that the signal to noise ratio does not depend on the step size, but the noise decreases (i.e. the quality of the simulation increases) with increasing number of markers, and the noise increases (i.e. the quality decreases) with increasing grid size.

A second measure of the quality of a simulation is the difference between the electrostatic and kinetic energies of the markers, which is a direct indication of the degree of the conservation of energy throughout the simulation. Using this criterion, it is observed that increasing the time step and decreasing the number of markers both degrade the quality of the solution, with the time-step being the more important effect. This is in agreement with previous studies [2].

The original goal was to perform a more detailed analysis of the various parameters that affect the performance of EUTERPE using hundreds of thousands of PEs. However, due to time and resource limitations this objective had to be scaled down.

2.5. Test evaluation

The initial intention was to run a simulation in line of those that had been performed in the framework of the project “Non-diffusive transport in ITG plasma turbulence”. The idea was to perform a detailed simulation in a cylinder or tokamak geometry with very good statistics.

However, owing to unforeseen delays in the porting to GPUs, it was not possible to realize these plans. Therefore, we performed a different simulation that was scientifically relevant.

The executed test case simulated the Wendelstein 7X stellarator with 256 million ions and 1024 million electrons on a grid of 100x128x128 with quadratic B-splines. Unfortunately, there was not enough time to advance further in this analysis. The numbers obtained so far are presented in Tab.2.

Table 2. Preliminary scaling results on JUGENE.

<table>
<thead>
<tr>
<th>PEs</th>
<th>Step time (s)</th>
<th>Speed up</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>684.3</td>
<td>1.00</td>
</tr>
<tr>
<td>1024</td>
<td>347.9</td>
<td>1.97</td>
</tr>
<tr>
<td>4096</td>
<td>95.4</td>
<td>7.17</td>
</tr>
<tr>
<td>8192</td>
<td>54.4</td>
<td>12.58</td>
</tr>
</tbody>
</table>
3. Conclusions

The hybrid version of EUTERPE, with the BJPCG solver, shows improved performance over the MPI version. Our initial findings on the porting to GPUs also indicate that particle-in-cell algorithms such as EUTERPE are suitable candidates for the new computing paradigms involving heterogeneous architectures. Therefore, we will continue the work in the direction to advance the porting of the code to GPUs. On the other hand, while I/O forwarding is found to be potentially beneficial, at the moment it comes at an inhibitive cost on the performance.

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-211528 and FP7-261557. The work has been achieved using the PRACE Research Infrastructure resources.

References