

# Complex Shallow Water Simulations through Lattice-Boltzmann-based Mesoscopic Numerical Treatment Exploiting Hybrid Compute Nodes

Markus Geveler

Institut für Angewandte Mathematik  
TU Dortmund, Germany  
[markus.geveler@math.tu-dortmund.de](mailto:markus.geveler@math.tu-dortmund.de)

GS11  
Shallow Water Simulations on GPUs  
Long Beach, March 23, 2011

# Motivation

---

## The real world is ...

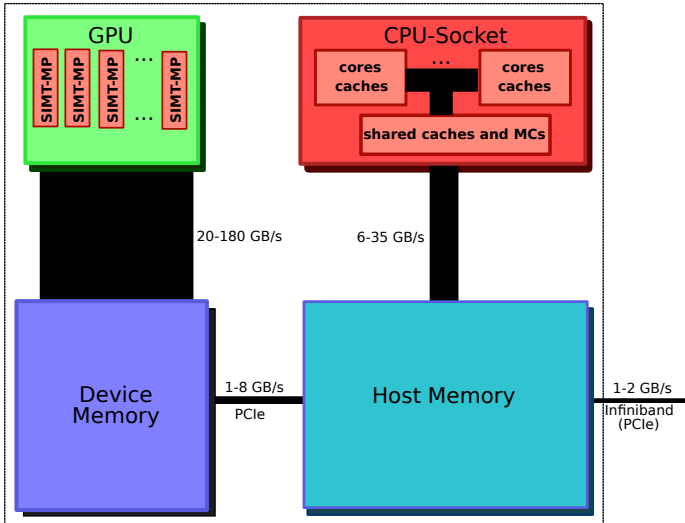
- multiphysics
- real-time

## Challenges in Fluid Simulation

- problems/functionality constraints constantly emerging
- computational resource/memory/storage/energy -demand increasing dramatically
- computational hardware undergoes paradigm-shift
  - parallel deprecates serial thinking
  - numerics must be fitted to hardware
  - multiple levels of parallelism (SIMD, shared on chip, distributed nodes,...)
  - chip-designs are heterogeneous
  - programming environments diverging
  - ...
- (software-)projects become hard to handle

# Motivation

When speaking of compute nodes...



# Motivation

---

## Simplification

- using SWE is a simplification on the model-level
- other levels:
  - *numerical method(s)*
  - computational accuracy

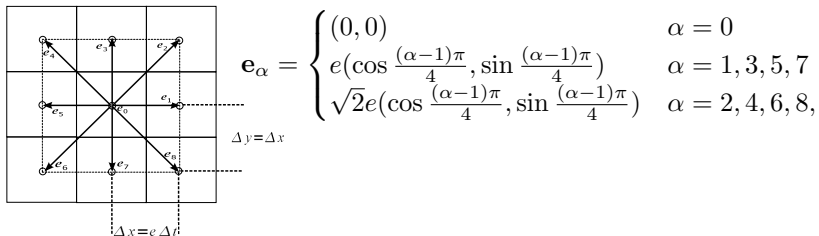
## Modification on the discrete and model levels and adjustment to hardware

- pure macroscopic approach: modification on the PDE-level → discretisation
- pure microscopic approach: modification on discrete-level → reconstruct continuum
- LBM is mesoscopic: allows both

# LBM for SWE

---

## Discrete phase space, general form of LBM: collision and streaming



## The LBE

$$f_\alpha(\mathbf{x} + \mathbf{e}_\alpha \Delta t, t + \Delta t) = f_\alpha(\mathbf{x}, t) + Q(f_\alpha, f_\beta), \quad \beta = 1, \dots, k .$$

# LBM for SWE

---

## Collision operator

$$f_{\alpha}^{\text{temp}}(\mathbf{x}, t) = f_{\alpha}(\mathbf{x}, t) - \frac{1}{\tau}(f_{\alpha} - f_{\alpha}^{\text{eq}})$$

