



Article

# Performance and Energy Assessment of a Lattice Boltzmann Method Based Application on the Skylake Processor <sup>†</sup>

Ivan Girotto <sup>1,2,3,\*</sup>, Sebastiano Fabio Schifano <sup>4,5</sup> , Enrico Calore <sup>5</sup> , Gianluca Di Staso <sup>2</sup> and Federico Toschi <sup>2</sup>

<sup>1</sup> The Abdus Salam, International Centre for Theoretical Physics, 34151 Trieste, Italy

<sup>2</sup> Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands; g.di.staso@tue.nl (G.D.S.); F.Toschi@tue.nl (F.T.)

<sup>3</sup> University of Modena and Reggio Emilia, 41125 Modena, Italy

<sup>4</sup> University of Ferrara, 44122 Ferrara, Italy; schifano@fe.infn.it

<sup>5</sup> INFN Ferrara, 44122 Ferrara, Italy; enrico.calore@fe.infn.it

\* Correspondence: igirotto@ictp.it

<sup>†</sup> This Paper Is an Extended Version of Our Paper Published in the Proceedings of the ParCo2019: Mini-Symposium on Energy-Efficient Computing on Parallel Architectures, Prague, Czech Republic, 10–13 September 2019.

‡ Current Address: International Centre for Theoretical Physics, Strada Costiera, 11-34151 Trieste, Italy.

Received: 6 March 2020; Accepted: 29 April 2020; Published: 8 May 2020



**Abstract:** This paper presents the performance analysis for both the computing performance and the energy efficiency of a Lattice Boltzmann Method (LBM) based application, used to simulate three-dimensional multicomponent turbulent systems on massively parallel architectures for high-performance computing. Extending results reported in previous works, the analysis is meant to demonstrate the impact of using optimized data layouts designed for LBM based applications on high-end computer platforms. A particular focus is given to the Intel Skylake processor and to compare the target architecture with other models of the Intel processor family. We introduce the main motivations of the presented work as well as the relevance of its scientific application. We analyse the measured performances of the implemented data layouts on the Skylake processor while scaling the number of threads per socket. We compare the results obtained on several CPU generations of the Intel processor family and we make an analysis of energy efficiency on the Skylake processor compared with the Intel Xeon Phi processor, finally adding our interpretation of the presented results.

**Keywords:** lattice boltzmann method; KNL; skylake; flat mode; energy efficiency; computational performances; intel processor family; bandwidth; flops

---

## 1. Introduction and Related Works

Stabilized multi-component emulsions, commonly found in many foods and cosmetic products, display a fascinating and rich phenomenology. How do such complex fluids flow? Despite considerable attention in the literature, many fundamental questions remain to be answered. Numerical simulations have been demonstrated to be effective methods to study those physical systems at high-resolution with a level of details that current experiments cannot achieve. Indeed, we have been granted about 100 millions core-hours combined on the Marconi-A2 partition, hosted at CINECA, and on the MareNostrum, hosted at the Barcelona Supercomputing Centre (BSC), to explore the dynamics of stabilized emulsions via mesoscale 3d numerical simulations (at various resolutions). To successfully complete the project we implemented a LBM based application introduced in Section 2, named LBE3D,

which includes optimized data layouts in order to achieve high efficiency. Both Marconi-A2 and MareNostrum are European Tier-0 systems for high-performance computing, equipped with different architectures of the Intel processor family. Marconi-A2 was equipped (out of production in early 2020) with the latest Knights Landing (KNL) processor while MareNostrum is equipped with the Skylake processor (SKL). The compute nodes of both the systems are interconnected with the same network (Intel OmniPath), even if not relevant for this work where we mainly focus on the performances measured on the single CPU sockets.

This paper extends a series of previous works from the authors on the LBE3D applications, by adding the analysis for the SKL processor and the comparison with other processors of the Intel processors family. In [1], we presented the analysis of the computational performance and energy efficiency of the LBE3D for the KNL processor. In [2], we focused on comparing the performance of LBE3D if configuring the KNL processor in either Flat and Cache mode. In [3,4], a similar analysis was presented for a two-dimensional lattice, using the D2Q37 Lattice Boltzmann scheme, on both CPU processors and NVIDIA GPUs, while in [5] an analysis of an OpenACC implementation of the Lattice Boltzmann main kernels was discussed. In regards to the analysis of energy efficiency, we aim to show the impact on energy consumption of the optimized data layouts by measuring at runtime the energy consumed by the LBE3D and comparing single socket performances of both the SKL and the KNL processors.

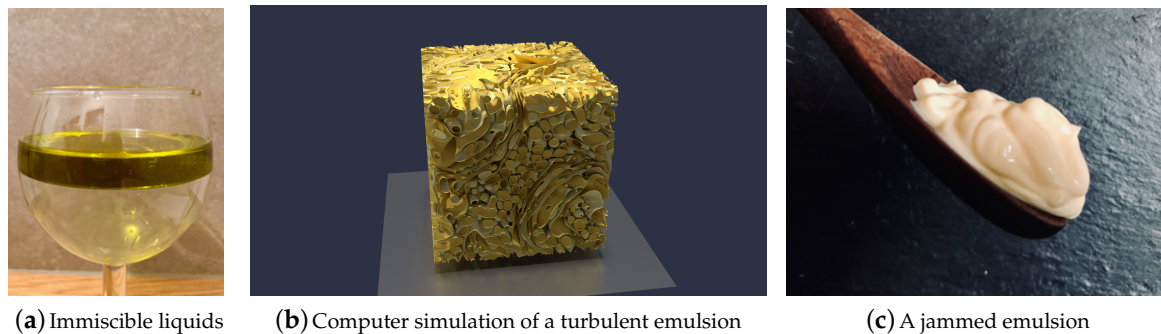
A relevant part of this work is also to show the impact in term of energy efficiency, when using highly optimized data layouts on LBM based applications with particular focus to three-dimensional systems. Indeed, energy consumption is relevant in computer engineering research to find the roadmap for a sustainable development of the next generation of large scale computing platforms. Number of tools have been developed to tackle the problem at several levels, from the system level [6], to the resource manager [7] and the runtime execution such as: triggering the clock frequency at runtime accordingly to the specific case (i.e., lowering the frequency if memory bound) [8] and performing fine-grained power measurements that enable energy-aware performance optimizations [9]. However, a significant improvement in term of energy efficiency is still much related to algorithms and software engineering. We focus on this part by measuring the improvement when using highly optimized data layouts in term of energy consumption (Joule) as well as in term of the average power absorbed (Watt) by the application. There are several metrics used to quantify energy efficiency of software applications for high-performance computing [10,11] offering a wide spectrum of options for a quantitative analysis. In this work, we consider more common quantities such as average power drain, time-to-solution and energy-to-solution [12].

