Cache Optimization for the Lattice Boltzmann Method in 3D

Computational Engineering Bachelorthesis
Author: Klaus Iglberger

Contents

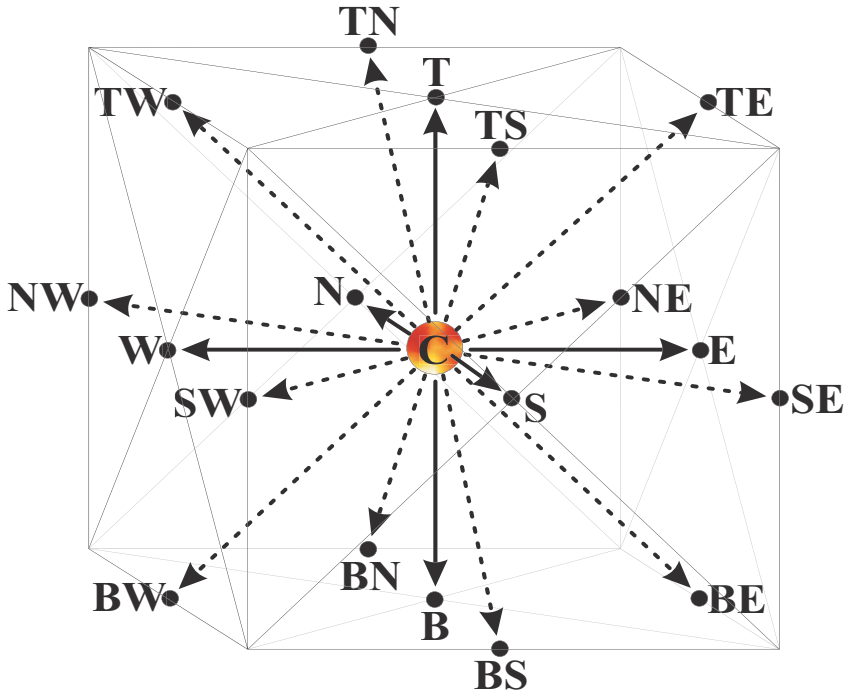
The Lattice Boltzmann Method

Cell-based computational fluid dynamic (CFD) simulation method on a uniform cartesian grid

D3Q19 Model for the 3D case:

19 directions/distribution functions f_α

- **faster than more directions**
- **enough accuracy**
- **less instability than fewer directions**



Memory Layout for one cell (Cell order approach):

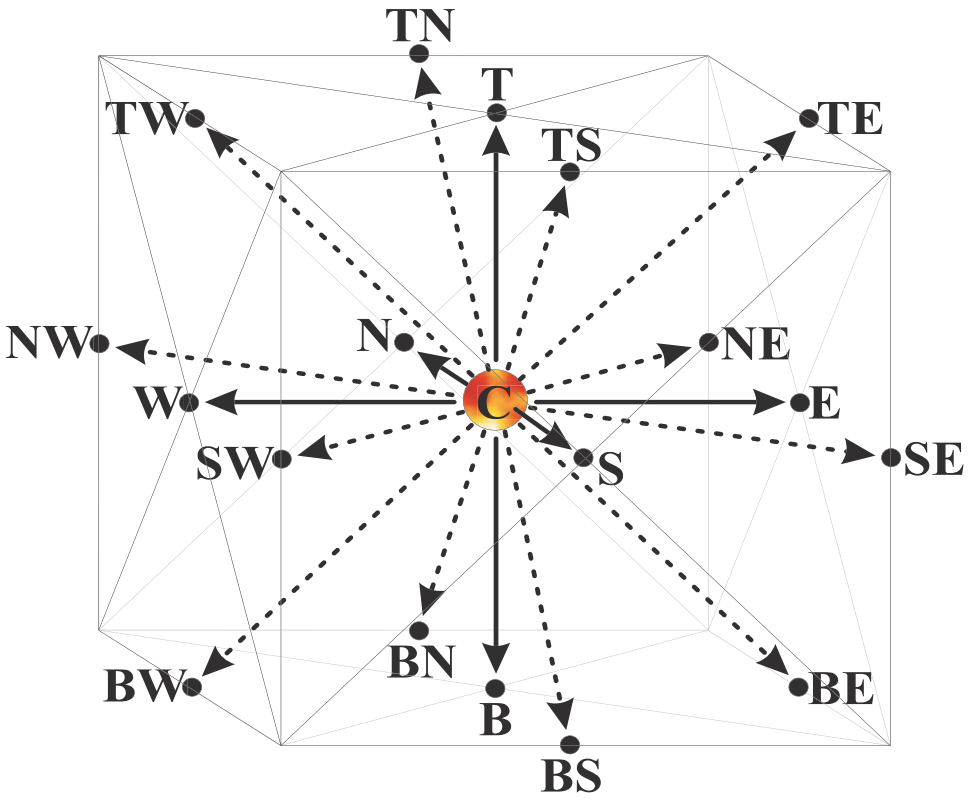
The distribution functions describe the behavior of the fluid:

Density:

$$\rho = \sum_{\alpha = 0}^{18} f_\alpha$$

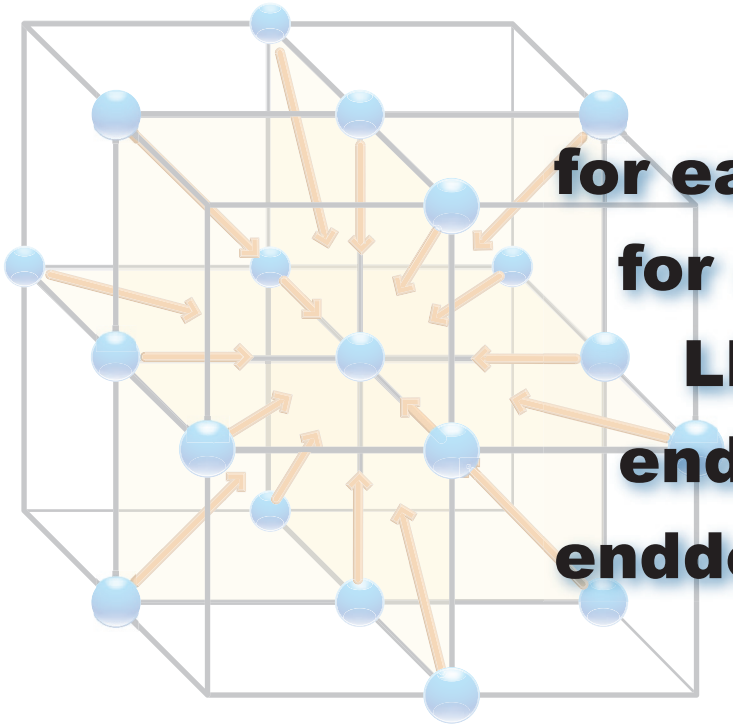
Momentum flux:

$$\rho u = \sum_{\alpha = 0}^{18} e_\alpha f_\alpha$$

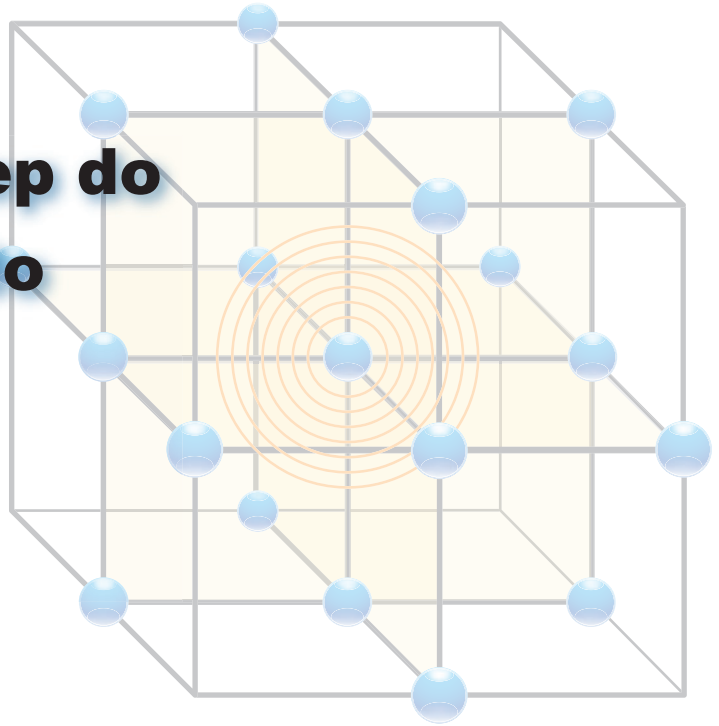


D3Q19 model for the 3D case:
19 distribution functions per cell

Basic Algorithm



```
for each timestep do
  for each cell do
    LBMSTEP
  enddo
enddo
```

