LBM performance in Exascale era

Accademia Discussion Group (Open power Foundation).
Giorgio Amati, g.amati@cineca.it
The aim of this talk...

✓ Exascale machine class will be delivered in the next few years.
✓ They will have order of 10’000 nodes with accelerators or more than 100’000 nodes without accelerators.
✓ Which performance can a real code achieve on these HW?
✓ We’ll try to make an “educated guess” about performance figure reasonable for a LBM code, also in terms of performance portability.
  ○ Playing with different HW
  ○ Just to have some figures to think about
  ○ We are aware that there are other more clever than us...
✓ LBM: brief introduction
✓ Code Description
✓ Single node performance
✓ Multi node performance
✓ Going to exascale
✓ Comments
√ CFD Kinetic based numerical scheme
  ○ Particle population to integrate not velocity & pressure
  ○ For classical lattice (D3Q19) 19 populations to take care of
  ○ Ideal to parallel computing
  ○ BW limited
  ○ No front tracking for two phase flows
  ○ Equilibrium depends only from local values of velocity and density

\[
    f_i(x + c_i, t + 1) - f_i(x, t) = \omega [f_i^{eq}(x, t) - f_i(x, t)]
\]

\[
    \rho(x, t) = \sum_{i=0}^{b} f_i(x, t)
\]

\[
    u(x, t) = \sum_{i=0}^{b} c_i f_i(x, t) / \rho(x, t)
\]
Euplectella aspergillum
(Venus’ flower Basket)

Evolutionary structures
Evolutionary Structures

Re = 2000
Re = 1500
Re = 1000
Re = 500
Re = 100

\( u_x \) vs. \( u_y \)

Re = 2000
Re = 1500
Re = 1000
Re = 500
Re = 100

\( u_x \) vs. \( u_y \)
✓ Full sponge run (marconi100, Nvidia V100)
  ○ Size: 5600x3400x2500  (7 TB RAM)
  ○ Sponge radius = 100 grid-points
  ○ Mlups: 680’000 (a.k.a. 170 Tflops, full simulation)
  ○ 5 Re Number, 512GPUs, MPI+OpenACC
  ○ 50’000 gpu-hours

✓ Evolutionary run (marconi, Intel KNL)
  ○ Size: 4000x2000x400 (.5 TB RAM)
  ○ Sponge radius = 100 grid-points
  ○ 4 geometries, 5 Re numbers each
    ➢ 128 KNL nodes, 8192 cores, MPI+OpenMP (for 3 geometries)
    ➢ 16 nodes, 64 GPUs, MPI+OpenACC (for 1 geometry)
  ○ 500’000 cpu-hours, 10’000 gpu-hours
✓ LBM: brief introduction
✓ Code Description
✓ Single node performance
✓ Multi node performance
✓ Going to exascale
✓ Comments
✓ BGK3D: D3Q19 Lattice, Single Relaxation Time (SRT)
  ○ Multiblock
  ○ Two implementation (Original/Fused)
  ○ Senderecv or non-blocking send
✓ Brute force approach
  ○ No grid refinement, next step
✓ Hybrid parallelization for performance portability
  ○ Intel KNL ➔ MPI+OpenMP
  ○ Nvidia V100 ➔ MPI+OpenACC
  ○ Fujitsu A64fx ➔ MPI+OpenMP
✓ Important to understand which is the limit that can achieved with up-to-now supercomputer for a single phase SRT LBM
✓ Metrics used is Mega-Lattice-Update-Per-Second (MLUPS)
  ○ Higher is better
✓ Arithmetic intensity (Roofline model) of the code, single precision, is
  ○ 1.64 for the fused implementation (one big subroutine)
  ○ 0.84 for the streaming implementation (two different subroutines)
✓ Collision step represents 100% of total Flops
  ○ About 250 flops per gridpoint
✓ Test case: 3D lid driven cavity
✓ Validation check: plotting velocity profile along z-direction
✓ Code written in Fortran90
✓ Hybrid Parallelization with
  ○ OpenMP
  ○ OpenACC
✓ OpenACC porting was not so painful due to previous OpenMP parallelization, about 40 loops has been parallelized using directives
✓ Further few 1D loops has been parallelized to avoid data movement between Host and device
✓ Flow validation was the most expensive step
  ○ PGI compiler 19.10 release
  ○ (preliminary) Gnu 9.X with OpenACC support tested but slow (4 times!!!)
✓ LBM: brief introduction
✓ Code Description
✓ **Single node performance**
✓ Multi node performance
✓ Going to exascale
✓ Comments
“Dirty Tricks”: single vs. double precision