The rest of the paper is structured as follows. In Section 2, we introduce the context of this work and the LBM. In Section 3 we provide an overview of the utilized processor architectures. In Section 4, we describe the implemented data layouts and the analyzed implementations. In Section 5, we present the results about the computing performance and the energy efficiency. Finally, we conclude with our interpretation of the results. Being an extension of [1,2], for completeness, Sections 2 and 4 are only partially expanded with respect to previous works.

## 2. The Lattice Boltzmann Method

The recently emerged LBM [13] has rapidly become a popular method for computational fluid-dynamics applications and it is currently widely used to study the behaviour of fluid flows. Due to its ability to fulfil both symmetry and conservation constraints needed to reproduce hydrodynamics in the continuum limit, the LB could be viewed as a kinetic solver for the Navier–Stokes equations. It aims at solving a discretized version of the Boltzmann Equation, describing the probability for (pseudo)-particles to move along given directions, with an assigned speed, so that at each time step they can move to nearest neighbour grid sites. In the LBE3D we implement a three-dimensional scheme, named D3Q19, where 19 predetermined molecular trajectories (populations) are arranged to form a lattice at each grid site. In particular, we also use a pseudo-potential model to be able to include

attractive and repulsive forces [14] needed to introduce the surface tension and the disjoining pressure between the two-component fluid [15]. We implemented those numerical models to explore the physics of complex fluid emulsions: from an initial condition (Figure 1a) where the two components are separated, to the phase of turbulent stirring where the emulsion is produced (Figure 1b) up to a final state achieved when turbulence is turned off and the resulting emulsion is in a jammed state (Figure 1c).



**Figure 1.** From two simple (Newtonian) fluids (a), to one complex (non-Newtonian) fluid (c). The production process of mayonnaise proceeds at home, as well as in food industry, via stirring of oil and water with addition of surfactants, in order to confer stability to the emulsion. With highly-optimized computer models, as the ones implemented in the LBE3D, we can study how to go from picture (a) to picture (c) using computer simulation, as illustrated from the screenshot in picture (b).

The project is composed of two phases: first a validation part and, second, a number of simulations are performed to collect large statistics for a detailed analysis of the emulsion morphology (e.g., droplets volume and surface distribution) by spanning a large parameter space to investigate the physical (statistical) behaviour of several systems. The main investigated parameters are the volume ratio between the two components, the amplitude of the turbulent forcing, used to mix the immiscible fluids, and the resolution. Because of the complexity of the model and the number of systems we aim to study, those computational fluid-dynamics simulations require a huge amount of computing resources, especially considering that the model evolves two lattices needed to describe the two component emulsions. To comply with such requirements and succeed in the goal of simulating stabilised emulsions on a three-dimensional domain on massively parallel infrastructure for high-performance computing, a largely-scalable LBM code was implemented and optimized, including a detailed analysis of the performance of the main kernels of our three-dimensional implementation of the LBM.

Computationally the LBM method is composed by two main steps: a free streaming step, where the populations move along a predetermined number of molecular trajectories, and a collision step, featuring a relaxation towards equilibrium and determined by the local flow properties. We indicate those two steps (implemented in two different kernels) as `propagate` and `collide`, respectively. The nature of those computationally relevant components of the LBM approach guarantees clear advantages in practical implementations as a relevant number of parallelisms can be identified, making LBM also an ideal tool to investigate performance bottlenecks of modern high-performance computing systems [16–18]. We use these two kernels to measure the peak performance and memory bandwidth, as the `propagate` kernel is characterised by memory data movement with no floating point operations, while the `collide` kernel includes a large number of floating point operations, the `collide` kernel in LBE3D is used to estimate the peak performance at runtime. However, the number of floating point operations implemented within the `collide` kernel changes accordingly to the physics of the numerical model and, therefore, the measurement presented in this paper is only a reference for LBM based application at large.

### 3. Computer Architectures

We briefly describe the CPU processors used to produce the results presented in Section 5, with particular focus on the SKL and on the KNL processor. A summary of all the architectures is given in Table 1 to facilitate the reading. There, the column “Short name” indicates how we will refer to a specific processor model for the rest of the paper, avoiding to mention the entire processor name and model repeatedly.

**Table 1.** The table reports a summary of all computer architectures of the Intel Processors Family mentioned in this paper. In all cases the version 2018 of the Intel Compiler has been used to compile the LBE3D application but for the ICTP cluster where the 2017 version is available.

Host	Proc. Name	Proc. Model	#Cores	Compiler Flags	Short Name	VL
ICTP	Westmere	E5620 @ 2.40GHz	4	-sse4	WEP	1
ICTP	Ivy Bridge	E5-2680 v2 @ 2.80GHz	10	-xavx	IB	2
CINECA	Broadwell	E5-2697 v4 @ 2.30GHz	18	-xavx2	BWD	4
CINECA	Knights Landing	Phi7250 @ 1.40GHz	68	-xMIC-AVX512	KNL	8
INFN	Skylake	Gold 6130 @ 2.10GHz	16	-xCORE-AVX512	SKL-G	8
INFN	Knights Landing	Phi7230 @ 1.30GHz	64	-xMIC-AVX512	KNL	8
BSC	Skylake	Platinum 8160 @ 2.10GHz	24	-xCORE-AVX512	SKL-P	8

The SKL is a member of the last generation of the 14nm Intel Xeon Processor Scalable Family. It is available on market in various versions categorized as Bronze, Silver, Gold and Platinum, on the base of different architectural aspects. Indeed, the metal indicates a growing computational capability from low performance consumer hardware to processors for high-performance computing. We are interested in measuring the performance of the LBE3D on the SKL-P processor as we are committed to a large number of simulations on MareNostrum at BSC. However, for the analysis on the energy consumption of the LBE3D on the SKL processor we refer to the SKL-G processor available in our lab, at the University of Ferrara, because the measurements require root privileges. We also highlight performances of both the SKL-P and the SKL-G processors as some interesting results came up while looking at the performance numbers obtained on the two architectures. The processors have comparable characteristics in terms of memory bandwidth being both equipped with six Double Data Rate fourth-generation (DDR4) channels of synchronous Dynamic Random-Access Memory (DRAM), but delivering a significant difference in peak performance since the SKL-P hosts 24-cores while the SKL-G version comes with 16-cores.

From the x86 based many-cores series of the Intel processor family, the KNL is a processor model designed for high-end computing. It is equipped with six DDR4 channels of DRAM, with a nominal bandwidth of 115.2 GB/s and 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. The processor can be configured in three different modes: Flat, Cache and Hybrid mode. Each mode is characterised by how the program execution accesses to the MCDRAM at runtime as, to achieve high-performance, it is fundamental to exploit the MCDRAM. 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. In a previous work [2] we analyzed in detail how the LBE3D, being a massively scalable code, can well exploit the MCDRAM memory if keeping the amount of memory per node around the 16GB while scaling up the number of nodes for large problems. For the analysis presented in this paper we focus on the KNL configured in Flat mode because it maximizes the usage of the MCDRAM at runtime. The Marconi-A2 partition (recently out of production) was based on the KNL processor 7250. However, for the analysis we refer to the 64-cores KNL processor 7230 available in our lab, because of the same reason we mentioned above for the SKL processor.

