Accelerating physical simulations from a multicomponent Lattice Boltzmann method on a single-node multi-GPU architecture

Julien Duchateau, François Rousselle, Nicolas Maquignon, Gilles Roussel, Christophe Renaud
Laboratoire d’Informatique Signal Image de la Côte d’Opale
Université du Littoral Côte d’Opale
Calais, France
Email: duchateau@lisic.univ-littoral.fr

Abstract—In this paper, we introduce an efficient method to accelerate flow simulations for an isothermal multiphase and multicomponent (MPMC) Lattice Boltzmann method (LBM) on a single-node multi-GPU architecture. Our objective is to propose an efficient way to improve performance of multiphase and multicomponent Lattice Boltzmann simulations by the use of Nvidia GPU Direct technology and Peer-to-Peer (P2P) data transfers. Optimization of Peer-to-Peer communications is also studied in this work by the use of a clustering algorithm. Several simulations are shown and performance is discussed in order to validate the method.

Keywords—single-node multi-GPU implementation, Peer-to-Peer, Lattice Boltzmann method, multiphase and multicomponent flows, K-means clustering;

I. INTRODUCTION

Lattice Boltzmann method (LBM) is a class of computational fluid dynamics (CFD) methods for fluid simulations. It is a relatively recent technique which is able to approximate Navier-Stokes equations by the use of a collision-propagation scheme [1]. It is an interesting alternative which allows to simulate complex phenomena by its mesoscopic nature. Its high parallelization capacity also makes this method attractive to perform fast simulations on parallel hardware. Moreover, the emergence of high-performance computing (HPC) using graphics processing units (GPU) [6] has attracted the interest of many researchers.

In this work, we introduce a method to accelerate flow simulations for an isothermal multiphase and multicomponent (MPMC) Lattice Boltzmann method (LBM) on a single-node multi-GPU architecture. This method is mainly based on the maximization of GPU parallelism and the use of Peer-to-Peer (P2P) memory transfer. A clustering algorithm is also used in order to maximize Peer-to-Peer data exchanges. At the best of our knowledge, no recent work deals with such optimizations for Lattice Boltzmann method. Moreover, the use of Peer-to-Peer data transfer for inter-GPU communications has never been exploited for LBM simulations.

Section II first introduces the multiphase and multicomponent model used for our simulations. Several recent existing works involving implementations of Lattice Boltzmann methods on GPUs are then presented in section III. Optimizations and the resulting algorithm concerning this model are proposed in section IV. This paper mainly focus on the acceleration of data transfer by the inclusion of Peer-to-Peer data exchanges. The use of a clustering algorithm based on the K-means method in order to maximize P2P communications is also introduced. An analysis concerning performance is done through several simulations in section V. A conclusion and future works are finally presented in the last section.

II. ISONTHERMAL MULTIPHASE AND MULTICOMPONENT LATTICE BOLTZMANN METHOD

A. The Lattice Boltzmann Bhatnagar-Gross-Krook (BGK) equation

The Lattice Boltzmann method is based on three main discretizations: space, time and velocities. Velocity space is reduced to a finite number of well-defined vectors. Figures 1(a) and 1(b) illustrate this discrete scheme for D2Q9 and D3Q19 model.

![Example of Lattice Boltzmann schemes](image_url)

(a) D2Q9 scheme
(b) D3Q19 scheme

The simulation grid is therefore discretized as a Cartesian grid and calculation steps are achieved on this entire grid. The classical LBM collision-propagation equation [1] with single relaxation time Bhatnagar-Gross-Krook (SRT-BGK) collision term is defined by the following equation:

$$f_i(x + e_i, t + \Delta t) = f_i(x, t) - \frac{1}{\tau} (f_i(x, t) - f_i^{eq}(x, t))$$

(1)

