Many-core and Accelerator-based Computing for Physics and Astronomy Applications
Nov 29-Dec 2, 2009, Stanford , US

Parallel implementation of LBM on GPU-based cluster

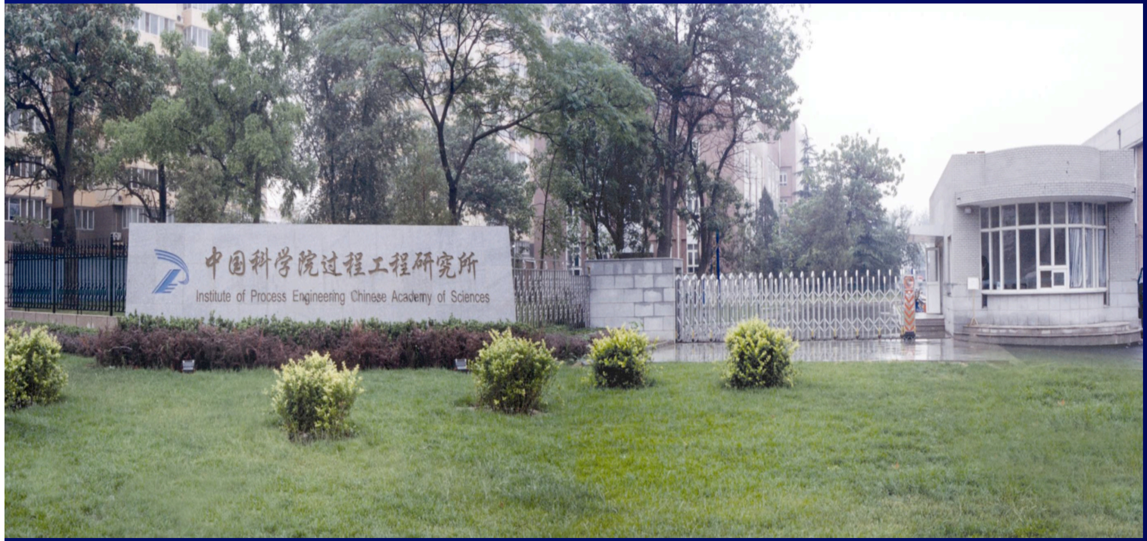
Xiaowei Wang, Xipeng Li, Yun Zhang, Bo Li, Wei Ge

Institute of Process Engineering, CAS

Outline

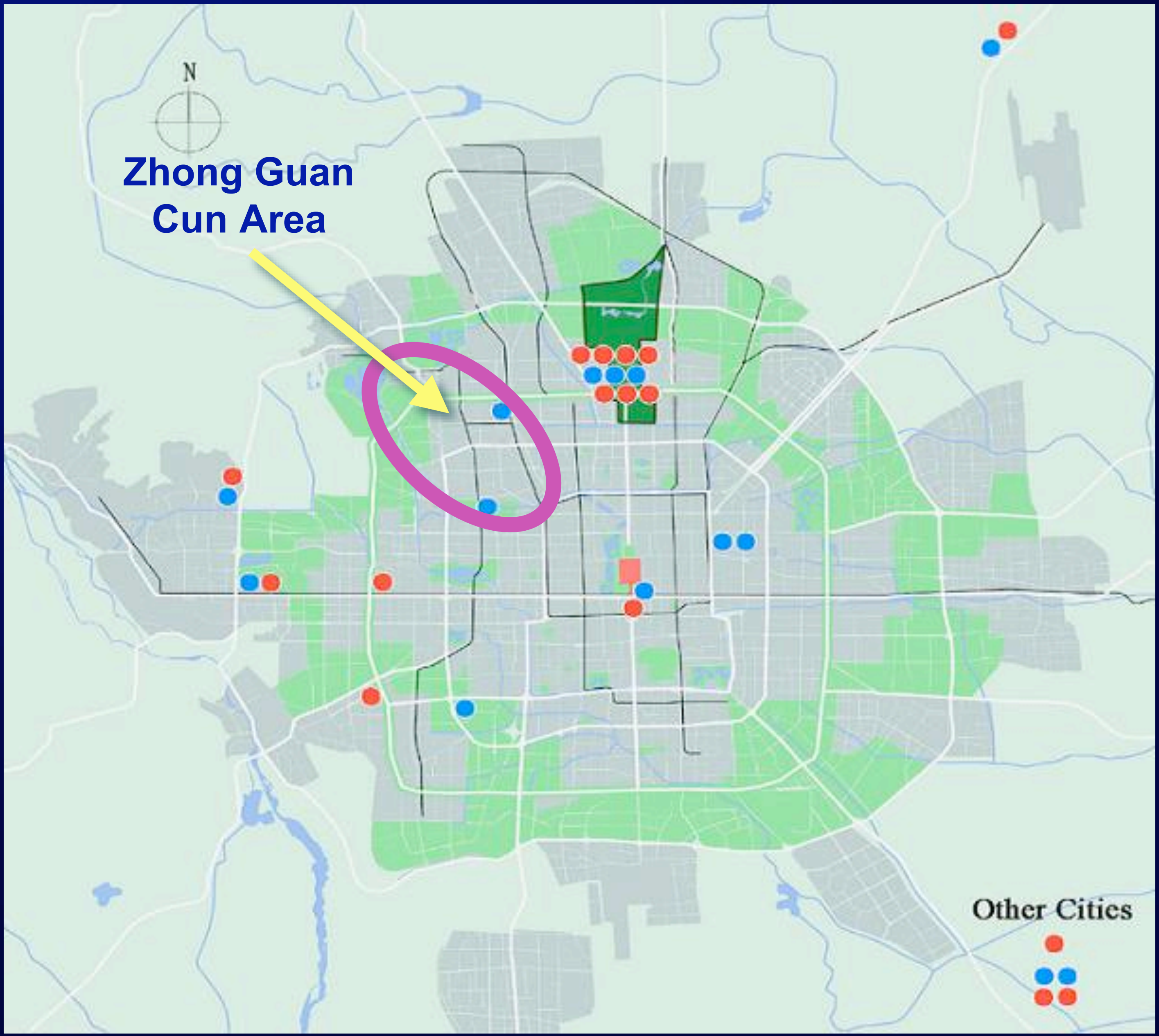
- About us
- Introduction
- Lattice Boltzmann method
- Serial implementation
- Parallel implementation
- Performance analysis
- Coupled implementation with AMD GPUs
- Application
- Conclusions