The Intel library implementing the Message Passing Interface (MPI) was used together with the Intel compiler to compile and build the LBE3D application which requires the MPI even if running

on single process. The Table 1 reports the Intel compiler version used to build the application on the various system. The same level of optimization (-O3) was also used in all cases on top of the specified flags for auto-vectorization. The “VL” column of the table shows the vector length (VL) used to compile the LBE3D, accordingly to the instruction set supported by a given CPU model, as it is described in the following Section. The LBE3D is a highly scalable application that implements a distributed memory approach for inter-node communication while multi-threading via OpenMP is implemented to exploit the intra-node compute capability by scaling with the number of threads. The threads affinity is set at run time with the Intel environment variables “KMP\_AFFINITY=compact” and “I\_MPI\_PIN\_DOMAIN=socket”. All presented results are related to a single socket execution therefore using a single MPI processes and a given number of threads accordingly to the number of cores available. The same approach was used for all benchmarks.

#### 4. Code Optimization for LBM Based Applications

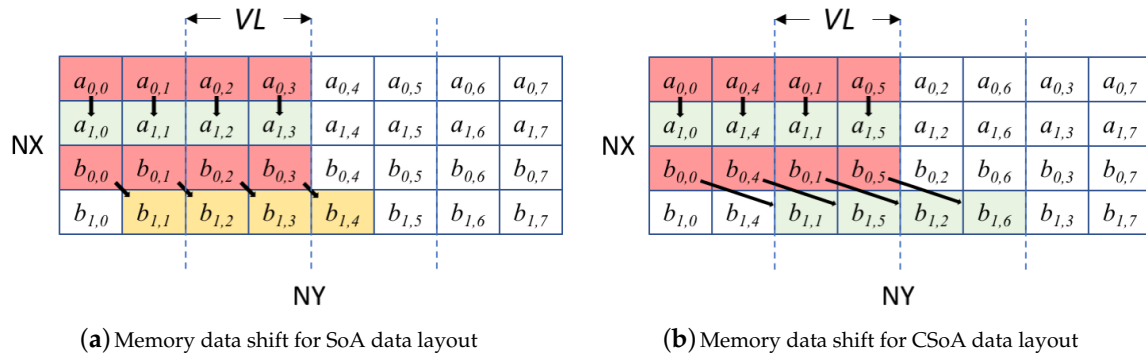
In this section, we describe the most relevant optimization steps we have applied in our code. The main options for data-structure used in many stencil-based codes, and also in LBM, are Array of Structures (AoS) and Structure of Arrays (SoA). In the AoS layout, the population data of each lattice site is stored in memory in consecutive locations, and then each lattice site is stored one after the other. Conversely, in the SoA scheme, populations with the same index of all sites are stored at consecutive addresses as an array, and then arrays of different population are stored one after the other. The first scheme in general optimizes cache accesses and data re-use, while the latter fits better data-parallelism processing appropriate for latest processors based on large vector SIMD data-paths.

However, applications are usually composed of many kernels, and each may have different computing requirements. This is for example the case of LBM, where the propagate kernel performs only memory operations, while the collide kernel is more compute intensive.

In [19], we have defined two additional data structures, named respectively CSoA and CAoSoA, to match the different computing requirements of several parts of LBM codes. The CSoA is a direct extension of SoA, where each population array is composed of clusters of VL data elements at distance  $L/VL$ , being  $L$  the size of the lattice, and  $VL$  the size of SIMD registers of CPUs. Cluster are then stored in memory as vectors at properly aligned addresses, and then processed using SIMD operations to exploit vectorization. A graphical representation of the improvement given by the CSoA structure, in regards to the SoA, is shown in Figure 2. In particular, Figure 2a represents a two-dimensional lattice ( $2 \times 8$ ) of two populations stored in row-major order, using the SoA layout and Figure 2b the CSoA layout, respectively. For simplicity the pictures show the data shift in place, however in the case of LBE3D the propagate kernel works on two lattices of equal dimensions. In the SoA classical scheme, site elements of different populations are contiguously placed in memory and all data elements of a single population move in the same direction (as described by a given LB scheme). In Figure 2a we represent the data shift of the two populations (a and b), with the population (a) moving along the x-coordinate and (b) moving along both coordinates. Supposing each lattice element is stored in double precision and the memory alignment as well as vector registers length being 128bits, when using SoA every data stream along the most nested coordinate (NY, in the described case) is misaligned in memory. In fact, when the data move along the x-direction both the operation of reading and writing are aligned in memory and can be vectorized while, if considering the case of the population (b), only the reading is aligned in memory but the operation of writing would result misaligned. On the other hand, in the case of using the CSoA data layout, the data shift in any direction consists in moving clusters of VL data being aligned in memory where a vectorized data shift can be always applied. Therefore, the CSoA effective improvement with respect to the SoA layout depends by the utilized LB scheme as each scheme describes a different number of shifts along the most nested direction. In the case of the LBE3D, implementing the D3Q19 scheme, only 9 of the 19 directions stream along the z-direction and therefore the improvement is only related to 50% of the data shift happening on a single LB loop (population 0 does not move). In the other direction, the CAoSoA



layout mixes the vectorization features of the CSoA with the locality of AoS. In practice, it stores at contiguous memory locations a cluster of  $VL$  populations of same index of different sites, and then all clusters of population for that  $VL$  sites are stored in memory one after the other. The rationale behind this scheme is that each block can be moved and processed using the SIMD instructions of the processor, keeping cluster corresponding to different populations indexes relatively close in memory to enhance localities of memory accesses. This choice potentially retains the benefits of the AoS and CSoA layouts, and therefore provides a compromise between the requirements of both propagate and collide kernels.



**Figure 2.** The figure shows the data shift happening for the propagate kernel on a  $2 \times 8$  two-dimensional lattice with two populations (a and b), using the SoA (a) and the CSoA (b) schemas, respectively. Data are considered stored in row-major order and the indexes indicating the site position in the two-dimensional lattice. In the picture the arrows describe the direction on which each given population moves. Data elements representing a specific data movement have been coloured: red for elements with correct memory alignment for the operation of reading, green for elements with a correct memory alignment for the operation of writing and finally, yellow for elements with a memory misalignment for the operation of writing. While in (a) only 50% of the operations of writing are aligned in memory, in (b) 100% of the copies of element required by the propagate kernel will result aligned in memory.

