PARALLEL CFD 2011
23rd International Conference on
Parallel Computational Fluid Dynamics 2011

GPU implementation of the LBM: Architectural Requirements and Performance Result

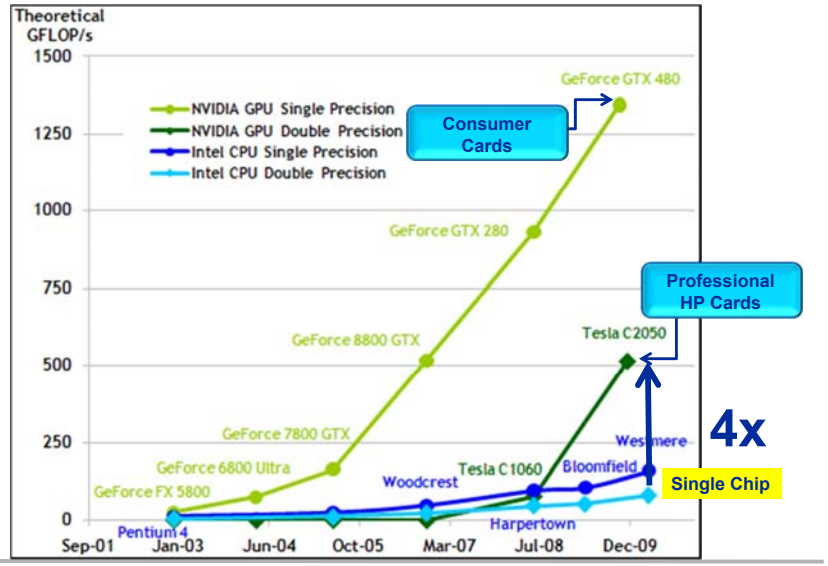
Parallel CFD 2011

J. Habich[a], C. Feichtinger[b], G. Wellein[a,b]

[a]HPC Services – Regional ComputingCenter Erlangen
[b]Department of Computer Science

University Erlangen-Nürnberg

Why GPUs? Peak Performance of CPU vs. GPU



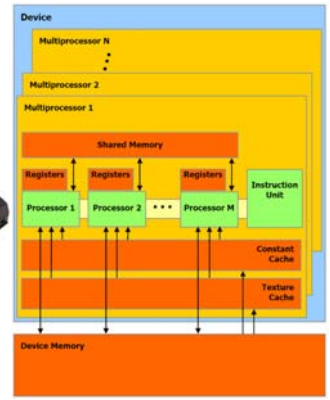
Outline

- GPGPU CUDA Hardware
- Architectural Requirements
  (Stream Benchmarks)
- Lattice Boltzmann on GPUs

Specifications of the NVIDIA Fermi GPU

- **14 Multiprocessors (MP); each with:**
  - 32 processors SP driven by :
    Single Instruction Multiple Data (SIMD)
    Single Instruction Multiple Thread (SIMT)
  - Explicit in-order architecture
  - 32K Registers
  - 48 KB of local on-chip memory
    (shared memory)
  - 1st and 2nd level Cache hierarchy
  - clock rate of 1.15 GHz