Institute of Process Engineering, CAS

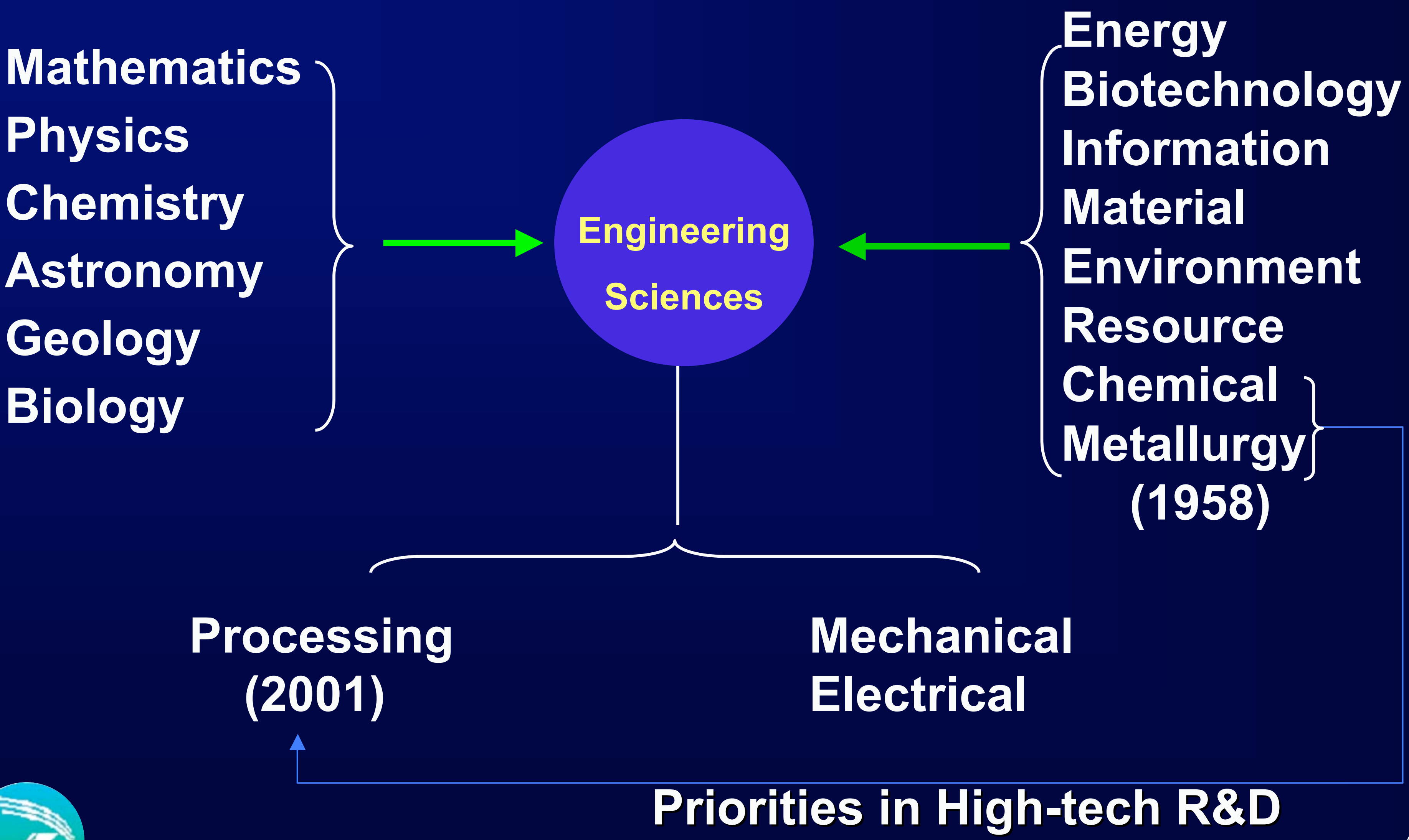

中国科学院过程工程研究所
Institute of Process Engineering Chinese Academy of Sciences

3

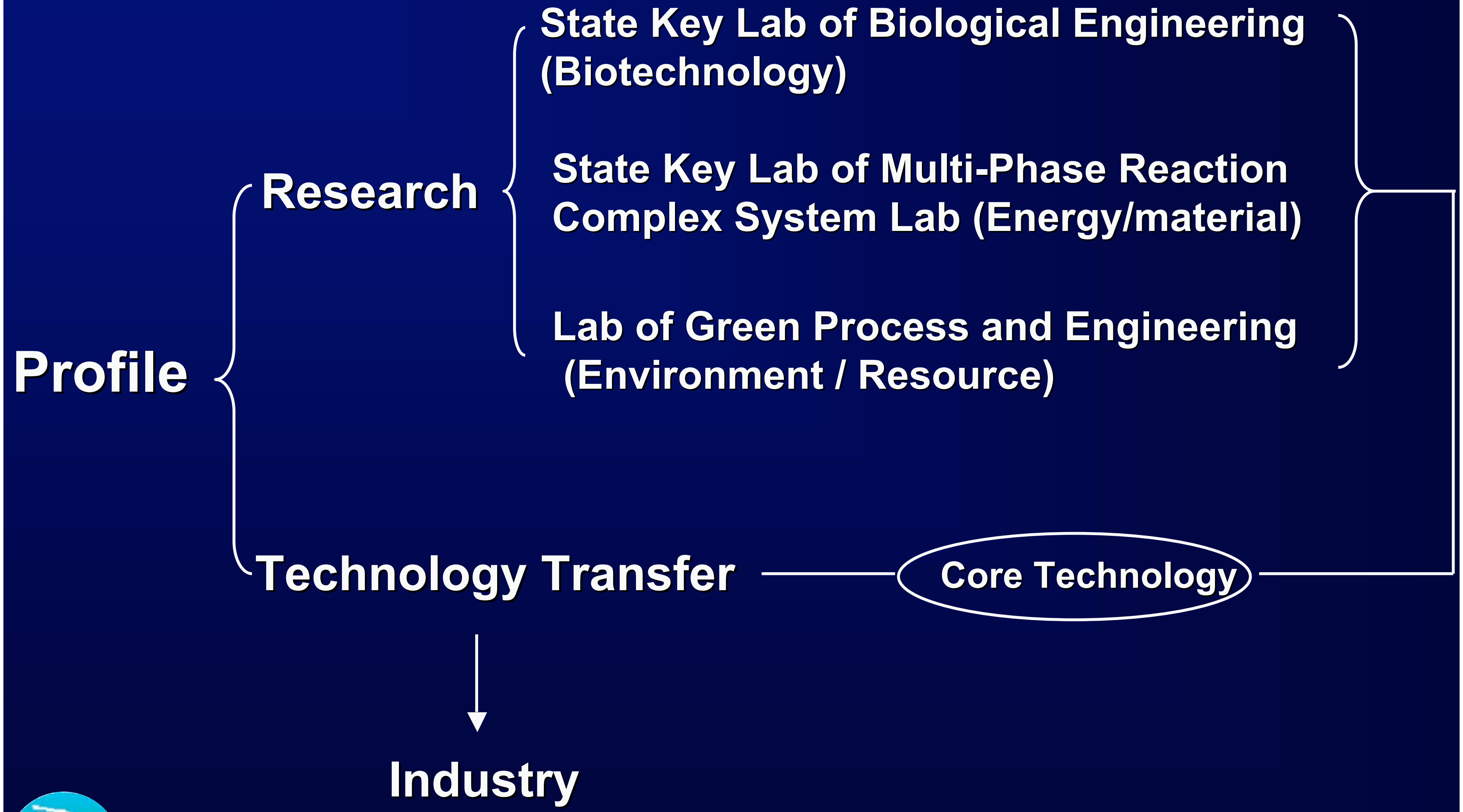
Beijing Map



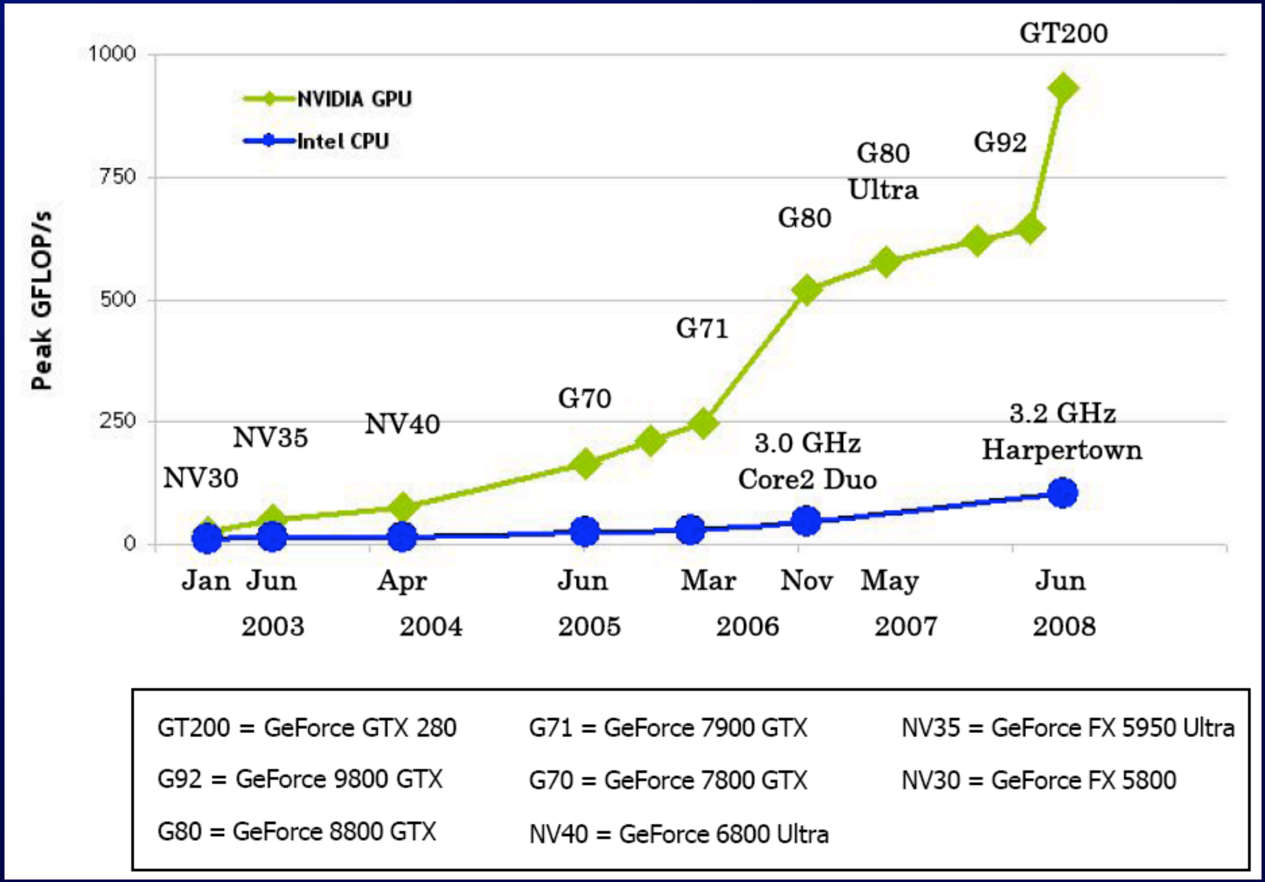
The position of our institute in CAS



5


Profile
Research
State Key Lab of Biological Engineering (Biotechnology)
State Key Lab of Multi-Phase Reaction Complex System Lab (Energy/material)
Lab of Green Process and Engineering (Environment / Resource)
Technology Transfer
Core Technology
Industry

Why GPU for Computing? –Peak Flops



GT200 = GeForce GTX 280	G71 = GeForce 7900 GTX	NV35 = GeForce FX 5950 Ultra
G92 = GeForce 9800 GTX	G70 = GeForce 7800 GTX	NV30 = GeForce FX 5800
G80 = GeForce 8800 GTX	NV40 = GeForce 6800 Ultra	

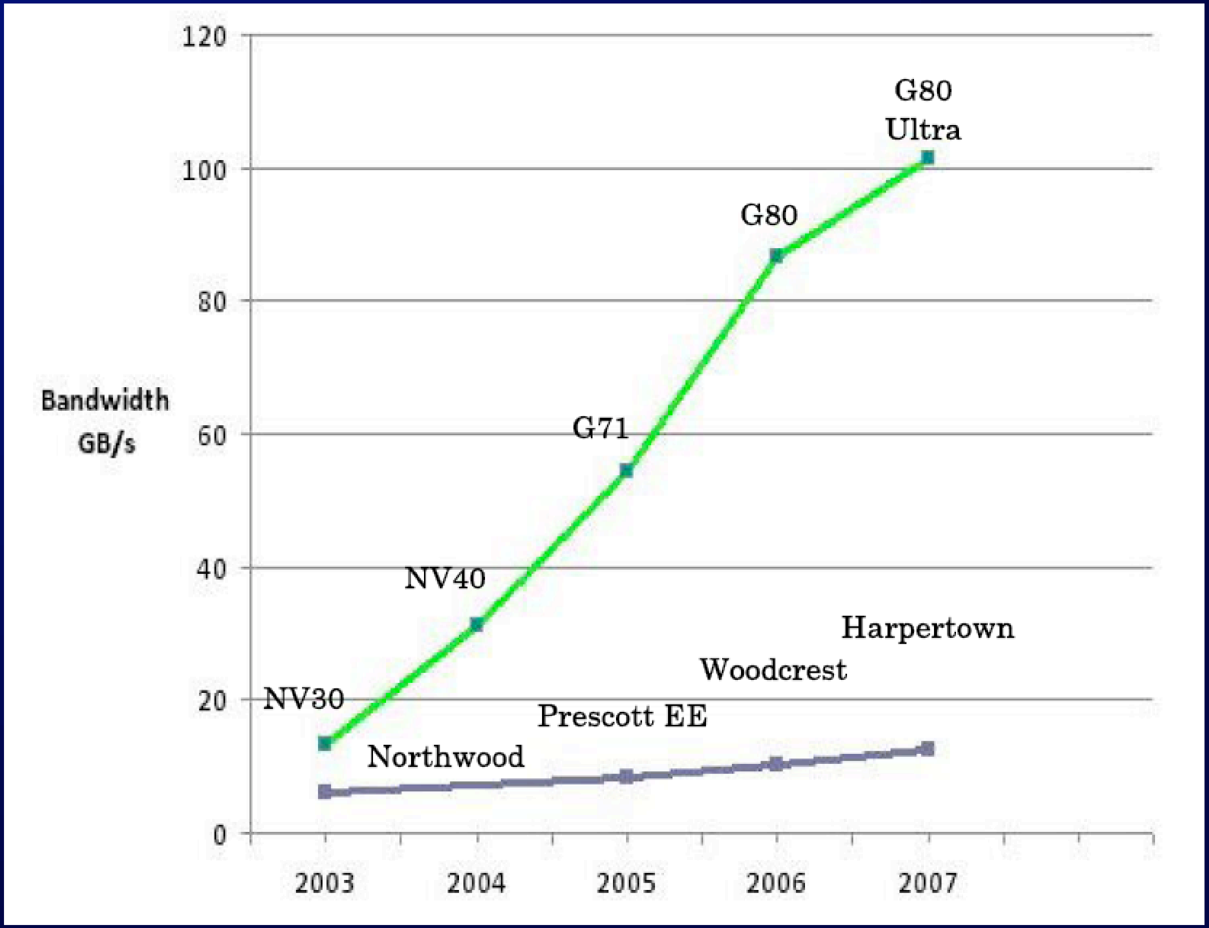
(from *CUDA Programming Guide 04/02/2009*)

Massive parallel processor

CPU : ~4 @ 3.2 Ghz (Intel Quad Core) 4*3.2*4*2 = 102.4 G

GPU : ~240 @ 1.296 Ghz (Nvidia GeForce GTX280) 1.296*240*3 = 933 G

Why GPU for Computing?—Memory BW



(from *CUDA Programming Guide 04/02/2009*)

High memory bandwidth

- CPU : 21 GB/s
- GPU : 141.7 GB/s (Nvidia GTX280)

Why GPU for Computing? –High Performance/Price Peak flops/Power

Super computer	Rpeak (Tflops)	Expenses	Power (KW)	Peak flops/Power (Tflops/KW)
Roadrunner	1456.7	> 0.1 Billon $	2483	0.587
Magic Cube	233.47	>0.1 Billon￥	720	0.324

GPU	Rpeak (Tflops)	Expenses	Power (KW)	Peak flops/Power (Tflops/KW)
NV Tesla C1060	0.933	~10000 ￥	0.188	4.96
NV Geforce GTX 280	0.933	~3000 ￥	0.236	3.89

GPU+CPU: Future architecture for HPC?