The Equation (1) generically describes the Boltzmann equation in a form where it is simple to identify the two operators of propagation and collision: left and right of the equal, respectively.

$$f_i(X + c_i \delta_t, t + \delta_t) - f_i(X, t) = -\frac{1}{\tau} [f_i(X, t) - f_i^{eq}(X, t)] \tag{1}$$

A naive implementation of the equation would require to apply first the propagate kernel, then all discretized hydrodynamic quantities needed for computing the equilibrium function  $f^{eq}$  and finally the collision operator. To improve this approach we have implemented two separate versions of the code where the collide and propagate kernels have then been fused in a single step, as illustrated in the Algorithm 1.

In the following, we refer with CF to the classical scheme implemented in LBM applications, where the propagate and collide kernels are kept separated, and the density is stored on a separate structure, and with FF the fully fused version where the two kernels are joined together and the density is computed as temporary when needed. As in the FF version all local quantities are stored locally, in vectors of  $VL$  dimension, the number of vectorized instructions within the FF kernel goes from 20% to 100% as we measured runtime using the Intel Vtune and described in [1].

**Algorithm 1:** Schematic description of the LBM main loop optimization

The LBM main loop implemented by simply coding the LB equation as shown in Equation (1)

```

for all time steps do
  < Set boundary conditions >
  for all lattice sites do
    < Propagate >
  end for
  for all lattice sites do
    < Hydrovar >
  end for
  for all lattice sites do
    < Equili >
  end for
  for all lattice sites do
    < Collis >
  end for
end for

```

The collisional operator (`collide`) is computed with only one iteration over the lattice sites, named CF version

```

for all time steps do
  < Set boundary conditions >
  for all lattice sites do
    < Propagate >
  end for
  for all lattice sites do
    < COLLIDE_FUSED >
  end for
end for

```

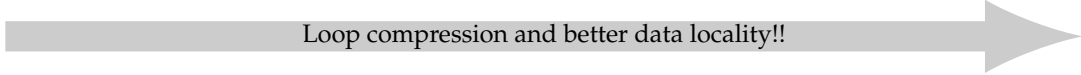
The `propagate` and the `collide` kernels are fused together. All intermediates quantities are only temporarily computed, named FF version

```

for all time steps do
  < Set boundary conditions >
  for all lattice sites do
    < FULLY_FUSED >
  end for
end for

```

Loop compression and better data locality!!

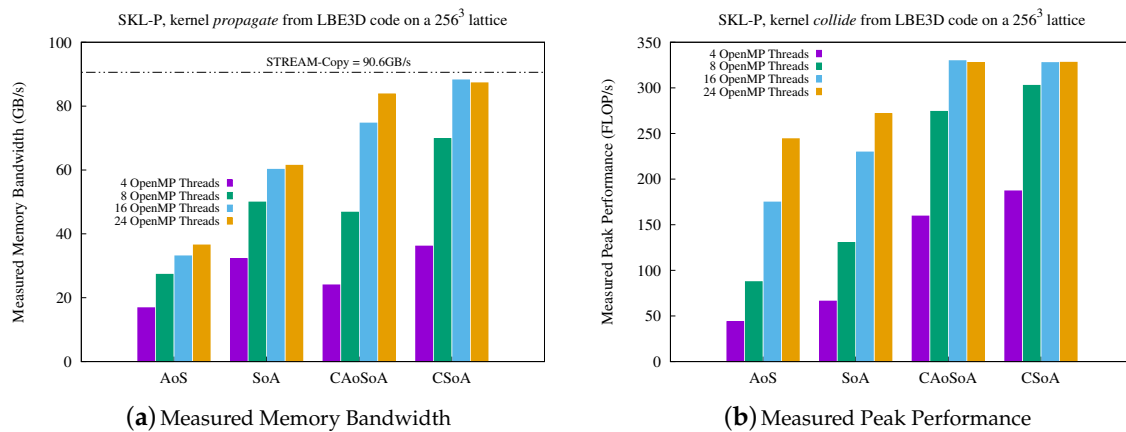


## 5. Results

In this Section we present the performance analysis of the LBE3D application on the SKL processor: first, the detailed performance of the kernels `propagate` and `collide` are presented for the SKL processor available at BSC; second, we present a comparison between the data obtained on two different SKL processors for comparing the market value in regards to the performance of the LBE3D (value for money) measured on those CPU systems; third we present the performance of the LBE3D application comparing several CPUs models of the Intel processor family; and finally we present an analysis of energy efficiency for the LBE3D application on the SKL processor also comparing with the result presented in [2] for the KNL processor.

### 5.1. Analysis of Performance

The performance analysis reported in Figure 3a shows the measured memory bandwidth of the `propagate` kernel, while scaling the number of threads within a single CPU socket. The obtained results confirms, as for the KNL processor [1,2], how the clustered data layouts CSoA and CASoA deliver good results in term of memory bandwidth on the SKL-P processor too. Peak of the measured memory bandwidth reaches almost 90GB/s which is about 70% of the nominal peak performance (128GB/s) and close to the results achieved using the STREAM-Copy benchmark. However, the thread scaling shows a saturation of the memory bandwidth already at 16 threads per socket despite the 24-cores available on socket. On the SKL-P processor the CSoA layout, implemented in the LBE3D, delivers more than 2 times higher performance if compared with the AoS implementation and it is 50% better if compared with the canonical SoA layout. The same performance saturation, Figure 3b, is reported for the `collide` kernel where we report the number of double precision floating-points operations executed in time (FLOP/s), despite as mentioned in Section 4 the `collide` kernel is expected to be more computing intensive. It is interesting to notice how good thread scaling is given within the socket by the AoS structure, which is overall less efficient if compared with other data layouts, confirming how scaling should not be considered a metric for analyzing high-performance computing applications unless coupled together with efficiency.



**Figure 3.** (a) measured memory bandwidth for the propagate kernel, (b) the measured peak performance for the collide kernel, for the SKL-P processor. The performance measurement is reported using a single MPI process on the socket and a number of threads equal to the maximum number of cores available on the given CPU model. Results are from a simulation with  $256^3$  lattice sites representing a real workload.