## Equilibrium distribution for Shallow Water Equations

$$f_{\alpha}^{\text{eq}} = \begin{cases} h(1 - \frac{5gh}{6e^2} - \frac{2}{3e^2}u_i u_i) & \alpha = 0 \\ h(\frac{gh}{6e^2} + \frac{e_{\alpha i}u_i}{3e^2} + \frac{e_{\alpha j}u_i u_j}{2e^4} - \frac{u_i u_i}{6e^2}) & \alpha = 1, 3, 5, 7 \\ h(\frac{gh}{24e^2} + \frac{e_{\alpha i}u_i}{12e^2} + \frac{e_{\alpha j}u_i u_j}{8e^4} - \frac{u_i u_i}{24e^2}) & \alpha = 2, 4, 6, 8 \end{cases}$$

## Physical quantities

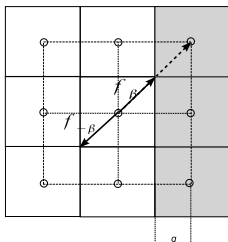
$$h(\mathbf{x}, t) = \sum_{\alpha} f_{\alpha}(\mathbf{x}, t) \quad \text{and} \quad u_i(\mathbf{x}, t) = \frac{1}{h(\mathbf{x}, t)} \sum_{\alpha} e_{\alpha i} f_{\alpha},$$

→ dry states?

# Boundary conditions

---

## Simple bounce-back



## Accuracy

- LBM in general: 2nd order in space, 2nd order in time
- bounce back gives 1st order in space for boundary sites
- often: porous media, non-axis-aligned geometry → sufficient?

# LBM for SWE

---

## Source terms become additive force terms

$$f_\alpha(\mathbf{x} + \mathbf{e}_\alpha \Delta t, t + \Delta t) = f_\alpha(\mathbf{x}, t) - \frac{1}{\tau}(f_\alpha - f_\alpha^{eq}) + \frac{\Delta t}{6e^2} e_{\alpha i} S_i, \quad \alpha = 0, \dots, 8.$$

with (bed slope and friction)

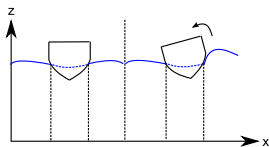
$$S_i = -g \left( h \frac{\partial b}{\partial x_i} + n_b^2 h^{-\frac{1}{3}} u_i \sqrt{u_j u_j} \right)$$

→ numerical scheme?

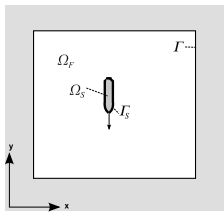
# SWE with FSI

---

## Idea



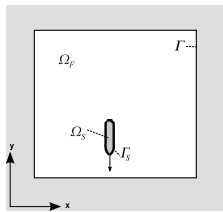
t-1



## Ingredients

- moving boundaries
- fluid initialisation
- momentum exchange
- *dynamic lattice*  $\rightarrow$  implementation

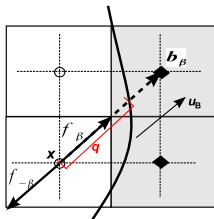
t



# SWE with FSI

---

## Boundary conditions for moving boundary: BFL



$$f_{-\beta}^{\text{temp}}(\mathbf{x}, t + \Delta t) = C_1(q)f_{\beta}(\mathbf{x}, t) + C_2(q)f_{\beta}(\mathbf{x} + \mathbf{e}_{-\beta}, t) + C_3(q)f_{-\beta}(\mathbf{x}, t) + C_4(q)2\Delta x w_{-\beta} c_s^{-2} [\mathbf{u}_B(\mathbf{b}_{\beta}) \cdot \mathbf{e}_{-\beta}],$$

