

Using OpenMP on a Hydrodynamic Lattice-Boltzmann Code

Salvatore Filippone — Computing Center, Università di Roma Tor Vergata,
Nicola Rossi — Università di Roma Tor Vergata,
Gino Bella — Dept. of Mech. Engineering, Università di Roma Tor Vergata,
Stefano Ubertini — Dept. of Mech. Engineering, Università di Roma Tor
Vergata

salvatore.filippone@uniroma2.it

Using OpenMP on a Hydrodynamic Lattice-Boltzmann Code

...or “How we fixed(?) a(n almost) real world code in a few weeks (??)” and lessons learned

Using OpenMP on a Hydrodynamic Lattice-Boltzmann Code

...or “How we fixed(?) a(n almost) real world code in a few weeks (??)” and lessons learned

- Work in progress...

Using OpenMP on a Hydrodynamic Lattice-Boltzmann Code

...or “How we fixed(?) a(n almost) real world code in a few weeks (??)” and lessons learned

- Work in progress...
- Multidisciplinary approach, or: how to get mechanical engineers to exploit their machines

Using OpenMP on a Hydrodynamic Lattice-Boltzmann Code

...or “How we fixed(?) a(n almost) real world code in a few weeks (??)” and lessons learned

- Work in progress...
- Multidisciplinary approach, or: how to get mechanical engineers to exploit their machines
- Using the best general purpose software analysis tool: the grad student

Using OpenMP on a Hydrodynamic Lattice-Boltzmann Code

...or “How we fixed(?) a(n almost) real world code in a few weeks (??)” and lessons learned

- Work in progress...
- Multidisciplinary approach, or: how to get mechanical engineers to exploit their machines
- Using the best general purpose software analysis tool: the grad student

...Platform: Silicon Graphics Origin3200, 6 CPUs MIPS R12000 at 400 MHz, 8 GB main memory, 8MB cache, IRIX 64 Version 6.5.

Lattice-Boltzmann Model

Solving fluid dynamics through a microscopic kinetic approach

$$\partial_t f + \xi \cdot \nabla f = \Omega$$

The collision operator Ω_i

$$\Omega_i = -\frac{1}{\tau} (f_i - f_i^{eq})$$

The discrete equation:

$$f_i(\vec{x} + \vec{c}_i \cdot \Delta t, t + \Delta t) - f_i(\vec{x}, t) = \frac{\Delta t}{\tau} [(f_i(\vec{x}, t) - f_i^{eq})]$$

Lattice Boltzmann equation with Bhatnagar-Gross-Krook (BGK) approximation

Coding issues

2D discretization lattice \Rightarrow discretization matrix.

Algorithm structure:

Streaming Implementation of the formula

$$f_i(\vec{x} + \vec{c}_i, t + 1) = f_i(\vec{x}, t)$$

Collision Correction of distribution function as in:

$$f_i(\vec{x}, t + 1) = f_i(\vec{x}, t + 1) - \omega(f_i(\vec{x}, t + 1) - f_i^{eq}(\vec{x}, t + 1))$$

Profile data

The initial profile for a free-channel simulation:

% time	cumulative seconds	self seconds	Sub-routine
71.62	271.14	271.14	hydrovar_equili_collis
25.03	365.91	94.77	move
2.05	373.66	7.75	diag0d
0.51	375.59	1.93	mbc
0.32	376.79	1.20	rec_f
0.28	377.85	1.06	error
0.19	377.86	0.71	other

Profile data

The initial profile with a cylindrical obstacle in a channel (with mesh refinement);

% time	cumulative seconds	self seconds	Sub-routine
68.78	940.50	940.50	hydrovar_equili_collis
22.58	1249.30	308.80	move
2.04	1277.26	27.96	obst_bf
1.91	1303.43	26.17	interp
1.68	1326.39	22.96	rescale_f
1.35	1344.90	18.51	diag0d
1.09	1359.74	14.84	rec_f
0.34	1364.36	0.71	mbc

Streaming phase initial implementation

```
do j = nyf,nyi,-1
  do i = nxi,nxf
    if (ntype(i,j).eq.0) then
      f(2,i,j) = f(2,i,j-1)
      f(6,i,j) = f(6,i+1,j-1)
    endif
  enddo
enddo
```

```
do j = nyf,nyi,-1
  do i = nxf,nxi,-1
    if (ntype(i,j).eq.0) then
      f(1,i,j) = f(1,i-1,j)
      f(5,i,j) = f(5,i-1,j-1)
    endif
  enddo
enddo
```

```
do j = nyi,nyf
  do i = nxf,nxi,-1
    if (ntype(i,j).eq.0) then
      f(4,i,j) = f(4,i,j+1)
      f(8,i,j) = f(8,i-1,j+1)
    endif
  enddo
enddo
```

```
do j = nyi,nyf
  do i = nxi,nxf
    if (ntype(i,j).eq.0) then
      f(3,i,j) = f(3,i+1,j)
      f(7,i,j) = f(7,i+1,j+1)
    endif
  enddo
enddo
```