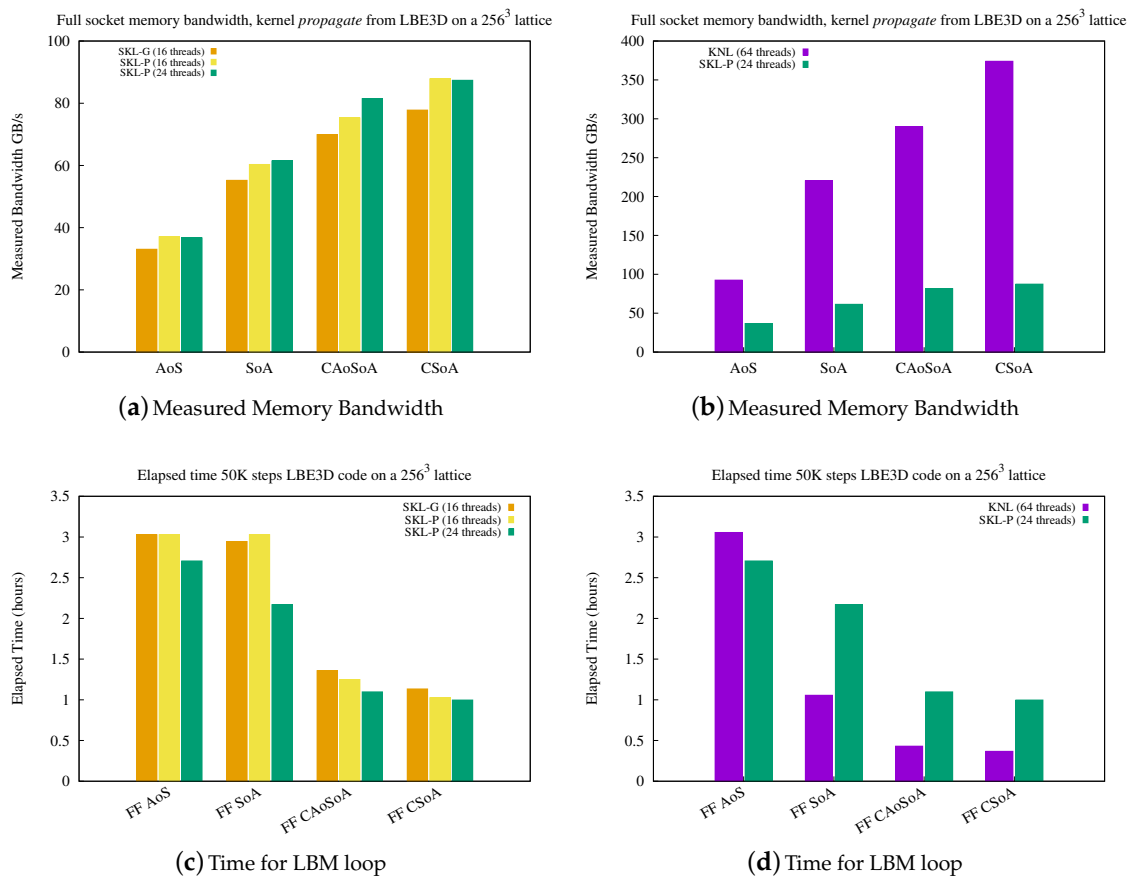
In Figure 4a we compare performances of the LBE3D on both the SKL-P and the SKL-G processors. The latest being expected to be a less performing processor as equipped with approximately 30% lower number of cores per socket if compared with the SKL-P. Our results show that if considering the most optimized version of the LBE3D, CSoA, there is only a really small performance gap between the two processors despite the SKL-P costing about 2.5 times more than the SKL-G. Indeed, the recommended market price for those two processors [20] is approximately €4700 for the SKL-P version and approximately €1900 for SKL-G. The relevance of the memory bandwidth limit of the LBE3D application on the SKL processor is also underlined in Figure 4b where we show that the performance gap in term of memory bandwidth between the two processors defines a similar performance trend when looking at the entire LBM main loop, in the most optimized version of the LBE3D (FF).

Figure 4 generally shows the impact of the memory bandwidth capability available on socket, on the LBE3D. Indeed, by looking at the comparison between the SKL-P and the KNL processors a factor of about 4 times in the performance per socket is measured, which is similar to the difference in term of memory bandwidth between the two systems, as for the KNL we consider the MCDRAM only (flat mode). Figure 4d reports the performance gap between the SKL and the KNL processor for the same simulation while considering the highly-optimized version (FF) of the LBE3D. For the KNL processor, installed in our lab, the recommended market price is approximately €1900 [20].

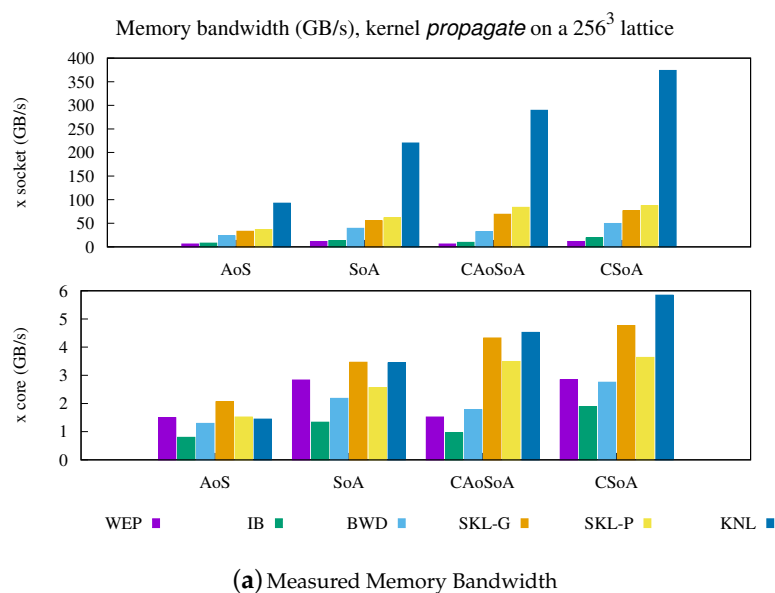
We have also analyzed the performance of the main kernels, propagate and collide, of the LBE3D application on various models of the Intel processor family. In particular we have analyzed the last four generations, from the Westmere to the Skylake, including the KNL many-core system. In Figure 5 we present the results obtained by the propagate kernel in 10 years of processors manufacturing by using one thread per core while filling all the cores available on the various sockets.

The analysis confirms how LBE3D is capable, when using the clustered data layout, to exploit the whole memory bandwidth available on the various systems, see Figure 5a. The memory per core would remain almost constant over the years, if considering the canonical layouts AoS and SoA, but the clustered structure show an increase of the memory bandwidth per core. On the other hand, Figure 5b presents the data for the collide kernel on several platforms, showing a higher value of FLOP/s per core for the SKL-G processor but again, showing how the KNL processor offers the best performances.



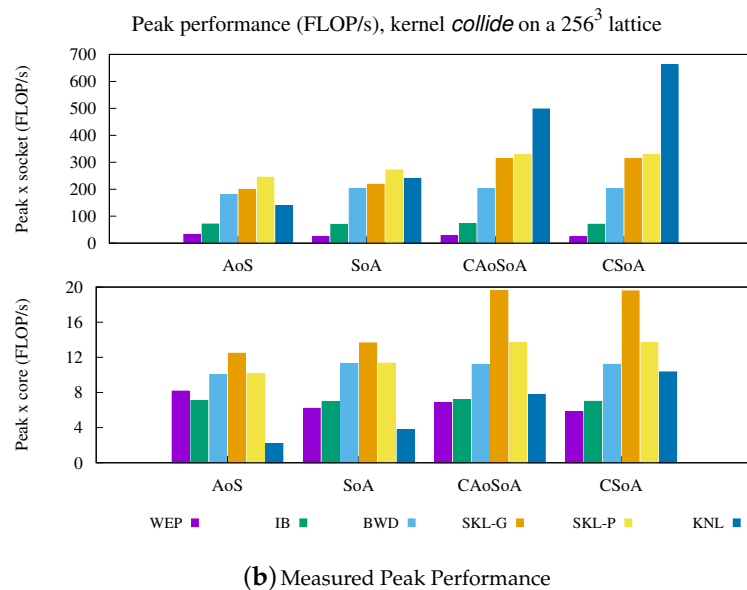


**Figure 4.** We report compute performance value of the optimized data layouts measured on both the SKL and the KNL processors. (a) reports the data related to the memory bandwidth measure with the propagate kernel on the SKL Gold 6230 and the SKL Platinum 8160. To emphasise the level of memory bandwidth saturation for the SKL Platinum 8160 we also report the numbers obtained using only 16 threads of the 24 available on socket. A comparison of the performance of memory bandwidth between the SKL and the KNL processor is reported in (b). (c,d) compare the performance, in terms of time to solution (full LB loop), of the three processors for the most optimized version of the LBE3D.