**Definition**: some tricks that in a particular situation can help to improve performance but are not always feasible…

- ✔ Double precision (Re=100)
- ✔ Single precision (Re=100)
“Dirty Tricks”: single vs. double precision

- Intel KNL, 64 Nodes
- Cylinder @Re=100

<table>
<thead>
<tr>
<th>Precision</th>
<th>Total time</th>
<th>MLUPS</th>
<th>MPI time</th>
<th>COLL time</th>
</tr>
</thead>
<tbody>
<tr>
<td>single</td>
<td>29591”</td>
<td>52717</td>
<td>12019” (40%)</td>
<td>16774 (47%)</td>
</tr>
<tr>
<td>double</td>
<td>51253”</td>
<td>30437</td>
<td>17064” (33%)</td>
<td>33224 (64%)</td>
</tr>
<tr>
<td>mixed</td>
<td>39820”</td>
<td>39175</td>
<td>11726” (30%)</td>
<td>27295 (69%)</td>
</tr>
</tbody>
</table>
“Dirty Tricks”: single vs. double precision

✓ Nvidia V100 GPU (16GB HMB)
✓ 3D Lid Driven Cavity

<table>
<thead>
<tr>
<th>Precision</th>
<th>Total time</th>
<th>MLUPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>single</td>
<td>493”</td>
<td>3403</td>
</tr>
<tr>
<td>double</td>
<td>931”</td>
<td>1802</td>
</tr>
<tr>
<td>mixed</td>
<td>544”</td>
<td>3085</td>
</tr>
</tbody>
</table>
✓ We are not so far from the max speed we can achieve (NVidia V100)
✓ Max → 1.36 TFlops
✓ Obtained → 0.87 TFlops

Here we are (GPU) !!!!
3D Driven, size=360^3, single precision

- 1 Fujitsu node, mpi+openmp, streaming implementation (5.4 TFlops peak)
- 1 GPU NVidia V100, pure openacc (15.7 TFlops peak)
- 1 Intel KNL node, pure openmp (~6 TFlops peak)

<table>
<thead>
<tr>
<th>Machine</th>
<th>Parallelization</th>
<th>Impl.</th>
<th>Mlups</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fujitsu a64fx</td>
<td>4task/12 threads</td>
<td>Original</td>
<td>1346</td>
<td>0.336 TF (6.2%)</td>
</tr>
<tr>
<td>Nvidia V100</td>
<td>Openacc</td>
<td>Fused</td>
<td>3419</td>
<td>0.854 TF (5.4%)</td>
</tr>
<tr>
<td>Intel KNL</td>
<td>64 threads</td>
<td>Fused</td>
<td>1421</td>
<td>0.355 TF (5.9%)</td>
</tr>
</tbody>
</table>

Caveat: for a64fx performance are still evolving
✓ LBM: brief introduction
✓ Code Description
✓ Single node performance
✓ Multi node performance
✓ Going to exascale
✓ Comments
## HW Comparison

✓ 3D Driven, 960*1440*960 gridpoint
  - Fujitsu: MPI+OpenMP, Tofu interconnect
  - M100: MPI+OpenAcc, IB-EDR Interconnect
  - Summit: MPI+OpenACC, IB-EDR interconnect
  - KNL: MPI+OpenMP, OPA interconnect

