High Performance Computing

Node-level performance of the lattice Boltzmann method on recent multicore CPUs

Gerhard Wellein[a,b], J. Habich[b] , G. Hager[b] , T. Zeiser[b]
[a]Department for Computer Science
[b]Erlangen Regional Computing Center
Friedrich-Alexander-University Erlangen-Nuremberg

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

ParCFD 2011 – Special Session " "

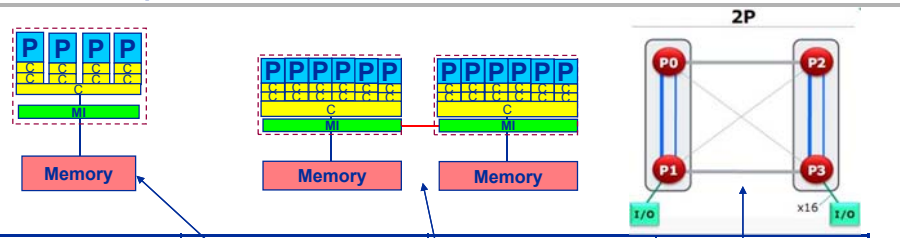
1. State of the art CPU nodes
2. Setting up a simple performance model for the basic LBM step
3. Performance capabilities of modern CPU based compute nodes

High Performance Computing

1. State of the art CPU nodes

Modern CPU compute nodes Hardware specifications

High Performance Computing



	Intel Core i5 – 2500	Intel X5650 DP	AMD Opteron
Cores@Clock	4 @ 3.3 GHz	2 x 6@2.66 GHz	2 x 8@2.3 GHz
Performance+/core	26.4 GFlop/s	10.6 GFlop/s	9.2 GFlop/s
L3 size	1 x 8 MB	2 x 12 MB	4 x 6 MB
Total performance SP/DP	105 / 210 GFlop/s	128 / 256 GFlop/s	147 / 294 GFlop/s
Stream BW	17 GB/s	41 GB/s	51 GB/s
	*"Sandy Bridge" (**SNB**)*	*"Westmere"*	*"Magny Cours"*

+ DOUBLE Precision (DP)

Trading single thread performance for parallelism
Hardware capabilities: GPGPUs vs. CPUs

High Performance Computing

GPU vs. CPU light speed estimate:

1. **Compute bound: 4-5 X**
2. **Memory Bandwidth: 2-5 X**

	Intel Core i5 – 2500 ("Sandy Bridge")	Intel X5650 DP node ("Westmere")	NVIDIA C2070 ("Fermi")
Cores@Clock	4 @ 3.3 GHz	2 x 6@2.66 GHz	448 @ 1.1 GHz
Performance+/core	52.8 GFlop/s	21.3 GFlop/s	2.2 GFlop/s
Threads@stream	4	12	8000 +
Total performance+	210 GFlop/s	255 GFlop/s	1,000 GFlop/s
Stream BW	17 GB/s	41 GB/s	90 GB/s (ECC=1)
Transistors / TDP	1 Billion* / 95 W	1.17 Billion / 95 W	3 Billion / 238 W

+ *Single Precision*

* *Includes on-chip GPU and PCI-Express*

High Performance Computing

2. Setting up a simple performance model for the basic LBM step

Estimating optimal performance (minimum runtime) must be the first step when optimizing codes!

Performance Modeling – Our approach

High Performance Computing

1. **Analyze the minimum computational requirements (data volume, FLOP-ops) of the algorithm**
2. **Analyze the computational requirements (data access in cache/main memory, FLOPS, instruction mix,..) of the implementation. Optimize if they do not fit to data from 1.**
3. **Analyze the available computational resources of the target hardware: Cache/Memory bandwidth, SIMD capabilities,..**
4. **Determine max. performance (min. runtime) based on 2 and 3.**
5. **Measure performance and compare with 4. Go back to 2. / 3. if numbers differ substantially**

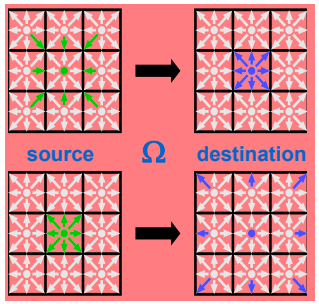
Main memory access analysis / optimization
Lattice Boltzmann method: Method / Implementation

High Performance Computing

collision step: $\tilde{f}_\alpha(x_i,t) = f_\alpha(x_i,t) - \omega\,[f_\alpha(x_i,t) - f_\alpha^{(eq)}(x_i,t)]$

streaming step: $f_\alpha(x_i + \vec{e}_\alpha \delta t, t+\delta t) = \tilde{f}_\alpha(x_i,t)$