(a) Measured Memory Bandwidth

Figure 5. Cont.



**Figure 5.** We report in (a) the measured memory bandwidth for the propagate kernel while in (b) the measured peak performance for the collide kernel. In this case, we compare several models representing 10 years of CPU manufacturing from the Intel product family, including the KNL many-core system. The performance measurement is reported using a single MPI process on the socket while scaling the number of threads up to the maximum number of core available on the given CPU socket, and it is performed on a lattice size of  $256^3$  representing a real workload.

### 5.2. Analysis of Energy Efficiency

As for the KNL processor in [2], we measured the energy efficiency on the SKL processor, on a real workload and analyzing all the discussed data layouts implemented in the LBE3D code. In particular, we measure the energy consumption of the processor package and of the DRAM memory system, using the RAPL PACKAGE\_ENERGY and DRAM\_ENERGY hardware counters. The PAPI [21] library is considered an established practice for accessing the counters, as already validated in other studies [22,23]. On top of the PAPI library, we have developed a custom library [24] to manage power/energy data acquisition from hardware registers by simply literally adding in the LBE3D three lines of code for the initialization of the runtime environment, the start and the stop of the measurement.

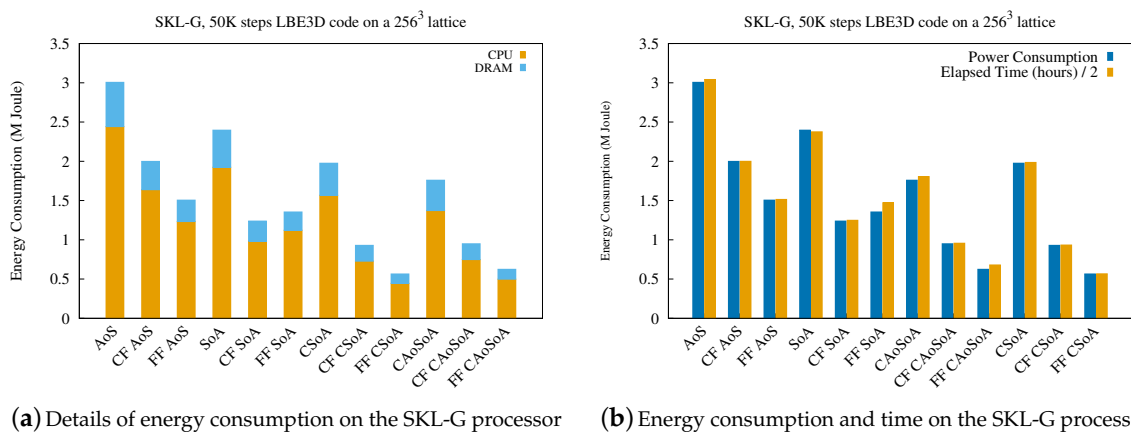
The library allows to read at runtime the register counters available in the SKL processor. In Figure 6a, we show in details the measured values of energy consumption (million Joule) for the LBE3D application, respectively, for the CPU and the DRAM memory. In Figure 6b, we report on the same scale the total (CPU and the DRAM memory together) energy consumption compared with the elapsed time of the simulation. 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. This is confirmed also by the data reported in Figure 7c where we compare the average power absorbed by the LBE3D in the FF version, comparing the SKL and the KNL processor. Therefore, the energy consumption of the LBE3D application on real workloads remains mostly proportional to the time to solution and, how reported in Figure 6b, there is almost a constant ratio of one million Joules of energy consumed every 2 h, if considering a single SKL-G socket.

Finally, we consider the energy efficiency when running the same simulation on the SKL-G and on the KNL processor, Figure 7. It is evident how the optimized clustered structures result to be from 2 times to 2.5 times more efficient, considering the two analyzed architectures singularly, if compared with the canonical SoA data layout, but even more if considering the AoS. It is also interesting to see how the KNL is the most efficient architecture for running LBM based application implementing the

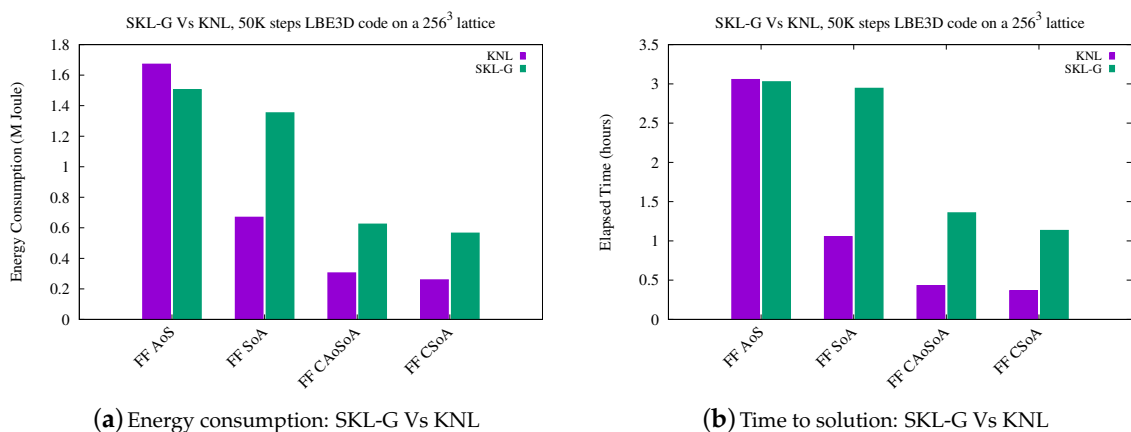
proposed clustered layouts considering power consumption, Figure 7a, as well as time to solution, Figure 7b. Indeed, the memory bandwidth limit of the application remains the main bottleneck and therefore, the KNL delivering a really high memory bandwidth when using efficiently the memory on cheap, it gives the best performances. Despite the SKL-G processor is 15–25% more efficient if looking at the average power drain, Figure 7c, it results twice less efficient in energy-to-solution. It confirms how lower average power drain is not much significant to improve efficiency of high-performance computing applications if not integrated over the application execution time (or *time-to-solution*,  $T_s$ ) to obtain the application consumed energy (or *energy-to-solution*,  $E_s$ ):