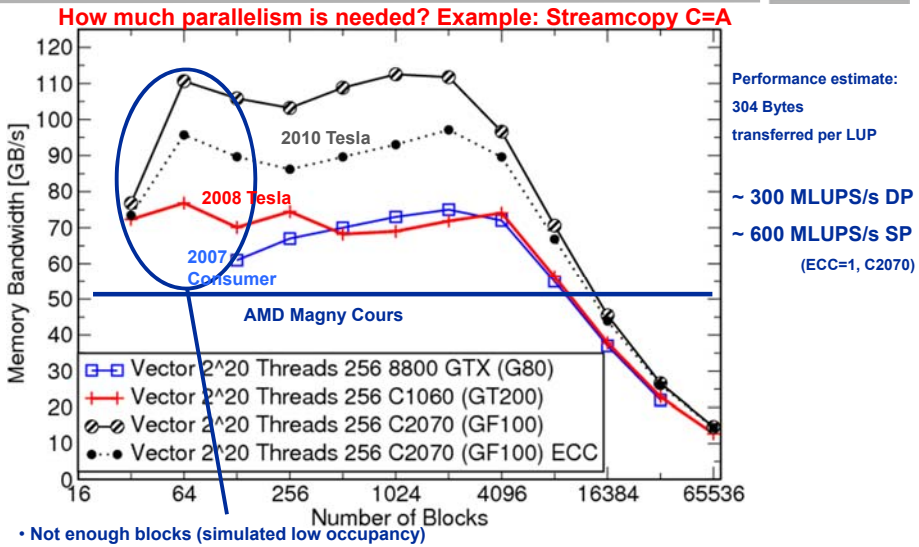
  1030 GFLOP/s (single precision)
  515 GFLOP/s (double precision)



- **8+8 GB/s PCIe 2.0 x16 interface to CPU**

	Clock (MHz)	Peak (GFLOPs)	Memory (GB)	Memory Clock (MHz)	Memory Interface (bit)	Memory Bandwidth (GB/sec)	Threads@Stream
Tesla C2070	1150	1030	6	1500	384	144	8000
GeForce GTX 280	1400	1000	1	1160	512	148.6	8000
GeForce 8800 GTX	1350	345	0.768	900	384	86	8000
Host (Westmere)	2.66	255	24	1333	3*64	63	12

Memory Bandwidth on GPUs



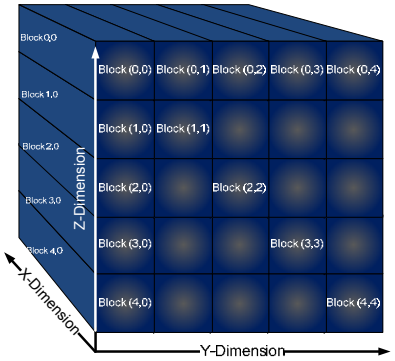
CUDA Scheduling impacted by Resource limits

- **Resources per Streaming Multiprocessor (MP)**
  - 32 000 (32bit) Register
  - 16 KB to 48 KB of Shared Memory
- **Paralellism aka. Occupancy is limited by resource usage per Block/Thread**
  - 1536 threads can be executed/scheduled parallel per MP
    → 20 Registers per Thread
    → 10 to 35 byte shared memory per Thread (GF100 allows switching)
  - Using more registers will decrease parallel threads per MP

Threads	1536	1024	512	384	256	128	64
Registers	20	30	62	83	125	250	500

Lattice Boltzmann on GPUs

- Physical discretization: D3Q19
- Pull / stream collide algorithm order
- Structure of Arrays data layout
- 1 GPU thread per lattice node



The lattice Boltzmann method algorithm

```
double precision F(0:18,0:iMax+1,0:jMax+1,0:kMax+1,0:1)
 x= threadidx%x  ! Lattice x coordinate
 y = blockidx%x  ! Lattice y coordinate
 z = blockidx%y  ! Lattice z coordinate
          if( fluidcell(x,y,z) ) then
             LOAD F( 0,x  ,y  ,z  ,t)
             LOAD F( 1,x+1,y+1,z  ,t)
             LOAD F( 2,x  ,y+1,z  ,t)
             LOAD F( 3,x-1,y+1,z  ,t)
             ...
             LOAD F(18,x  ,y-1,z-1,t)
             Relaxation (complex computations)
             STORE F(0:18,x,y,z,t+1)
          endif
```

