Parallel Computing: Technology Trends
I. Foster et al. (Eds.)

doi:10.3233/APC200090

Computational Performances and Energy Efficiency Assessment for a Lattice Boltzmann Method on Intel KNL

Ivan GIROTTO [a,c,d,1], Sebastiano Fabio SCHIFANO [b] Enrico CALORE [b]
Gianluca DI STASO [c] and Federico TOSCHI [c]

[a] *The Abdus Salam, International Centre for Theoretical Physics. Italy*
[b] *University of Ferrara and INFN Ferrara, Italy*
[c] *Eindhoven University of Technology, The Netherlands*
[d] *University of Modena and Reggio Emilia, Italy*

Abstract. In this paper we report results of the analysis of computational performances and energy efficiency of a Lattice Boltzmann method (LBM) based application on the Intel KNL family of processors. In particular we analyse the impact of the main memory (DRAM) while using optimised memory access patterns to accessing data on the on-chip memory (MCDRAM) configured as cache for the DRAM, even when the size of the data of the simulation fits the capacity of the on-chip memory available on socket.

Keywords. Lattice Boltzmann method, KNL, Cache mode, Flat mode, energy efficiency, computational performances

1. Introduction

The LBM [1] is widely used in computational fluid-dynamics to study the behaviour of fluid flows. Computational fluid-dynamics studies requires a huge amount of computational resources when simulating 3-dimensional systems at high-resolution. To study the behaviour of multicomponent emulsions, an amount of 40 millions core-hours was allocated, for a year, on the KNL partition of the MARCONI Tier-0 system, hosted at CINECA, within the PRACE 17th call, project named "TurEmu - The physics of (turbulent) emulsions". To successfully complete the project and perform the simulations on a large number of KNL processors, we developed a LBM based application which delivers good scaling performances on distributed systems while optimising the memory access through a data organisation that enables high computing efficiency. The overall code optimisation, as well as the optimised data layouts were introduced in [2], including a performance analysis in terms of both computing and energy consumption on the KNL processor. Other similar analysis were also previously studied on different computer architectures [3,4,5].

[1] Corresponding Author: Ivan Girotto; The Abdus Salam, International Centre for Theoretical Physics; Strada Costiera, 11 - 34151 Trieste - IT; E-mail: igirotto@ictp.it

In this paper we present a new analysis of computational performances and energy efficiency of our code on the Intel KNL family processors when using the on-chip memory as cache for the main memory even when the size of the data of the simulation fits the capacity of the on-chip memory available per socket. We show how optimized memory access patterns are efficient in using the on-chip memory. Indeed, accessing only the faster on-chip memory with generic data layouts, does not bring any benefit over using it as an additional cache level between the cores and the DRAM. On the other hand, we demonstrate that optimized data layouts allow our LBM application to fully exploit the on-chip memory, obtaining an higher performance and making it more efficient to use this memroy exlcusively, without involving the DRAM.

The rest of the paper is composed as follow: in section 2 we describe the main technical aspects of the KNL processor relevant to our analysis, in section 3 we introduce the scientific context we have been working on underlining the relevance of the performance analysis, in section 5 we present the performance results of the code on the KNL processors while, in section 6, an analysis on energy efficiency is reported.

2. The Marconi Tier-0 System and The KNL Processor

The KNL processor is equipped with 6 Double Data Rate fourth-generation (DDR4) channels at support of up to 384 GB, depending on the size of the memory modules, of synchronous Dynamic Random-Access Memory (DRAM) with a peak raw bandwidth of 115.2 GB/s. The processor also includes four high-speed memory banks based on the Multi-Channel DRAM (MCDRAM) that provides 16 GB of memory, capable to deliver an aggregate bandwidth of more than 450 GB/s when accessing the on-package high-bandwidth memory. Therefore, maximize the usage of the MCDRAM is a key aspect to achieve great performance for memory bounded applications. MCDRAM on a KNL can be configured at boot time in Flat, Cache or Hybrid mode. The Flat mode defines the whole MCDRAM as addressable memory allowing explicit data allocation, whereas Cache mode uses the MCDRAM as a last-level cache.