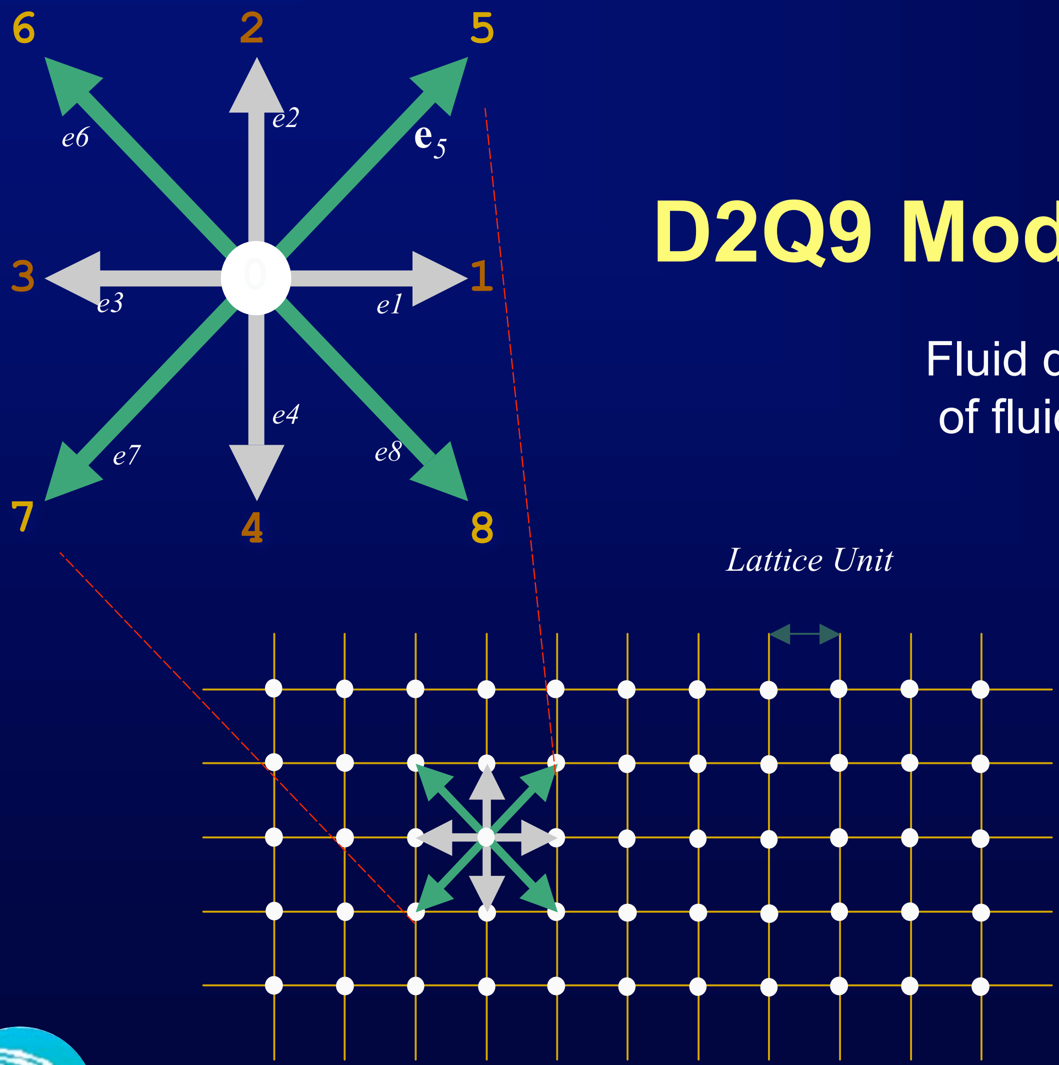
“GPUs have evolved to the point where many real-world applications are easily implemented on them and run significantly faster than on multi-core systems. Future computing architectures will be hybrid systems with parallel-core GPUs working in tandem with multi-core CPUs."

——Jack Dongarra （Linpack, Top500, US）

“A serious competitor for the multi-core CPU is represented by graphical processing units (GPUs), which are graphic cards used for scientific computing. There are four basic things about GPUs. They are fast and will get a lot faster. They are cheap, measured on a performance-per-dollar basis. They use less power than CPUs when compared on a performance-per-watt basis. ………For the near future, we expect that the hardware architecture will be a combination of specialised CPU and GPU type cores.”

——Hans Meuer（Chairman of ISC，Top500 , Germany）

LBM Basics



D2Q9 Model

Fluid described by a distribution function of fluid (quasi-)particles.

Lattice Boltzmann Algorithm

- Very similar to an explicit finite difference scheme

- Macroscopic fluid quantaties are derived from microscopic quantaties

density [illegible]

velocity [illegible]

- Collision

[illegible]

[illegible]

- Streaming

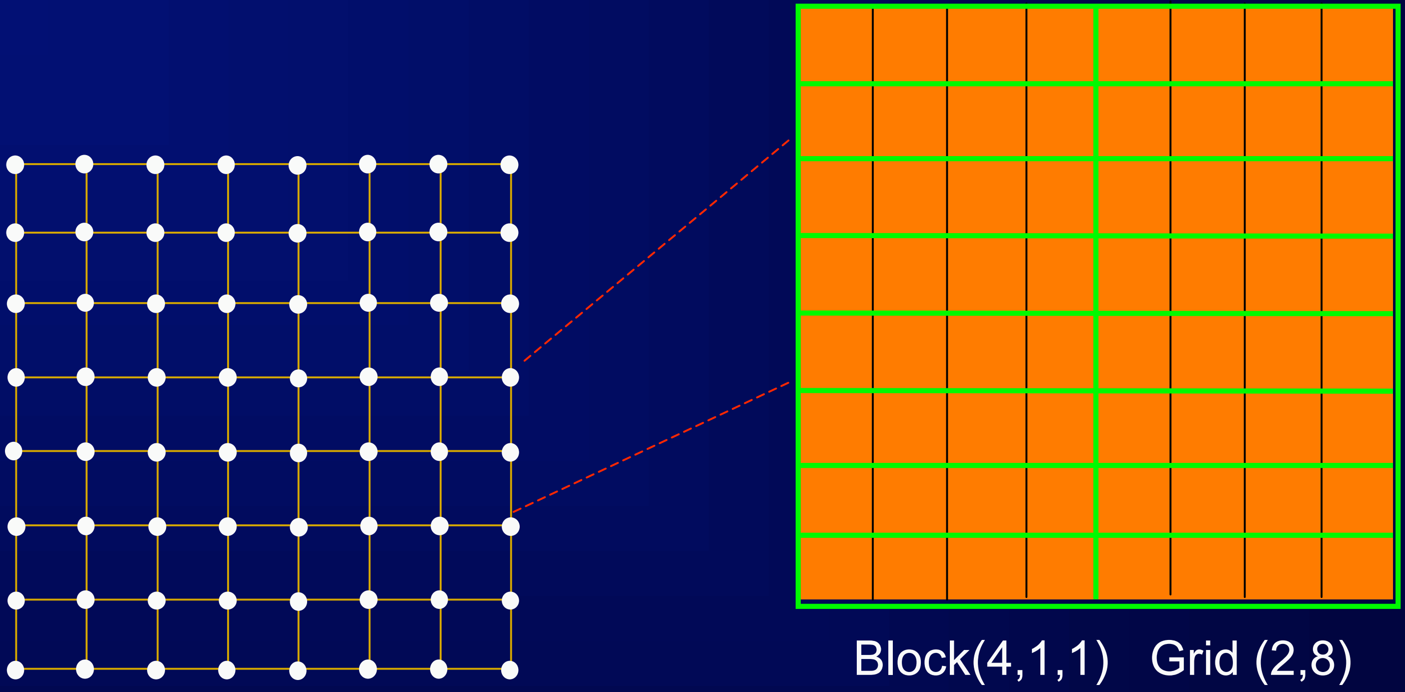
[illegible]

Serial implementation

Jonas Tolke, Computing and Visualization in Science, 2008: p. s00791-008-0120-2

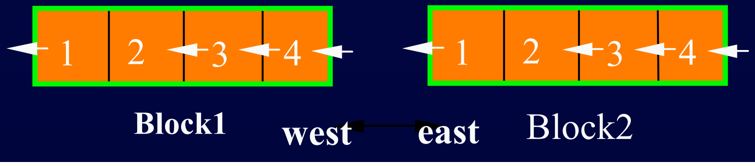
- ♦ two 1D array for the distribution function of each direction
- ♦ one for current step and next step
- ♦ three main kernel function
  - ♦ LBCollProp( ) : collosion and propagation
  - ♦ LBExchange( ) : exchange between thread blocks
  - ♦ LBBC( ): for the inlet and outlet boundary

LBCollProp( ) kernel



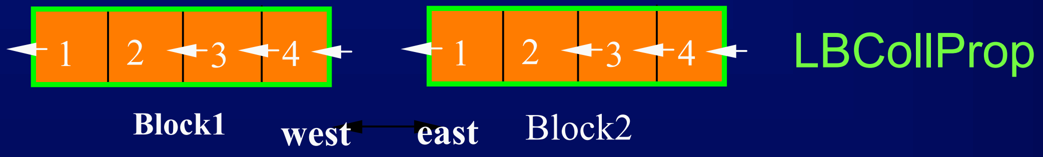
Lattice 8X8

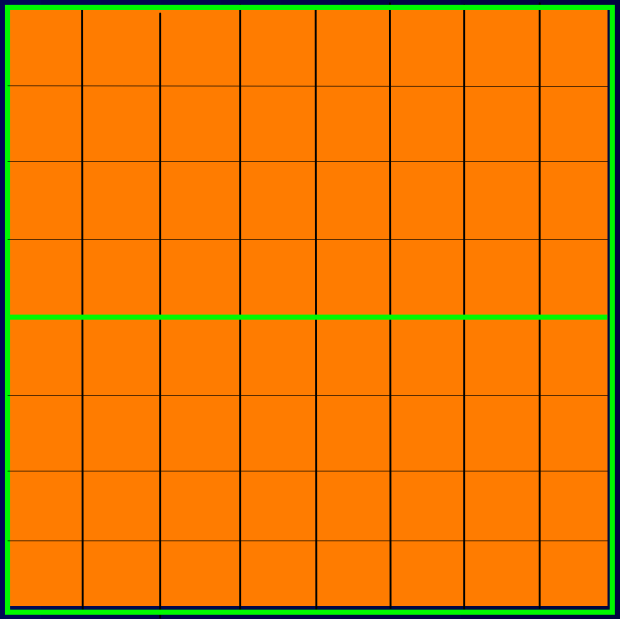
- each thread compute one lattice separately
- unite collision and propagation in the kernel
- share memory for the propagation along east and west