The function $f_i(x, t)$ corresponds to the discrete density distribution function along velocity vector $e_i$ at a location $x$. 
and a time \( t \). The parameter \( \tau \) corresponds to the relaxation time of the simulation. The equilibrium distribution function is defined as follows:

\[
f_{eq}^i(x,t) = \omega_i \rho(x,t)[1 + \frac{e_i u(x,t)}{c_s^2} + \frac{(e_i u(x,t))^2}{2c_s^4} - \frac{u(x,t)^2}{2c_s^2}]
\]  

(2)

With \( \omega_i \) taking the values in a D3Q19 scheme:

\[
w_0 = \frac{1}{3}; \quad w_{1-6} = \frac{1}{18}; \quad w_{7-18} = \frac{1}{36};
\]  

(3)

Note that for D2Q9 and D3Q19 scheme \( c_s^2 = \frac{1}{3} \left( \frac{3}{2} \right)^2 \) with \( d_t \) as time step and \( d_x \) as spatial step. The first two moments are density \( \rho \) and velocity \( u \), with space and time dependency:

\[
\rho(x,t) = \sum_i f_i(x,t); \quad \rho(x,t)u(x,t) = \sum_i f_i(x,t)e_i;
\]  

(4)

**B. Isothermal multiphase and multicomponent (MPCM) model**

Several models have been developed from BGK-LBM in order to perform complex physical simulations. In this section, a MPCM-LBM model based on the work achieved by Bao & Schaeffer [5] is presented. The first step of the algorithm consists in calculating the pressure for each component \( \sigma \) by the use of the adimensionalized Peng-Robinson equation of state:

\[
p_{\sigma}(x,t) = \rho_{\sigma}(x,t)R_{\sigma}T_{\sigma} - \frac{a_{\sigma} \alpha(T_{\sigma}) \rho_{\sigma}^2(x,t)}{1 + b_{\sigma} \rho_{\sigma}(x,t) - b_{\sigma} \rho_{\sigma}^2(x,t)}
\]  

(5)

A pseudo-potential is then deduced from the pressure term. At a location \( x \), it has the following form:

\[
\psi_{\sigma}(x,t) = \sqrt{\frac{2(p_{\sigma}(x,t) - c_s^2 \rho_{\sigma}(x,t))}{c_s^2 g_{\sigma}}}
\]  

(6)

Forces are then computed by the use of the pseudo-potential. The internal fluid interaction force is expressed as follows [3] [4]:

\[
F_{\sigma\sigma}(x) = -\beta \psi_{\sigma}(x) \sum_{x'} G_{\sigma\sigma}(x,x') \psi_{\sigma}(x')(x' - x)
\]

\[
-\frac{(1 - \beta)}{2} \sum_{x'} G_{\sigma\sigma}(x,x') \psi_{\sigma}^2(x')(x' - x)
\]  

(7)

The inter-component force is also introduced. The equation introduced by Bao & Schaeffer in [5] is:

\[
F_{\sigma\sigma'}(x) = -\psi_{\sigma}(x) \sum_{x'} G_{\sigma\sigma'}(x,x') \psi_{\sigma'}(x')(x' - x)
\]  

(8)

Some additional forces such as gravity or fluid structure interactions can be added into the model. The incorporation of the force term is then achieved by an Exact Difference Method (EDM). It takes a modified collision operator as follows:

\[
f_{\sigma,i}(x+e_i \delta t, t+\delta t) - f_{\sigma,i}(x,t) = - \frac{f_{\sigma,i}(x,t) - f_{\sigma,i}^{eq}(x,t)}{\tau_{\sigma}} + \Delta f_{\sigma,i}
\]  

(9)

\[
\Delta f_{\sigma,i}(x,t) = f_{\sigma,i}^{eq}(\rho_{\sigma}(x,t), u_{\sigma} + \Delta u_{\sigma}) - f_{\sigma,i}^{eq}(\rho_{\sigma}(x,t), u_{\sigma})
\]  

(10)

\[
\Delta u_{\sigma} = F_{\sigma} \delta t / \rho_{\sigma}
\]  

(11)

The density and velocity of component \( \sigma \) finally correspond to the moments of \( f_{\sigma} \):

\[
\sum_i f_{\sigma,i}(x,t) = \rho_{\sigma}(x,t)
\]  

(12)

\[
\sum_i f_{\sigma,i}(x,t)e_i = \rho_{\sigma}(x,t)u_{\sigma}(x,t)
\]  

(13)

### C. Algorithm

This section gives details about the algorithm concerning the model described previously. This paragraph is also useful to clarify and summarize the previous sections.