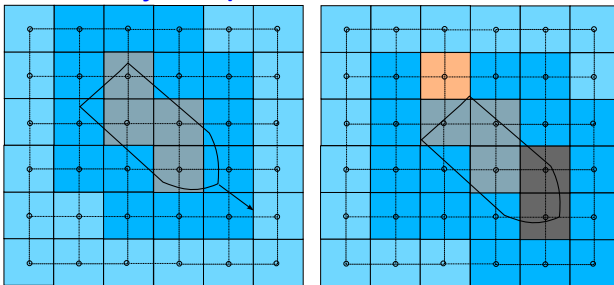
quality-down:  $q = 1/2 \Rightarrow$

$$f_{-\beta}^{\text{temp}}(\mathbf{x}, t + \Delta t) = 6\Delta x w_{-\beta} (\mathbf{u}_B(\mathbf{b}_{\beta}) \cdot \mathbf{e}_{-\beta}).$$

# SWE with FSI

---

## Fluid initialisation by extrapolation



$$\tilde{h}(\mathbf{x}, t + \Delta t) = 3h(\mathbf{x} + \mathbf{e}_{-\beta}\Delta t, t + \Delta t) - 3h(\mathbf{x} + 2(\mathbf{e}_{-\beta}\Delta t), t + \Delta t) + h(\mathbf{x} + 3(\mathbf{e}_{-\beta}\Delta t), t + \Delta t)$$

with  $q = 1/2$ :

$$\begin{aligned}\tilde{\mathbf{u}}(\mathbf{x}, t + \Delta t) &= 8/15\Delta x \mathbf{u}_B(\mathbf{b}_\beta, t + \Delta t) + 2/3\mathbf{u}(\mathbf{x} + \mathbf{e}_{-\beta}\Delta t, t + \Delta t) \\ &- 2/5\mathbf{u}(\mathbf{x} + 2(\mathbf{e}_{-\beta}\Delta t), t + \Delta t)\end{aligned}$$

# SWE with FSI

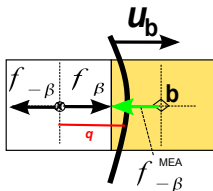
---

## Momentum Exchange Algorithm

$$f_{-\beta}^{\text{MEA}}(\mathbf{b}, t) = e_{\beta i} (f_{\beta}^{\text{temp}}(\mathbf{x}, t) + f_{-\beta}^{\text{temp}}(\mathbf{x}, t + \Delta t)).$$

The forces can be aggregated into the total force acting on  $\mathbf{b}$ :

$$F(\mathbf{b}, t) = \sum_{\alpha} f_{\alpha}^{\text{MEA}}(\mathbf{b}, t).$$



# Implementation: A word on HPC software

---

## HONEI

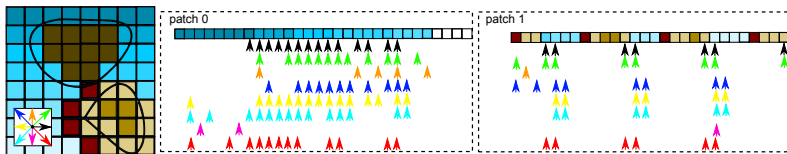
- primary design goals: abstract from target-hardware, exploit all resources in a node
- support natively:
  - x86 SIMD (handcrafted SSE2 using intrinsics)
  - x86 multi-core (pthreads)
  - distributed memory clusters (MPI)
  - NVIDIA GPUs (CUDA 3) + multiple GPUs + hybrid computations
  - Cell BE (libspe2)
- unittest + benchmark frameworks, visualisation
- build system to create SPE kernels for Cell BE, CUDA-kernels
- thread-management, MPI
- support for RTTI and exception-handling, memory transfers
- RPC system to call SPE programs from the PPE
- templates to facilitate development of new callable SPE functions and registering them with the RPC system
- automatic job scheduling
- ...

