The TheLMA Project:
Multi-GPU Implementation of the Lattice Boltzmann Method

Christian Obrecht, Frédéric Kuznik, Bernard Tourancheau,
and Jean-Jacques Roux

CCGSC – September 8, 2010
Motivations

- Due to computational cost, fluid dynamics is often neglected in building physics. Modelling of energy efficient buildings requires to take these effects into account.

- The lattice Boltzmann method (LBM) is an innovative approach in CFD. Besides other advantages, parallel implementations of the LBM are rather straightforward.

- GPUs, e.g. CUDA capable hardware, provide an inexpensive and efficient way to perform parallel computations.

- In real life computations, the device memory of a single GPU is too small to store a whole lattice, hence multi-GPU implementations are necessary.
I – Lattice Boltzmann Method
Lattice Boltzmann Method

Mass transfer is performed in discrete time and space using a finite set of velocities, as the D3Q19 stencil:
The fluid is represented by a discrete distribution $f_i$ associated to the velocities $e_i$, and obeying to the following equation:

$$f_i(x + \delta t e_i, t + \delta t) - f_i(x, t) = \Omega_i(f(x, t))$$

where $\Omega_i$ is a collision operator.

The macroscopic quantities are given by:

$$\rho = \sum_i f_i \quad \quad \quad \quad u = \frac{1}{\rho} \sum_i f_i e_i$$
Algorithmic Aspect

The LBM breaks up in two elementary steps, i.e. collision and propagation:

\[ \tilde{f}_i(x, t) = f_i(x, t) + \Omega_i(f(x, t)) \]

\[ f_i(x + \delta t e_i, t + \delta t) = \tilde{f}_i(x, t) \]
for each time step \( t \) do
    for each lattice node \( x \) do
        read velocity distribution \( f_i(x, t) \)
        if node \( x \) is on boundaries then
            apply boundary conditions
        end if
        compute updated distribution \( \tilde{f}_i(x, t) \)
        propagate to neighbouring nodes \( x + \delta t e_i \)
    end for
end for
II – Implementing the LBM on the GPU
Single-GPU Implementation of the LBM

- Assign one node per thread to take advantage of massive parallelism.
- Store the velocity distribution in global memory using a structure of array (SoA) like layout to enable coalescing.
- Launch one kernel for each time step to ensure global synchronisation.

The limiting factor for single-GPU implementation is the global memory maximum throughput.
Misaligned Memory Transactions

The major issue in optimising single-GPU implementation is to reduce the impact of misaligned memory transactions.
Shared Memory Approach

Normal case

Block boundaries
In-Place Propagation

Misaligned reads being far less expensive than misaligned writes, an alternative to the shared memory approach is to use in-place propagation.
III – Multi-GPU Implementation
The TheLMA Framework

param.c
init.c
stat.c
output.c

main.c

thelma.cu

gometry.cu
init.cu
compute.cu
results.cu
To validate our code, we implemented the lid-driven cubic cavity test case. The lattice is split in cuboid sub-domains along the major dimension.
Inter-GPU Communication Scheme

Communication between GPUs is performed using page-locked CPU memory and zero-copy memory transactions.
Communication between GPUs is performed using page-locked CPU memory and zero-copy memory transactions.
IV – Performance Study
We measured up to 2.17 GLUPS (94% efficiency) using 6 Tesla C1060 on a $192^3$ lattice in single precision.
Splitting the lattice along the minor dimension leads to sub-domains composed of non contiguous memory cells, which has dramatic effects on performance.
To measure inter-GPU maximum sustained throughput, we used a stripped-off version of our program.
## Maximum Sustained Throughput vs Required Throughput

<table>
<thead>
<tr>
<th>GPUs</th>
<th>Maximum (MB/s)</th>
<th>Kernel duration (µs)</th>
<th>Required (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8958</td>
<td>9145</td>
<td>322</td>
</tr>
<tr>
<td>3</td>
<td>7661</td>
<td>6096</td>
<td>968</td>
</tr>
<tr>
<td>4</td>
<td>7270</td>
<td>4572</td>
<td>1935</td>
</tr>
<tr>
<td>6</td>
<td>6650</td>
<td>3048</td>
<td>4838</td>
</tr>
</tbody>
</table>

The required throughput is always less than the maximum available. It should be mentioned that the devices are able of efficient communication/computation overlapping.
Summary and Future Work

In this contribution:

- We present an implementation of a multi-GPU LBM solver.
- Our implementation shows good performance and scalability.
- We study effective throughput of inter-GPU communications.

Future work will include:

- Multi-GPU thermal LBM solver.
- Extension of the TheLMA framework.
- MPI based implementation.
Thank you for listening!