Performing collision and streaming separately doubles main memory transfer! → Do it in a single step / loop



Stream-Collide (Pull-Method)

Get the distributions from the neighboring cells in the source array and store the relaxated values to one cell in the destination array

Collide-Stream (Push-Method)

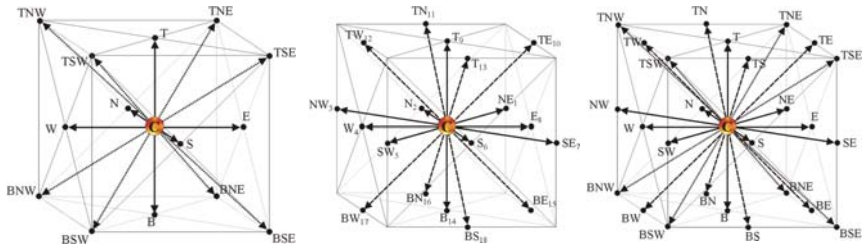
Take the distributions from one cell in the source array and store the relaxed values to the neighboring cells in the destination array

Writing data is more expensive than reading → Choose Stream-Collide in what follows

Main memory access analysis / optimization
Lattice Boltzmann method: Method

High Performance Computing

- **Different discretization schemes in 3D**
  - Numerical accuracy and stability
  - Computational speed and simplicity
  - Schemes differ in data volume and computational complexity



D3Q15 D3Q19 D3Q27

We choose the widely used D3Q19 discretization
→ For each node (I,J,K) 19 values (Q) must be stored

Main memory access analysis / optimization
LBM: A basic implementation

High Performance Computing

```
double precision F(0:18,0:iMax+1,0:jMax+1,0:kMax+1,0:1)
do k=1,kMax
   do j=1,jMax
      do i=1,iMax
         if( fluidcell(i,j,k) ) then
            LOAD F( 0,i  ,j  ,k  ,t)
            LOAD F( 1,i+1,j+1,k  ,t)
            LOAD F( 2,i  ,j+1,k  ,t)
            LOAD F( 3,i-1,j+1,k  ,t)
            ...
            LOAD F(18,i  ,j-1,k-1,t)
            Relaxation (complex computations)
            STORE F(0:18,i,j,k,t+1)
         endif
      enddo
   enddo
enddo
```

Data layout F(Q , I , J , K)

19 Cachelines

Collide Step

~200 FLOPs

Stream Step

LD 1-2 Cachelines (cont. access)

If cache line of store operation is not in cache it must be loaded first ("write allocate")!

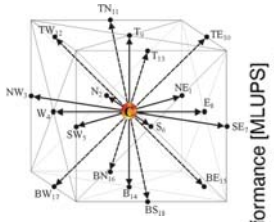
#load from main memory: 19*iMax*jMax*kMax + 19*iMax*jMax*kMax

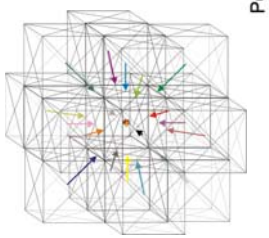
#store to main memory: 19*iMax*jMax*kMax

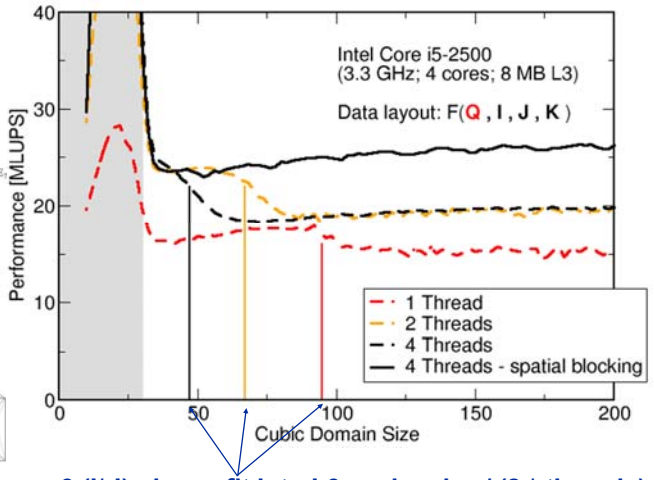
Main memory access analysis / optimization
LBM: Performance measurement

High Performance Computing

D3Q19 stencil







3 (I*J) planes fit into L3 cache size / (2 * threads)