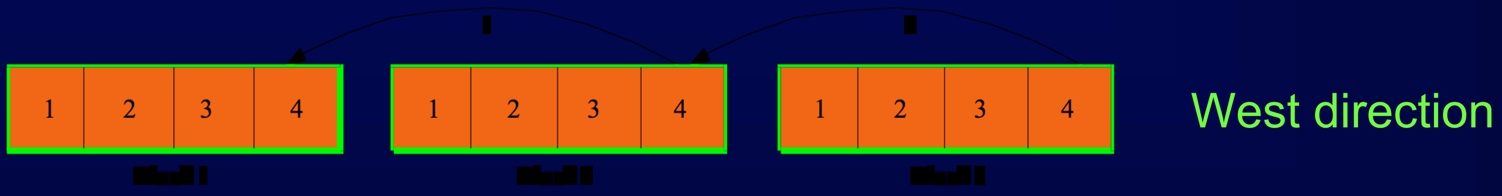
LBExchange ( ) kernel

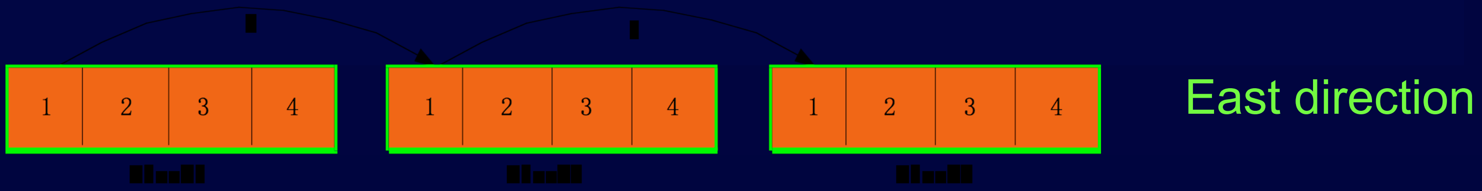
- exchange the distribute function along east and west
- each thread compute one line sequentially





Block(4,1,1) Grid (1,2)

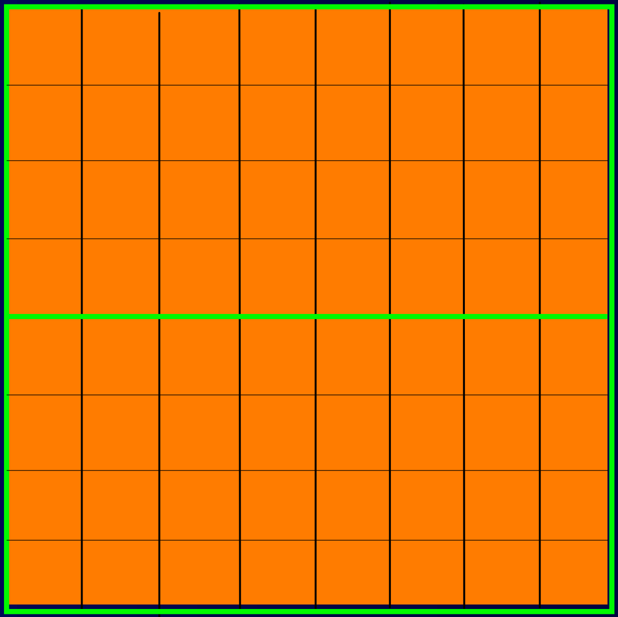




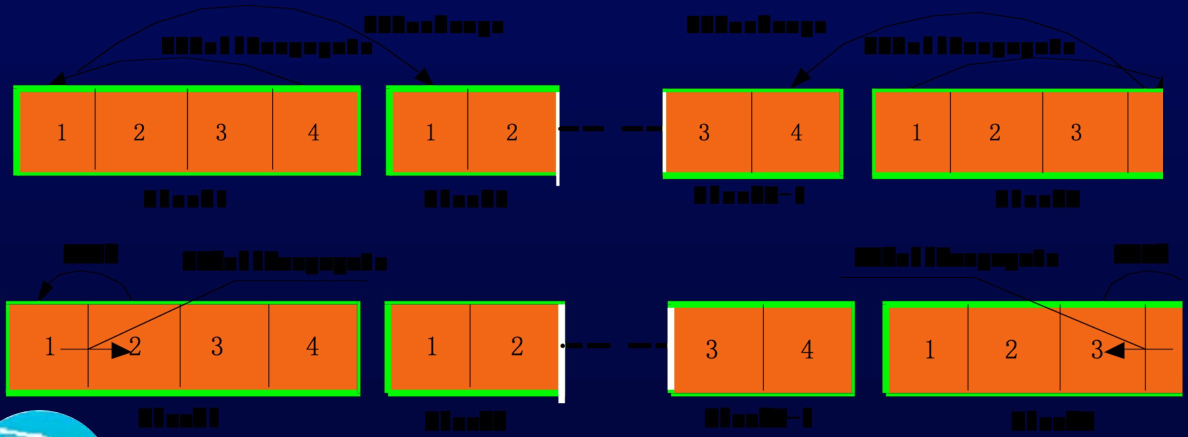
LBBC ( ) kernel

- modify the distribute function at inlet and outlet
- each thread compute one line sequentially,

two lattice (inlet and outlet)

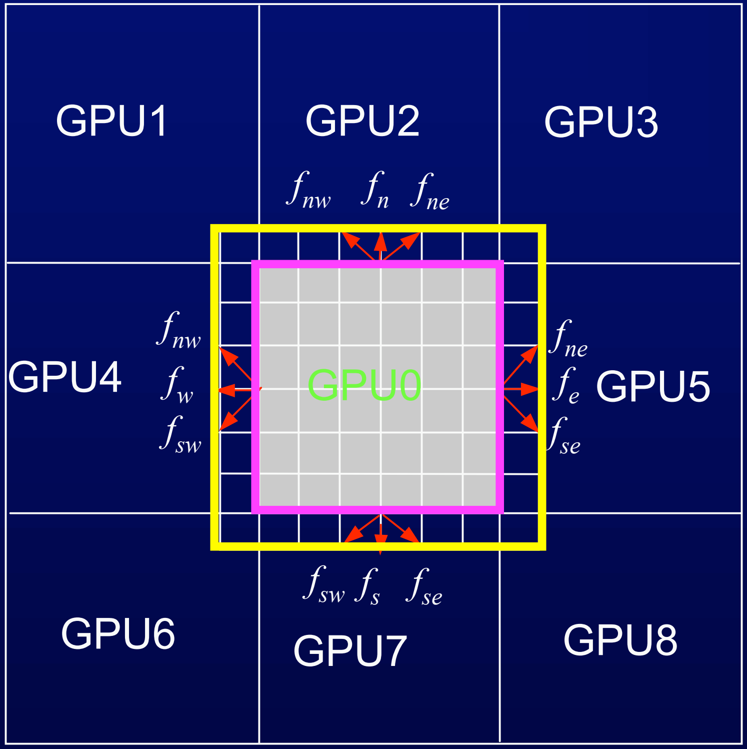


Block(4,1,1) Grid (1,2)



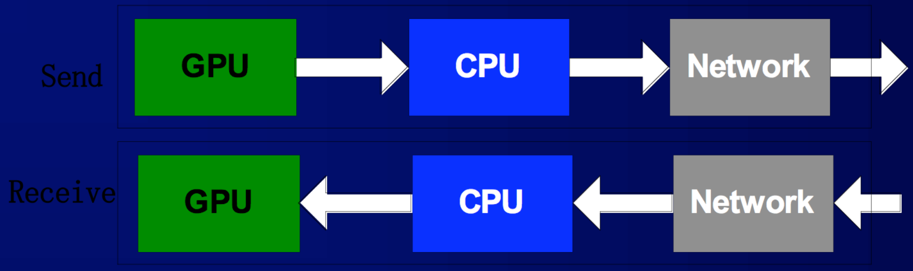
distributation function of inlet and outlet change

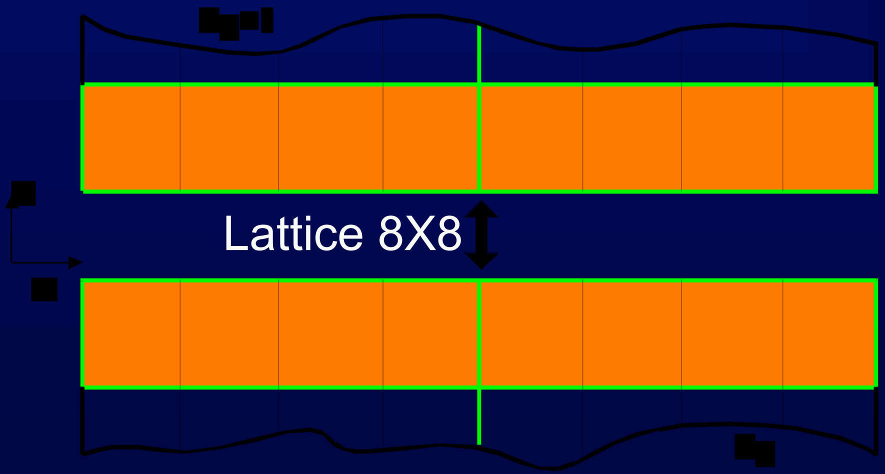
Parallel implementation

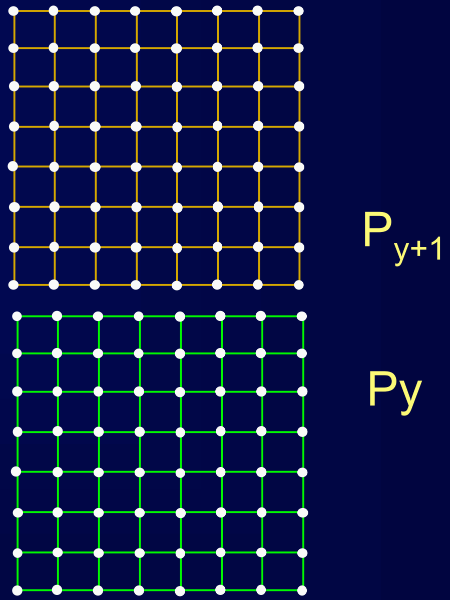


- Neighbor lattice transfer before collision
  - boundary lattice collision overlapped
  - communication scheme same
- Neighbor lattice transfer after collision
  - no overlapped computation
  - communication scheme complex

