High Performance Computing

Multicore-aware parallelization strategies for efficient temporal blocking

(*BMBF project: SKALB*)

G. Wellein, G. Hager, M. Wittmann, J. Habich, J. Treibig

Department für Informatik

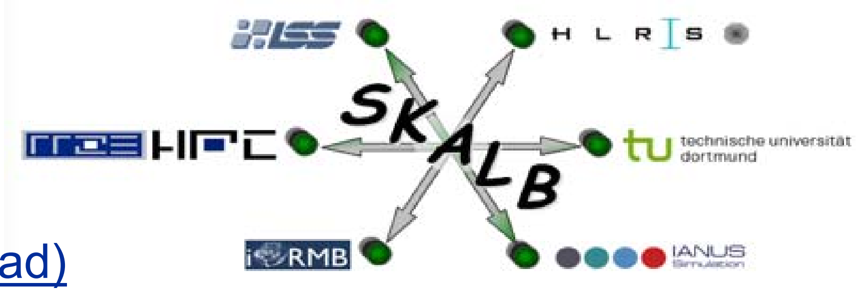
HPC Services, Regionales Rechenzentrum Erlangen

Friedrich-Alexander-Universität Erlangen-Nürnberg

SKALB project survey

High Performance Computing

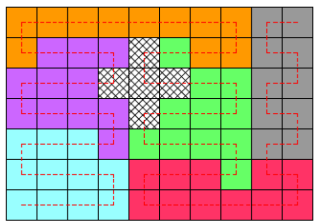
- Partners:
  - TU Braunschweig
  - TU Dortmund / IANUS
  - HLRS Stuttgart
  - Erlangen: LSS / RRZE (projectlead)

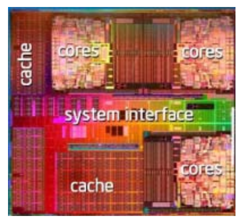


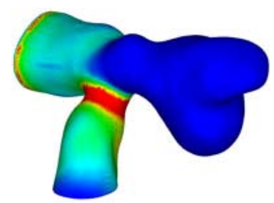
- Ass. partners: Intel, CRAY, BASF, Sulzer, hhpberlin, HP

- Goal: Efficient implementation of Lattice-Boltzmann CFD solvers for complex multi-physics applications on petascale supercomputers

- Main tasks of RRZE:
  - Multicore-specific implementation & optimization of kernels & full applications.
  - Static domain decomposition
  - Benchmarking and showcases







- 1.1.2009-31.12.2011
  (BMBF Call „HPC-Software für skalierbare Parallelrechner“)

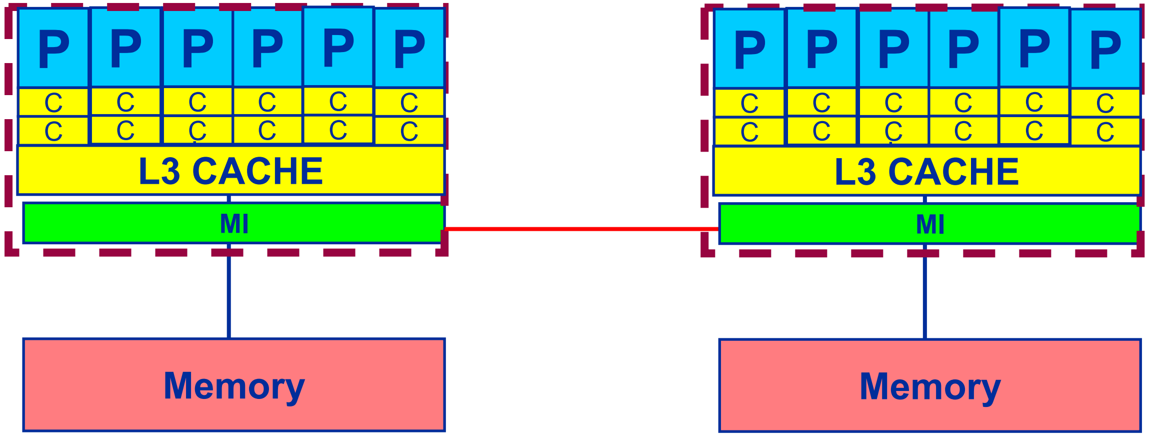
  Funding: 1,8 M€

Bundesministerium für Bildung und Forschung

- **Multicore-aware parallelization approach:**
  - Efficient, parallel, multicore-aware temporal blocking
  - Proof of concept: Simple iterative method w. regular stencil updates (Jacobi)
  - Elaborate spatial blocking techniques have been implemented (not shown)
  - Socket-/node level implementation + multi-halo data exchange for multi-node computations

1. **M. Wittmann, G. Hager, J. Treibig and G. Wellein: *Leveraging shared caches for parallel temporal blocking of stencil codes on multicore processors and clusters.* Submitted to Parallel Processing Letters. arXiv:1006.3148**
2. **J. Treibig, G. Wellein and G. Hager: *Efficient multicore-aware parallelization strategies for iterative stencil computations.* Submitted to Journal of Computational Science. arXiv:1004.1741**
3. **M. Wittmann, G. Hager and G. Wellein: *Multicore-aware parallel temporal blocking of stencil codes for shared and distributed memory.* Accepted for Proceedings of LSPP10, the Workshop on Large-Scale Parallel Processing at IPDPS 2010, April 23rd, 2010, Atlanta, GA. arXiv:0912.4506**
4. **G. Wellein, G. Hager, T. Zeiser, M. Wittmann, H. Fehske: *Efficient temporal blocking for stencil computations by multicore-aware wavefront parallelization*. Proceedings of 2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC2009), IEEE Computer Society, p. 579-586. DOI 10.1109/COMPSAC.2009.82. BEST PAPER AWARD**