<table>
<thead>
<tr>
<th>Processor</th>
<th>Nodes</th>
<th>Parallelization</th>
<th>Mlups</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fujitsu A64Fx</td>
<td>48</td>
<td>4 task + 12 threads per node</td>
<td>49760</td>
</tr>
<tr>
<td>m100/Nvidia V100</td>
<td>12 (4 GPU per node)</td>
<td>1 task per GPU</td>
<td>80282</td>
</tr>
<tr>
<td>Summit/Nvidia V100</td>
<td>8 (6 GPU per node)</td>
<td>1 task per GPU</td>
<td>67936</td>
</tr>
<tr>
<td>Marconi/Intel KNL</td>
<td>48</td>
<td>1 task x 64 threads per node</td>
<td>62789</td>
</tr>
</tbody>
</table>

✓ Caveat: for a64fx/Summit performance are still evolving
CINECA

✓ 740³ gridpoint per Node

<table>
<thead>
<tr>
<th>Node</th>
<th>Gpu</th>
<th>Mlups</th>
<th>PFlops</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>10769</td>
<td>-</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>20167</td>
<td>-</td>
<td>0.93</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>37203</td>
<td>-</td>
<td>0.86</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>70032</td>
<td>-</td>
<td>0.81</td>
</tr>
<tr>
<td>16</td>
<td>64</td>
<td>128640</td>
<td>0.03</td>
<td>0.74</td>
</tr>
<tr>
<td>32</td>
<td>128</td>
<td>250963</td>
<td>0.06</td>
<td>0.73</td>
</tr>
<tr>
<td>64</td>
<td>256</td>
<td>506892</td>
<td>0.13</td>
<td>0.73</td>
</tr>
<tr>
<td>128</td>
<td>512</td>
<td>1011420</td>
<td>0.25</td>
<td>0.73</td>
</tr>
<tr>
<td>256</td>
<td>1024</td>
<td>2028480</td>
<td>0.51</td>
<td>0.73</td>
</tr>
<tr>
<td>512</td>
<td>2048</td>
<td>3999260</td>
<td>1.00</td>
<td>0.72</td>
</tr>
<tr>
<td>936</td>
<td>3744</td>
<td>7507385</td>
<td>1.87</td>
<td>0.74</td>
</tr>
</tbody>
</table>
✓ 740^3 gridpoint per Node
✓ 0.73 Efficiency in petascale range
✓ LBM: brief introduction
✓ Code Description
✓ Single node performance
✓ Multi node performance
✓ Going to exascale
✓ Comments
“Back of envelope” figures

With the figure obtained so far we can make an educated guess of real performance for a LBM code with these assumptions

✓ Efficiency will be around 70% in the exascale range
✓ A 2x HW performance improvement

Using an Exascale machine with 15’000 nodes and 60’000 accelerators we can achieve

→ About 60 PFlops sustained
→ About 240 TLUPs → 240’000’000’000’000 Lattice Updated
→ Order of 6 Tera Grid-points (or 1 PB of RAM)
→ I/O can introduce a 15% overhead
→ Obstacle (sponges) can introduce another 15% overhead (to optimize)
→ A “(10’000)^3” simulation should ask 0.005” per timestep
Further steps

✓ Running using Summit (more figures)
✓ Running using Nvidia A100
✓ Running using OpenMP support for GPU
✓ Grid refinement
✓ Try CudaFortran performance respect OpenACC
✓ Advanced MPI
  ○ MPI: overlapping communications vs. computation
  ○ MPI3 one sided communications
✓ Giacomo Falcucci (univ. Tor Vergata)
✓ Sauro Succi (IIT)
✓ Vittorio Ruggiero (Cineca - Roma)
✓ Isabella Baccarelli (Cineca - Roma)
✓ Ingolf Stark & Fujitsu staff
5. www.top500.org
Thanks for your attention!