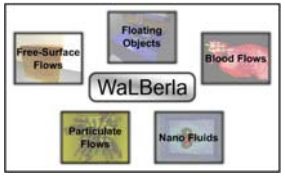
Collide Step

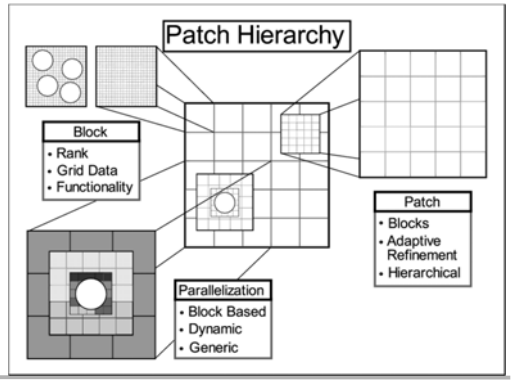
Stream Step

- **Initial 100 Registers per Thread → Occupancy below 50%**
- **Manual array stepping led to 32 Registers per Thread → Occupancy ~ 66 %**

WaLBerla

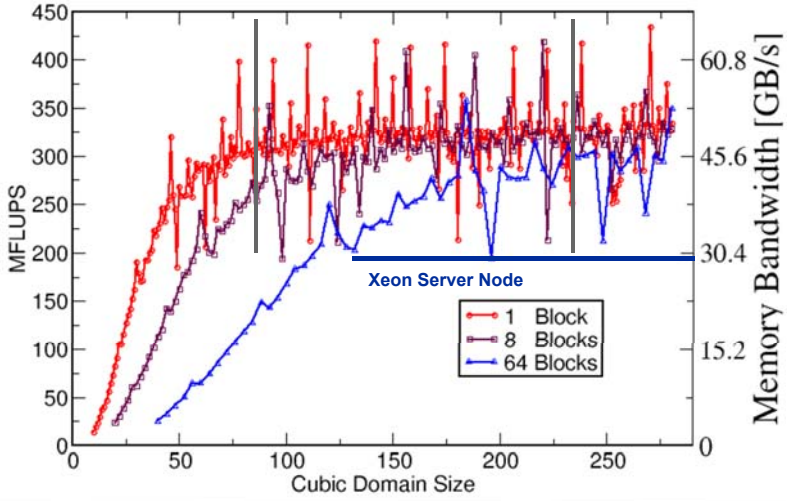
- Patch and Block based domain decomposition for grid refinement and complex flows
- Block contains Simulation Data and Metadata e.g. for parallelization, advanced models
- Block can be algorithm or architecture specific
- Processes can have one or multiple blocks





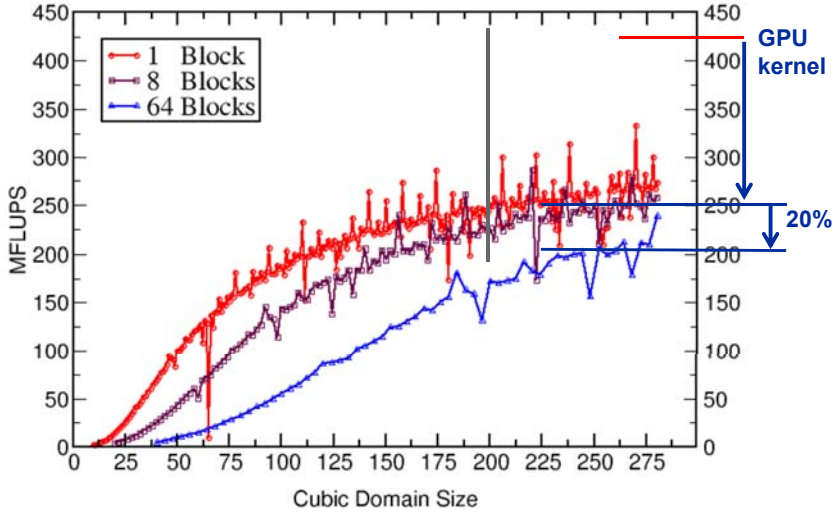
GPU kernel (SP), no PCIe transfer

- Decomposition to Blocks influence kernel only marginally
  (for Domains < 200x200x200)