The Marconi KNL (A2 partition) was deployed at the end of 2016 and consists of 3600 Intel server nodes integrated by Lenovo. Each node contains 1 Intel Knights Landing processor with 68 cores. The entire system is connected via the Intel OmniPath network. CINECA provides to PRACE users, as well as to all other users the KNL nodes configured in Cache mode. Probably this is mainly due to the fact that performances of common applications are comparable on the KNL configured either in Cache mode or Flat mode when the size of the data of the application fits the 16 GB capacity of the MCDRAM, and the on-chip memory is used as cache of the DRAM memory. Indeed, as we show in this work, this is also confirmed by our analysis on LBM based applications when using common data layouts. On the other hand the Cache mode configuration is more flexible giving users applications the opportunity to exploit an higher capacity of memory per socket while relying on the MCDRAM as cache to maximise the memory throughput.

As we want to measure the impact of the DRAM when the KNL is configured in Cache mode and the dataset fits the memory available on the MCDRAM, we perform the performance analysis as well as the measures of energy efficiency on a 64-core Intel Xeon Phi CPU 7230 processor installed in our laboratory. The authors consider the sys-

tem equivalent to the one installed on Marconi as far as the present analysis is concerned. However, using a stand alone processor gives us the flexibility to make a number of tests by rapidly switching the configuration of the processor between Cache and Flat mode along with the opportunity to use the processor with root privileges that are required for measurements of energy efficiency. We only consider the Quadrant cluster configuration in which the 64-cores available are divided in four quadrants, each directly connected to one MCDRAM bank. The same configuration is used on Marconi for the PRACE projects production.

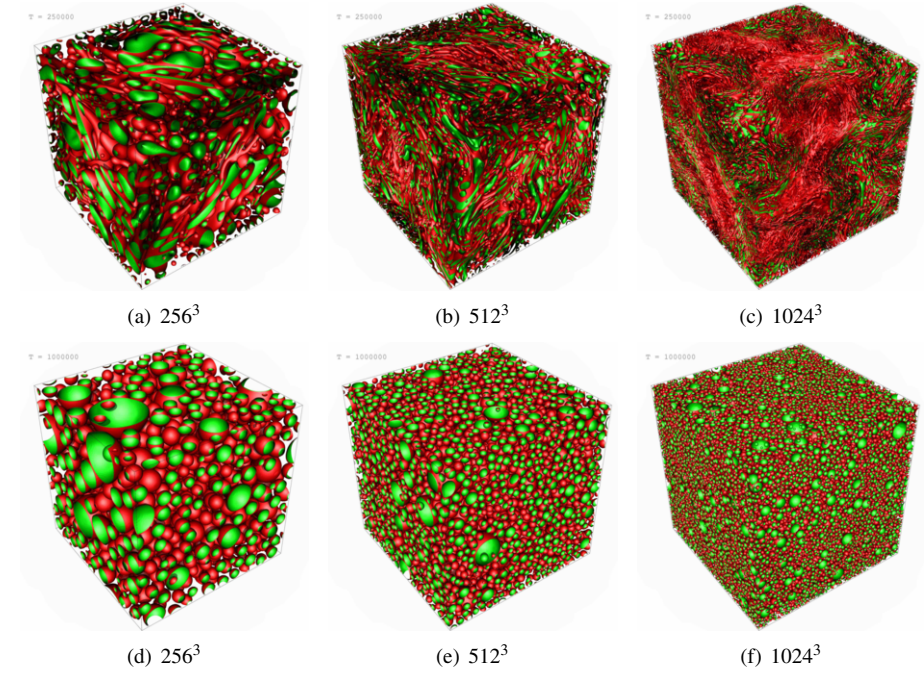
In this work, we used Intel library for Message Passing Interface (MPI) to compile the LBE3D application. Compiler auto-vectorization is activated at compile time using the -xMIC-AVX512 Intel compiler option. Multi-thread version of LBE3D is implemented using OpenMP and enabled at compile time. Threads affinity at run time is obtained with "KMP_AFFINITY=compact" and "I_MPI_PIN_DOMAIN=socket" environment variables. Memory allocation for all results related to the KNL configured in Flat mode is made by using the "`numactl -m 1 mpirun ./lbe3d`" command.

