Description of the Codes

Over the last years, the lattice Boltzmann method proved very successful in modeling fluids in many different applications for science and engineering. Compared to traditional Navier Stokes solvers, the method allows an easy implementation of complex boundary conditions and—due to the high degree of locality of the algorithm—is well suited for the implementation on parallel supercomputers.

LB3D is a general purpose lattice Boltzmann implementation developed over the last 12 years at three research institutes residing in the UK, Germany, and the Netherlands. Among other applications, the code is used to study the rheology of ternary fluid mixtures [1] and blood flow under various conditions [2]. The main benchmark systems during the workshop are emulsions stabilized by colloidal particles (Pickering emulsions) [3], for which a study of droplet size distributions demands simulations of considerable spatial and temporal dimensions.

HYPO4D is a four-dimensional extension to the existing lattice Boltzmann implementation, which parallelizes in time as well as in space. The parallelization in time aids us in identifying unstable periodic orbits (UPOs) in turbulent flow. These UPOs provide a countable sequence of orbits and can therefore allow us to characterize the structure and dynamics of the attractor of the system [4]. To accomplish this, we apply our solver with a novel spacetime variational algorithm, which minimizes a candidate orbit to a UPO.

Results

First benchmarks of LB3D with additional timing functions performed during the workshop showed that the coupling of suspended particles to the fluid involved a subroutine that scaled linearly with the number of cores. Due to the small pre-factor this was not recognized earlier. Another finding was that the (serial) output of the Lagrangian particle trajectories would fail for large core counts since it relied on point-to-point communication between every rank and
the root process instead of collective MPI calls. Turning off Lagrangian output, we could still successfully run the code on 294912 cores for the first time and produce parallel HDF5 output of velocity and density fields for a volume of $1024^2 \times 1152$ lattice sites containing two fluid components and 454508 colloidal particles at a volume concentration of 20%.

In discussions, mainly with Pascal Vezolle (IBM), we learned that our code could be optimized by better exploiting the network topology of JUGENE and implementing a hierarchical output scheme for HDF5 data in which only part of the tasks perform actual file IO. However, due to the limited time, it was not possible for us to implement major optimizations at the workshop. Still, in the following weeks, we enabled LB3D for manual matching of the physical geometry of the problem to the hardware topology. This change is the main cause for the improved speedup for a benchmark system of $1024^2 \times 2048$ lattice nodes containing only one fluid species as shown in Figure 2(a). Further improvement seems possible by integration of recent optimizations performed for HYPO4D. Also the linear contribution to the computational cost of the particle-fluid coupling was eliminated. The effect of both modifications together is visible in Figure 2(b) comparing the speedup before and after the workshop for a system of $1024^2 \times 2048$ lattice sites carrying two fluid components and 4112895 colloidal particles.

We also had the opportunity to test and diagnose a number of optimizations which we recently inserted in the HYPO4D code. During the workshop we found that the hand-written collective communication and File I/O operations posed a barrier to perform diagnostics on the code for runs beyond 65536 cores and to scale the code in general beyond 131072 cores. As a result, we spent a large part of the workshop simplifying and rewriting these collectives. This effort has been partially successful as we are now able to run across 131072 cores with diagnostics enabled. Unlike previous version, the use of the collective operations in the diagnostics no longer adversely affects the performance or stability of the code. In addition, we eliminated some of the errors which prevented us from
Figure 2: Strong scaling of LB3D on JUGENE before and after the workshop. (a) relates to a system with only one fluid component, (b) to one with two fluid species and suspended particles similar as in Figure 1(b). The absolute execution times for small core counts did not change significantly.

running across 262144 cores, and are in the process of diagnosing and resolving several other issues which prevent the code from completing successfully at that core count.

Converting the Lagrangian output of LB3D to collective operations cost unexpectedly large effort: first, sending custom data types of zero lengths in MPI \texttt{gatherv()} led to errors which were hard to identify since the MPI call still pretended to have finished successfully. Secondly, MPI \texttt{gatherv()} proved unacceptably slow. Using MPI \texttt{allgatherv()} instead, we could speedup the communication by a factor of 77 on 131072 cores at the cost of allocating large receive buffers for every task. In first tests, the new implementation successfully wrote trajectories and full checkpoints for 435600 particles on 131072 cores.

**Outlook**

While all groups using LB3D profit from its increased efficiency the improvement of the Lagrangian output is a crucial prerequisite for massively parallel production runs. These will be employed soon for the study of Pickering emulsions. The know-how obtained at the workshop is likely to foster further optimizations in the future.

The knowledge acquired at the workshop also led to a number of revisions in the HYPO4D code already, but we have planned a large number of additional changes. First priority is to eliminate the remaining obstacles which prevent us from running across 262114 cores. We also intend to implement the hierarchical file I/O solution described earlier in this report, and to optimize the 4D relaxation scheme both for accuracy and performance.
Figure 3: Weak scaling of HYPO4D on JUGENE

Acknowledgments

Besides support from the experts from JSC and IBM present at the workshop we acknowledge fruitful discussions with our colleagues Luis Fazendeiro, Stefan Frijters, Sebastian Schmieschek, and Florian Günther. Additional computing resources on JUGENE were granted by PRACE (project pra036, PI F. Toschi). Our visit was partially funded by the European MAPPER project (FP7:RI-261507).

References