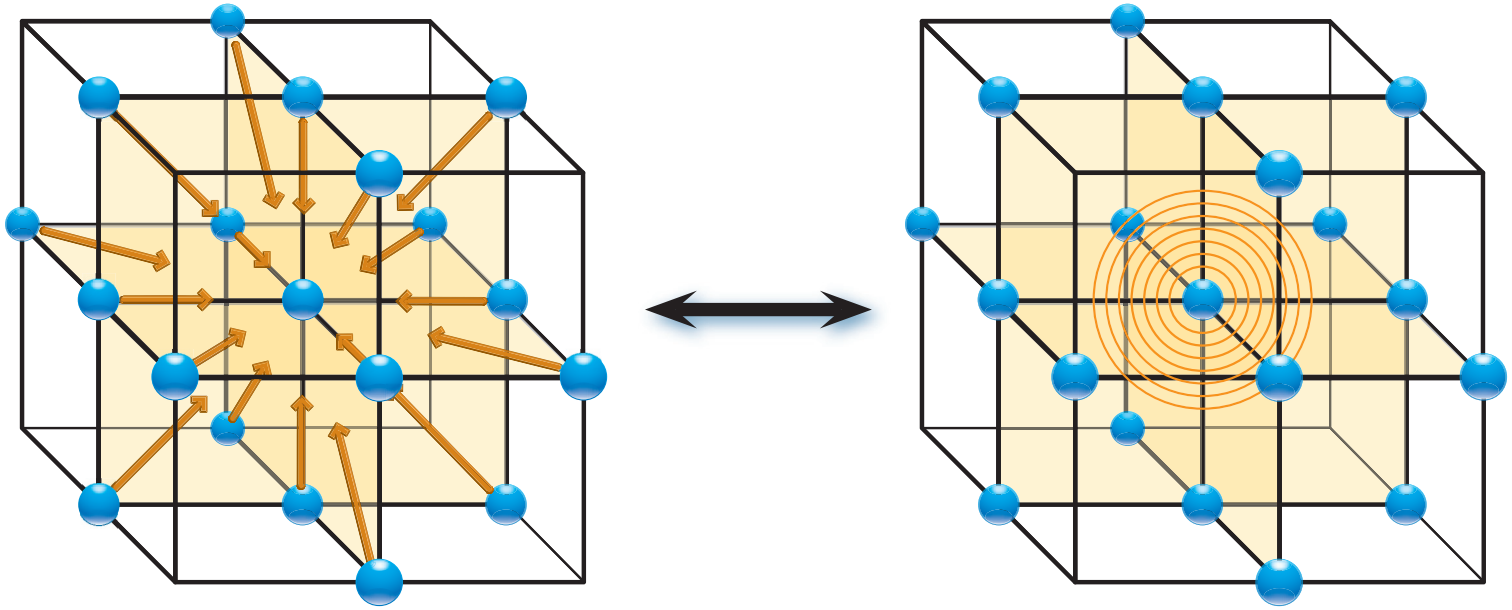


Stream Step: Moves fluid particles from one cell to another according to their velocity

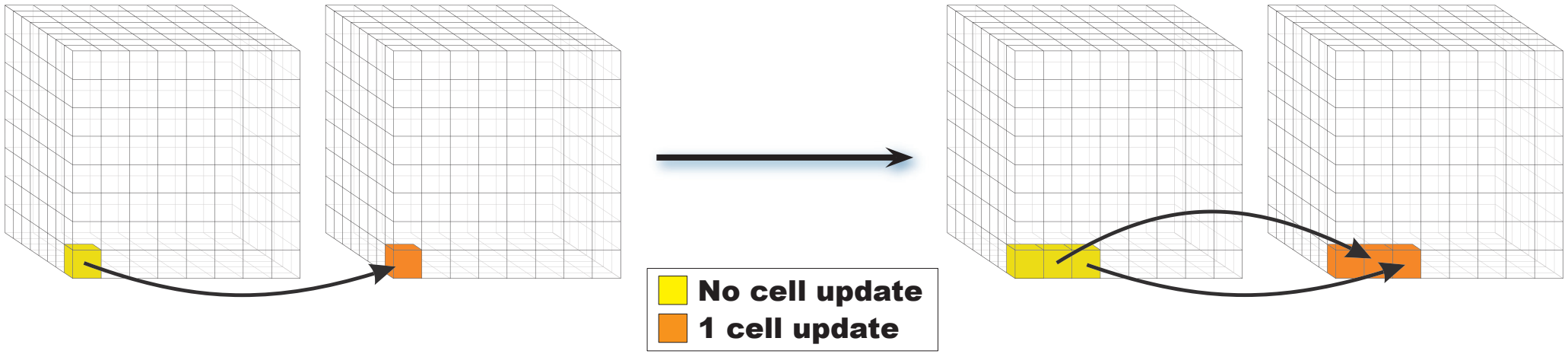
Collide Step: Models various interactions among fluid particles and calculates new distribution functions

Stream Step: Distributed reading/writing from/to other cache lines

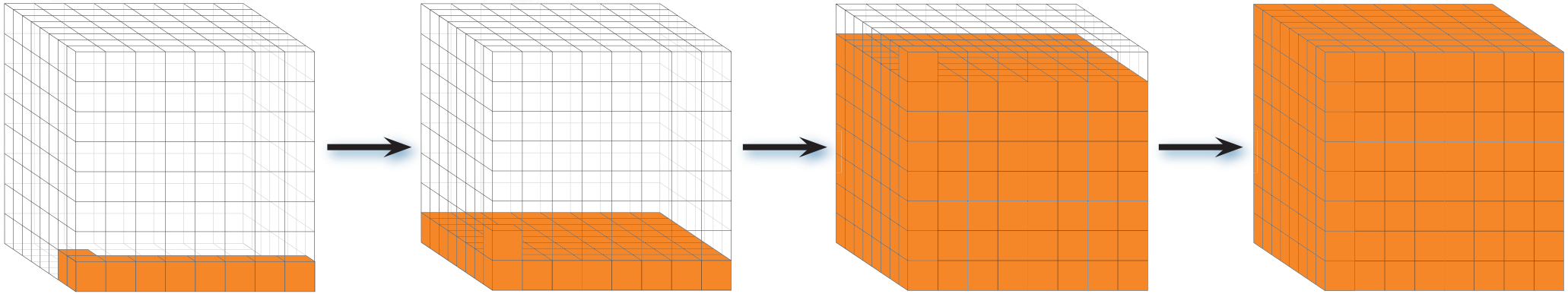
- → **Strong memory dependency**
- → **Switching the Stream Step and the Collide Step to change the read/write behavior**
- → **Cache Optimizations**



Basic Implementation with 2 Grids



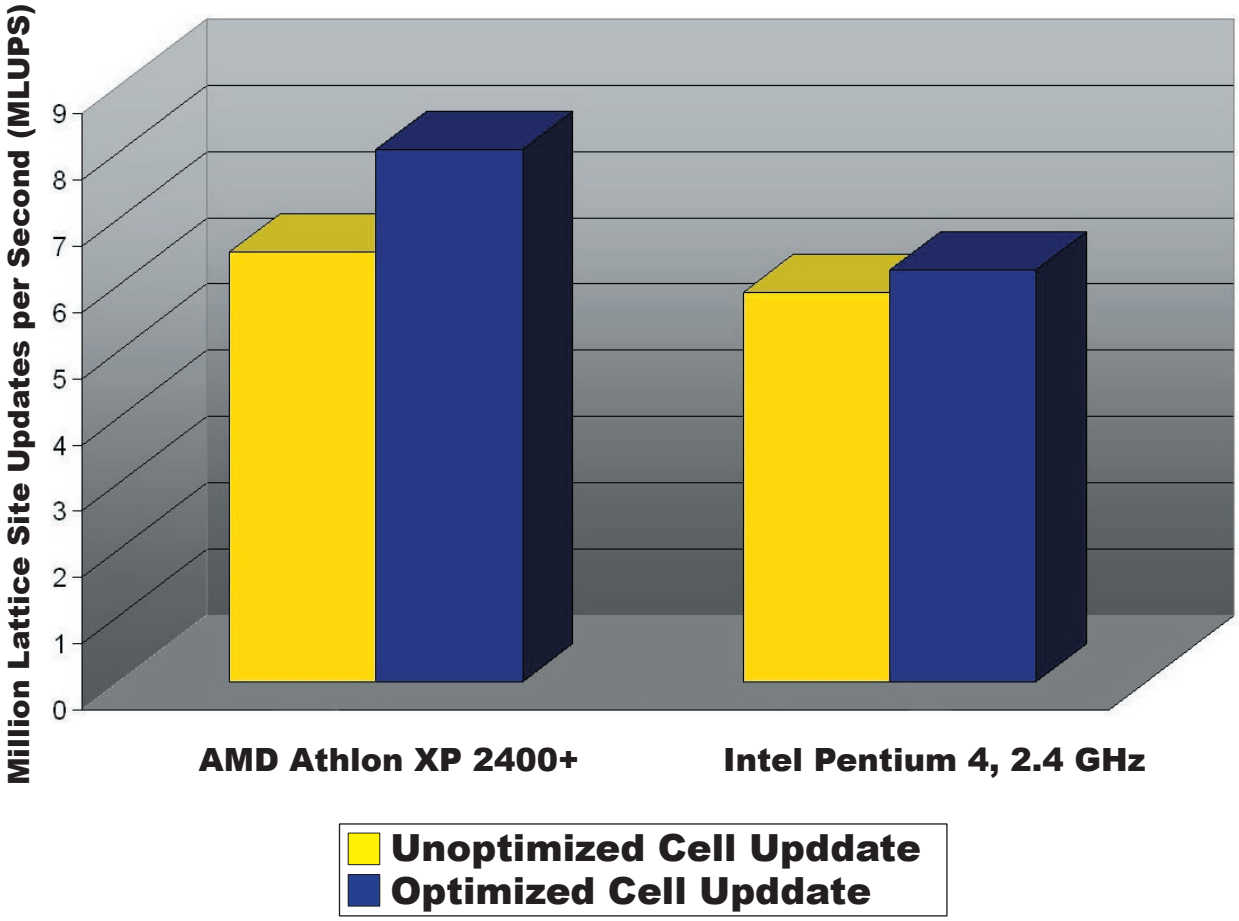
Update steps in the 2nd grid:



Memory Requirement for 2 Grids with 100^3 cells: 339.5 MByte

Arithmetic Optimizations

Arithmetic optimizations only concern one cell update



- Choosing the best data access scheme for data cells
  - 4 index array
  - hand-linearised 1 index array
- Fusion of stream and collide step
- Reduction of function call overhead
- Reduction of floating point operations
- Precalculation of intermediate values
- Loop Unrolling

Cache Optimizations

Increase of performance by considering the underlying memory hierarchy

Goals:

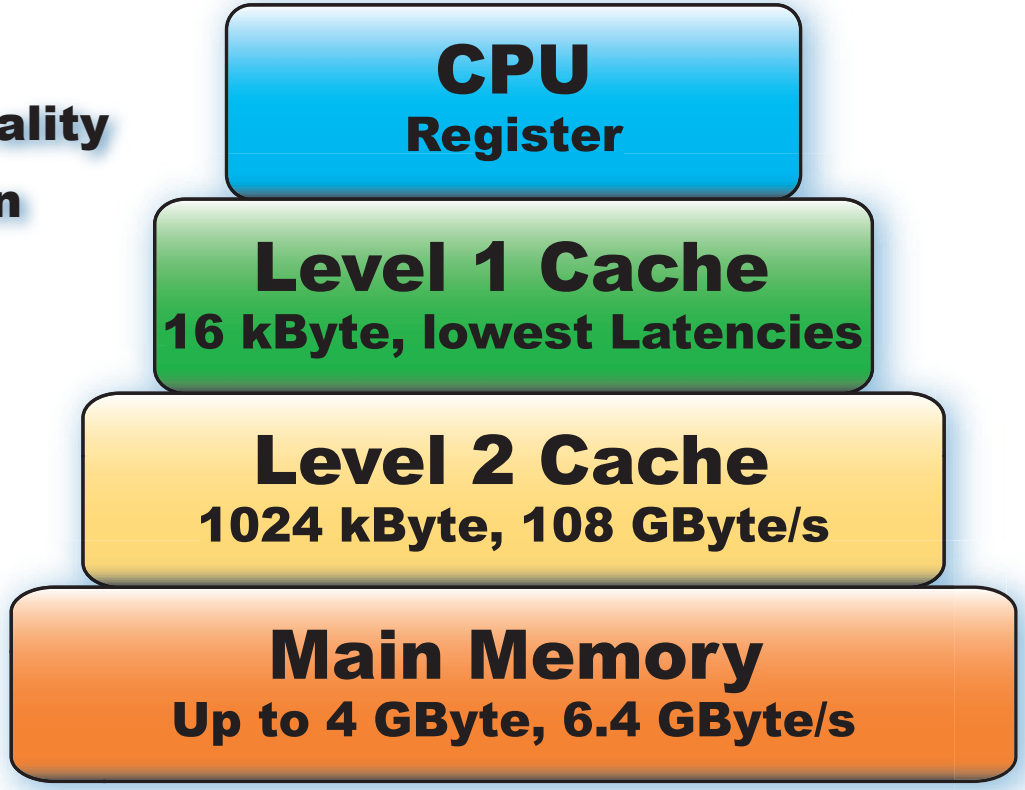
- **Increase of spatial and temporal locality**
- **Reduction of memory traffic between CPU and Main Memory**

Improvements required for:

- **Data Layout**
- **Cell Access Pattern**

Independent from each other

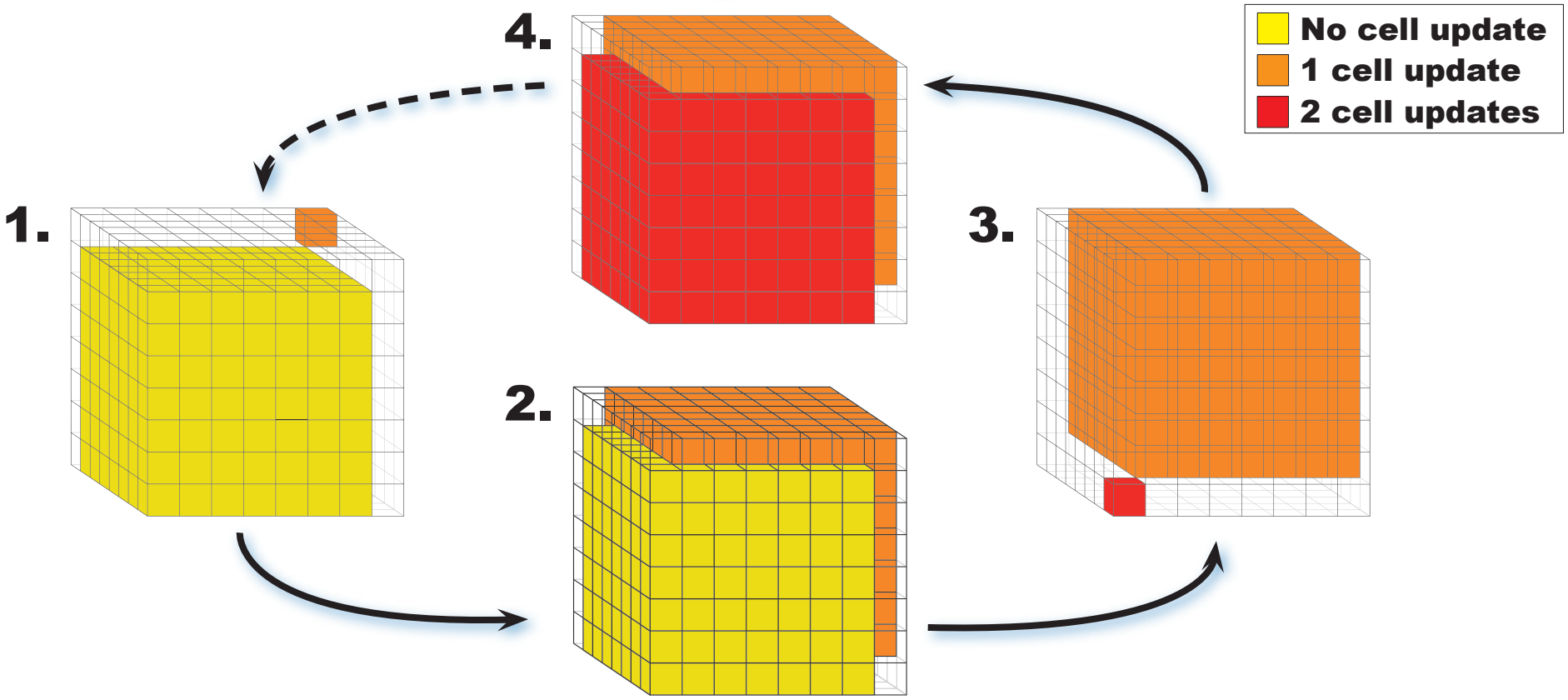
→ "LBM Toolbox"



Pentium 4 Prescott, 3.4 GHz

Grid Compression

- Reduction of memory requirements to nearly 50%
- Efficient reduction of memory traffic