3. The Lattice Boltzmann Method

Multi-component fluids are extremely common in industrial as well as natural processes. In particular, multi-component fluids emulsions are an important ingredient of many foods and cosmetics, and at the same time fascinating systems from the point of view of fundamental science, due to the rich fluid dynamic phenomenology. To explore the physics of complex fluid emulsions we employ high-resolution and high-statistics simulations, via optimised computational codes based on the multicomponent LBM.

We implemented a largely-scalable and highly-optimised LBM based code to study high-resolution stabilised emulsions on a 3-dimensional domain. In particular the objective is to explore the physics of complex fluid emulsions using disjoining pressure at the surface between the two components [6]: from their production, via turbulent stirring, to their (statistical) behaviour under flowing, as well as, under resting jammed conditions (figure 1). We are interested in collecting a large statistics aimed at providing a detailed analysis of the emulsion morphology (e.g. droplet size distribution) of the various investigated systems by varying the turbulent forcing, the resolution, as well as the volume fraction of the two components.

LBM, recently emerged as a popular numerical method for computational fluid dynamics applications, aims to solve a discretized version of the Boltzmann Equation. Due to its ability to fulfil both symmetry and conservation constraints needed to reproduce hydrodynamics in the continuum limit, LB could be viewed as a kinetic solver for the Navier-Stokes equations. From an algorithmic point of view, the method is based on a splitting approach: a free flow streaming step (`propagate`) is followed by a collision step (`collide`). During the streaming step, the discrete probability distribution functions (frequently named as populations), arranged to form a lattice at each grid site, propagate along a predetermined number of molecular trajectories and with an assigned speed so that at each time step they can only move to next-nearest neighbor grid sites. The collision step, instead, features a relaxation process towards equilibrium determined according to the local flow properties. Such splitting approach guarantees clear advantages in practical implementations as a relevant number of parallelisms can be identified. This



(a) 256^3 (b) 512^3 (c) 1024^3

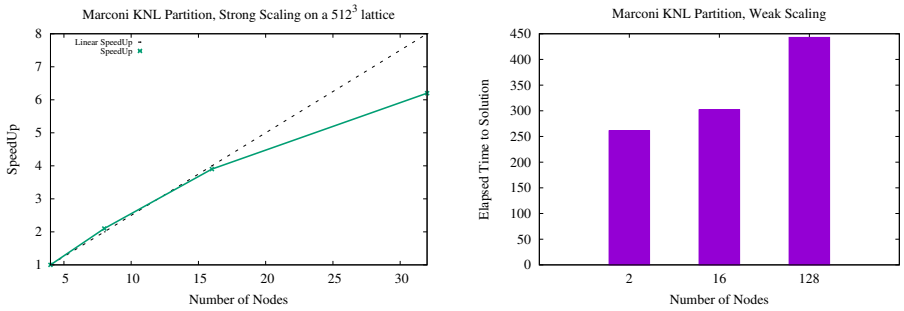
(d) 256^3 (e) 512^3 (f) 1024^3

Figure 1. Via turbulence stirring we could produce emulsions at different numerical resolutions, from top to bottom, 256^3, 512^3 and 1024^3. As it can be observed we obtain stretched droplets when turbulence is turned on (a-c) and jammed compositions of spherical droplets (d-f), with a volume fraction of approximately 40% of the droplets phase, in absence of turbulence.