2-socket shared-memory node of cache-coherent Non-Uniform Memory Access (ccNUMA) type



Shared Caches:

- Data exchange
- Thread synchronization

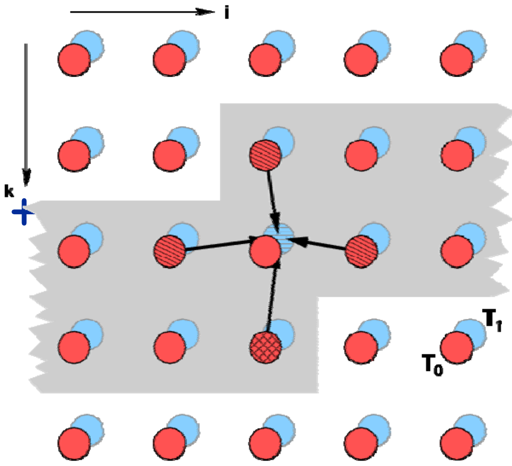
- Data Coherency!
- L3 cache traffic?
- Scalable?
- MPI parallel?

AMD Opteron „Istanbul“	Intel Xeon „Westmere“
6 cores@2.8 GHz	6 cores@2.93 GHz
L1: 64 KB / L2: 512 KB / L3: 6 MB	L1: 32 KB; L2: 256 KB; L3: 12MB
2 X 2 X DDR2-800 → 25.6 GB/s	2 X 3 X DDR3-1333 → 63.6 GB/s
HT2000 → 8 GB/s/dir	2 X QPI6.4 → 12.8 GB/s/dir

Multicore-aware parallelization

Jacobi solver – a simple prototype

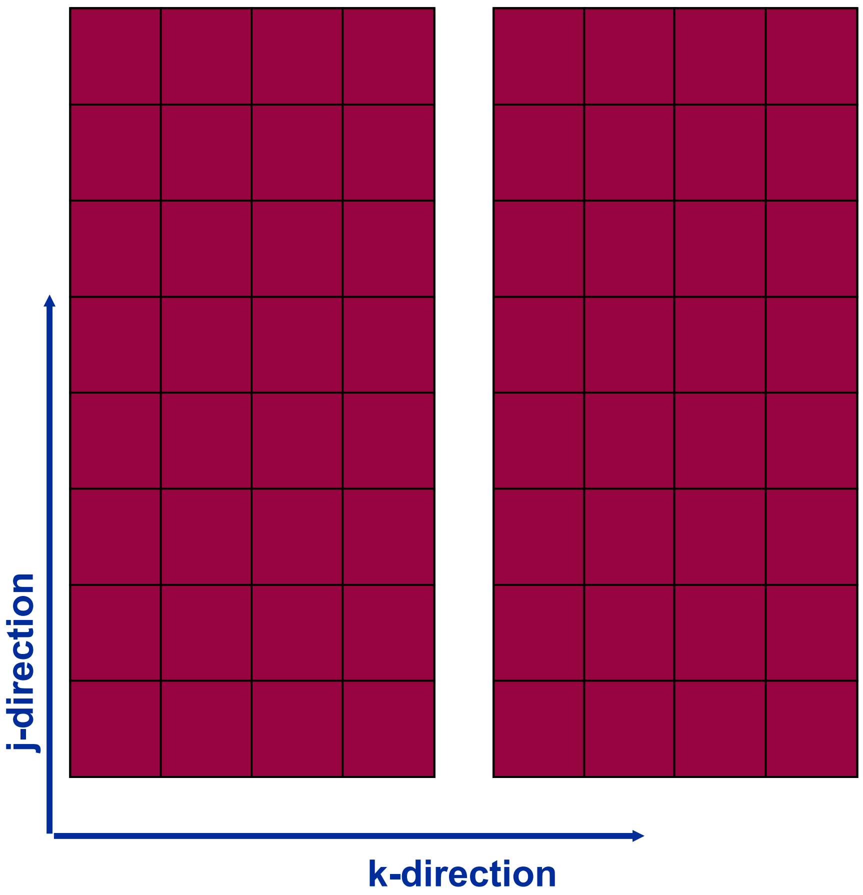

```
do k = 1 , N
  do j = 1 , N
       do i = 1 , N
          y(i,j,k) = b* (x(i-1,j,k)+x(i+1,j,k)+
                         x(i,j-1,k)+x(i,j+1,k)+
                         x(i,j,k-1)+x(i,j,k+1))
       enddo
  enddo
enddo
```

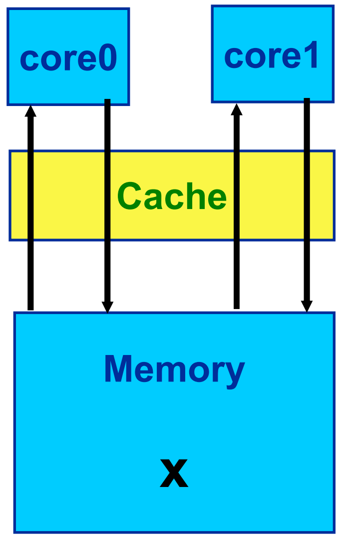


- **A simple but representative test**
- Performance Metric: Million Lattice Site Updates per second (MLUPs)
  Equivalent MFLOPs: 6 FLOP/LUP * MLUPs
- Bandwidth requirements: **16 Byte / Lattice Site Update** (LUP) if write allocate on y is surpressed (24 Byte/LUP otherwise)
- Performance estimate for best baseline: **B_M / (16 Byte/LUP)**
  (**B_M** : attainable memory bandwidth as measured with `stream`)