# Implementation: packed lattice

---

## One datastructure fits all

- compress lattice: stationary obstacles
- store contiguously in memory
- instationary obstacles (moving solids): colouring
- cut domain in one dimension



# Implementation: GPU

---

## Shared memory: general

- cache lattice velocities  $e_{\alpha i}$

## Memory arbitration

- external device memory management
- works 'behind the scenes'
- transfers only if necessary

## Complex kernels (force, FSI)

- avoid branch-divergence
- cache factors in force terms
- cache extrapolation weights

# Implementation: Inhomogeneous SWE

---

## Dry-states

- 1 define fluid sites with  $h < \epsilon$  as *dry*:  $h \leftarrow 0, u_i \leftarrow 0$
- 2 confine local oscillations:

$$\phi_U(x) = \max(-U, \min(U, x))$$

## Centred scheme for slope term

$S_i = S_i(\mathbf{x}, t)$  insufficient,  $S_i = S_i(\mathbf{x} + \frac{1}{2}\mathbf{e}_\alpha \Delta t, t + \Delta t)$  prohibitively expensive!

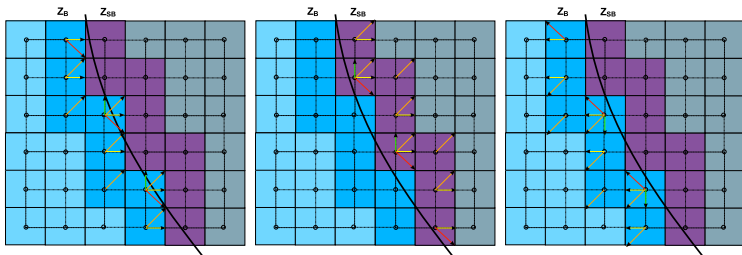
Solution:  $S_i = S_i(\mathbf{x} + \frac{1}{2}\mathbf{e}_\alpha \Delta t, t)$  ([Zhou 2004])

# Implementation: dynamic lattice

---

## Backward streaming

- repacking of lattice per timestep far too expensive
- fuse functionality, preserve parallel efficiency and reusability
- GPU: avoid branch-divergence:

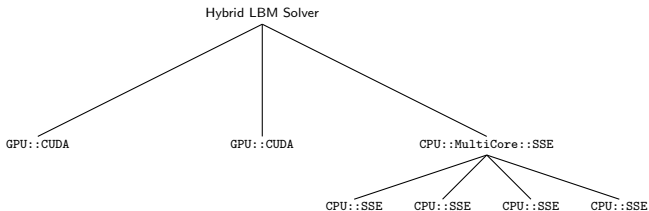


# Implementation: hybrid compute nodes

---

## Composition of hybrid solver: example

- 3 patches of domain initially
- 1 patch subdivided for the MultiCore backend
- each of the four MC-threads use the SSE backend
- resulting in utilisation of two GPUs and all cores of the (quadcore) CPU

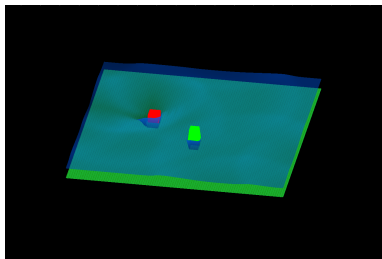


# Results

---

## Numerical properties

- approved: many different simulations
- well suited for CG
- finding solver parameters is hard
- timestep usually small
- stabilisation still experimental

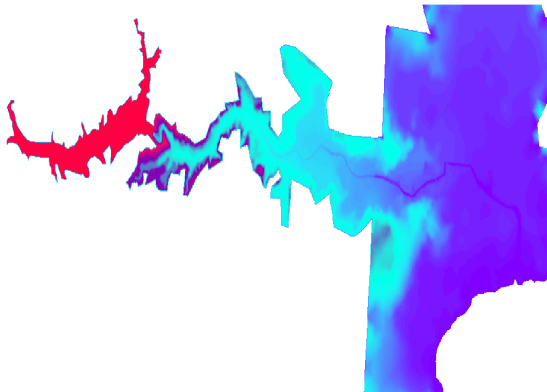


# Results

---

## Method even works for 'real-world' problems

- Example: Malpasset, France - arch dam-break tragedy (1959): 451 kills, ~\$70 M worth of damage