Main memory access analysis / optimization
LBM: Alternative data layout

High Performance Computing

```
double precision F(0:iMax+1,0:jMax+1,0:kMax+1, 0:18,0:1)
do k=1,kMax
   do j=1,jMax
      do i=1,iMax
         if( fluidcell(i,j,k) ) then
            LOAD F( i  ,j  ,k  , 0,t)
            LOAD F( i+1,j+1,k  , 1,t)
            ...
            LOAD F(i  ,j-1,k-1 ,18,t)
            Relaxation (complex computations)
            STORE F(i,j,k, 0, t+1)
            STORE F(i,j,k, 1, t+1)
            ...
            STORE F(i,j,k,18, t+1)
         endif
      enddo
   enddo
enddo
```

Data layout F(I , J , K, Q)

Collide Step

Stream Step

A min. of 38 cache lines (~2.5 KB) must be held in cache
We further assume the use of this layout

LBM implementation analysis
Performance model: data transfer (1)

High Performance Computing

- **MFLOP/s is not a good performance measure for LBM**
  - Depends on the programmer 150,…,500 FLOP / fluid cell update
  - Depends on the compiler and hardware
- **Good measure for performance: Inverse runtime for a well defined problem to be solved**
- **LBM performance measure: Million fluid cell updates per second**

$$\textit{MFLUP/s:} \quad \frac{\text{sweeps}*\text{iMax}*\text{jMax}*\text{kMax}}{10^6 * \text{Time}_{\text{sweeps}}}$$

$\text{Time}_{\text{sweeps}}$: **Wallclock time to perform *sweeps* LBM iterations**

LBM implementation analysis
Performance model: data transfer (2)

High Performance Computing

- **Assume: Data set is too large to fit into cache + Double Precision**
- **Data transfer between main memory and CPU for a single fluid cell:**
  (19 + 19 + 19) * 8 Byte = 456 Byte
  [Without Write Allocate: 304 Byte]
- **Determine achievable main memory bandwidth with kernel benchmark (e.g. STREAM):** Mem_BW [MByte/s]
- **Performance estimate (BW):** $\frac{\text{Mem\_BW [MByte/s]}}{\text{456 Byte/FLUP}}$
- **Example (SNB):** Mem_BW = 17,000 MByte/s → **~37 MFLUP/s**
- **Very simple approach**:
  - Ignoring intra cache data transfer
  - 19 Concurrent READ and 1 WRITE streams (STREAM: 1 READ; 1 STORE)
  - Perfect prefetching assumed; no associativity conflicts in cache

LBM implementation analysis
Performance model: Floating point operations

High Performance Computing

- **Assume 200 FP operations per cell update**
- **Performance estimate (FP):** $\frac{\text{FP\_Performance [MFLOP/s]}}{\text{200 FLOP/FLUP}}$
- **Example (SNB):** FP_performance = 100,000 MFLOP/s → **500 MFLUP/s**
- **Here we assume:**
  - No overhead from main memory transfer
  - No complex FP instructions
  - Evenly balanced MULT – ADD instruction mix
  - No other instruction mix bottlenecks (LDT / ST,…)
  - Fully vectorized SIMD kernel
- **But if kernel is not vectorized: Performance shrinks by 8x – SNB implements AVX instruction set!**

LBM implementation analysis
Performance model: Putting it all together

High Performance Computing

- **Determine maximum performance for a single Sandy Bridge (SNB) CPU (4-cores) with**
  - **Mem_BW = 17,000 MByte/s**
  - **FP_performance = 105,000 MFLOP/s (double precision)**
  - **FP_performance = 210,000 MFLOP/s (single precision)**

	Mem_BW	Mem_BW (No Write allocate)	FP_performance (optim. SIMD)	FP_performance (scalar FP)
Double precision	**37 MFLUP/s**	**56 MFLUP/s**	**500 MFLUP/s**	**62 MFLUP/s**
Single precision	**74 MFLUP/s**	**112 MFLUP/s**	**1,000 MFLUPs**	**62 MFLUP/s**

- **SIMD instructions are a must at least for SP kernels for SNB!**
- **Performance estimates are upper qualitative boundaries**
- **Single socket numbers, i.e. 4-cores**