Multicore-aware parallelization

Towards multicore-awareness: Naive / calssical approach

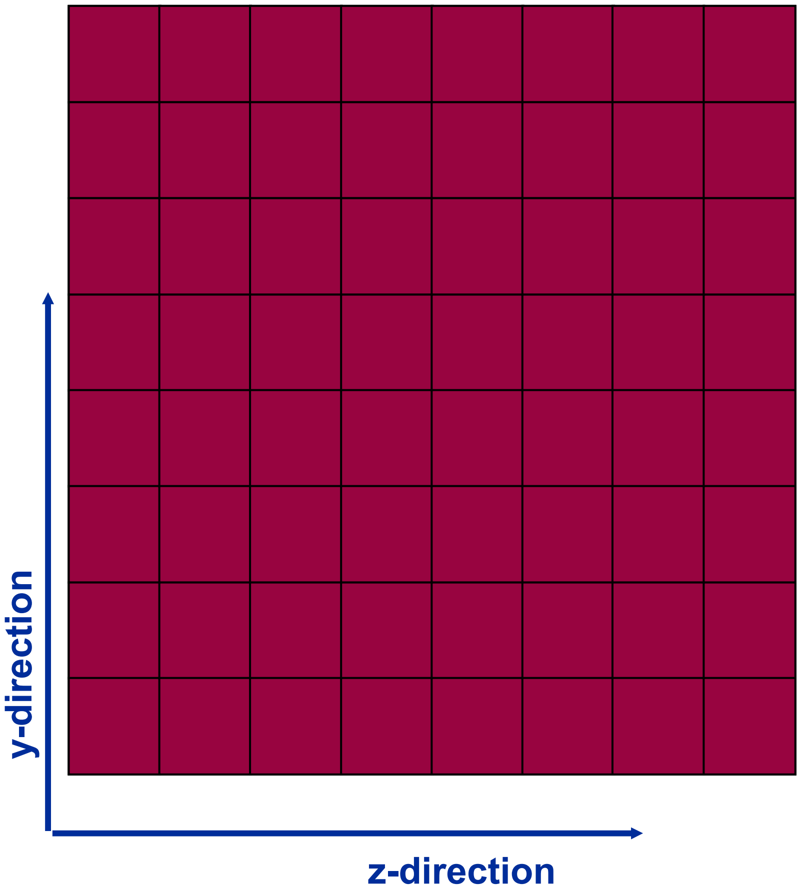


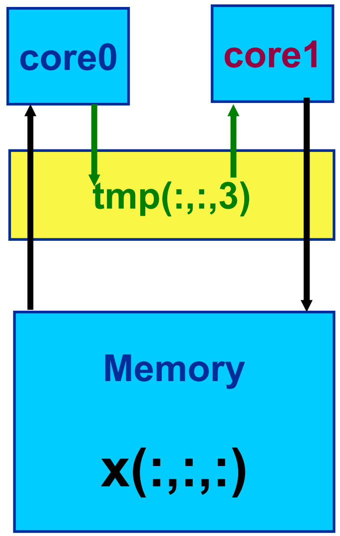



```
do t=1,tMax
!$OMP PARALLEL DO private(…)
  do k=1,N
     do j=1,N
        do i=1,N
           y(i,j,k) = …
        enddo
     enddo
  enddo
  y <-> x
enddo
```

Multicore-aware parallelization

Reuse of in-cache data between threads: wavefronts





Reduce main memory accesses by a factor 2!

`y(:,:,:)` is obsolete!

Save main memory data transfers for `y(:,:,:)`!

Use ring buffer `tmp(:,:, 0:3)` which fits into the cache

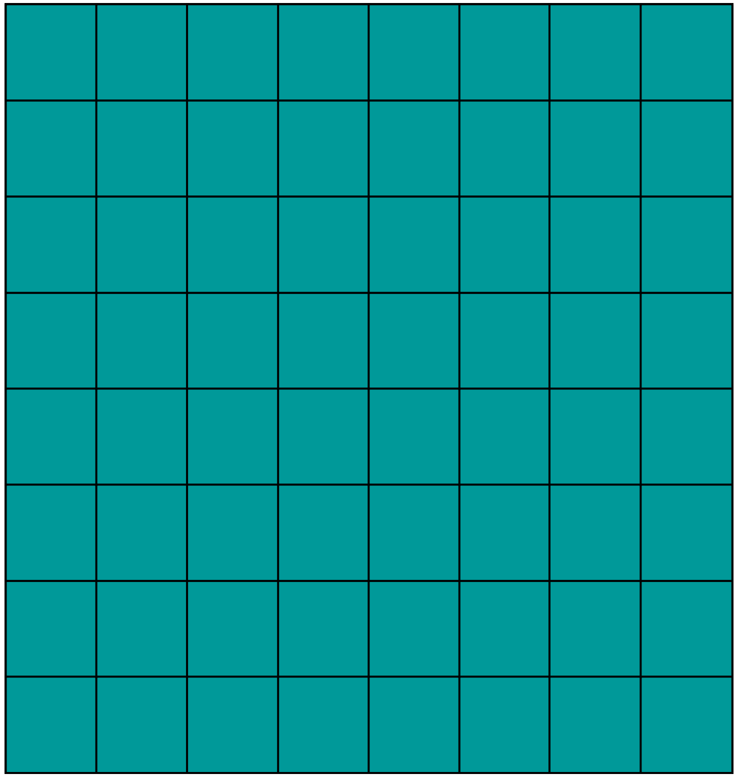
Sync threads/cores after each k-iteration