Collision operator

$$f(0, i, j) = f(0, i, j) * (1.0d0 - \omega) + \omega * feq(0, i, j)$$

$$f(1, i, j) = f(1, i, j) * (1.0d0 - \omega) + \omega * feq(1, i, j)$$

$$f(2, i, j) = f(2, i, j) * (1.0d0 - \omega) + \omega * feq(2, i, j)$$

$$f(3, i, j) = f(3, i, j) * (1.0d0 - \omega) + \omega * feq(3, i, j)$$

$$f(4, i, j) = f(4, i, j) * (1.0d0 - \omega) + \omega * feq(4, i, j)$$

$$f(5, i, j) = f(5, i, j) * (1.0d0 - \omega) + \omega * feq(5, i, j)$$

$$f(6, i, j) = f(6, i, j) * (1.0d0 - \omega) + \omega * feq(6, i, j)$$

$$f(7, i, j) = f(7, i, j) * (1.0d0 - \omega) + \omega * feq(7, i, j)$$

$$f(8, i, j) = f(8, i, j) * (1.0d0 - \omega) + \omega * feq(8, i, j)$$

The two matrices idea

```
do j=nyi,nyf
  do i=nxi,nxf
    if (ntype(i,j) .eq. 0)then
      f_odd(1,i,j) = f_even(1,i-1,j)
      f_odd(2,i,j) = f_even(2,i,j-1)
      f_odd(3,i,j) = f_even(3,i+1,j)
      f_odd(4,i,j) = f_even(4,i,j+1)
      f_odd(5,i,j) = f_even(5,i-1,j-1)
      f_odd(6,i,j) = f_even(6,i+1,j-1)
      f_odd(7,i,j) = f_even(7,i+1,j+1)
      f_odd(8,i,j) = f_even(8,i-1,j+1)
    end if
  enddo
enddo
```

Serial speedup (15–20%)!

Code structure

```
do istep = 1,nsteps

    if (mod(istep,2).eq.1) then
!$OMP PARALLEL

c ----- static & collide -----
    call step_complete (ff0_A,ff0_B,feqf0,rhof0,uf0,vf0,1,nx,1,ny,
.      nx,ny,nodetype_f0,omegaf0,vett_resolv,n_resolv)
c ----- Boundary conditions -----
    call mbc(ff0_B,rhof0,uf0,vf0,1,nx,1,ny,nx,ny)
c ----- Obstacle -----
    call obst_bf (ff0_B,feqf0,omegaf0,rhof0,uf0,vf0,nodetype_f0,
.      delta_f0,0,nx+1,0,ny+1,nx,ny,vett_obst,n_obst)

!$OMP END PARALLEL
    else
```

Code structure

```
!$OMP PARALLEL
c ----- static & collide -----
    call step_complete(ff0_B,ff0_A,feqf0,rhof0,uf0,vf0,1,nx,1,ny,
.      nx,ny,nodetype_f0,omegaf0,vett_resolv,n_resolv)
c ----- Boundary conditions -----
    call mbc(ff0_A,rhof0,uf0,vf0,1,nx,1,ny,nx,ny)
c ----- Obstacle -----

    call obst_bf (ff0_A,feqf0,omegaf0,rhof0,uf0,vf0,nodetype_f0,
.      delta_f0,0,nx+1,0,ny+1,nx,ny,vett_obst,n_obst)
!$OMP END PARALLEL
endif
```

Speedup Results

Run with no obstacle, schedule STATIC

CPU	Time (s)	Speedup	Efficiency
1	268.0		
2	143.4	1.87	0.94
3	99.83	2.68	0.89
4	81.1	3.3	0.83
5	75.92	3.53	0.71
6	51.66	5.19	0.87

Speedup Results

Run with no obstacle, schedule DYNAMIC

CPUs	Time (s)	Speedup	Efficiency
1	268.0		
2	138	1.94	0.97
3	102	2.63	0.88
4	85.1	3.15	0.79
5	70.67	3.79	0.76
6	65.8	4.07	0.68

Not exactly earth-shattering!