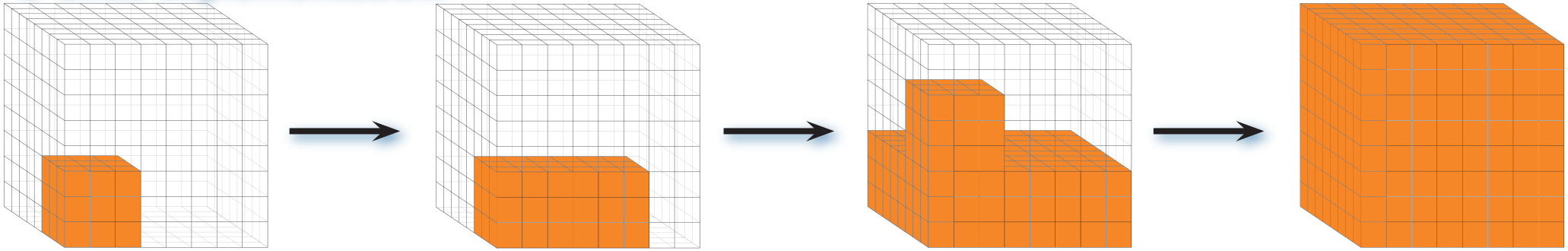


3-way Blocking

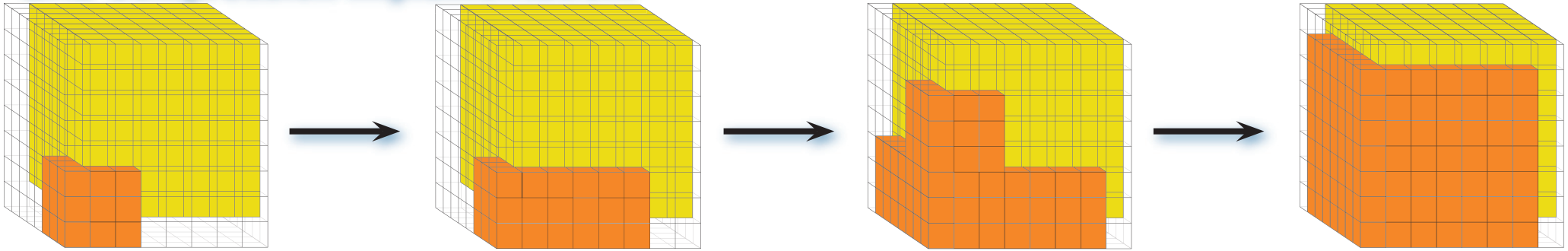
- Increase of the spatial locality
- Simple implementation
- Nearly always applicable

No cell update
1 cell update

2 Grids implementation:

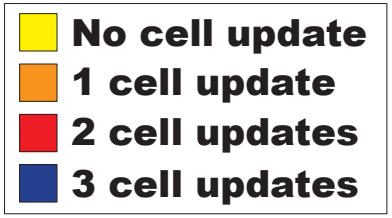


Grid Compression implementation:

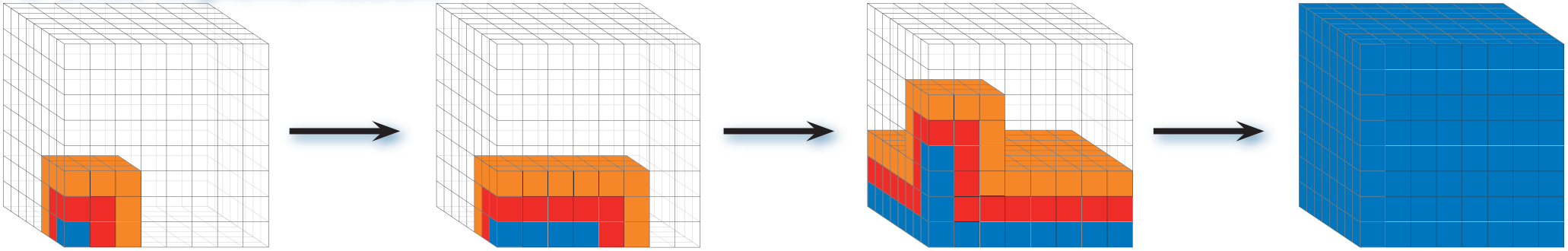


4-way Blocking

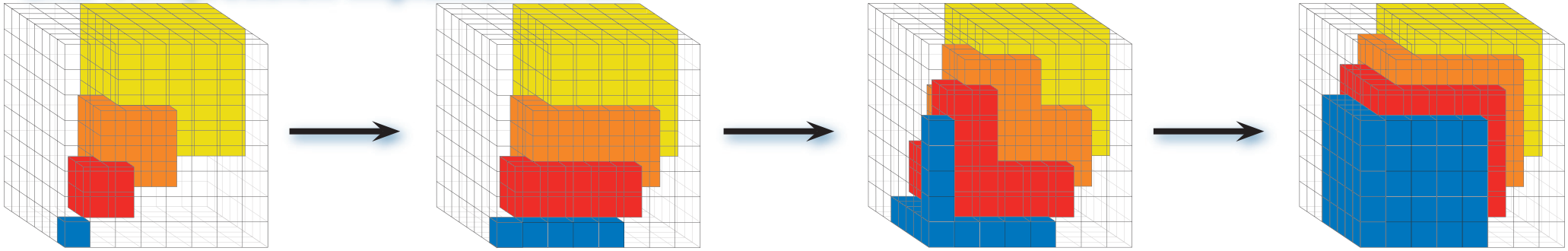
- Additional increase in temporal locality
- Promises even more performance than 3-way blocking
- Not always applicable



2 Grids implementation:



Grid Compression implementation:

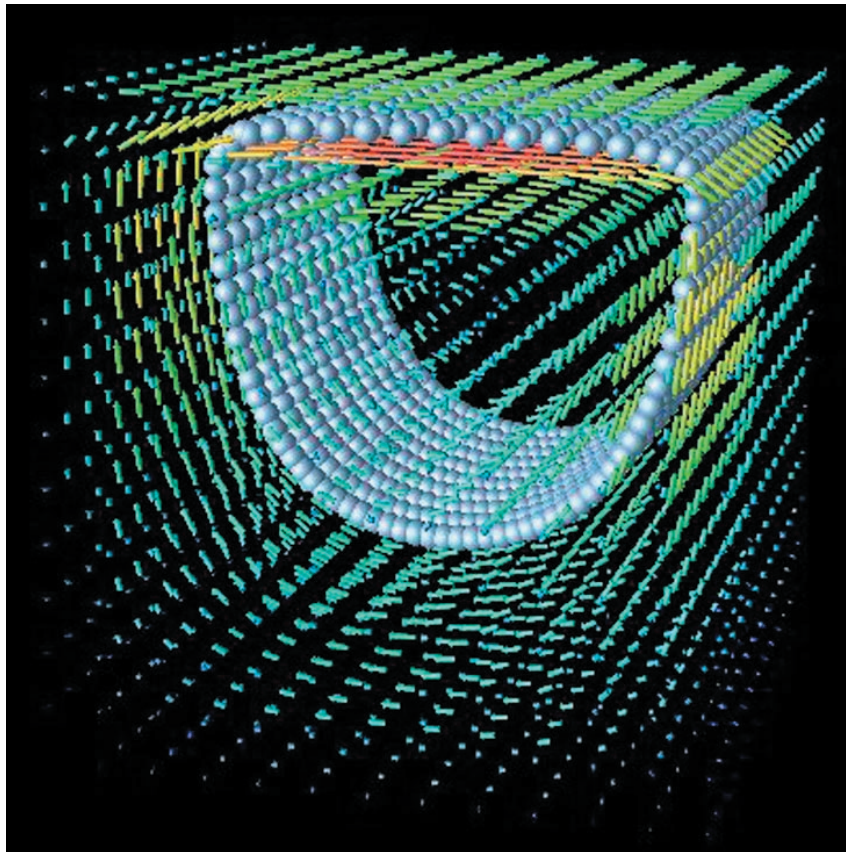


Results

Test Problem: Lid Driven Cavity

- Well known numerical test case
- Acceleration cells at the top of the domain
- Use of the C programming language