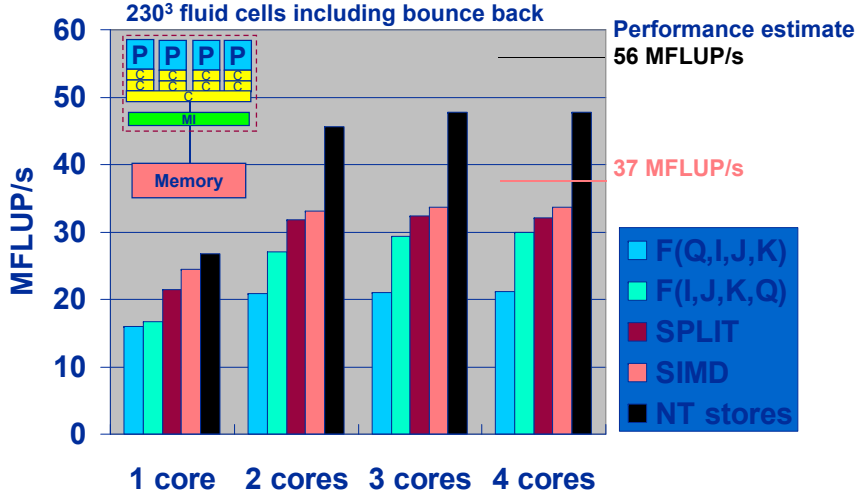
High Performance Computing

3. Performance capabilities of modern CPU based compute nodes

Performance capabilities
Intel Sandy Bridge (SNB): 4-core desktop CPU

High Performance Computing

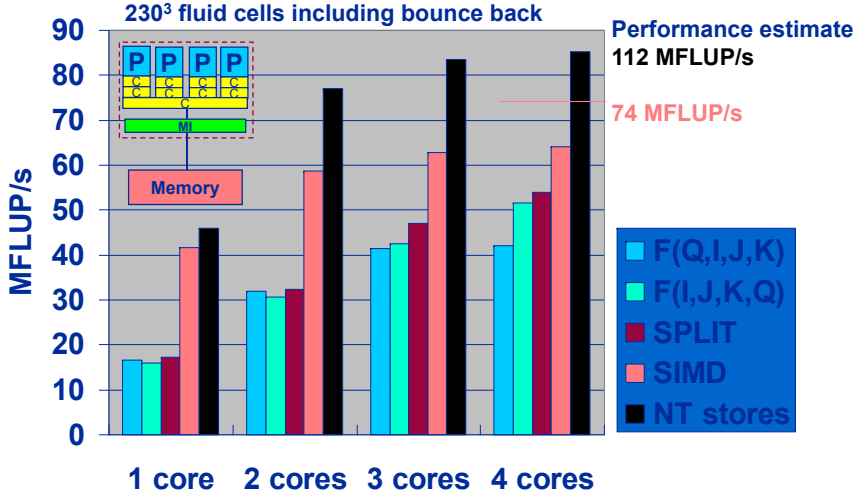
Double precision (DP) computations for lid driven cavity with 230^3 fluid cells including bounce back



Performance capabilities
Intel Sandy Bridge (SNB): 4-core desktop CPU

High Performance Computing

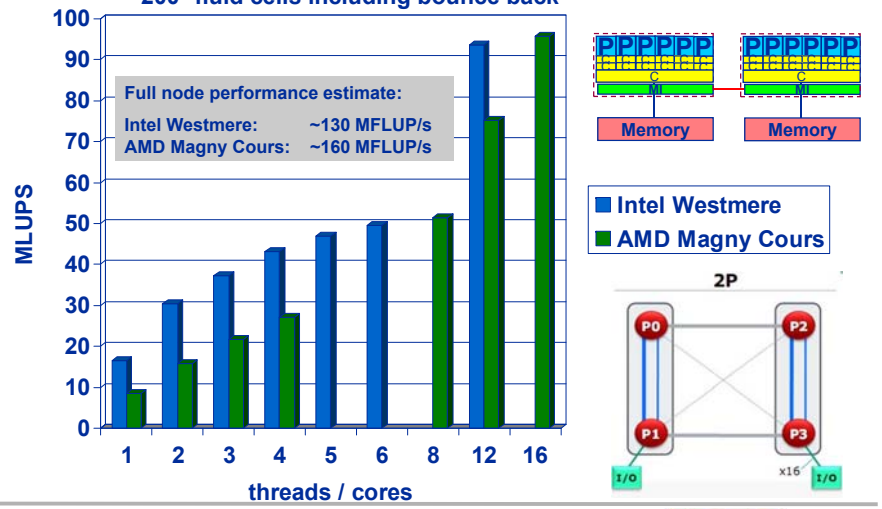
Single precision (SP) computations for lid driven cavity with 230^3 fluid cells including bounce back



Performance capabilities
2-way compute nodes: Intel vs. AMD

High Performance Computing

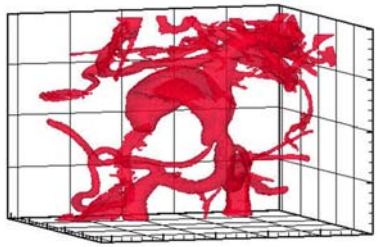
Double precision (DP) computations for lid driven cavity with 200^3 fluid cells including bounce back

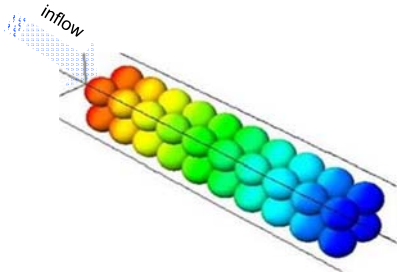


Main memory access analysis / optimization What about complex geometries?

