Congress on Numerical Methods in Engineering
Lisboa, June 29 to July 02, 2015

RESUMO N° 313

PERFORMANCE ANALYSIS OF A PARTICLE-IN-CELL PLASMA PHYSICS CODE ON HOMOGENEOUS AND HETEROGENEUS HPC SYSTEMS

Xavier Sáez, xavier.saez@bsc.es
BSC, Spain

Alejandro Soba, soba@cnea.gov.ar
CNEA, Argentina

Edilberto Sánchez, edi.sanchez@ciemat.es
CIEMAT, Spain

Mervi Mantsinen, mervi.mantsinen@bsc.es
ICREA, BSC, Finland

José María Cela, josem.cela@bsc.es
BSC, Spain

Keywords: PIC Code, HPC, Performance Analysis, GPU, Multi-Core, Plasma Physics

During the last few decades, high-performance computing (HPC) has been dominated by the rapid scaling of the CPU clock frequency (c.f. Moore's Law). Currently, the performance of next-generation supercomputers is limited by the power efficiency and, as a result, several novel hardware designs have emerged.

Originally, many scientific codes were not developed for present designs of supercomputers based on multi-core and heterogeneous architectures. Basically, they were only leveraging task level parallelism through message passing models such as MPI. Unfortunately, a hand-tuning of these codes is often required to exploit the modern platforms capabilities.

We explore the particle-in-cell (PIC) method through a production plasma physic code (EUTERPE) which is a gyrokinetic PIC code which simulates fusion plasma instabilities in three-dimensional (3D) geometries. The PIC method is one of the most used methods in plasma simulations. Macroscopically, it describes the plasma dynamics by a system of partial differential equations (continuous model) while microscopically a set of discrete particles is used. The quality of the results achieved with this method relies on tracking a very large number of particles. Therefore, PIC codes require intensive computation and their adaptation to new computing platforms is of particular interest.

We will present a comprehensible evaluation of the PIC code performance on four current parallel platforms: IBM PowerPC, Intel Nehalem, Intel Sandy Bridge and ARM. The behavior of computational algorithms and data structures are analyzed to deduce which code optimizations will make the best use of each platform.

Moreover, in order to check the ported code for a realistic problem, we perform a number of tests in 3D geometries. Traditionally, 3D simulations have been very time-consuming even in the simplified linear limit but are becoming more accessible given the code optimization on modern computing platforms such as those analyzed here.