→ Very fast implementation possible



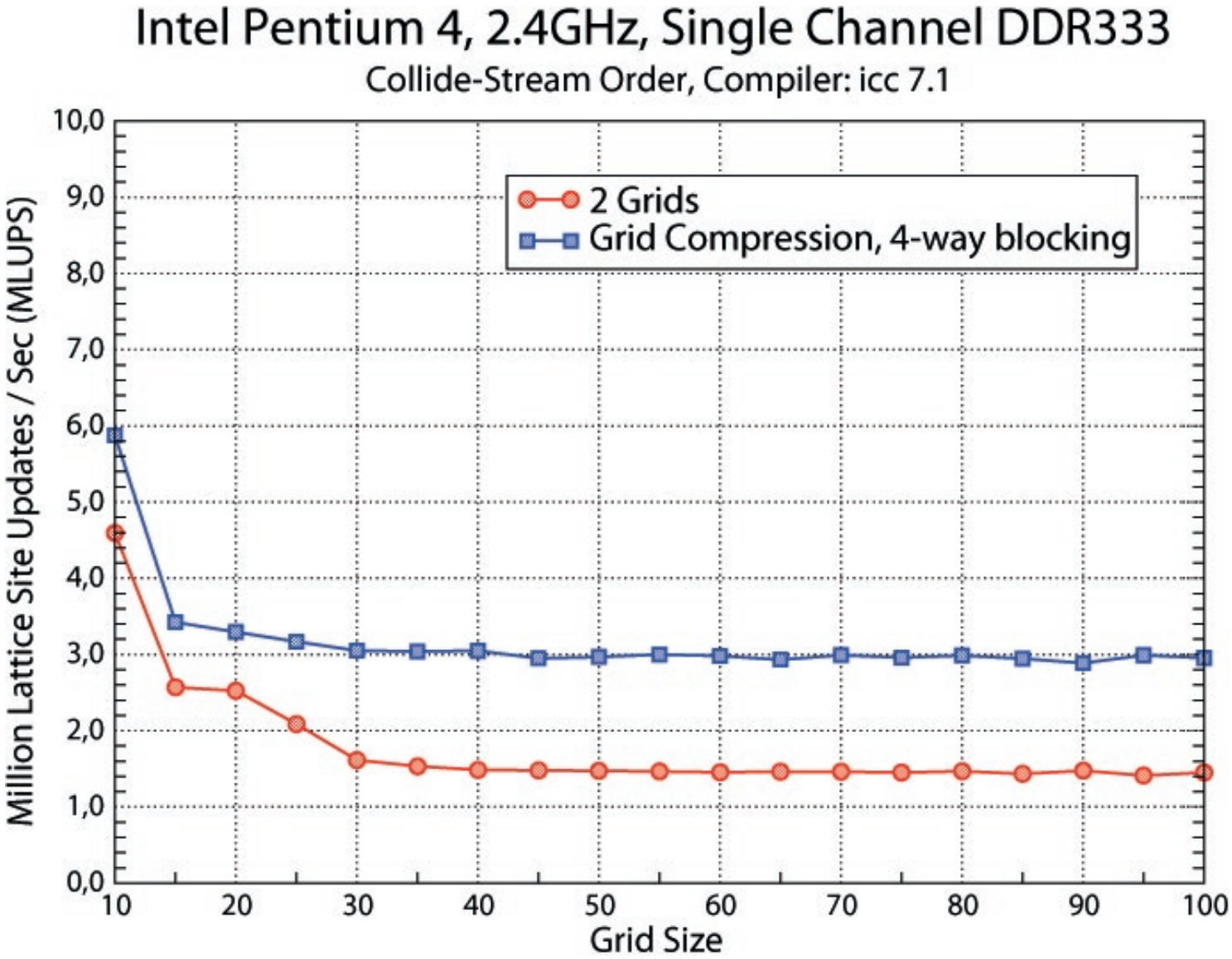
Test Platforms:

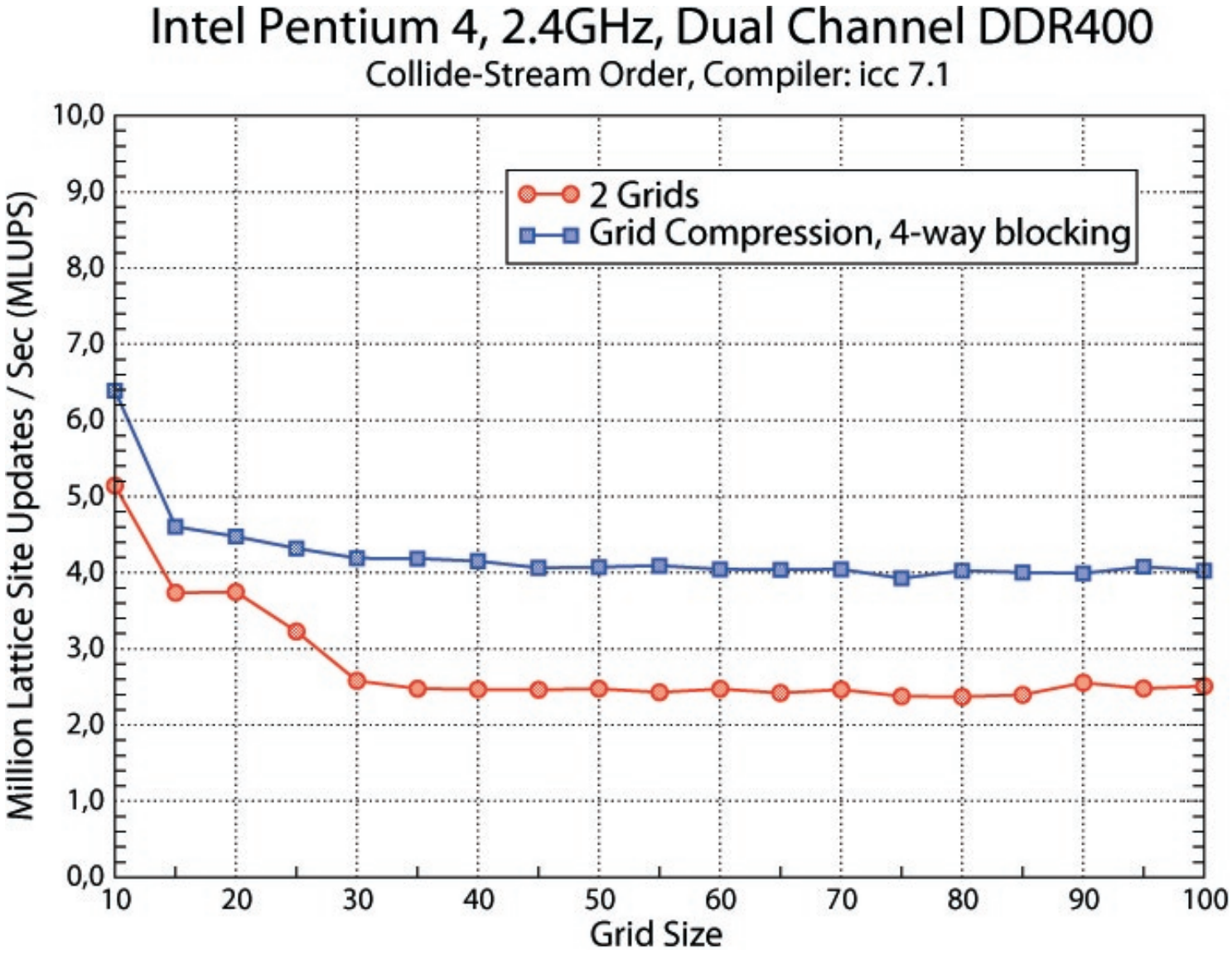
- Alpha 21164
- AMD Athlon K75, 700 MHz
- AMD Athlon XP 2400+
- Intel Pentium 4, 2.4 GHz, Single Channel DDR333
- Intel Pentium 4, 2.4 GHz, Dual Channel DDR400
- AMD Opteron, 1.6 GHz
- Intel Itanium 2, 900 MHz

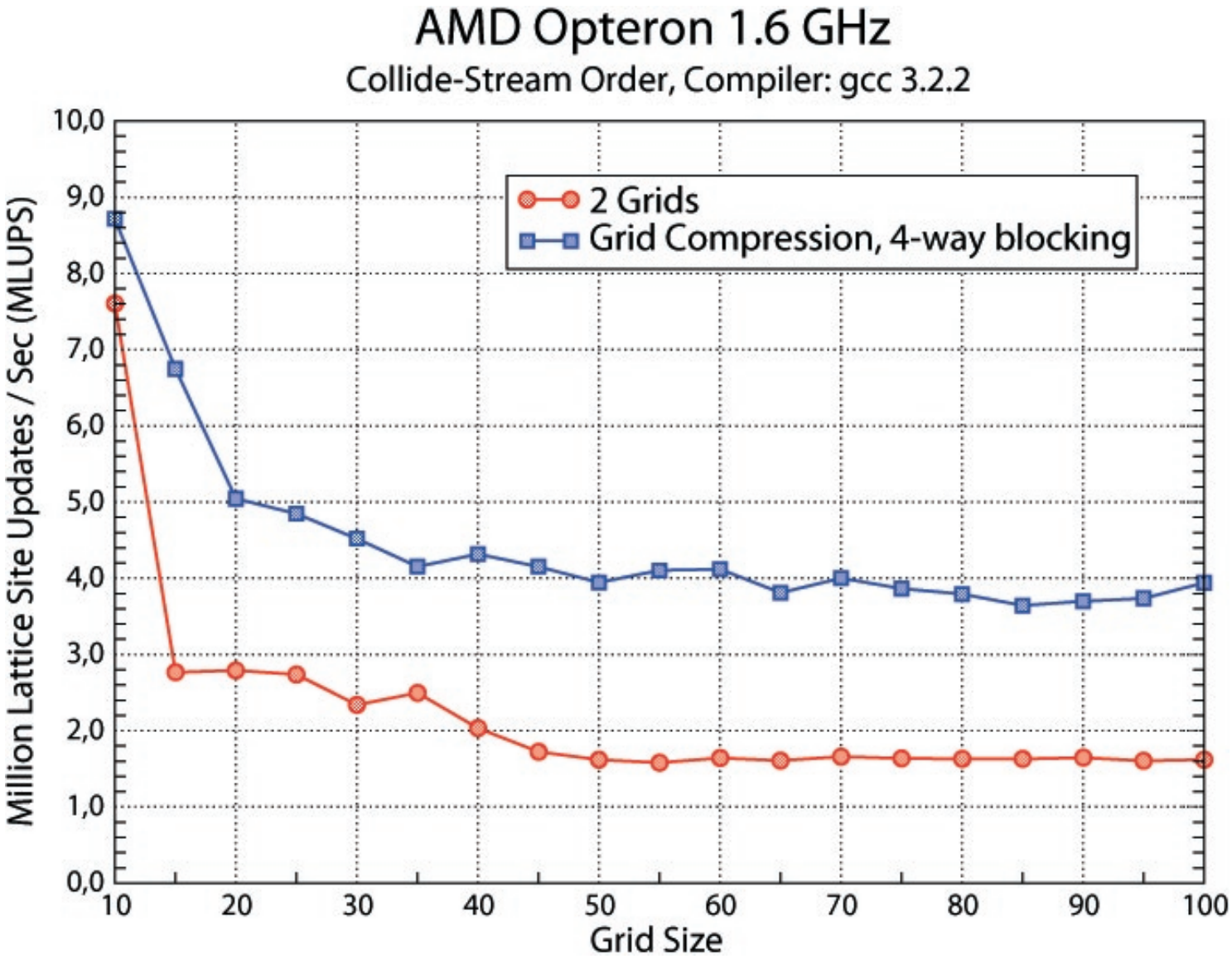
Performance Unit:

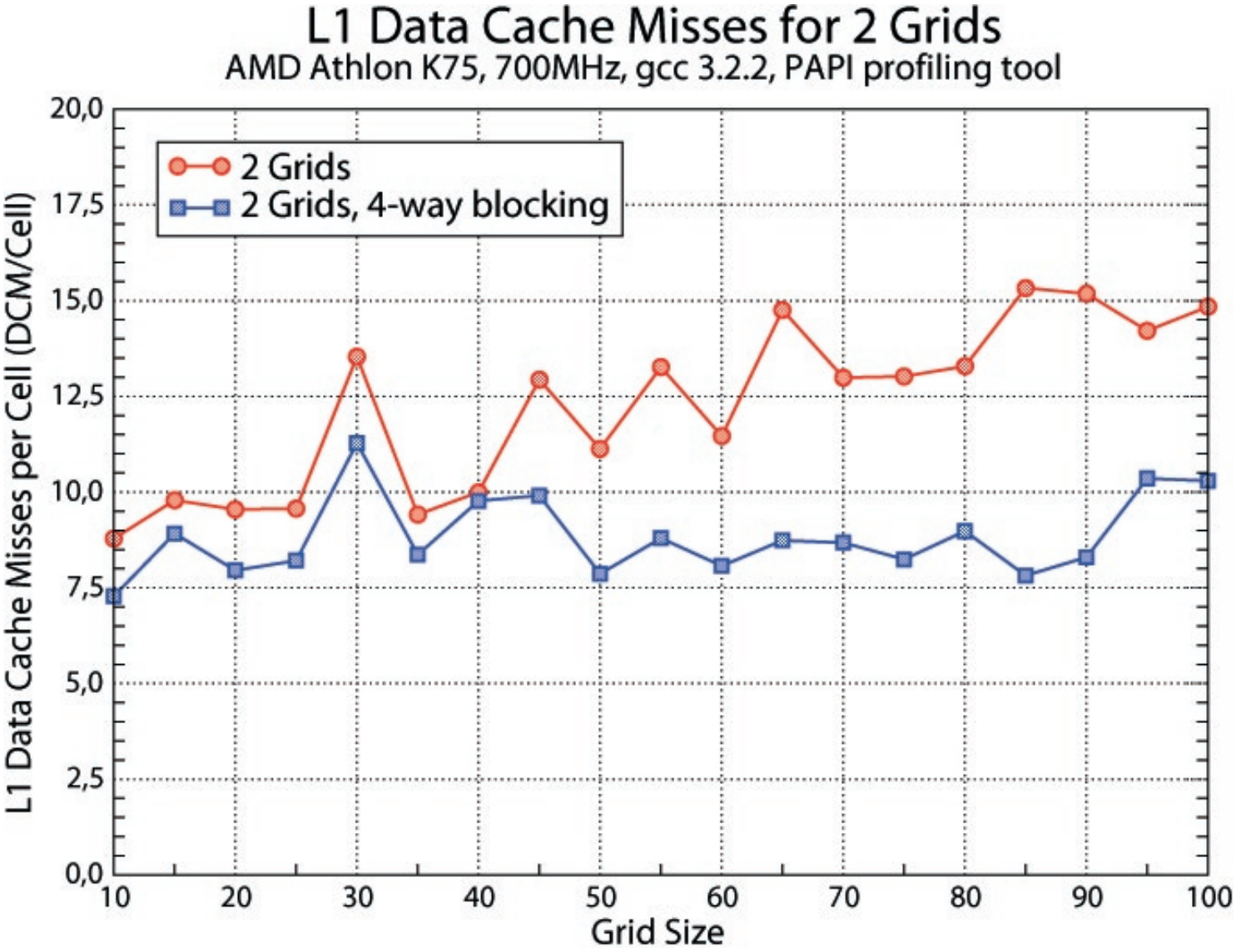
MLUPS = Million Lattice Site Updates per Second