$$E_s = T_s \times P_{avg}$$

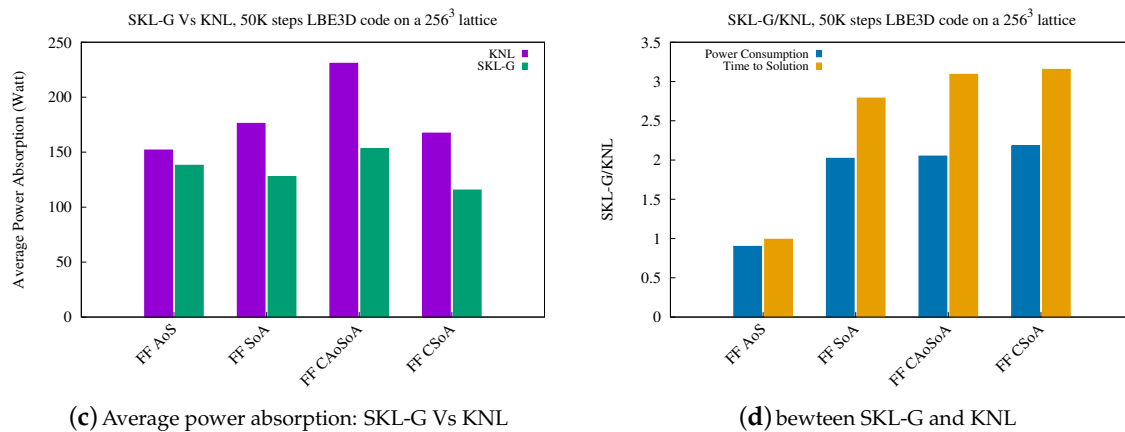
thus an increase in  $T_s$  may lead to an  $E_s$  increase, despite a lower average power drain  $P_{avg}$ . The performance gap between the SKL-G and the KNL processors is also summarized in Figure 7d, where we report the ratio, in both terms of power consumption and time to solution, while comparing the two architectures on the same workload.



**Figure 6.** Measure of energy consumption for the LB main loop on SKL-G processor. In (a) the measured energy efficiency is reported for 50K time steps of the LB main loop on a 256<sup>3</sup> lattice measuring all considered data layout.



**Figure 7.** Cont.



**Figure 7.** The figure reports an overview of the energy efficiency, when comparing the SKL-G and the KNL processors on the same simulation using the LBE3D. In all cases only the the fully fused version of the LBE3D is reported, to compare the two architectures in respect to the most optimized configuration. In (a) we compare the energy consumption, while in (b) we provide a view of the time to solution. In (c) we report the average power absorption and finally, in (d), we summarize the performance gap between the two architectures.

## 6. Conclusions

We extended the analysis of computing performance and energy efficiency on both the SKL and the KNL processors, two high-end computer architecture equipping Tier-0 European systems for high-performance computing such as Marconi-A2 partition, hosted at CINECA and MareNostrum, hosted at BSC. On both processors the highly-optimized data layouts, previously introduced [1,19], and successively implemented on the LBE3D for studying turbulent multicomponent emulsions at high-resolution, have demonstrated to deliver high memory bandwidth for the propagate kernel and good efficiency for the collide kernel, especially on the KNL when using the MCDRAM efficiently [2].

The LBE3D application offers high-memory bandwidth for the propagate kernel but showing a memory bounded limit also for the case of the collide kernel. This is confirmed also from our analysis when we compare the SKL-G processor with the higher-end version, the SKL-P. The LBE3D does not properly exploit the additional cores available on the SKL-P version. The two architectures are mostly similar but due to the fact that the SKL-P delivers higher peak performance (as equipped with 30% more core if compared with the SKL-G), the LBE3D benefits really little of the peak extension. Therefore, for the LBE3D application the SKL-G results more efficient of the SKL-P version in the value for money as the price ratio between the two computer platforms is about a factor of 2.

In all the proposed cases the energy consumption proportionally follows the time to solution without major unexpected behaviours such that the most efficient architecture is the one delivering the fastest time solution. Results highlighted the KNL is the most efficient processor among the ones analyzed from the Intel processor family, at least for our LBM based application. LBE3D is a largely scalable application that allows to reduce the amount of memory per process by increasing the number of processes and it is ideal to fit the MCDRAM capacity of the KNL processor. Indeed, our analysis shows that for the LBE3D application the KNL socket is four times more efficient than the SKL-G processor in terms of value for money as it turned to be approximately two times faster when comparing time to solution and being a factor of 2 cheaper in comparing the price market value.

In a further work, we will consider to extend the performances assessments of the LBE3D application on more high-end systems for high-performance computing, and include also the analysis of energy efficiency at level of whole computer node.

**Author Contributions:** Conceptualization, all authors; methodology, all authors; software, all authors; validation I.G., S.F.S. and F.T.; formal analysis, S.F.S. and F.T.; investigation, I.G., E.C. and S.F.S.; resources, S.F.S. and F.T.;

data curation, I.G.; writing original draft preparation, I.G., G.D.S., E.C. and S.F.S.; review and final editing, all authors; visualization, I.G.; and funding acquisition, I.G., S.F.S. and F.T. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by INFN within the COKA and COSA projects, and by the University of Ferrara in the context of “Grandi Attrezzature 2015”. E.C. was supported by “Contributo 5 per mille assegnato all’Università degli Studi di Ferrara - dichiarazione dei redditi dell’anno 2014” and by the EuroExa Project (grant agreement No. 754337), funded by the European Union’s Horizon 2020 Research and Innovation Programme.