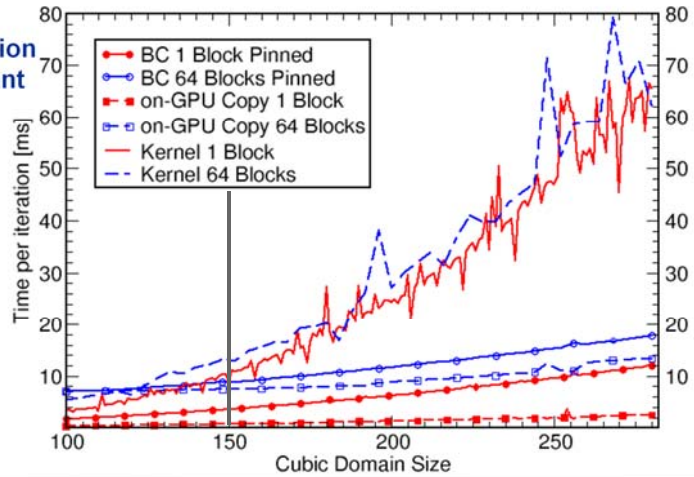
Kernel with boundary transfer (SP) over PCIe

- Decomposition to Blocks influence kernel with any domainsize.



Time measurements of kernel with 1 and 64 blocks

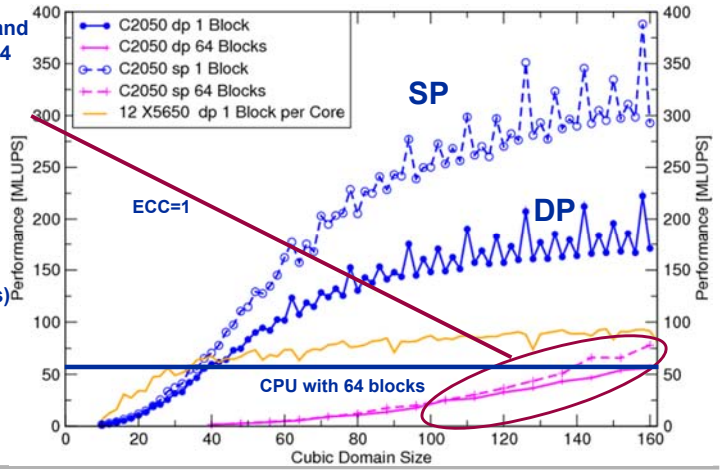
- Domains > 250^3 → about 50% of execution time is spent in non-kernel parts
- Kernel execution time is constant



Performance on GPU

- C2050 speedup of 2 compared to full 12 core Intel Westmere in DP
- 75% of attainable memory bandwidth in DP

- Difference of sp and dp vanishes for 64 blocks
- Algorithm gets communication bound (75% loss)
- CPU impact is smaller (28% loss)

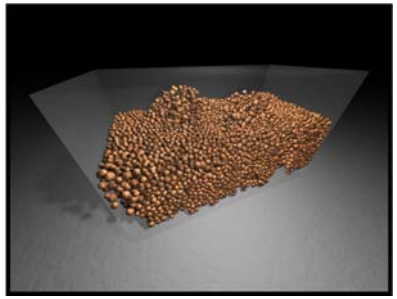


Summary

- High attainable performance on GPUS is achievable by hardware aware optimizations
- Data locality and access patterns play important role
- PCIexpress is still a major bottleneck
- First very promising heterogeneous CPU/GPU results cf.:

WaLBerla: Heterogeneous Simulation of Particulate Flows on GPU Clusters - C. Feichtinger MS3 - LBM II ~ 16:50

Thank you very much for your attention



KONWIHR-II projects
OMI4papps ,
HQS@HPC

Bundesministerium für Bildung und Forschung

BMBF Project
SKALB (01 IH08003A)