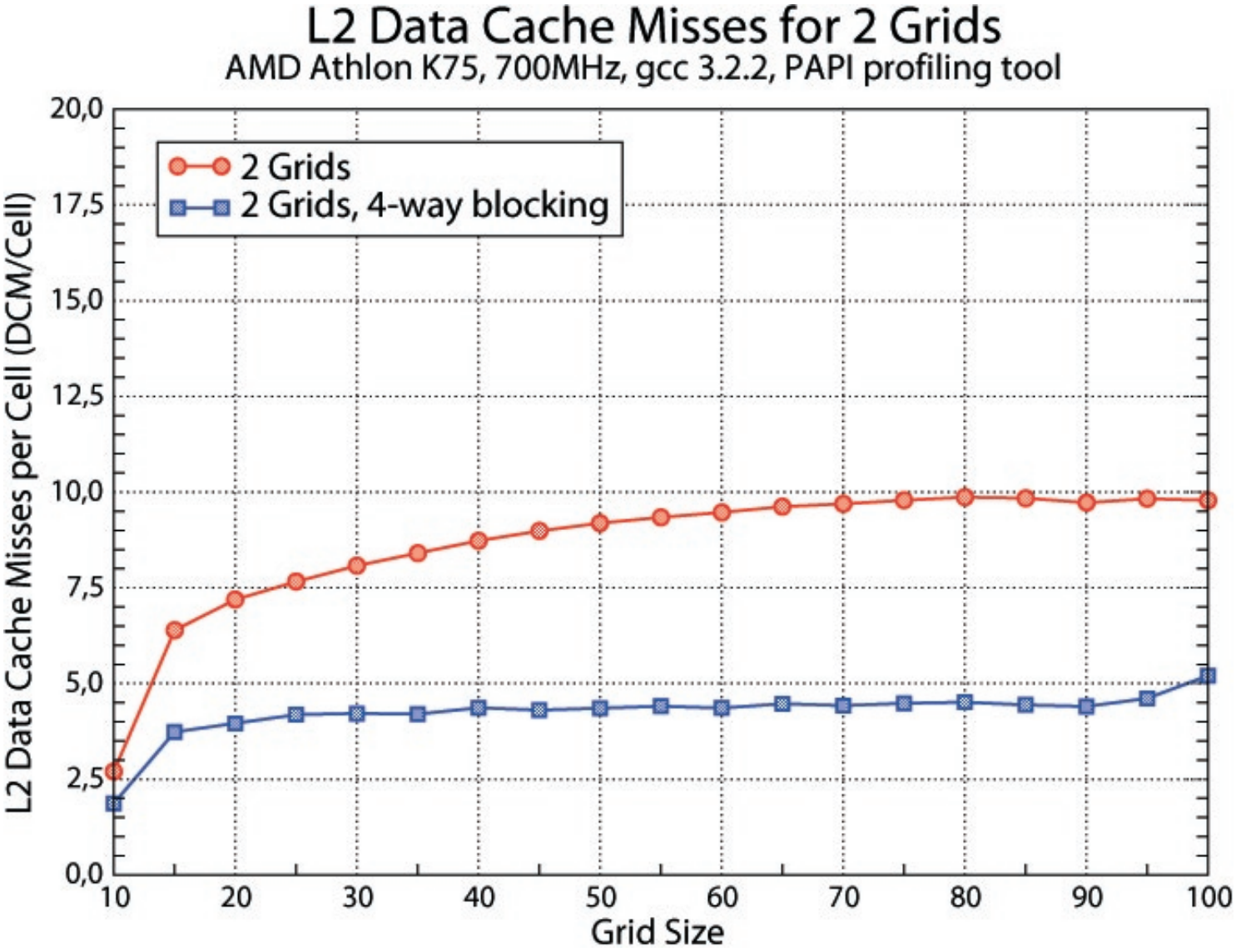
1 MLUPS = 273 MFLOPS

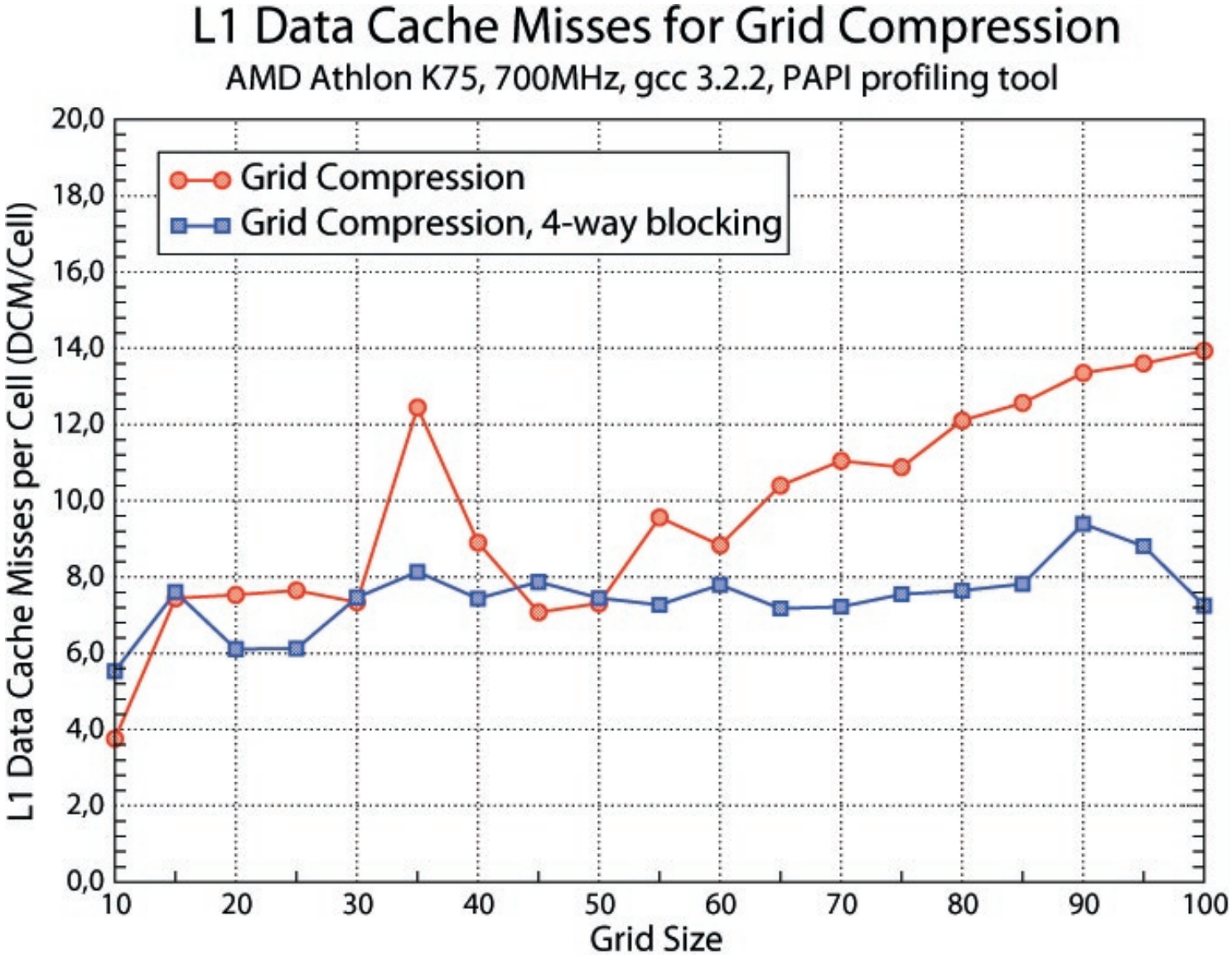

Intel Pentium 4, 2.4GHz, Single Channel DDR333
Collide-Stream Order, Compiler: icc 7.1
2 Grids
Grid Compression, 4-way blocking
Million Lattice Site Updates / Sec (MLUPS)
Grid Size
10,0
9,0
8,0
7,0
6,0
5,0
4,0
3,0
2,0
1,0
0,0
10
20
30
40
50
60
70
80
90
100

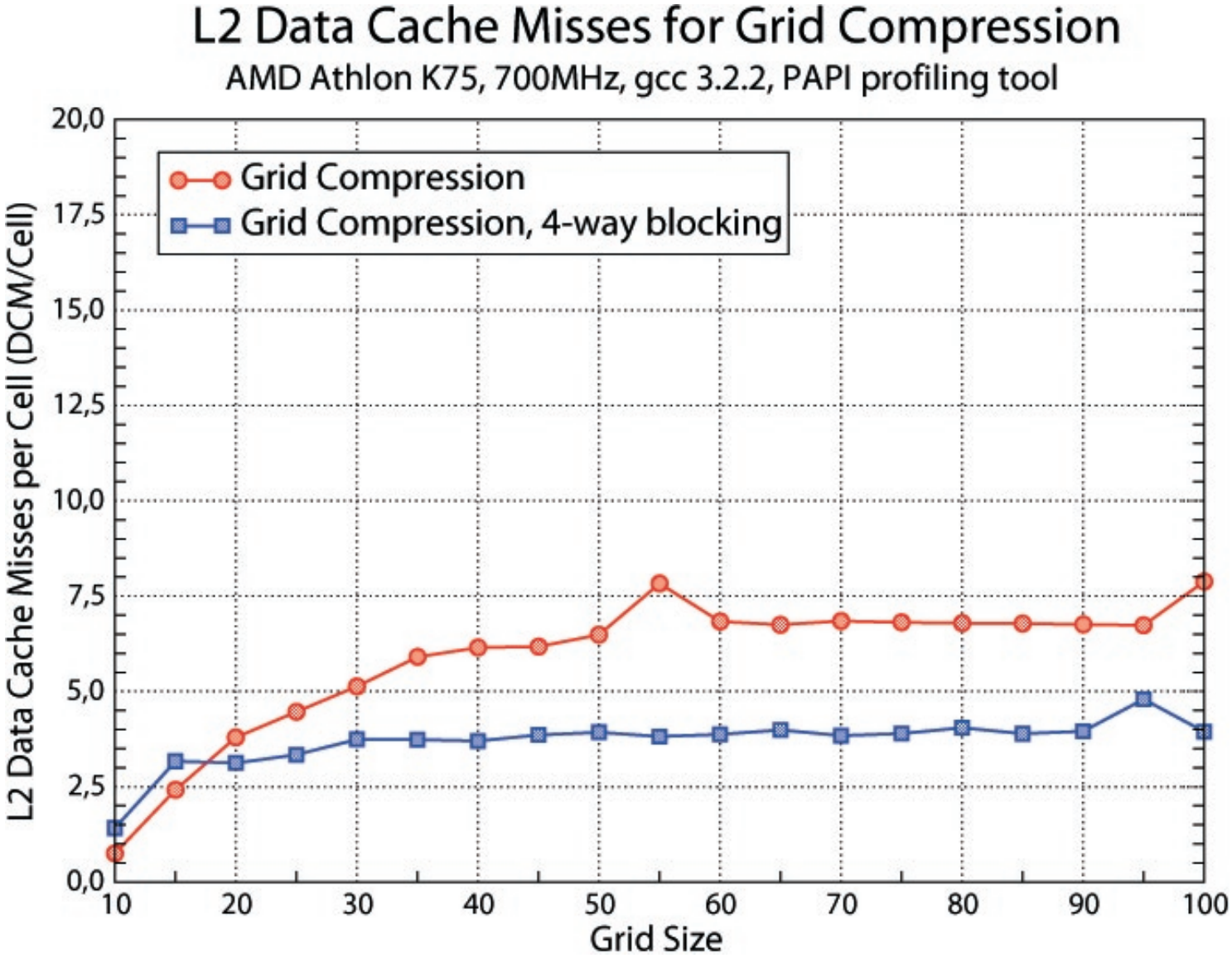

Intel Pentium 4, 2.4GHz, Dual Channel DDR400
Collide-Stream Order, Compiler: icc 7.1
Million Lattice Site Updates / Sec (MLUPS)
Grid Size
2 Grids
Grid Compression, 4-way blocking
10,0
9,0
8,0
7,0
6,0
5,0
4,0
3,0
2,0
1,0
0,0
10
20
30
40
50
60
70
80
90
100


AMD Opteron 1.6 GHz
Collide-Stream Order, Compiler: gcc 3.2.2
2 Grids
Grid Compression, 4-way blocking
Million Lattice Site Updates / Sec (MLUPS)
Grid Size
10,0
9,0
8,0
7,0
6,0
5,0
4,0
3,0
2,0
1,0
0,0
10
20
30
40
50
60
70
80
90
100


L1 Data Cache Misses for 2 Grids
AMD Athlon K75, 700MHz, gcc 3.2.2, PAPI profiling tool
2 Grids
2 Grids, 4-way blocking
L1 Data Cache Misses per Cell (DCM/Cell)
20,0
17,5
15,0
12,5
10,0
7,5
5,0
2,5
0,0
10
20
30
40
50
60
70
80
90
100
Grid Size