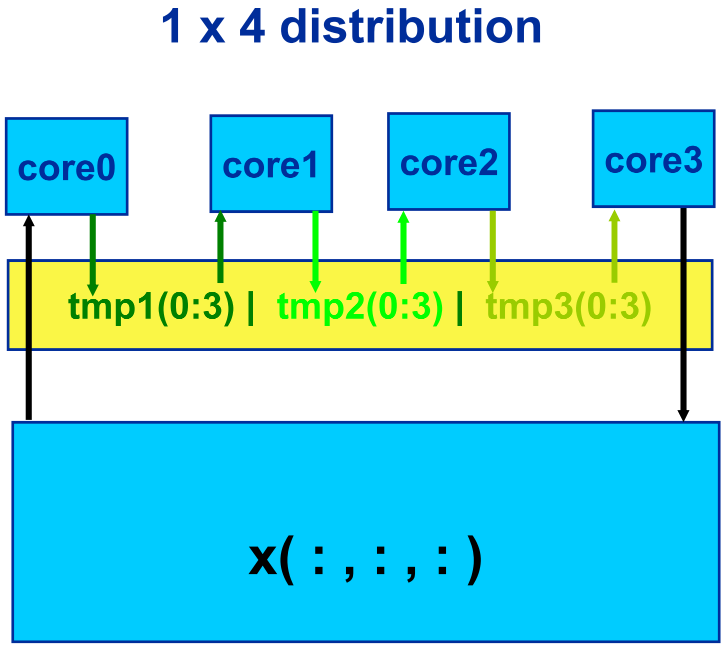
core0: `x(:,:,k-1:k+1)` $_t$ → *`tmp(:,:,mod(k,4))`*