makes LBM an ideal tool to investigate performances of modern high-performance computing systems [7,8,9]. In particular we use these two kernels to measure peak performance and memory bandwidth. Indeed, the `propagate` kernel is characterised by a data movement in memory with no floating point operations while the `collide` kernel includes high number of floating point operations. By counting the floating points operation used to implement the (`collide`) kernel we can estimate the peak performance at runtime. However, the measurement presented in this paper is only a reference for a generic LBM based application as the number of floating point operations implemented within the `collide` kernel changes accordingly to the physics of the numerical model. We adopt the D3Q19 LBM stencil, a 3-dimensional model with a set of 19 population elements corresponding to (pseudo)-particles moving one lattice point away along all 19 directions. Therefore, due to the complexity of the model and the number of simulations we plan to investigate the physical (statistical) behaviour of the system by varying the initial configuration on a parameter space as well as the resolutions, we are strongly motivated to optimise and analyse in details the performance of the main kernel of our 3d-dimensional implementation of the LBM. Indeed, this requires a large amount of computational resources, especially considering the two lattices needed to describe multicomponent emulsions.

4. Code Optimisations

Legacy data structure such as Array of Structures (AoS) or Structure of Arrays (SoA) are commonly used to implement stencil based applications, including LBM. In [2] we have described two additional data structure that can be used for LBM based application but delivering better performances if compared with canonical AoS or SoA data layouts. The two new layouts called CSoA and CAoSoA are seen as an extension of the SoA where VL lattice-site data at distance L/VL (L dimension of major order store) are clustered in consecutive elements for each population array, with VL equal to the number of double precision elements that can be stored in the vector register available on the given architecture. The newly designed layouts for high-performance LBM based codes allow to store data properly aligned in memory and aiding the compiler in the process of auto-vectorization of the steps required to compute the LBM main loop. The CAoSoA structure is a mix between CSoA and AoS, and allows to exploit the benefit of the VL clusterization of lattice sites element as introduced by the CSoA schema but with the benefit of higher locality in regards to the populations. In [2] we have also described the optimisation implemented in our LBM code by fusing the `collide` and `propagate` kernels, avoiding to store intermediate hydrodynamic quantities when not needed. In the following analysis we refer with CF to the classical schema implemented in high-performance LBM based applications where the `propagate` and the `collide` kernel are separated and the density stored on a separate structure, and with FF the fully fused version where the two kernels are compact and the density only temporary computed.



(a) Strong Scaling (b) Weak Scaling

Figure 2. Strong and weak scaling of the LBE3D code using the CSoA data layout on the KNL partition of Marconi. The strong scaling chart shows the scaling achieved on a 512^3 dataset by increasing the number of nodes with a reference initial point of 4 KNL nodes. The weak scaling chart reports how the code scales in function on the number of node by fixing the amount of work per process. In this case a 256^3 lattice is for the benchmark on 2 nodes, a 512^3 lattice on 16 nodes and a 1024^3 lattice on 128 nodes

5. Performance Measurements

In [2] we have been reporting performance measurements mainly related to the KNL when configured in flat mode and analysing the scaling as function of the number of thread used per SMT core. As that work demonstrated how the best configuration was