**Algorithm 1**: Algorithm for multiphase and multicomponent Lattice Boltzmann model

III. LATTICE BOLTZMANN METHOD ON GPUs

The important interest concerning GPUs has been quickly exploited in order to perform fast 2D [8] and 3D [9] simulations using Lattice Boltzmann method. Recent works have shown that GPUs are also used with multiphase and multicomponent models [17] [15]. The main aspects of GPU optimizations are decomposed into several categories [11] [10]:

- Thread Level parallelism.
- GPU memory accesses.
- Use of registers.
- Overlap of memory transfers with computations.

Data coalescence is needed in order to optimize global memory bandwidth. This implies two conditions:
• Alignment of data in GPU memory.
• The $k^{th}$ thread in a half warp must access the $k^{th}$ element in a block being read.

Concerning LBM, an adapted data structure such as the Structure of Array (SoA) has been well studied and has proven to be efficient on GPU [8]. The second condition is generally obtained by the fusion of the simulation domain with CUDA execution grid [11] [12] [10]. For this, CUDA kernels are run on a two-dimensional grid of blocks and each block is composed of one dimensional group of threads.

Several access patterns are described in the literature. The first one, named A-B access pattern, consists of using two calculation grids in GPU global memory in order to manage the temporal and spatial dependency of the data (Equation 9). The simulation steps alternate between reading distribution functions from A and writing them to B, and reading from B and writing to A reciprocally. This pattern is commonly used and offers very good performance [11] [12] [10] on a single GPU. Several techniques are however presented in literature in order to reduce significantly the computational memory cost without loss of information such as grids compression [7], Swap algorithm [7] or A-A pattern technique [13].

Recent works involving implementation of Lattice Boltzmann method on a single-node composed of several GPUs are also available. A first solution, proposed in [14] [18], consists in dividing the simulation domain into sub-domains according to the number of GPUs and performing LBM kernels on each sub-domain in parallel. CPU threads are used to handle each CUDA context. Communications between sub-domains are performed using zero-copy memory transfers. Zero-copy feature allows to perform efficient communications by a mapping between CPU and GPU pointers. Data must however be read and written only once in order to obtain good performance.

Some approaches have finally been proposed to perform simulations on several nodes constituted of multiple GPUs by the use of MPI in combination with CUDA [20] [19] [22] [16]. In our case, we only dispose of one computing node with multiple GPUs thus we don’t pay too much attention on these architectures in this paper.

IV. ACCELERATION OF DATA TRANSFER BETWEEN GPUs

All works described in Section III concerning single node multi-GPU implementation of LBM are generally limited to the use zero-copy memory transactions in order to achieve communications between GPUs. Some references [18] [14] have shown that zero-copy transactions can offer good performance for standard lattice Boltzmann schemes. However, the model described previously implies more exchanges and synchronizations between GPUs than a standard model. We therefore propose to exploit Nvidia GPUDirect feature and Peer-to-Peer data transfer in order to increase communications efficiency. In addition to this, the use of a clustering algorithm is also proposed in order to automatically and efficiently assign the GPUs on the simulation domain. This clustering has for goal to maximize Peer-to-Peer data transfers. The use of Peer-to-Peer data transfer for inter-GPU communications has never been exploited for LBM simulations and provides excellent scalability according to the number of GPUs at our disposal.