core1: *`tmp(:,:,mod(k-3,4):mod(k-1,4))`* → `x(:,:,k-2)` $_{t+2}$

Multicore-aware parallelization

Wavefront parallelization: Multiple updates

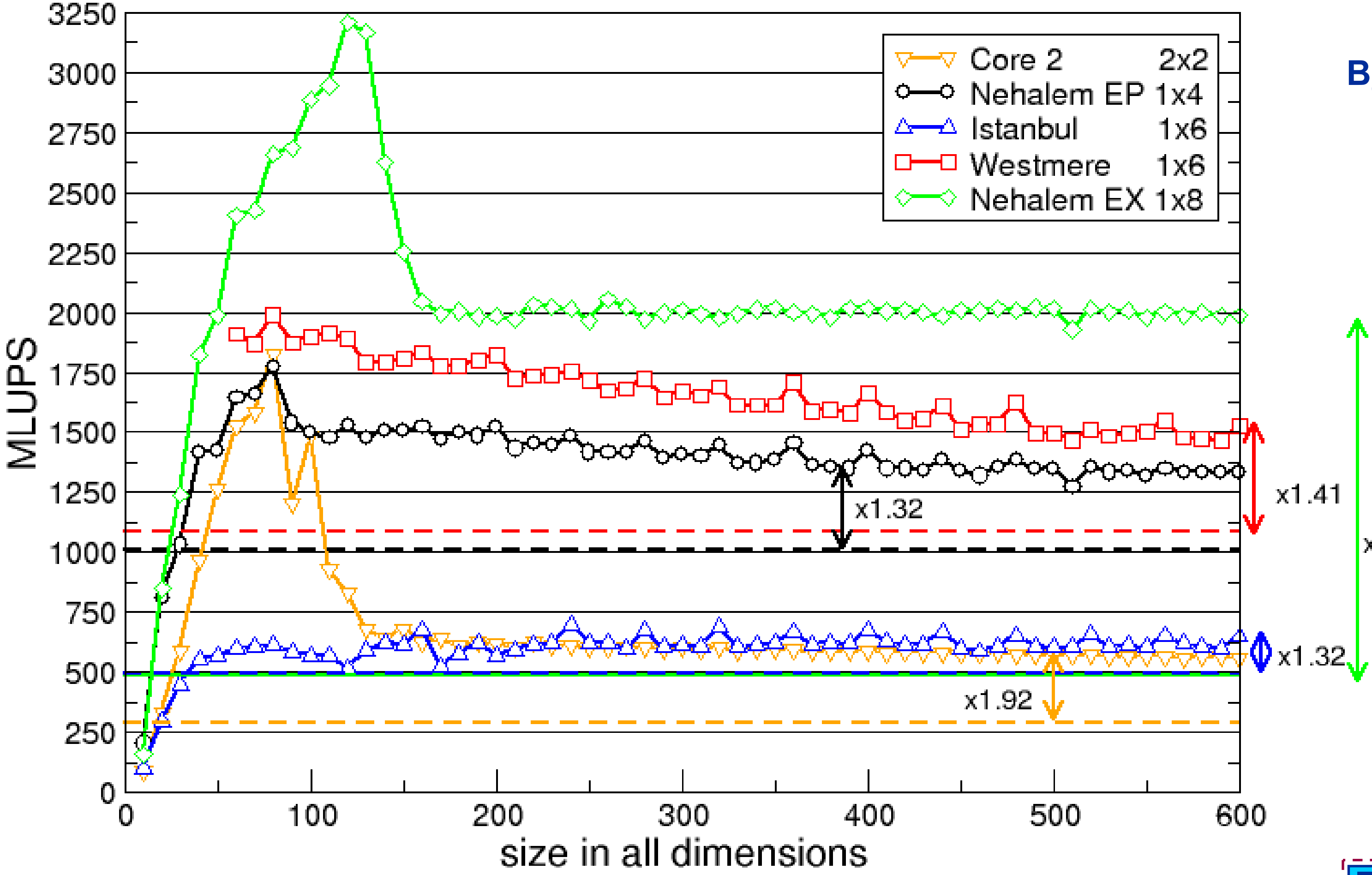


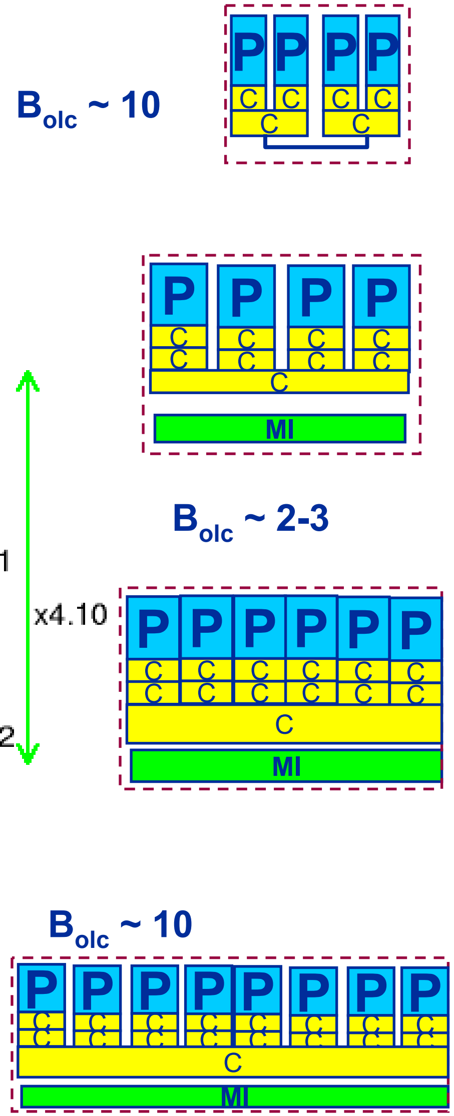


Running `tb` wavefronts requires `tb-1` temporary arrays `tmp` to be held in cache!

Max. performance gain: `tb = 4`

But extensive use of cache bandwidth!





Compare against optimal baseline!

Performance gain ~ B_{olc} = L3 bandwidth / memory bandwidth

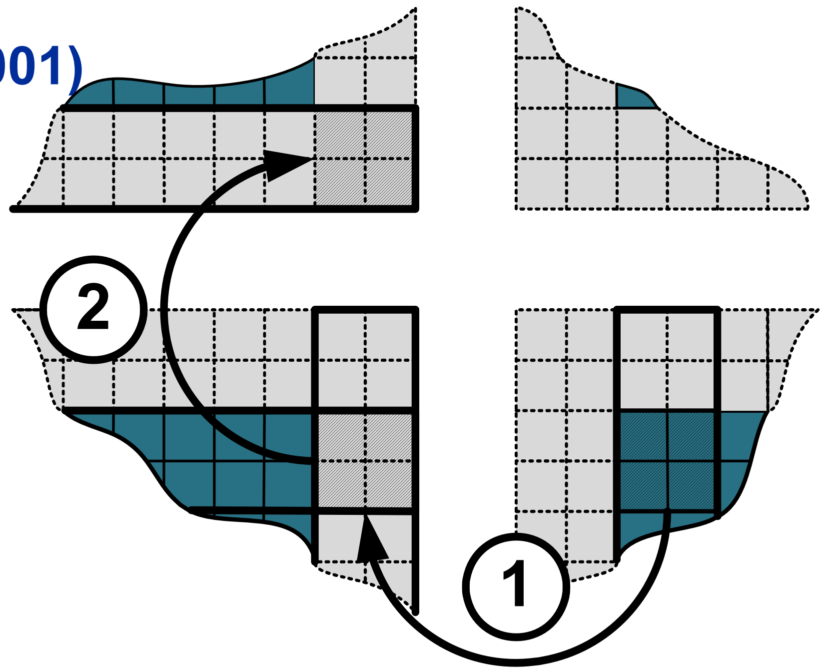
- **Fast shared on-chip resources: Room for new ideas!**
  - Fast thread synchronisation through shared caches
    - Wavefront works fine with Intel compilers but fails with gcc: OMP BARRIER: 4-core Core i7:
      - 11,000 cycles (gcc 4.3.3) vs. 800 cycles (icc 11.0)

  → Ability to exploit parallel structures at finer granularity (shorter loops, frequent synchronisation)

- **Technique can easily be extended to many regular stencil based iterative methods, e.g.**
  - Gauß-Seidel (→ done)
  - Lattice-Boltzmann flow solvers (→ work in progress)
  - Multigrid (→ BMBF proposal EMMMA)