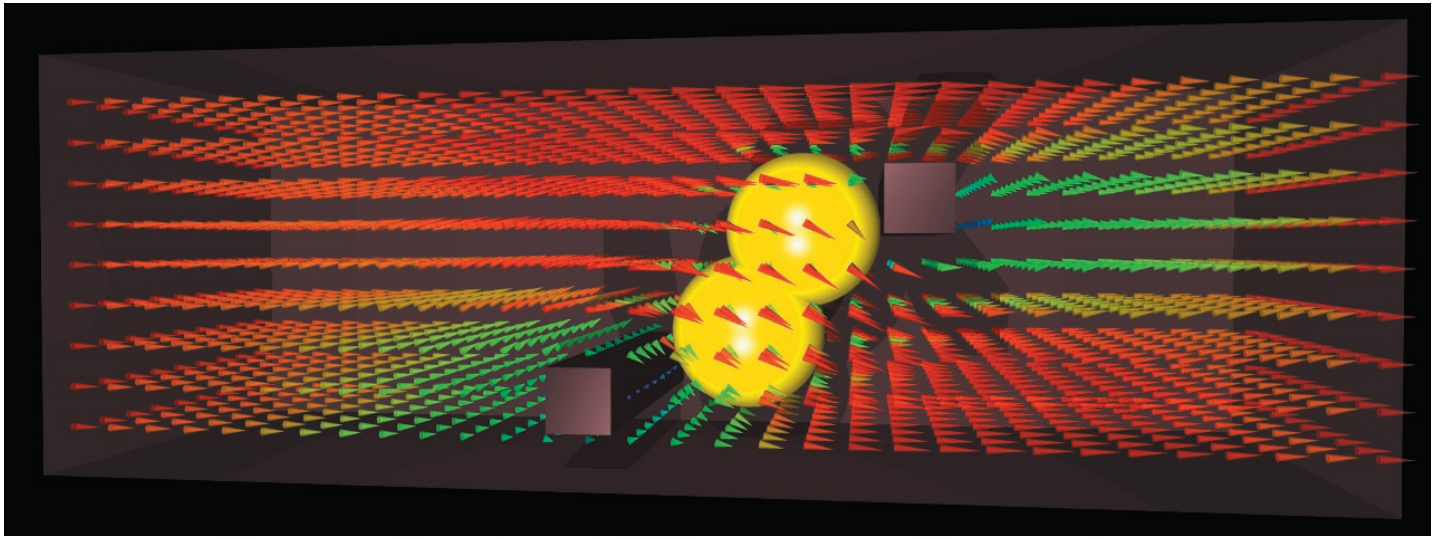

L2 Data Cache Misses for 2 Grids
AMD Athlon K75, 700MHz, gcc 3.2.2, PAPI profiling tool
2 Grids
2 Grids, 4-way blocking
L2 Data Cache Misses per Cell (DCM/Cell)
0,0
2,5
5,0
7,5
10,0
12,5
15,0
17,5
20,0
10
20
30
40
50
60
70
80
90
100
Grid Size


L1 Data Cache Misses for Grid Compression
AMD Athlon K75, 700MHz, gcc 3.2.2, PAPI profiling tool
Grid Compression
Grid Compression, 4-way blocking
L1 Data Cache Misses per Cell (DCM/Cell)
20,0
18,0
16,0
14,0
12,0
10,0
8,0
6,0
4,0
2,0
0,0
10
20
30
40
50
60
70
80
90
100
Grid Size


L2 Data Cache Misses for Grid Compression
AMD Athlon K75, 700MHz, gcc 3.2.2, PAPI profiling tool
Grid Compression
Grid Compression, 4-way blocking
L2 Data Cache Misses per Cell (DCM/Cell)
20,0
17,5
15,0
12,5
10,0
7,5
5,0
2,5
0,0
10
20
30
40
50
60
70
80
90
100
Grid Size

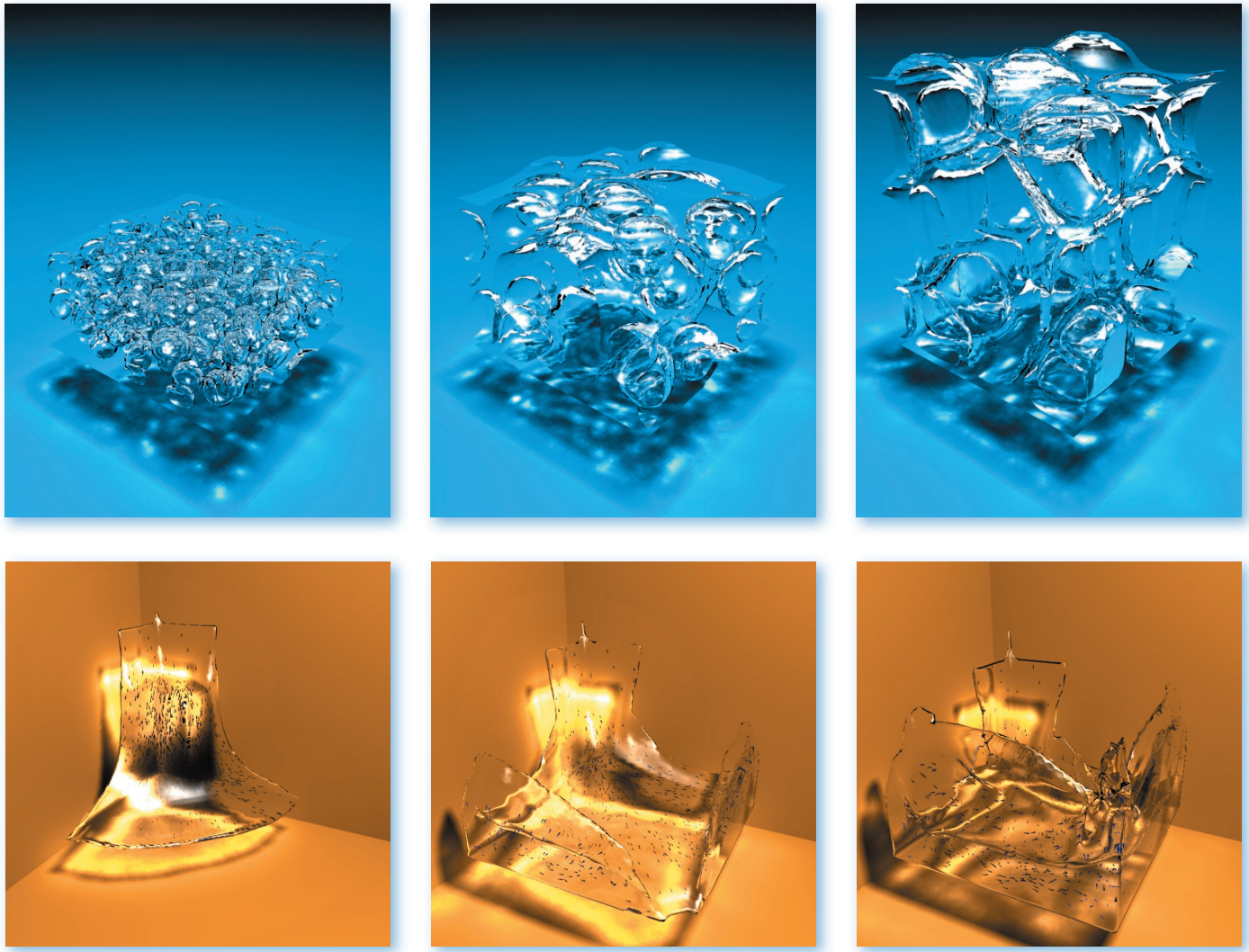
Conclusions

- The Lattice Boltzmann Method contains a lot of performance potential
- Cache Optimization unleashes much of this performance and is applicable on every architecture
- Higher performance increases possible for individually tuned block sizes
- Easy performance gains for real applications



Moving Particle LBM (current Masterthesis)

Further Impressions for real applications: Free Surface LBM



Related Works

Web Pages of the Chair for System Simulation in Erlangen/Germany:

http://www10.informatik.uni-erlangen.de

Lattice Boltzmann publications:

- **Performance Optimization of Lattice Boltzmann Methods, Jens Wilke**
- **On Optimized Implementations of the Lattice Boltzmann Method on Contemporary Architectures (Stefan Donath)**
- **A single-phase free-surface Lattice Boltzmann Method (Nils Thürey)**
- **Force Evaluation on Particles with the Lattice Boltzmann Method (Christian Feichtinger)**
- **Cache Performance Optimizations for Parallel Lattice Boltzmann Codes in 2D (J. Wilke, T. Pohl, M. Kowarschik, U. Rüde)**
- **Parallel Performance of Large-Scale Lattice Boltzmann Applications, (T. Pohl, N. Thürey, F. Deserno, U. Rüde, P. Lammers)**
- **Optimization and Profiling of the Cache Performance of Parallel Lattice Boltzmann Codes in 2D and 3D (T. Pohl, M. Kowarschik, J. Wilke, K. Iglberger, U. Rüde)**