A. Computations/communications overlap

Perform Lattice Boltzmann simulations on a single-node multi-GPU architecture implies an optimization of data exchanges between the GPUs. Efficiency of inter-GPU communications is surely the most difficult task in order to obtain good performance. Indeed, the simulation domain is divided into sub-domains according to the number of GPUs as shown on Figure 2 and this implies therefore to exchange data on interfaces between connected sub-domains.

![Fig. 2. Division of the simulation domain according to the number of GPUs: each GPU has in charge one sub-domain of the simulation grid](image)

Many exchanges are indeed needed for this type of model. The computation of interaction $F_{int}$ and inter-component $F_{ext}$ implies to have access to neighboring values of the pseudo-potential. The propagation step of LBM also implies to communicate several distribution functions between GPUs. To perform communications, aligned buffers may be used for data transactions (Figure 3).

![Fig. 3. Inter-GPU communications example in a 2D case: boundary distribution functions are transferred between two neighboring sub-domains](image)

In order to obtain a simulation time as short as possible, it is necessary to overlap data transfer with algorithm calculations. Indeed, overlapping computations and communications allows significant performance gain by reducing the waiting time of data. The idea is to separate the computation process...
into 2 steps: boundary and interior. Computations on the needed boundaries are firstly done. Communications between neighboring sub-domains are also done while computing the interior. The different communications are thus performed simultaneously with calculations which allows good efficiency. We therefore obtain the algorithm described in Algorithm 2.

Algorithm 2: Algorithm for multiphase and multicomponent Lattice Boltzmann model with communications/computations overlap.

B. Inclusion of Peer-to-Peer data transfer

In most cases for Lattice Boltzmann method, memory is transferred via zero-copy transactions to page-locked memory which allow good overlapping between communications and computations [18] [14] [16]. We propose in this paper a different approach concerning inter-GPU communications. In most recent HPC architectures, several GPUs can be connected to the same PCIe. To improve performance, Nvidia launched GPUDirect with CUDA 4.0.

This technology allows to perform Peer-to-Peer transfers and memory accesses between two compatible GPUs. The idea is to perform data transfer using Peer-to-Peer data transactions for GPUs sharing the same I/O hub and zero-copy transactions for others. This method allows to communicate data by bypassing the use of the CPU and therefore to accelerate the transfer (Figure 4). The use of this type of transaction improves performance and the efficiency of the simulation code.

C. Maximization of Peer-to-Peer data transfer

The use of Peer-to-Peer data transfer allows faster communications than zero-copy transactions. This type of transactions however requires compatibility between two GPUs. It is therefore important to promote this type of communications in order to maximize efficiency. The use of a simple clustering algorithm as K-means method [24] [23] is therefore proposed in order to maximize the use of Peer-to-Peer exchanges.

The number of clusters must firstly be defined. This can be done by the use of CUDA which is able to know if GPUs can communicate in Peer-to-Peer. The number of clusters is thus defined according to the possibility to communicate in Peer-to-Peer. Each group of GPUs able to communicate in Peer-to-Peer defines a cluster. The idea is then to connect spatially closest sub-domains to GPUs which can perform Peer-to-Peer communications, as shown on Figure 5.

We propose to use an adapted K-means clustering algorithm to define our clusters and then assign GPUs to clusters. It is described as follows:

Algorithm 3: K-means clustering algorithm and assignment of GPUs

A constraint is added to the clustering algorithm in order to have an uniform distribution of GPUs. Indeed, each GPU has

\(^{1}\)The cluster term always refers in this paper to the result of a clustering algorithm.
to be assigned to one sub-domain.

The main advantage of this algorithm is that Peer-to-Peer communications are maximized through the spatial locality of connected sub-domains. This algorithm is also simple to implement, automatic, efficient, and can be used on many applications.