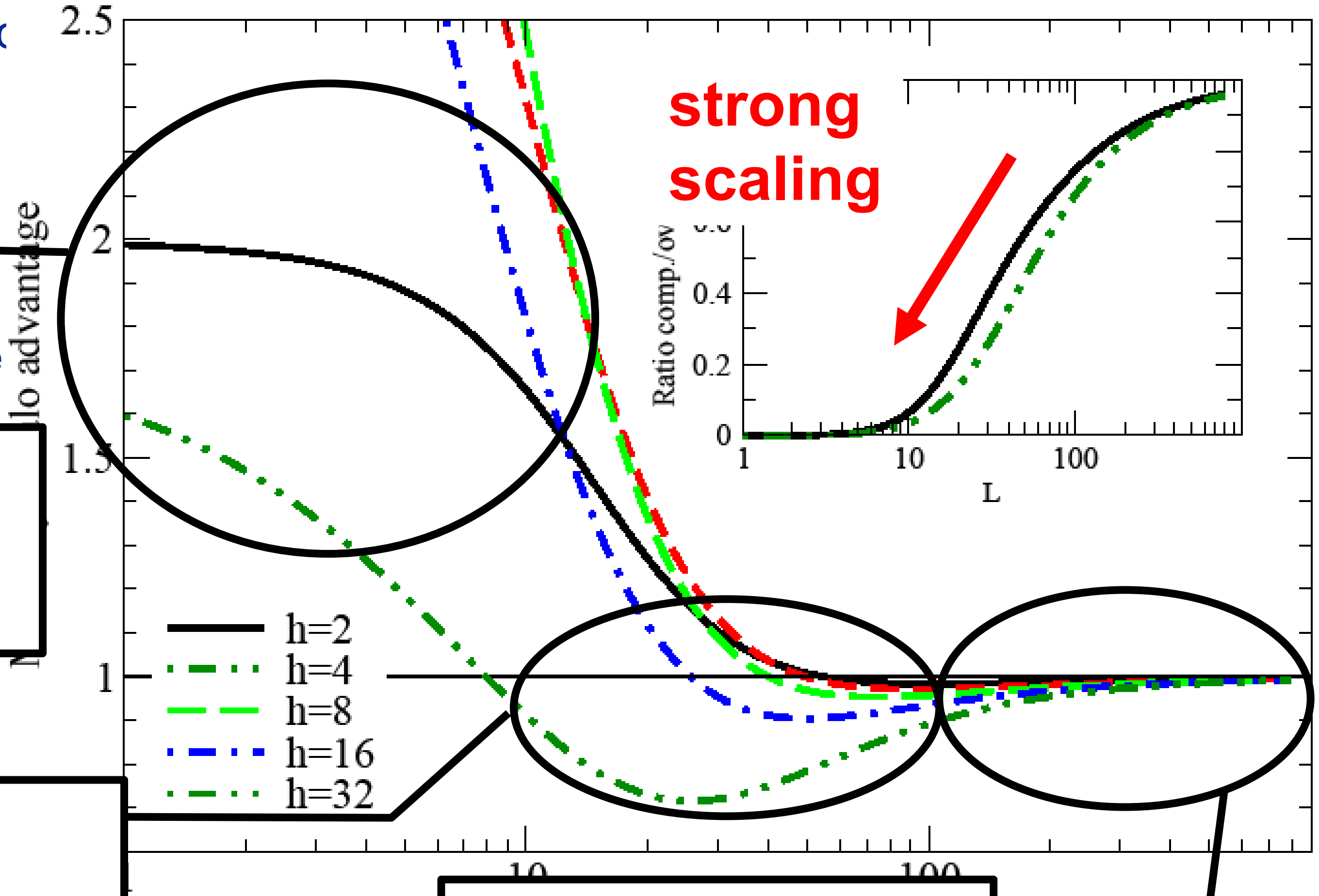
- **For multi-node applications an hybrid MPI+OpenMP approach needs to e implemented (next slides)**

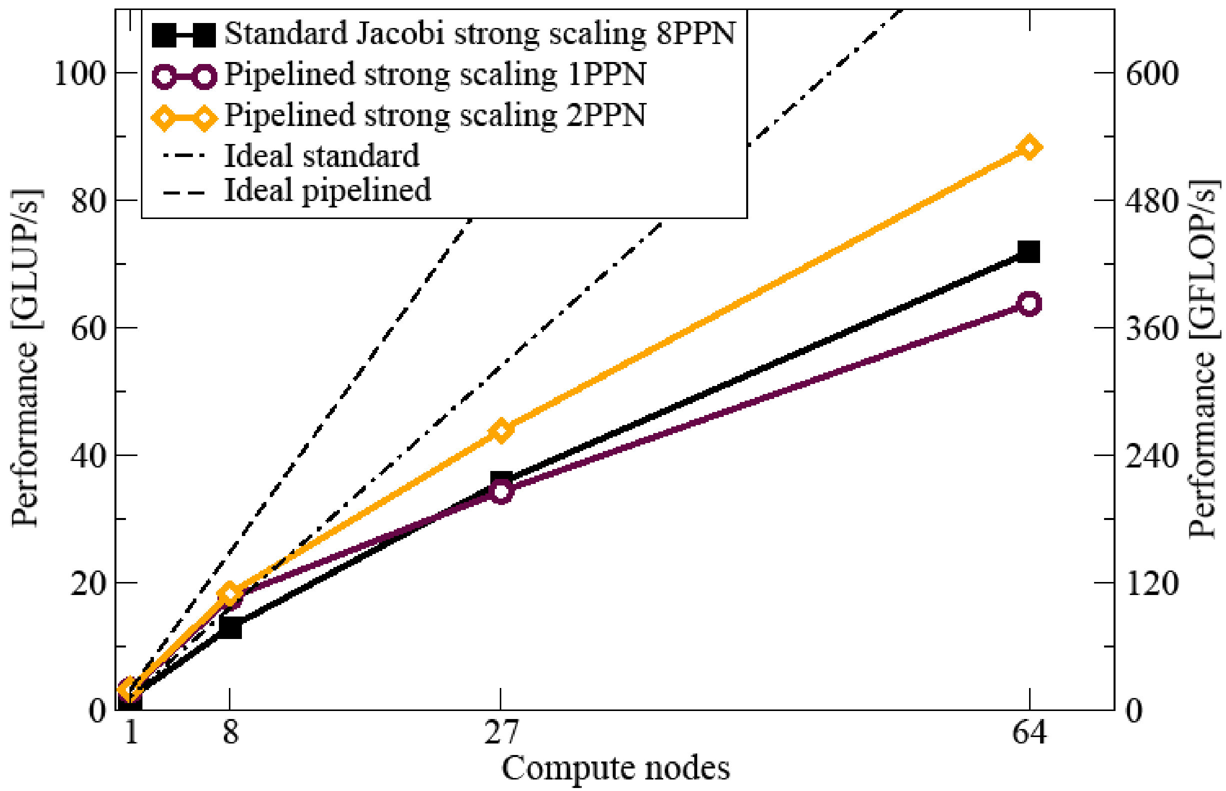
- **Temporal blocking requires multi-layer halo**
- **Using diagonal communication elimination (DCE) (Ding/He SC 2001)**
- **Exchanging halo with neighbors done only along the coordinate directions**