Communication for 1D partition

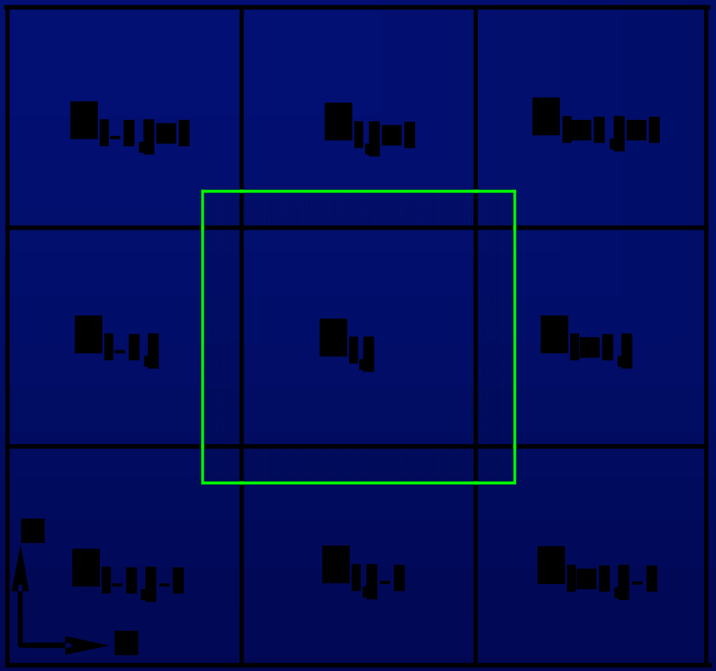


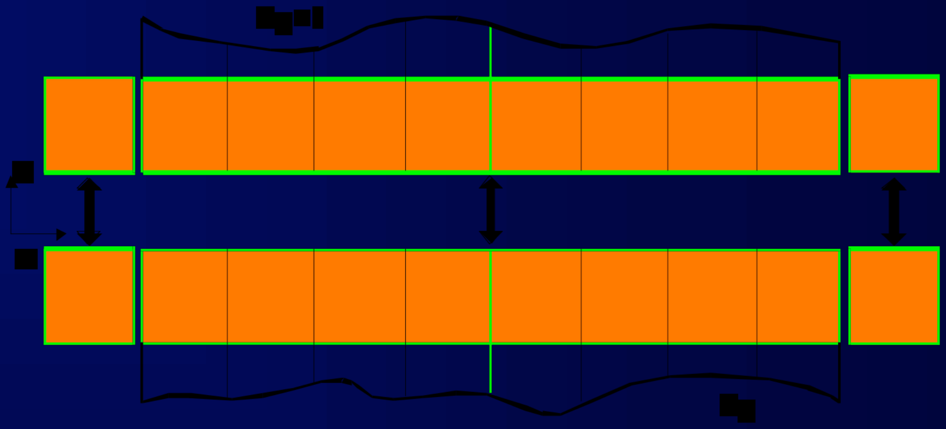




Block(4,1,1) Grid (2,1)

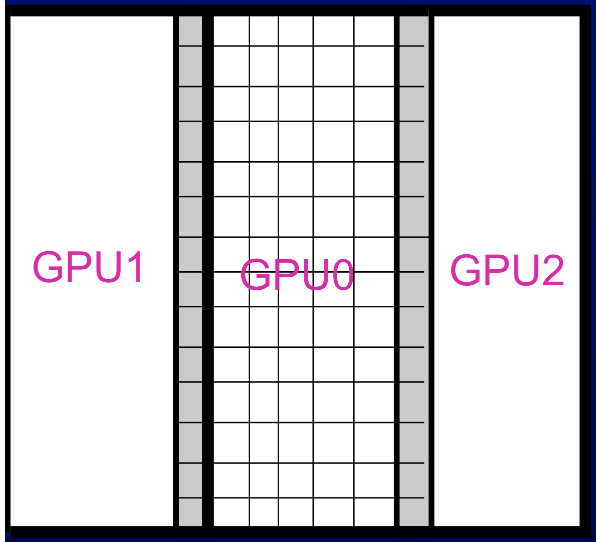
Communication for 2D partition


Shift communication scheme

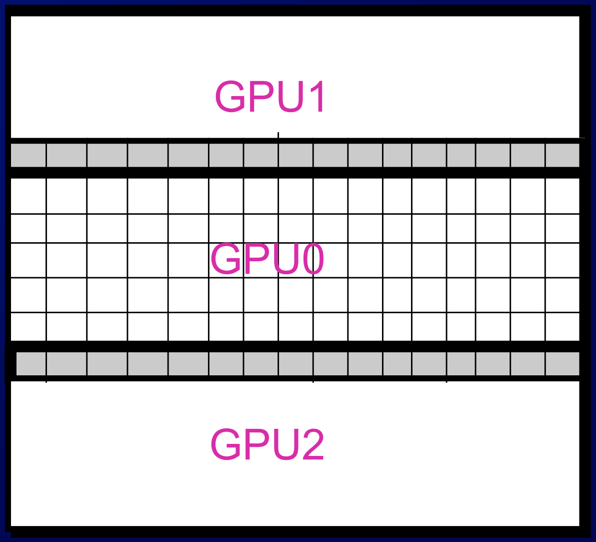

Block(4,1,1) Grid (2,1)

Kernel for corner lattice transfer in y direction
Block (2,1,1) Grid(1,1)

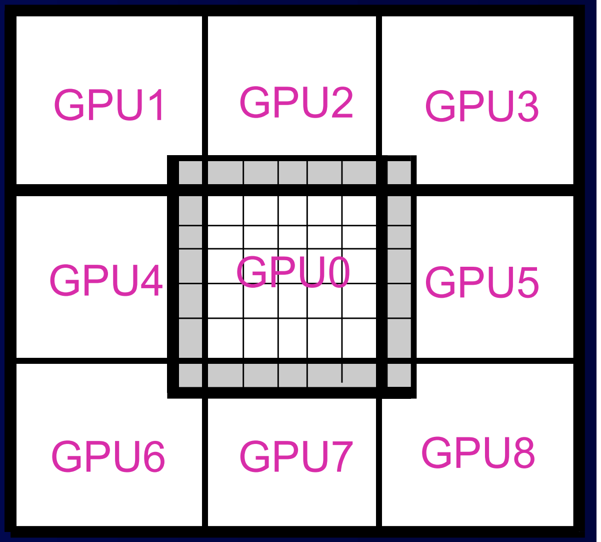
Domain of GPU with different partitions



Partition only along X

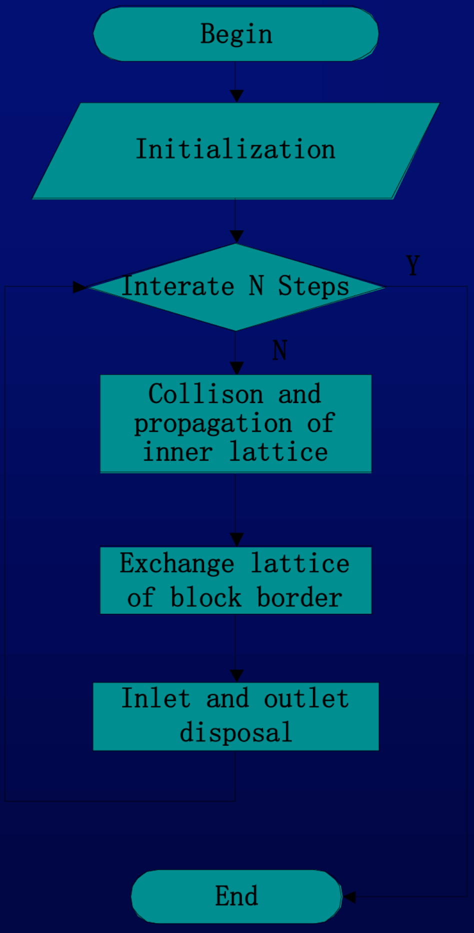


Partition only along Y

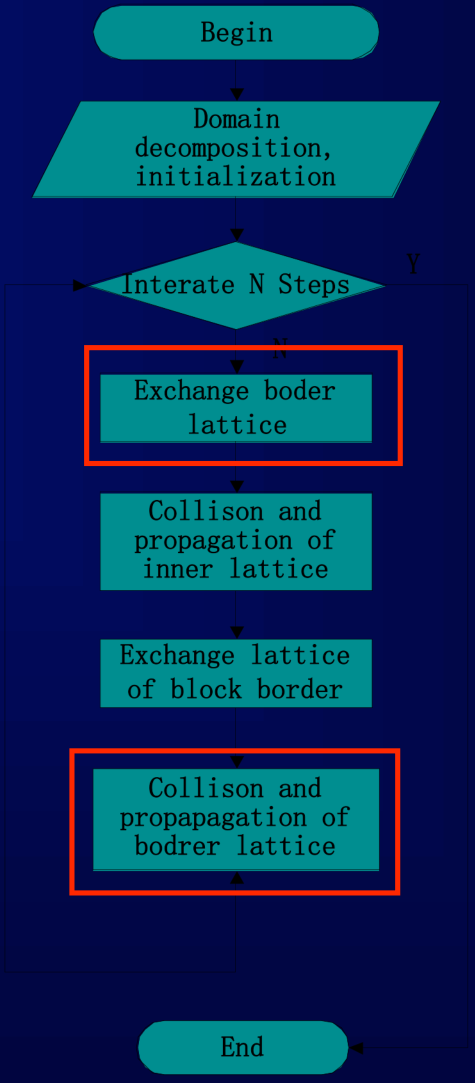


Partition along X and Y

Flow chart of GPU implementation

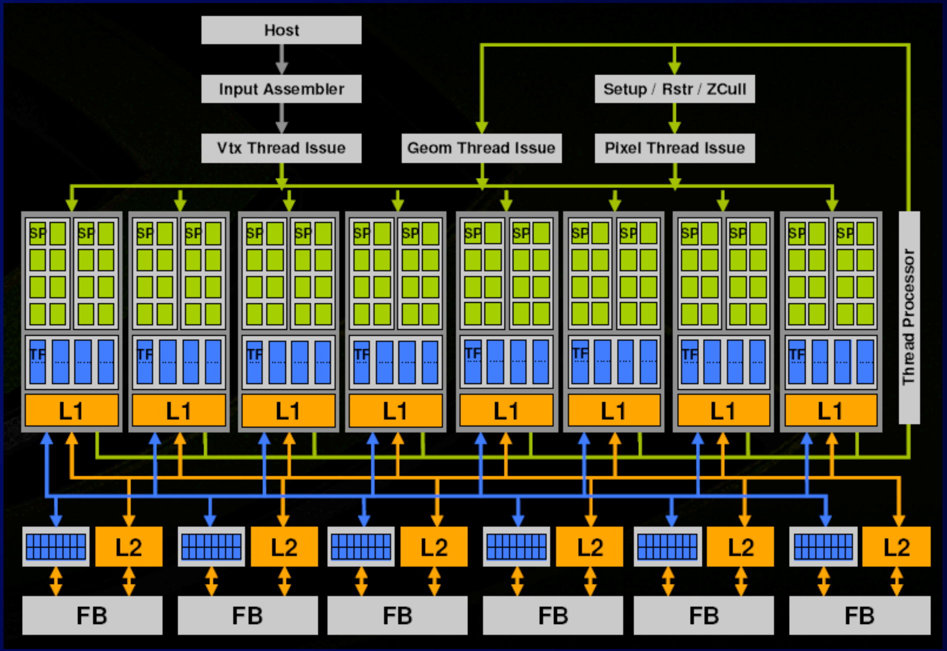

Begin
Initialization
Interate N Steps
Y
N
Collison and propagation of inner lattice
Exchange lattice of block border
Inlet and outlet disposal
End