D. Resulting algorithm

This section summarizes briefly the algorithm able to perform accelerated physical simulations from a multiphase and multicomponent lattice Boltzmann model with the inclusion of our contribution based on Peer-to-Peer data transfer and K-means clustering.

for each physical component do
    initialize simulation parameters;
end
divide the simulation grid according to the number of GPUs;
adapted K-means algorithm in order to assign efficiently GPUs to sub-domains;
for each iteration do
    for each sub-domain do
        for each physical component do
            compute $p_\sigma$ and $\psi_\sigma$ using (5) and (6) on boundaries;
        end
    end
    for each sub-domain do
        for each physical component do
            communications of $\psi_\sigma$ on boundaries;
            compute $p_\sigma$ and $\psi_\sigma$ for the interior;
        end
    end
    for each sub-domain do
        for each physical component do
            compute force terms with (7) and (8);
            perform a correction of velocity field by calculating $u_\sigma + \Delta u_\sigma$ from (11);
            calculate $\Delta f_{\sigma,i}$ (10);
        end
    end
    for each sub-domain do
        for each physical component do
            perform collision, propagation on boundaries;
        end
    end
    for each sub-domain do
        for each physical component do
            • communicate $f_{\sigma,i}$ needed values on boundaries;
            • perform collision, propagation and boundary conditions for $f_{\sigma,i}$ (9) for the interior;
            compute macroscopic quantities (12) (13);
        end
    end
end

Algorithm 4: Algorithm for multiphase and multicomponent Lattice Boltzmann model including K-means clustering in order to maximize Peer-to-Peer communications.

V. RESULTS AND PERFORMANCE

A. Simulations and Hardware

Several simulations have been performed in order to validate our contribution and study its performance (Table I and Figures 6(a), 6(b), 6(c)).

for each physical component do
    initialize simulation parameters;
end
divide the simulation grid according to the number of GPUs;
adapted K-means algorithm in order to assign efficiently GPUs to sub-domains;
for each iteration do
    for each sub-domain do
        for each physical component do
            compute $p_\sigma$ and $\psi_\sigma$ using (5) and (6) on boundaries;
        end
    end
    for each sub-domain do
        for each physical component do
            • communications of $\psi_\sigma$ on boundaries;
            • compute $p_\sigma$ and $\psi_\sigma$ for the interior;
        end
    end
    for each sub-domain do
        for each physical component do
            compute force terms with (7) and (8);
            perform a correction of velocity field by calculating $u_\sigma + \Delta u_\sigma$ from (11);
            calculate $\Delta f_{\sigma,i}$ (10);
        end
    end
    for each sub-domain do
        for each physical component do
            perform collision, propagation on boundaries;
        end
    end
    for each sub-domain do
        for each physical component do
            • communicate $f_{\sigma,i}$ needed values on boundaries;
            • perform collision, propagation and boundary conditions for $f_{\sigma,i}$ (9) for the interior;
            compute macroscopic quantities (12) (13);
        end
    end
end

Algorithm 4: Algorithm for multiphase and multicomponent Lattice Boltzmann model including K-means clustering in order to maximize Peer-to-Peer communications.

V. RESULTS AND PERFORMANCE

A. Simulations and Hardware

Several simulations have been performed in order to validate our contribution and study its performance (Table I and Figures 6(a), 6(b), 6(c)).

We used 8 NVIDIA Tesla C2050 graphics cards Fermi architecture based machine to perform simulations. Table II describes some Tesla C2050 hardware specifications. The possibility of providing Peer-to-Peer communications is described
in Table III (X means that Peer-to-Peer is available between the two corresponding GPUs).

**Tesla C2050**

| CUDA Capability Major/Minor version number: | 2.0 |
| Total amount of global memory: | 2687 MBytes |
| (14) Multiprocessors, (32) CUDA Cores/MP: | 448 CUDA Cores |
| GPU Clock rate: | 1147 MHz (1.15 GHz) |
| L2 Cache Size: | 786432 bytes |
| Total amount of shared memory per block: | 49152 bytes |
| Total number of registers available per block: | 32768 |

**TABLE II**

**TESLA C2050 HARDWARE SPECIFICATIONS**

<table>
<thead>
<tr>
<th>GPU</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE III**

**PEER-TO-PEER COMMUNICATIONS ACCESSIBILITY BETWEEN GPUs (X FOR AVAILABLE)**

**B. Peer-to-Peer communications efficiency**

This section describes performance obtained by the use of Peer-to-Peer communications instead of usual zero-copy transactions. The hardware and model used being different from references previously presented, it seems very hard to make a comparison with their results.