- **More complex stencils, e.g. occurring at lattice Boltzmann methods, need more attention for deciding which data to communicate**

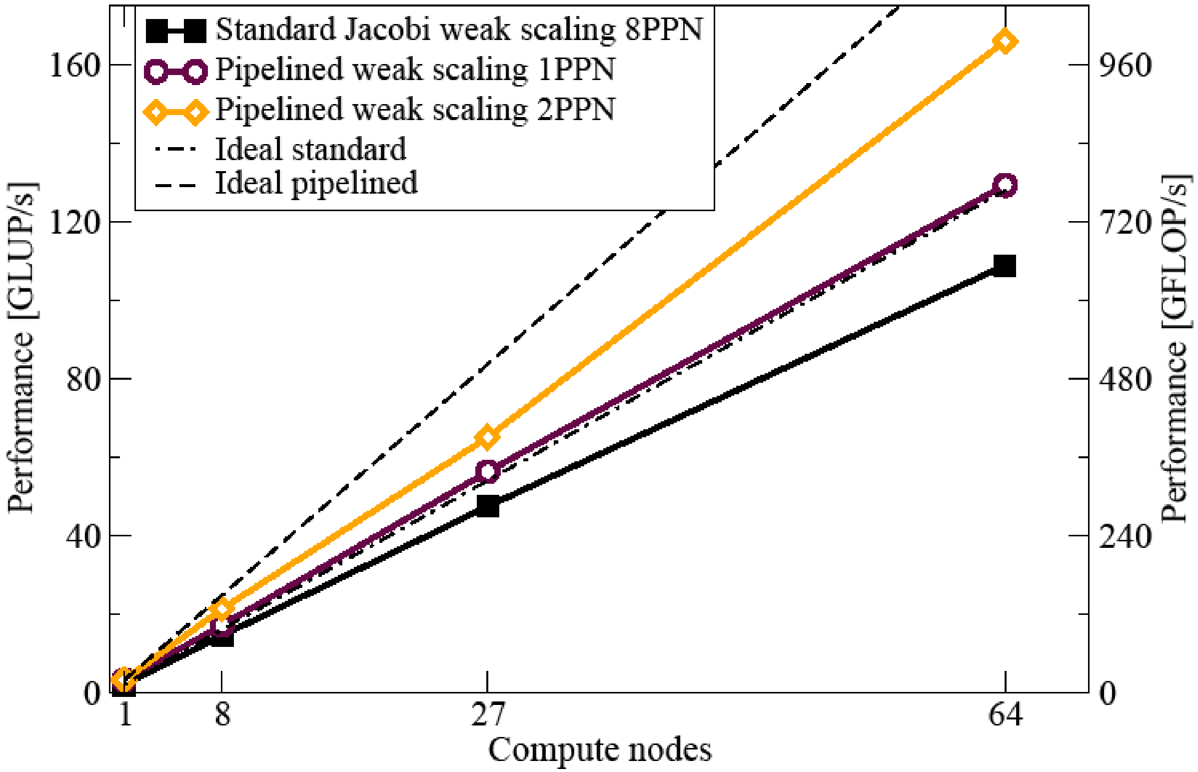
- **Assumptions for model:**
- No overlap between communication and computation
- QDR-InfiniBand
  - 3.2 GB/s
  - 1.8 µs latency
- Node performance 2 GLUPs





- **Nehalem EP cluster QDR Infiniband**
- **Domain size: $600^3 \approx$ 1.6 GB**
- **Benefit of temporal blocking eaten by communication overhead (~ 60-70%)**
- **Pipelining 1PPN suffers from bulk-synchronized communication**

High Performance Computing



- **Nehalem EP Cluster QDR Infiniband**
- **Subdomain size per node: 600^3**
- **Benefit from temporal blocking can be maintained**