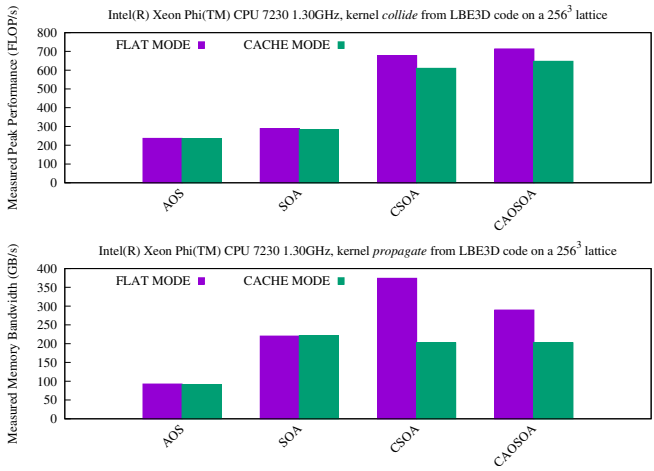


Figure 3. For the KNL processor configured in both FLAT mode and CACHE mode, we report in the upper chart the measured peak performance for the `collide` kernel while in the bottom chart the measured memory bandwidth for the `propagate` kernel. The performance measurement is reported using a single MPI process and 64-threads, one per each SMT KNL core. The performance measurement is performed on a lattice size of 256^3 which represent a real workload but at the same time fits the 16 GB capacity of the MCDRAM available on a single KNL socket.

achieved when using a single thread per SMT core, in this new analysis we only refer to configuration where the used number of thread is equivalent to the number of available cores on socket. Only a brief reference to the cache mode configuration was reported to show the performance impact when considering larger data-set that would not fit the 16 GB capacity of the MCDRAM memory. In other words, we were considering the KNL configured in cache mode only as the unique alternative when having to deal with dataset larger than 16 GB. However, as the Marconi KNL partition is configured in cache mode and we were allocated a significant amount of computer resources on the system for the PRACE project production we analysed performances also when using smaller data structures. Indeed, the LBE3D code demonstrated to efficiently scale on a large number of nodes and while scaling up we can reduce the amount of data per single compute node up to the point that we can always fit the 16 GB capacity of the MCDRAM memory, figure 2.

The first analysis was performed to measure the performance of the memory bandwidth using the `propagate` kernel in both configurations while using the same data-set and varying the presented data layouts. In figure 3 the result are reported. It is interesting to see how, only when using highly optimised data layouts such as CSoA and CAoSoA, there is a significant boost in terms of memory bandwidth moving from the Cache mode to the Flat mode. In particular for the CSoA data layout we could achieve a 50% of performance improvement in memory bandwidth for the same data set and the same code, simply switching the KNL configuration. Much lower impact is instead measured if considering the peak performance (FLOP/s) as reported for the `collide` kernel in figure 3. In this case we still register a significant boost of the highly-optimised data layouts if compared with the more common layouts but there is not the same performance lost when comparing the KNL processor configured in Flat or in Cache mode. In this case

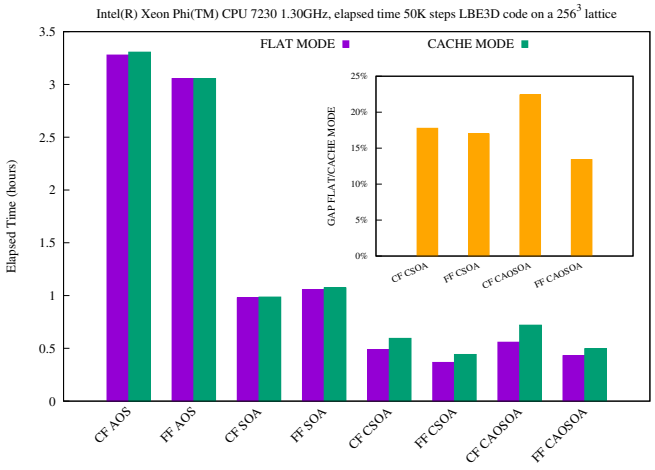


Figure 4. Measured time to solution for the LB main loop for all considered data layout in the fused fashion. Data are reported for KNL configured in cache mode as well as in flat mode. The inset reports the ratio between Flat and Cache mode of both CF and FF versions of the code for the optimised data layouts. The same setting and lattice size as for figure 3 is used also here.

only a performance benefit of about 10% is registered if comparing the two configurations.