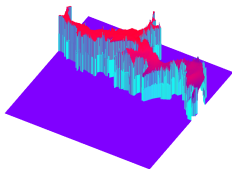
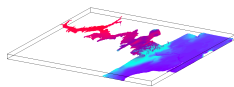
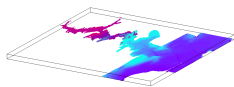
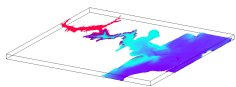


# Results

---

## Results

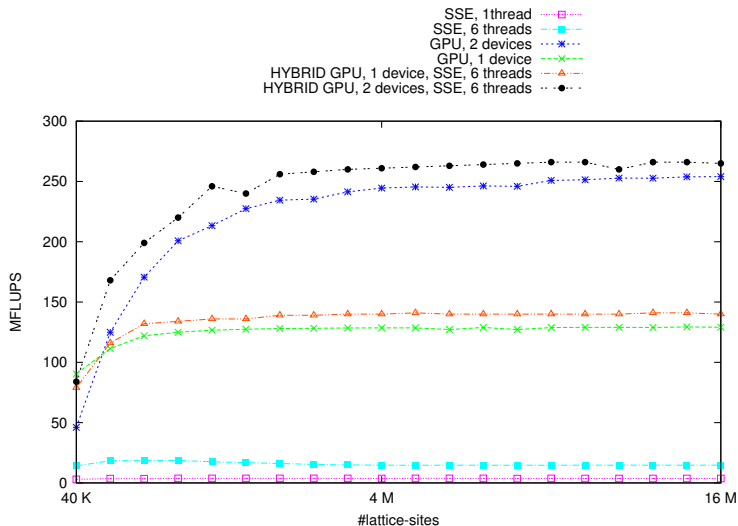
- Example: Malpasset dam-break tragedy (1959)



→ solution smooth, local discontinuities confined

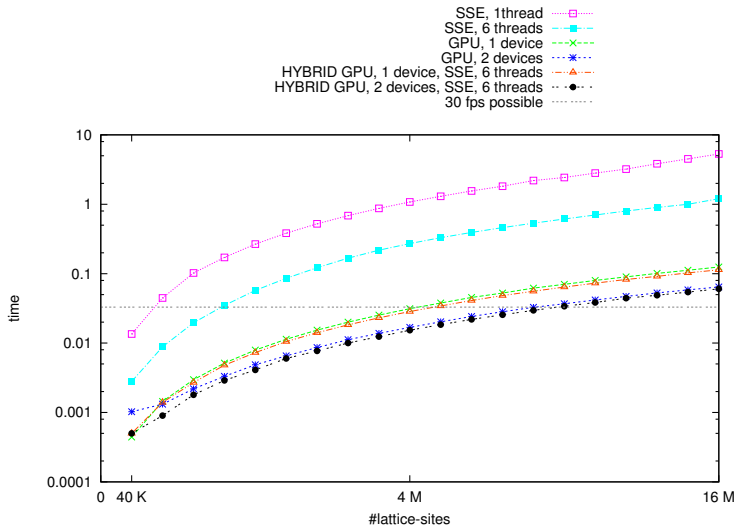
# Results

Performance: core i7 Gulftown 980 X, 3.33GHz Hexacore + 2 × FERMI (Tesla C2070)



# Results

## Simple Real-time CFD



# Conclusions

---

## Real-time and beyond

- LBM well suited for multi-level parallelism due to data locality
- up to 9 M lattice sites in real time possible (30 fps)
- Malpasset dam break simulation running at 1,100 fps
- GPU + CPU works well, only for very small problems → synchronisation too expensive

## Remarks

- basic LB-methods easy to implement, extend, maintain, ...
- very sophisticated HPC LBM codes available → [www.ska1b.de](http://www.ska1b.de)

# Acknowledgements

---

Supported by BMBF, *HPC Software für skalierbare Parallelrechner: SKALB project 01H08003D.*

Thanks to Dirk Ribbrock and all contributors to HONEI. Thanks to RRZE for hardware access.