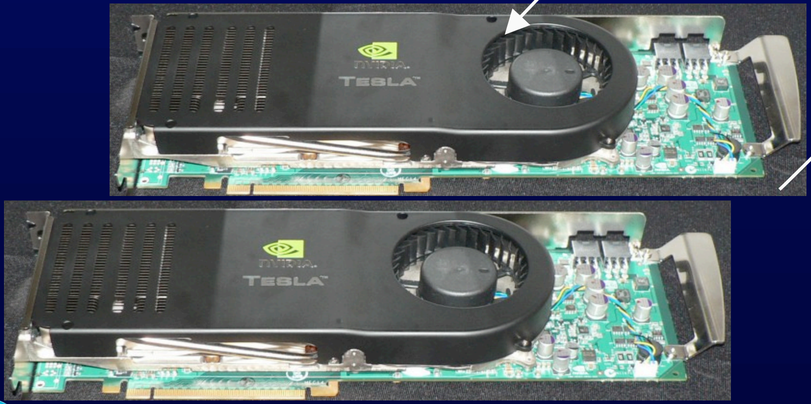
Serial


Begin
Domain decomposition, initialization
Interate N Steps
Y
N
Exchange boder lattice
Collison and propagation of inner lattice
Exchange lattice of block border
Collison and propapagation of bodrer lattice
End

Parallel

HARDWARE-- GPU Cluster





Tesla C870

16 Multiprocessors

Per MP:
8 Functional units
8192 Registers
16KB shared Memory

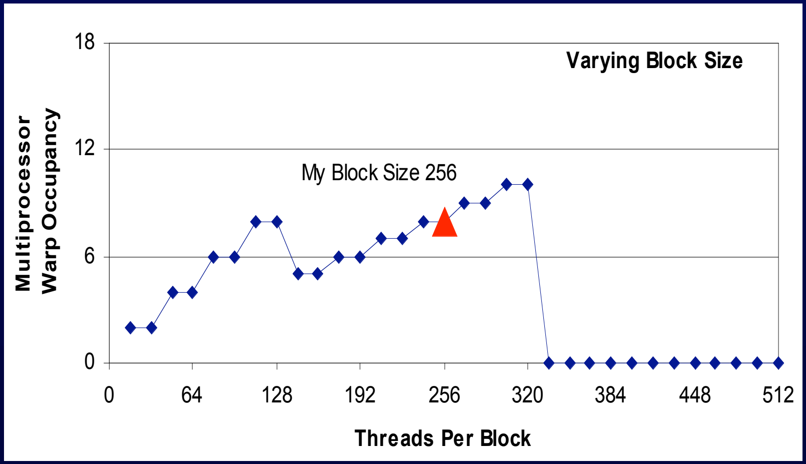
System Configuration

Peak performance：	127 Tflops Single Precision
Compute Node：	126×HP8600 Workstation
CPU：	252×Intel Xeon E5430 2.66GHz
GPU：	200×NV Tesla C870
	NVTesla c1060
	NV Gefroce GTX280
	NV Gerorce GTX295
	AMD HD4870X2
Network：	Gigabyte Ethernet
	DDR Infiniband
Switch：	H3C 7506R
Operating system：	CentOS 5.1
Software：	icc/gcc ,MPI, Cuda

Performance analysis –Block size

Average time per step with different mesh size and number of threads (ms)

Number of Thread	32	64	128	256
Mesh size				
1024×1024	5.42	3.26	2.88	2.71
2048×2048	28.47	16.13	12.84	11.15
4096×4096	115.44	64.66	51.67	45.96



利用Xeon 2.66GHz CPU计算的时间结果 (ms)

Mesh size	Time (ms)
1024×1024	72.20
2048×2048	288.34
4096×4096	1154.45

Performance analysis – GPU topology

Average time per step for multi GPUs with different topology (ms)

Processor Topology	Total Time	Comm time
1×2^1	24.08	1.56
1×2^2	27.50	4.52
2×1^1	108.07	1.09
2×1^2	111.46	5.00
1×4	25.60	13.28
4×1	69.20	12.28
2×2	77.51	25.66
2×4	54.74	22.25
4×2	56.01	27.50
1×8	20.85	9.19
8×1	42.56	13.42
2×8	42.79	17.53
8×2	41.21	14.67
4×4	60.09	38.50

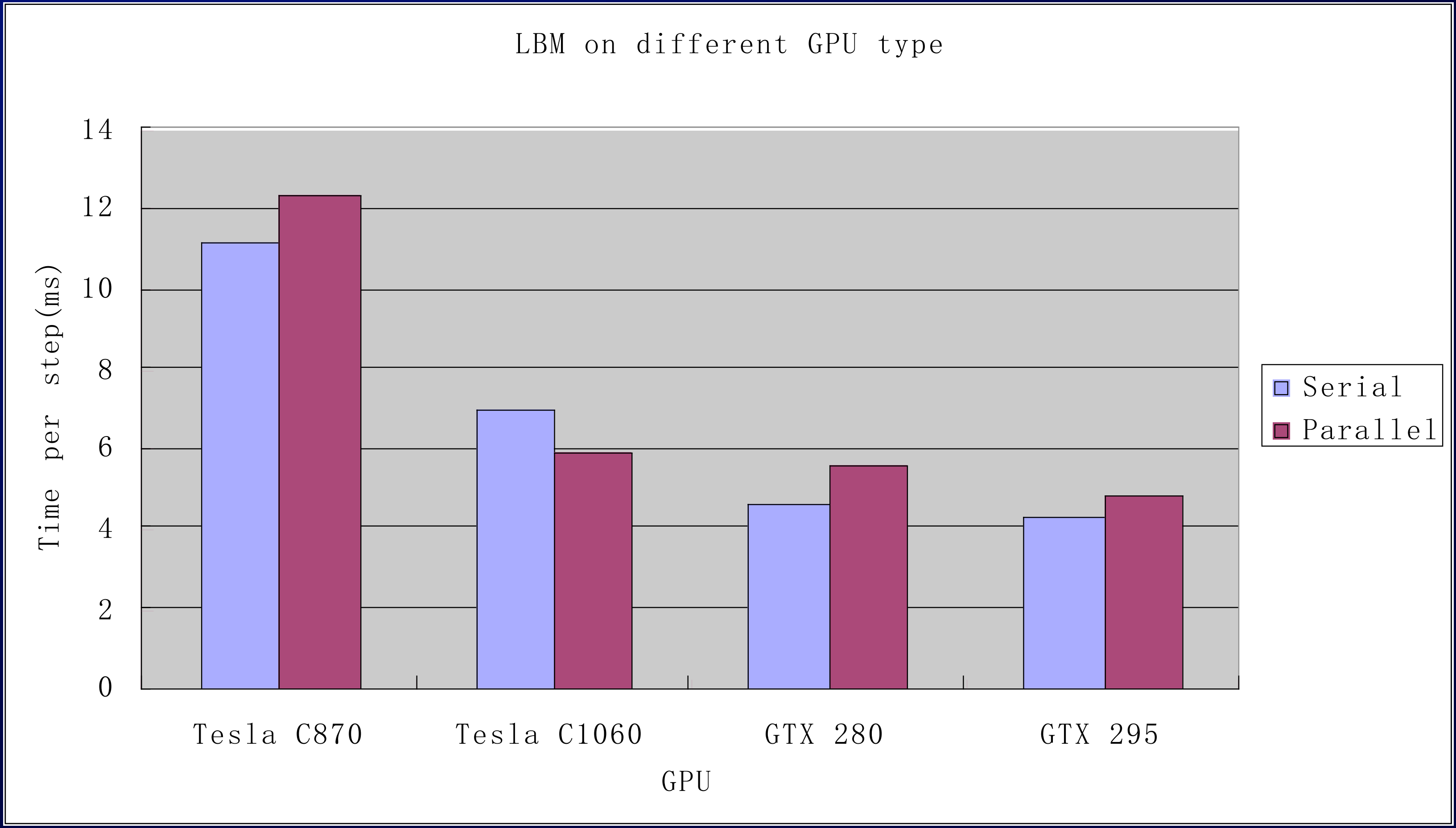
1-two GPU in same node, 2-two GPU in different nodes

Profiler for LBCollProp()

1X2 gst_coherent=[2358532]
gst_incoherent=[2]

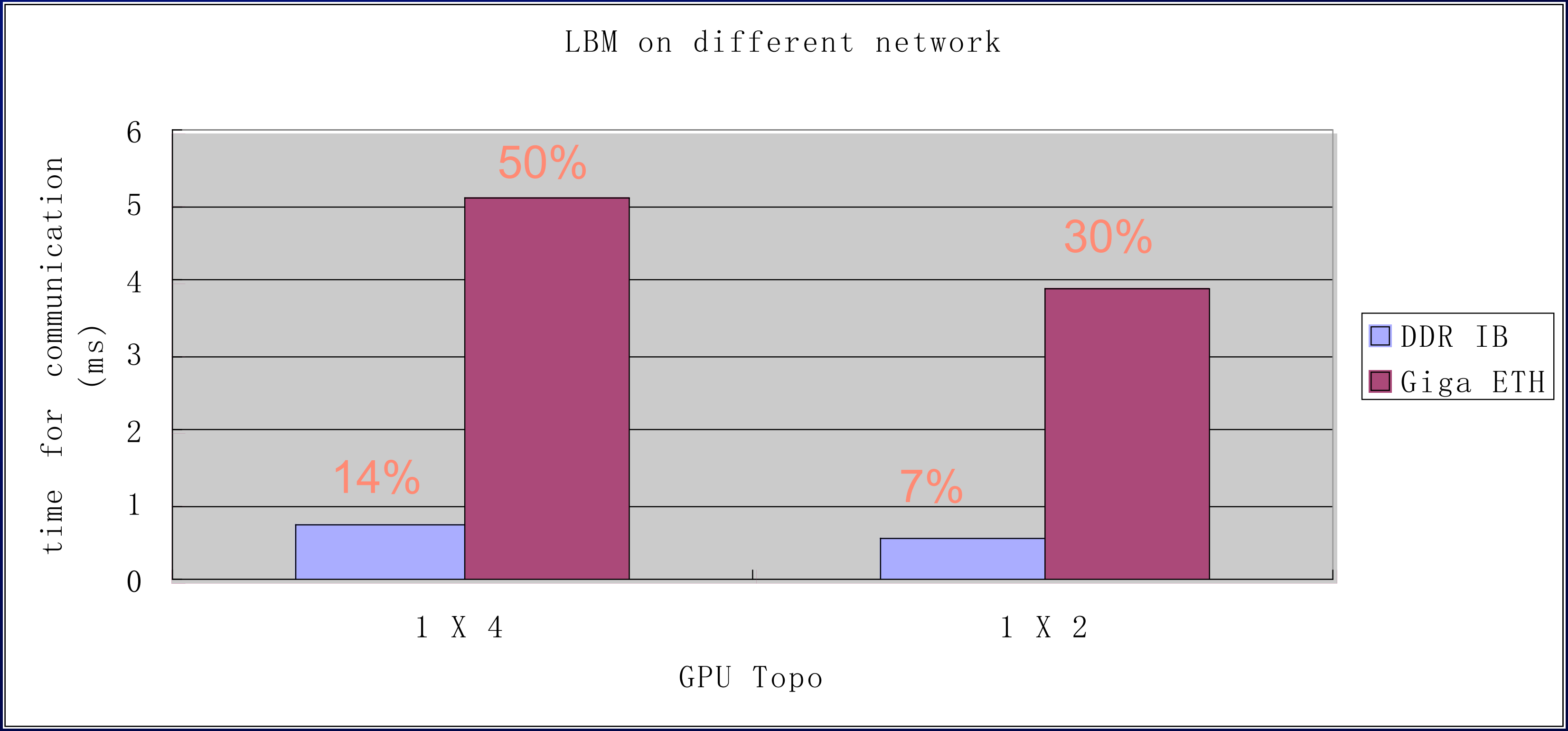
2X1 gst_coherent=[4]
gst_incoherent=[18871298]