The objective is to highlight the inter-GPU communications efficiency including the use of Peer-to-Peer data exchanges associated with the K-means clustering algorithm by a comparison with the use of standard zero-copy transfers.

A Million Lattice nodes Update Per Second (MLUPS) is the performance measurement currently used for LBM. Tables IV and V describe the performance obtained for the simulations previously described. Note first that we perform approximatively 150 MLUPS for a single GPU simulation, which is 29 times faster than our CPU OpenMP parallelized simulation on two Intel Xeon X5660 (12 hyperthreaded cores). For 8 GPUs working in parallel, we therefore obtain a gain of 226 times as compared to the same CPU implementation.

Performance is now studied through two different ways of inter-GPU communications. The first one consists in using only zero-copy memory transactions. We can note that we obtain relatively good scalability for this type of communication with a slight decrease when the number of GPUs increases.

Inclusion of Peer-to-Peer memory transactions into the simulation code is now studied. We note that this approach obtains better performance and better scalability than the zero-copy method. This better way to communicate data allows to improve performance between 8% and 12% approximatively according to the number of GPUs. Our results also demonstrate that we can have performance very close to perfect scaling with an efficiency of 98.5% on 8 GPUs (Figure 7).

**C. Impact of the clustering algorithm on performance**

Good performance of Peer-to-Peer data transactions can also be explained by an efficient assignment of GPUs on the simulation domain. In this section, a comparison is made between a random distribution of GPUs and the use of the clustering algorithm presented previously. Figures 8 and 9 provide some results concerning the clustering algorithm for 8 GPUs. As our architecture is currently limited to 8 GPUs, we are unable to perform tests for more GPUs currently even if the clustering algorithm can provide good results for more GPUs.

The random distribution implies only a few Peer-to-Peer communications because the spatial locality of connected subdomains is not considered in the distribution. However, K-means algorithm offers a distribution of GPUs which maxi-
mize the use of Peer-to-Peer exchanges because clusters are defined according spatial locality.

Easing Peer-to-Peer communications with a simple clustering algorithm is therefore an important factor to optimize in order to obtain best performance. This approach will be generalized for future works in order to be applied with multiple nodes architectures.

VI. CONCLUSION

We have introduced an efficient implementation of an isothermal multiphase and multicomponent Lattice Boltzmann on a single-node multiple GPUs architecture. We have been able to perform a simulation of a two-component flow on a grid size of $128 \times 256 \times 2048$ cells on 8 GPUs with a mean performance of 1180 MLUPS which is really satisfying concerning the model complexity.

The optimization of GPU computations have been well studied in literature but we have improved the efficiency of the developed algorithm by speeding up data transfer between different GPUs by the use of Peer-to-Peer data transactions. We demonstrated that the use of Peer-to-Peer data transfers have an important impact on the simulations and allows better performance and better scalability than zero-copy transactions.

We also demonstrated that the use of a clustering algorithm in order to maximize Peer-to-Peer exchanges have an important impact on the performance of the simulations. The use of a graph partition algorithm could provide better results for more GPUs and it will be studied in the coming months.

Extensions of this work to cases which include multiple nodes of GPUs is also now under investigation. Data transfer optimizations between the nodes will therefore be essential to keep good performance. A clustering algorithm allowing to minimize transfers between the nodes while maximizing Peer-to-Peer data transfer inside the node is currently in development.

ACKNOWLEDGMENT

This work has been made possible thanks to a collaboration between academic and industrial groups, gathered by the INNOCOLD association.

REFERENCES