**Acknowledgments:** We would like to thank PRACE for the granted project (ID: 2018184340 & 2019204899) “TurEmu - The physics of (turbulent) emulsions” along with CINECA, BSC, INFN and The University of Ferrara for access to their HPC systems.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Girotto, I.; Schifano, S.F.; Calore, E.; Di Staso, G.; Toschi, F. Performance Optimization of D3Q19 Lattice Boltzmann Kernels on Intel KNL. In Proceedings of the INFOCOMP 2018: The Eighth International Conference on Advanced Communications and Computation, Barcelona, Spain, 22–26 July 2018; pp. 31–36.
2. Girotto, I.; Schifano, S.F.; Calore, E.; Di Staso, G.; Toschi, F. Computational Performances and Energy Efficiency Assessment for a Lattice Boltzmann Method on Intel KNL. *Adv. Parallel Comput.* **2019**, *36*, 605–613. [[CrossRef](#)]
3. Calore, E.; Demo, N.; Schifano, S.F.; Tripicciono, R. Experience on Vectorizing Lattice Boltzmann Kernels for Multi- and Many-Core Architectures. In *PPAM 2015, Proceedings of the Parallel Processing and Applied Mathematics: 11th International Conference, Krakow, Poland, 6–9 September 2015*; Revised Selected Papers, Part I; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2016; pp. 53–62. [[CrossRef](#)]
4. Calore, E.; Gabbana, A.; Schifano, S.F.; Tripicciono, R. Early experience on using Knights Landing processors for Lattice Boltzmann applications. In Proceedings of the Parallel Processing and Applied Mathematics: 12th International Conference (PPAM 2017), Lublin, Poland, 10–13 September 2017; Lecture Notes in Computer Science; Volume 1077, pp. 1–12. [[CrossRef](#)]
5. Calore, E.; Gabbana, A.; Kraus, J.; Schifano, S.F.; Tripicciono, R. Performance and portability of accelerated lattice Boltzmann applications with OpenACC. *Concurr. Comput. Pract. Exp.* **2016**, *28*, 3485–3502. [[CrossRef](#)]
6. Eastep, J.; Sylvester, S.; Cantalupo, C.; Geltz, B.; ArdanazAsma, F.; Livingston, A.R.; Keceli, F.; Maiterth, M.; Jana, S. Global Extensible Open Power Manager: A Vehicle for HPC Community Collaboration on Co-Designed Energy Management Solutions. In *International Supercomputing Conference; High Performance Computing, ISC 2017, Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2017; Volume 10266, pp. 394–412. [[CrossRef](#)]
7. Patki, T.; Lowenthal, D.K.; Sasidharan, A.; Maiterth, M.; Rountree, B.L.; Schulz, M.; de Supinski, B.R. Practical Resource Management in Power-Constrained, High Performance Computing. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '15)*; Association for Computing Machinery: New York, NY, USA, 2015; pp. 121–132. [[CrossRef](#)]
8. Vysocky, O.; Beseda, M.; Říha, L.; Zapletal, J.; Lysaght, M.; Kannan, V. MERIC and RADAR Generator: Tools for Energy Evaluation and Runtime Tuning of HPC Applications. In *High Performance Computing in Science and Engineering (HPCSE 2017)*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018; Volume 11087. [[CrossRef](#)]
9. Hackenberg, D.; Ilsche, T.; Schuchart, J.; Schöne, R.; Nagel, W.E.; Simon, M.; Georgiou, Y. HDEEM: High Definition Energy Efficiency Monitoring. In Proceedings of the 2014 Energy Efficient Supercomputing Workshop, New Orleans, LA, USA, 16 November 2014; pp. 1–10.
10. Roberts, S.I.; Wright, S.A.; Fahmy, S.A.; Jarvis, S.A. Metrics for Energy-Aware Software Optimisation. In Proceedings of the 32nd International Conference, ISC High Performance 2017, Frankfurt, Germany, 18–22 June 2017; Volume 10266. [[CrossRef](#)]
11. Roberts, S.I.; Wright, S.A.; Fahmy, S.A.; Jarvis, S.A. The Power-Optimised Software Envelope. *ACM Trans. Archit. Code Optim.* **2019**, *16*. [[CrossRef](#)]



12. Padoin, E.L.; de Oliveira, D.A.G.; Velho, P.; Navaux, P.O.A. Time-to-Solution and Energy-to-Solution: A Comparison between ARM and Xeon. In Proceedings of the 2012 Third Workshop on Applications for Multi-Core Architecture, New York, NY, USA, 24–25 October 2012; pp. 48–53.
13. Succi, S. *The Lattice Boltzmann Equation: For Fluid Dynamics and Beyond*; Clarendon University Press: Oxford, UK, 2001; ISBN 978-0-19-850398-9.
14. Pierre-Gilles de Gennes, F.B.W.; Quéré, D. *Capillarity and Wetting Phenomena—Drops, Bubbles, Pearls, Waves*; Springer: Cham, Switzerland, 2004; ISBN 978-1-4419-1833-8. [[CrossRef](#)]
15. Sbragaglia, M.; Benzi, R.; Bernaschi, M.; Succi, S. The emergence of supramolecular forces from lattice kinetic models of non-ideal fluids: applications to the rheology of soft glassy materials. *Soft Matter* **2012**, *8*, 10773–10782. [[CrossRef](#)]
16. Williams, S.; Carter, J.; Olikier, L.; Shalf, J.; Yelick, K. Optimization of a Lattice Boltzmann computation on state-of-the-art multicore platforms. *J. Parallel Distrib. Comput.* **2009**, *69*, 762–777. [[CrossRef](#)]
17. Williams, S.; Carter, J.; Olikier, L.; Shalf, J.; Yelick, K.A. Lattice Boltzmann simulation optimization on leading multicore platforms. In Proceedings of the 2008 IEEE International Symposium on Parallel and Distributed Processing, Miami, FL, USA, 14–18 April 2008; pp. 1–14. [[CrossRef](#)]
18. Bernaschi, M.; Fatica, M.; Melchionna, S.; Succi, S.; Kaxiras, E. A flexible high-performance Lattice Boltzmann GPU code for the simulations of fluid flows in complex geometries. *Concurr. Comput. Pr. Exper.* **2009**, *22*, 1–14. [[CrossRef](#)]
19. Calore, E.; Gabbana, A.; Schifano, S.F.; Tripiccione, R. Optimization of lattice Boltzmann simulations on heterogeneous computers. *Int. J. High Perform. Comput. Appl.* **2017**, 1–16. [[CrossRef](#)]
20. Product Specifications. Available online: <https://ark.intel.com/content/www/us/en/ark.html> (accessed on 8 March 2020).
21. Weaver, V.; Johnson, M.; Kasichayanula, K.; Ralph, J.; Luszczek, P.; Terpstra, D.; Moore, S. Measuring Energy and Power with PAPI. In Proceedings of the 2012 41st International Conference on Parallel Processing Workshops (ICPPW), Pittsburgh, PA, USA, 10–13 September 2012.
22. Hackenberg, D.; Schone, R.; Ilsche, T.; Molka, D.; Schuchart, J.; Geyer, R. An Energy Efficiency Feature Survey of the Intel Haswell Processor. In Proceedings of the Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015 IEEE International, Hyderabad, India, 25–29 May 2015; pp. 896–904. [[CrossRef](#)]
23. Desrochers, S.; Paradis, C.; Weaver, V.M. A Validation of DRAM RAPL Power Measurements. In Proceedings of the Second International Symposium on Memory Systems, 2016 (MEMSYS '16), Alexandria, VA, USA, 3–6 October 2016; pp. 455–470. [[CrossRef](#)]
24. Calore, E.; Gabbana, A.; Schifano, S.F.; Tripiccione, R. Evaluation of DVFS techniques on modern HPC processors and accelerators for energy-aware applications. *Concurr. Comput. Pract. Exp.* **2017**, *29*, 1–19. [[CrossRef](#)]



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).