Performance analysis –GPU type



Serial: 2048x2048, Block size=256
Parallel: 4096X4096, GPU topo (1X4), Block size =256

Performance analysis –Network type

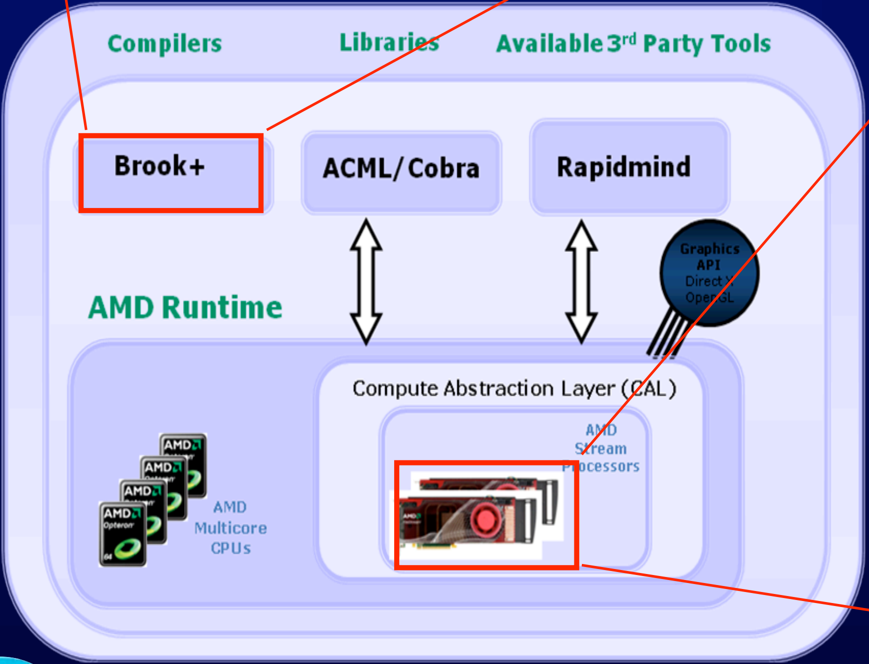


Mesh size: 4096x4096, only one GPU on each node

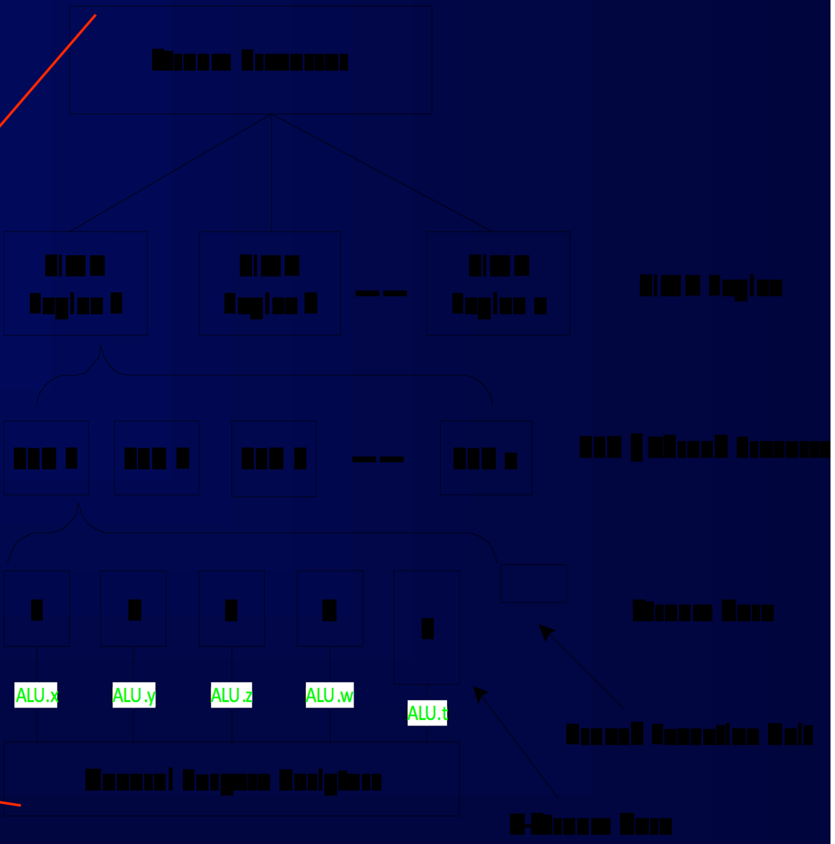
Implementation of LBM on AMD GPU

- Brcc
- Stream (Normal s<>,
  Gather input[][],
  Scatter
  output[][])
- Kernel
- Runtime API

CAL/IL, OpenCL?