High Performance Computing

Parallel GPU computing with Lattice-Boltzmann solvers– a focus of SKALB

Preliminary comparison Blue-Gene/P – GPU-Cluster

LUDWIG@iRMB-TuB

96 GPUs: NVIDIA Tesla

24 host nodes

D3Q13 discretization

	#cores	precision	# grid nodes	absolute performance
Blue-Gene/P	16.384	double	10^9	~20 GLUPS
Ludwig (96 GPUs)	23.040	single	3.2×10^9	~45 GLUPS

WALBERLA:
Block-/patch-structured heterogeneous CPU/GPU

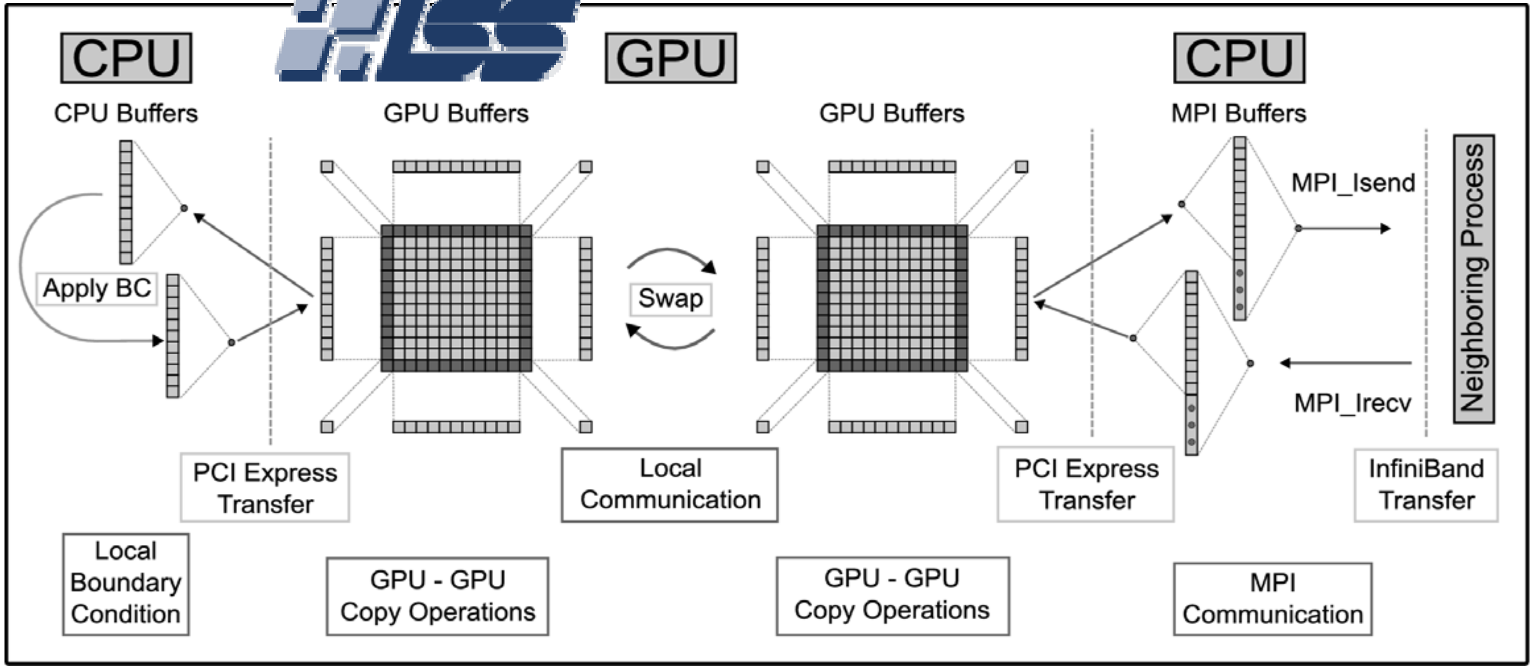
High Performance Computing

D3Q19

Halo exchange between blocks

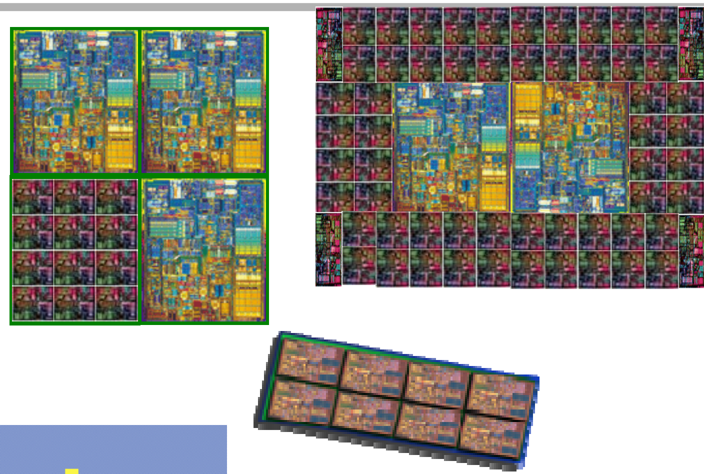
Single precision computations

GPU cluster at HLRS



Blocks	GPU: 1		GPU: 22, CPU:1			
Nodes	1	30	1	30	60	90
Processes	2 x GPU	60 x GPU	2 x GPU + 6 x CPU	60 x GPU + 180 x CPU	60 x GPU + 420 x CPU	60 x GPU + 660 x CPU
MFLUPS	476	14480	459	13267	15684	17846

~3 TFlop/s



Besten Dank

- **Financial support through**
  KONWIHR-II projects
  OMI4papps , HQS@HPC

KONWIHR II

BMBF Project
SKALB (01 IH08003A)
www.skalb.de

Bundesministerium für Bildung und Forschung