High Performance Computing

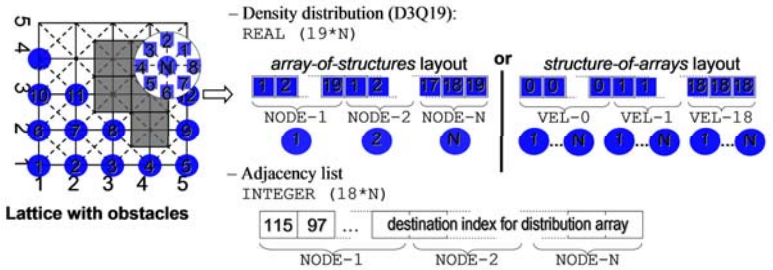
- Simple “matrices” and a marker-and-cell approach to block solid parts are a natural choice
- loops over the Cartesian coordinate can be fused if necessary to achieve long loops
- neighboring cells are known through simple index shift operations (-1/0/+1)
- but …
- large overhead (in terms of CPU and memory) if major parts of the domain are solid!



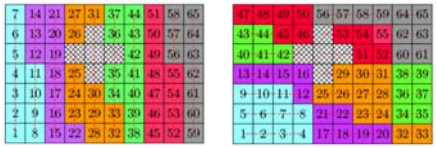


Alternative structure of lattice Boltzmann solver for complex geometries: Fluid Cells only

High Performance Computing



- Sort of “sparse” data structure
- explicit adjacency information required
- → **indirect addressing either during “read” or “write” (gather or scatter)**
- arbitrary order of the nodes within the list
- → implicit blocking possible



Lattice Boltzmann solver for flows in complex geometries: Towards optimal performance

High Performance Computing

Main memory bandwidth is bottleneck

Optimal Performance model (D3Q19):
→ **Loads: 19x8=152 bytes**
+ 9x4= 36 bytes
Stores: 19x8=152 bytes

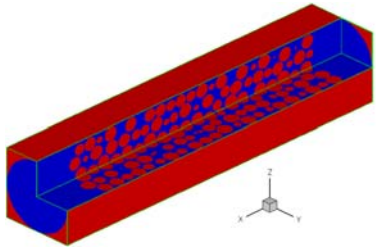
→ **340 Byte/LUP**

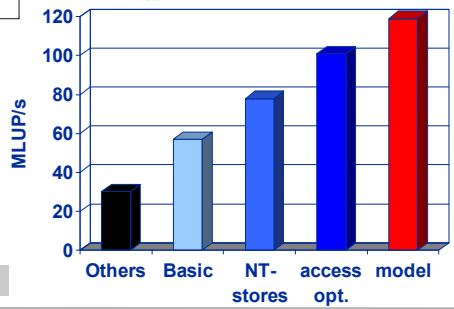
→ **1 GB/s is equiv. to 2.9 MLUP/s**

Million Lattice Site Updates per second (MFLUP/s): LBM specific performance metric

For a single Westmere node:
41 GB/s → 119 MFLUP/s

Cf. MS-3 talk of Markus Wittmann: Tue. 15:30





High Performance Computing

Acknowledgements

RRZE group
J. Treibig
M. Wittmann
waLBerla group
C. Feichtinger,…

Funding received trough

Bundesministerium für Bildung und Forschung
www.skalb.de
SKALB (01 IH08003A)

http://www.rrze.uni-erlangen.de/hpc/

References

- **G. Wellein, T. Zeiser, G. Hager, and S. Donath, Computer & Fluids 35, 910 (2006).** *On the Single Processor Performance of Simple Lattice Boltzmann Kernels*
- **T. Zeiser, G. Hager and G. Wellein. Parallel Processing Letters 19 (4), 491-511 (2009) DOI:10.1142/S0129626409000389** *Benchmark analysis and application results for lattice Boltzmann simulations on NEC SX vector and Intel Nehalem systems*

KONWIHR II
OMI4papps
KONWIHR-II projects
www.konwihr.uni-erlangen.de