A better schedule

Create a new data structure recording the type of mesh node and the coordinate:

```
do i=1,nx
  do j=1,ny
    if (nodetype(i,j).eq.0) then
      Add_node_(i,j)_to_type_0_structure
    else if (nodetype(i,j).eq.1) then
      Add_node_(i,j)_to_type_1_structure
    endif
  enddo
enddo
```

A better schedule

```
do ij=1,n_resolv
  i=v_resolv(0,ij)
  j=v_resolv(1,ij)
  Work_with_node(i,j)
enddo
```

```
do ij=1,n_obst
  i=v_obst(0,ij)
  j=v_obst(1,ij)
  Work_with_obstacle_node(i,j)
enddo
```

A better schedule

```
!$OMP DO SCHEDULE(STATIC,DIM_WORD_SCHD)
!$OMP.PRIVATE(ij,i,j,usq,vsq,sumsq,sumsq2,u2,v2,ui,vi,uv)
  do ij=1,num_resolv
    i=v_resolv(0,ij)
    j=v_resolv(1,ij)
c----- Move
    f(0,i,j) = f2(0,i,j)
    f(1,i,j) = f2(1,i-1,j)
    ...
c ----- Density -----
    rho(i,j)=f(0,i,j)+f(1,i,j)+f(2,i,j)+f(3,i,j)+f(4,i,j)+
    .           f(5,i,j)+f(6,i,j)+f(7,i,j)+f(8,i,j)
c ----- Velocity -----
u(i,j)=(f(1,i,j)-f(3,i,j)+f(5,i,j)-f(6,i,j)-f(7,i,j)+
    .           f(8,i,j))/rho(i,j)
    v(i,j)=(f(5,i,j)+f(2,i,j)+f(6,i,j)-f(7,i,j)-f(4,i,j)-
    .           f(8,i,j))/rho(i,j)
c ----- equilibrium
    feq(0,i,j)=c4d9*rho(i,j)*(1.0d0-sumsq)
    feq(1,i,j)=rho(i,j)*c1d9*(1.0d0-sumsq+u2+ui)
    ...
c----- collision
    f(0,i,j)=f(0,i,j)+omega*(feq(0,i,j)-f(0,i,j))
    f(1,i,j)=f(1,i,j)+omega*(feq(1,i,j)-f(1,i,j))
    .....
```

New results

Just a simple STATIC schedule suffices.

CPUs	Time (s)	Speedup	Efficiency
1	23.02		
2	15.82	1.455	0.7
3	11.8	1.951	0.65
4	9.93	2.318	0.58
5	8.37	2.75	0.55
6	9.09	2.532	0.42

Run with domain 50×40 , no obstacle

New results

CPU's	Time (s)	Speedup	Efficiency
1	188.89		
2	103.67	1.822	0.9
3	73.15	2.582	0.86
4	52.56	3.594	0.90
5	40.29	4.688	0.94
6	31.94	5.914	0.99

Run with domain 500×400 , no obstacle

New results

CPU's	Time (s)	Speedup	Efficiency
1	78.7		
2	40	1.9675	0.98
3	27	2.9148	0.97
4	20.05	3.9252	0.98
5	16.74	4.7013	0.94
6	16.42	4.7929	0.80

Run with domain 200×200 , obstacle $R = 24.9$

New results

CPUs	Time (s)	Speedup	Efficiency
1	177		
2	103.3	1.7135	0.86
3	67.8	2.6106	0.87
4	50.3	3.5189	0.88
5	36.26	4.8814	0.98
6	29.56	5.9878	0.99

Run with domain 500×400 , obstacle $R = 24.9$

Lessons learned

- No silver bullet in OpenMP vs. MPI: No matter what the paradigm, you better understand your architecture/application match

Lessons learned

- No silver bullet in OpenMP vs. MPI: No matter what the paradigm, you better understand your architecture/application match
- Fight with compiler, fight with scheduler (both machine and human)...

Lessons learned

- No silver bullet in OpenMP vs. MPI: No matter what the paradigm, you better understand your architecture/application match
- Fight with compiler, fight with scheduler (both machine and human)...
- Hardest part: transfer the knowledge to: the practitioners? the compilers?

Lessons learned

- No silver bullet in OpenMP vs. MPI: No matter what the paradigm, you better understand your architecture/application match
- Fight with compiler, fight with scheduler (both machine and human)...
- Hardest part: transfer the knowledge to: the practitioners? the compilers?
- ... value for us CS guys ?

Future work

- Why increasing efficiency?

Future work

- Why increasing efficiency?
- Is DYNAMIC working the way it should ? (probably not). We don't have a complete performance model yet....

Future work

- Why increasing efficiency?
- Is DYNAMIC working the way it should ? (probably not). We don't have a complete performance model yet....
- What happens on Linux (INTONE and other compilers)?

Future work

- Why increasing efficiency?
- Is DYNAMIC working the way it should ? (probably not). We don't have a complete performance model yet....
- What happens on Linux (INTONE and other compilers)?
- Special version for very small cases?

Future work

- Why increasing efficiency?
- Is DYNAMIC working the way it should ? (probably not). We don't have a complete performance model yet....
- What happens on Linux (INTONE and other compilers)?
- Special version for very small cases?
- 3D version: memory space problems, adapt Federico's approach?