Report on the 8/3/2019 Meeting

Marcello Sega, Manuel Zellhöfer

May 2019

For most of the investigations performed in our group, we make use of our LB3D software package to simulate soft matter systems at the mesoscopic scale. The package is built around a lattice Boltzmann engine, which takes care of computing the evolution of the hydrodynamic fields and includes, among others, solvers for the electrokinetic (EK) equations, point-like particle dynamics, rigid and deformable bodies. Within LB3D it is possible to simulate multiphase and multicomponent flows, EK phenomena and the dynamics of solid and deformable finite size suspended particles. All the functionalities are properly coupled such that, for example, it is possible to study the dynamics of charged colloidal suspensions in binary electrolyte solutions. Particles are implemented using a fully parallel molecular dynamics (MD) solver and coupled through the fluid via boundary conditions. Concerning EK, ion fluxes are resolved at the continuum level and coupled to the fluid and particles through electrostatic forces computed using a parallel solver of the Poisson equation. A great advantage of the LB method is that it is intrinsically local in space, therefore allowing for a straightforward and efficient parallelization, which makes the code excellently scalable and perfectly suitable for large HPC machines. LB3D has been so far employed on clusters such as JUQUEEN/JURECA to simulate systems of 1024^3 lattice nodes on 16384/2048 cores in parallel routinely.

In the course of the last months, our code underwent considerable changes in data layout and source code structure, with a focus on the core lattice Boltzmann routines, to adapt the code to the features of machines with SIMD processing

units. Besides moving to an object-oriented paradigm, the reorganization of the code involved rewriting the more computationally intensive routines dedicated to the advection, collision, calculation of fields and communication. The data layout was changed from arrays of structures to structures of arrays. In particular, the populations of the D3Q19 lattice are now stored in a single contiguous array in the order

$$p_1(1,1,1)...p_1(n_x,1,1)...p_1(n_x,n_y,1)...p_1(n_x,n_y,n_z)...p_{19}(n_x,n_y,n_z),$$

for a box with (n_x, n_y, n_z) nodes along the x, y and z edges.

However, during the development and testing phase, it became evident that, at the single core level, the SIMD units were underperforming, even though we could not spot the origin of this problem. To profile the bandwidth usage and the amount of calculation performed in the SIMD units, we decided to use the LIKWID library. In particular, the MarkerAPI was appealing for us to use, in order to understand which part of the code had potential for improvement.

The meeting with Thomas Gruber on the 8th of March 2019 was dedicated to the following:

- An introduction to the capabilities of LIKWID and its MarkerAPI, including hands-on examples tailored to the data structures used in LB3D.
- In-depth explanations on how to interpret the output of LIKWID, in particular for cases relevant to the profiling of LB3D.
- Extension of the development branch of LB3D to include calls to the LIKWID library
- Performance measurements of specific hotspots in LB3D

The performance analysis made together with Thomas Gruber allowed spotting a problem in the generation of AVX2 instructions by the GNU Fortran compiler. The compiler was not able to generate AVX2 code due to a missing declaration as `contiguous` of a pointer to memory allocated using `malloc`.

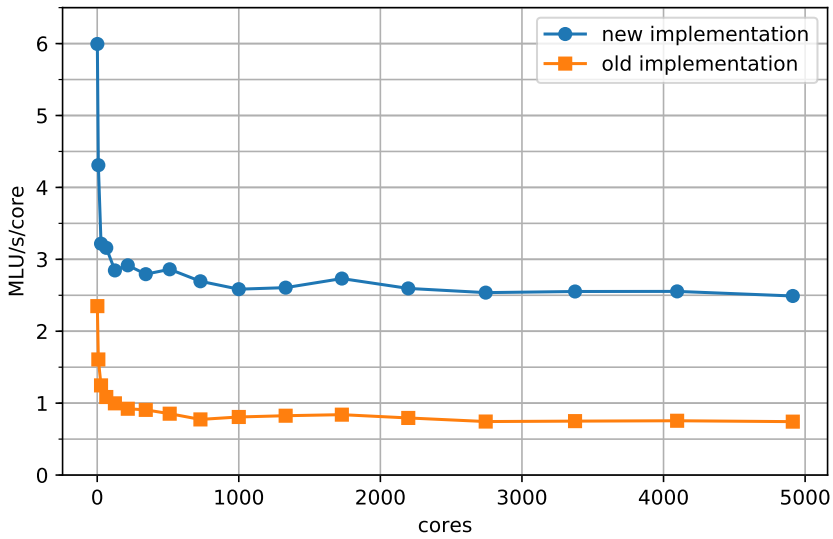

Figure 1: Weak scaling on Hazel Hen in MLU/s/core for the single fluid lattice Boltzmann in LB3D, old and new implementation. The whole domain is a cube of side 16 and 32 nodes/core, corresponding to the optimal sizes for the new and old implementations, respectively.

Thanks to the visit of Thomas Gruber, we increased the performances of about 20%. This is however not the only outcome, as clearing this issue allowed to start optimizing several other regions of the code, bringing the new implementation to peak performances of about 7.5 million lattice updates per second (MLU/s) on the Skylake machines on our local cluster, to be compared with the older implementation, which reached 2.5 MLU/s.

The results of weak scaling benchmarks are reported in Fig.1, where it is possible to appreciate the more than three-fold improvement in performance, which is roughly constant over a range of roughly 5000 cores employed on the Hazel Hen cluster at HLRS (Haswell). To perform the weak scaling, we kept the number of lattice nodes per core fixed to the value that gives optimal performance on a single cluster node (16^3 and 32^3 lattice nodes per core, for the new and old implementations, respectively).

Given the extremely positive experience in using LIKWID to analyze the performance of our code, we added it to our testing framework in order to automatically monitor the performance of the code in the course of future developments.