Other than comparing the single kernels of `propagate` and `collide` we have also compared the performance gap between two configuration while considering the entire LB loop which includes boundary exchange and other minor kernels, figure 4. Again the results confirm the trend reported by measuring the single kernels. First it is possible to notice how the common layout AoS and SoA do not show any performance difference between the two KNL configurations. On the other hand, as reported in the inset of figure 4 the highly-optimized layouts provide a performance benefit between 15% to 23% when considering the KNL configured in Flat mode rather then in Cache mode.

6. Energy Efficiency

We now consider energy efficiency for the LBE3D across the multiple data layouts presented in both the CF and the FF version of the code. Again we want to analyse the energy consumption in both Flat and Cache mode using a data-set that fits the on-chip memory capacity and try to understand what is the impact of the DRAM memory utilisation. We use data from the Running Average Power Limit (RAPL) register counters available in the KNL read through the custom library developed in [10]. In Figure 5, we show the measured values of energy consumption (million Joule) for the LBE3D application, respectively, for the processor and for the off-chip DRAM memory. The DRAM energy consumption is generally lower as the KNL is configured in Flat mode and during the simulation data are all stored into the MCDRAM. Indeed, energy measurements for the DRAM memory takes into account just the idle consumption, while the MCDRAM energy consumption is accounted in the CPU Package value, since MCDRAM is an on-package memory, such as caches.

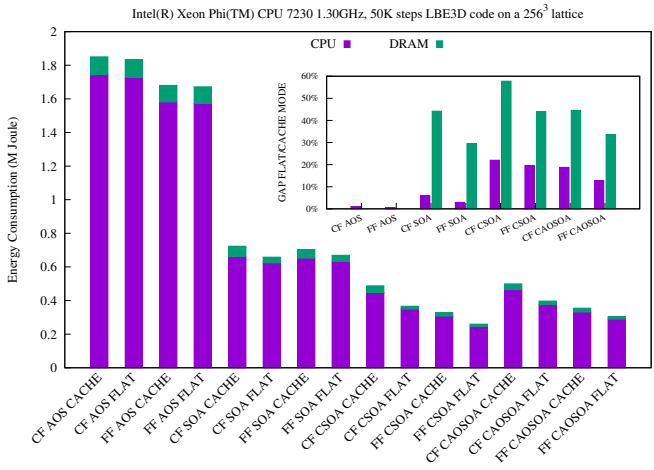


Figure 5. Measure of energy consumption for the LB main loop for all considered data layout in the fused fashion. Data are reported for KNL configured in cache mode as wall as in flat mode. The inset reports the ration between the fused versions in term of both CPU and DRAM. The same setting and lattice size as for figure 3 is used also here.

It is relevant to notice that, despite the CSoA and CAoSoA data layouts are expected to stress the CPU system more than the AoS and SoA (higher utilisation of the VPU), we can assume that the absorbed power remains approximately constant when considering different data layouts. Indeed, it is evident how the energy consumption remains mostly proportional to the time to solution, figure 4. On the other end it is impressive to see how we measure a much higher energy consumption as far as the DRAM is concerned. Our analysis reports that when using the highly optimized data layouts on the KNL configured in Cache mode, more energy is required due to DRAM if compared to the Flat mode.

7. Conclusions

Our LBM based application implements highly optimised data structures capable to deliver high-performance on modern many-cores processors systems such as the Intel KNL processor. In this work we have demonstrated that the introduced data layouts CSoA and CAoSoA are also extremely efficient in exploiting memory on-chip such as the MCDRAM. Indeed, while canonical structure such as AoS and SoA do not show any benefit when switching from Flat mode to Cache mode, the clustered data structure are much more efficient in Flat mode than in Cache mode. Highly scalable optimised code based on LBM can enable computer simulations at the frontier of science by reducing amount of memory required per core up to the capacity of the on-chip memory and excluding completely the main memory (DRAM). While this effort would be useless if considering common data layouts for stencil codes, our analysis shows that, when the clustered data layouts are used, a benefit between 15% to 25% is achieved by excluding the main memory DRAM. Moreover, results have confirmed the efficiency of highly-optimized data layouts also in term of energy consumption. The unnecessary access to the DRAM, in case the size of the data of the simulation fits the on-chip memory, translates in a significant overhead in energy consumption when considering the clustered data layouts.