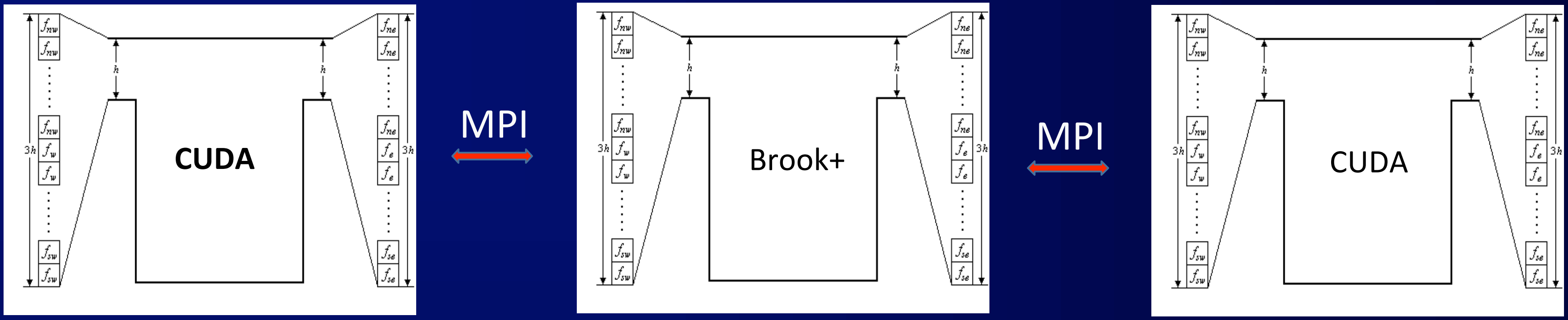


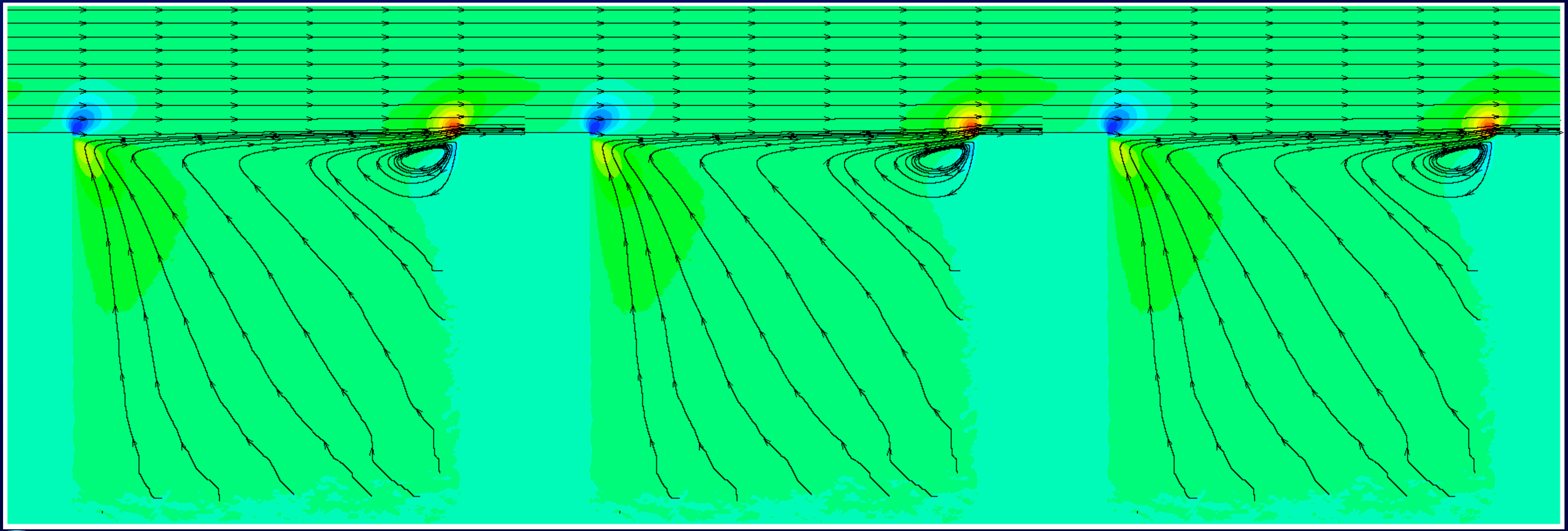
Stream Computing *



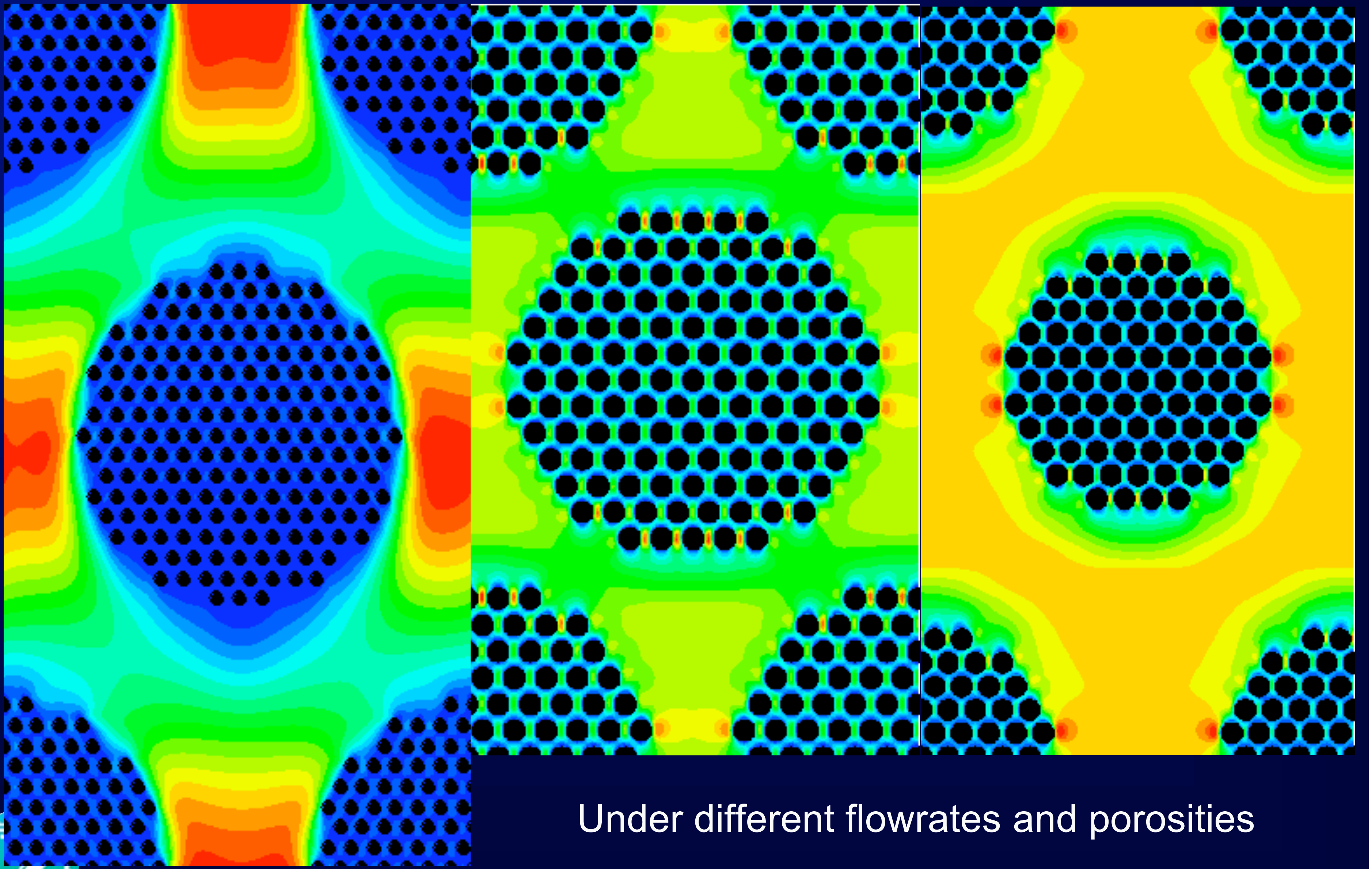
Work by Xipeng LI

Coupled computation with Brook+





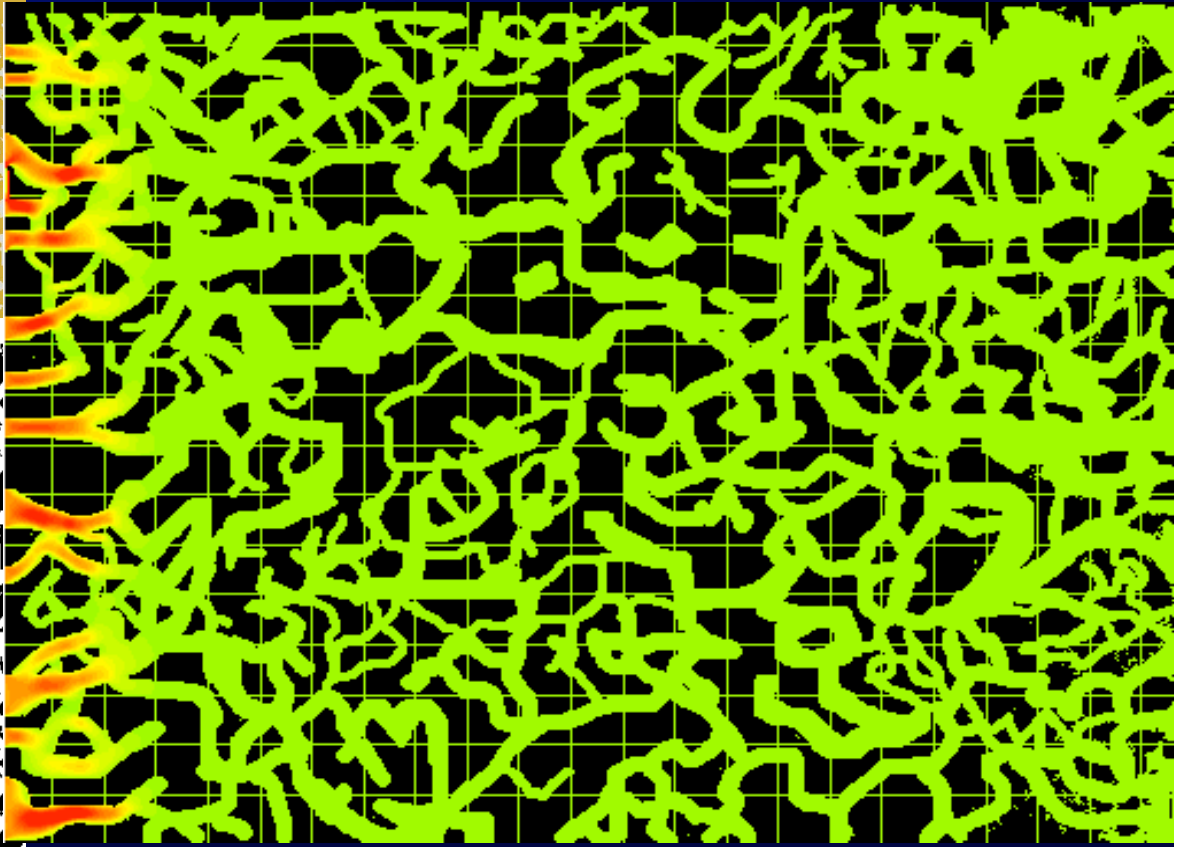
Meso-Scale Flow in Porous Media



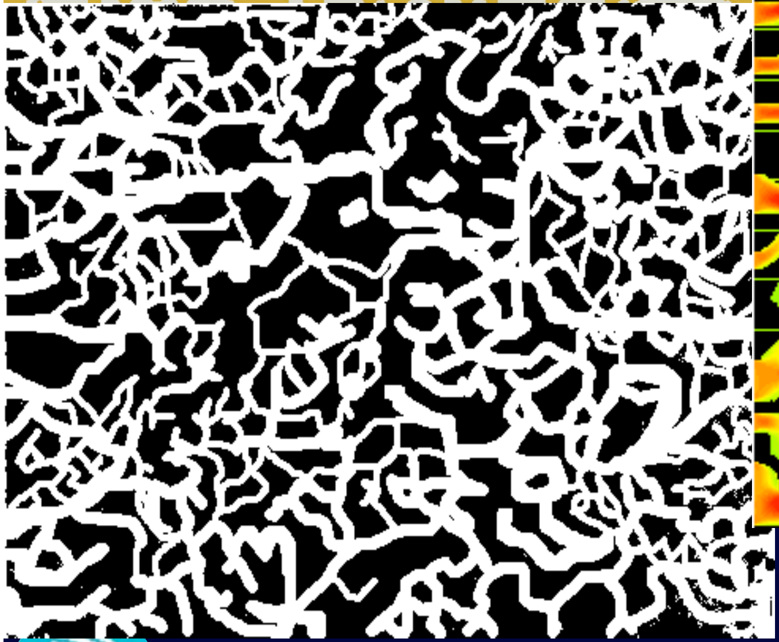
Under different flowrates and porosities

Picture of rock sample

picture

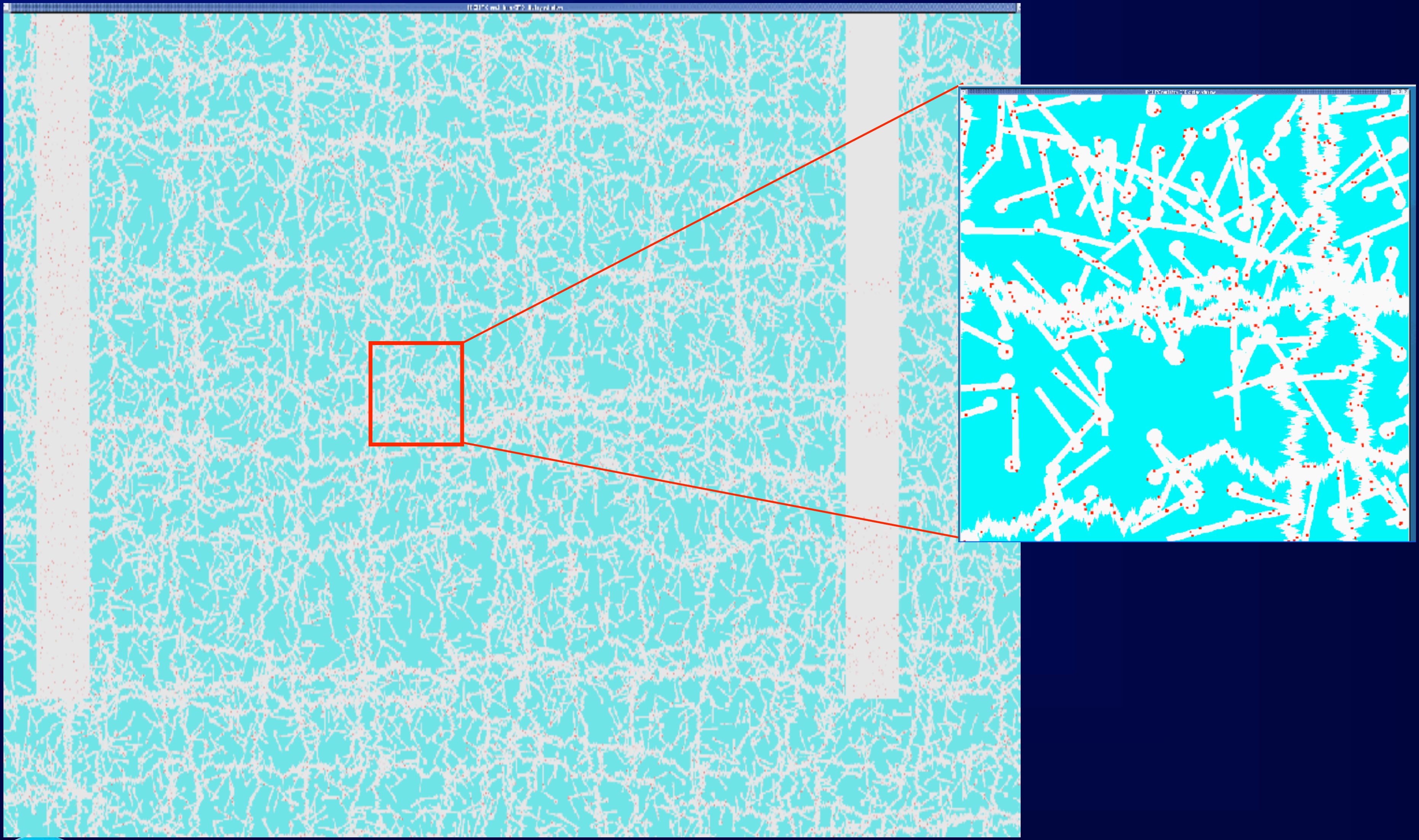


Flow field



initial condition

Simulation of fractured reservoir



Simulation of fractured reservoir using tracing particles on 64 GPUs
(16384x16384) (2048x2048 per GPU)

Conclusions

- ♦ LBM is suitable to be run large scale on GPU cluster
- ♦ The performance will be affected by some factors, such as thread block size, topology, GPU type, network
- ♦ Optimization is important to gain high performance
- ♦ The implementation of LBM coupled on both NVIDIA GPU and AMD GPU

Thank you !