8. Acknowledgements

We would like to thank PRACE for the granted project (ID 2018184340) "TurEmu - The physics of (turbulent) emulsions" along with CINECA, INFN and The University of Ferrara for access to their HPC systems.

References

[1] S. Succi, *The Lattice Boltzmann Equation: For Fluid Dynamics and Beyond*. Clarendon Press, 2001, ISBN: 978-0-19-850398-9.

[2] I. Girotto, S. F. Schifano, E. Calore, G. D. Staso, and F. Toschi, "Performance Optimization of D3Q19 Lattice Boltzmann Kernels on Intel KNL," in *INFOCOMP 2018: The Eighth International Conference on Advanced Communications and Computation*, 2018, pp. 31–36.

[3] E. Calore, N. Demo, S. F. Schifano, and R. Tripiccione, "Experience on Vectorizing Lattice Boltzmann Kernels for Multi- and Many-Core Architectures," in *Parallel Processing and Applied Mathematics: 11th International Conference, PPAM 2015, Krakow, Poland, September 6-9, 2015. Revised Selected Papers, Part I*, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 53–62. doi: 10.1007/978-3-319-32149-3_6

[4] E. Calore, A. Gabbana, S. F. Schifano, and R. Tripiccione, "Early experience on using knights landing processors for lattice boltzmann applications," in *Parallel Processing and Applied Mathematics: 12th International Conference, PPAM 2017, Lublin, Poland, September 10-13, 2017*, ser. Lecture Notes in Computer Science, vol. 1077, 2018, pp. 1–12. doi: 10.1007/978-3-319-78024-5_45

[5] E. Calore, A. Gabbana, J. Kraus, S. F. Schifano, and R. Tripiccione, "Performance and portability of accelerated lattice Boltzmann applications with OpenACC," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 12, pp. 3485–3502, 2016. doi: 10.1002/cpe.3862

[6] M. Sbragaglia, R. Benzi, M. Bernaschi, and S. Succi, "The emergence of supramolecular forces from lattice kinetic models of non-ideal fluids: applications to the rheology of soft glassy materials," *Soft Matter*, vol. 8, pp. 10 773–10 782, 2012. doi: 10.1039/C2SM26167G

[7] S. Williams, J. Carter, L. Oliker, J. Shalf, and K. Yelick, "Optimization of a Lattice Boltzmann computation on stateof-the-art multicore platforms," *Journal of Parallel and Distributed Computing*, vol. 69, pp. 762–777, 2009. doi: 10.1016/j.jpdc.2009.04.002

[8] S. Williams, J. Carter, L. Oliker, J. Shalf, and K. A. Yelick, "Lattice Boltzmann simulation optimization on leading multicore platforms," *2008 IEEE International Symposium on Parallel and Distributed Processing*, pp. 1–14, 2008. doi: 10.1109/IPDPS.2008.4536295

[9] M. Bernaschi, M. Fatica, S. Melchionna, S. Succi, and E. Kaxiras, "A flexible high-performance lattice Boltzmann gpu code for the simulations of fluid flows in complex geometries," *Concurrency Computat.: Pract. Exper.*, pp. 22: 1–14, 2009. doi: 10.1002/cpe.1466

[10] E. Calore, A. Gabbana, S. F. Schifano, and R. Tripiccione, "Evaluation of dvfs techniques on modern hpc processors and accelerators for energy-aware applications," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 12, pp. 1–19, 2017. doi: 10.1002/cpe.4143