Lattice-Boltzmann Simulation of the Shallow-Water Equations

with Fluid-Structure Interaction on Multi- and Many-core Processors

Markus Geveler, Dirk Ribbrock, Dominik Göddeke, Stefan Turek

Angewandte Mathematik, Technische Universität Dortmund

March 19, 2010

Overview

1. High performance CFD simulations
   - Hardware
   - Software
2. Shallow Water simulations with Fluid Structure Interaction
   - Composition of the method
   - Implementational issues
3. Results
4. Conclusion + Future Work

Overview

1. High performance CFD simulations
   - Hardware
   - Software

2. Shallow Water simulations with Fluid Structure Interaction
   - Composition of the method
   - Implementational issues

3. Results

4. Conclusion + Future Work

technische universität
dortmund

Different levels of parallelism

1. *microprocessor-level* (and beyond): IL, SIMD, multi-threading on a chip
2. *node-level*: parallel microprocessors in a compute-node
3. *clusters*: parallel compute-nodes in a cluster

Multi- and many-Core architectures

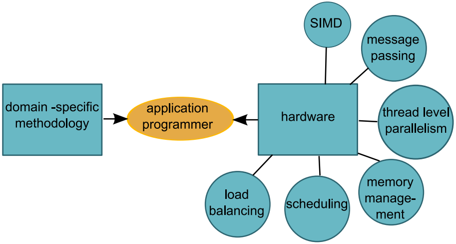
1. doubling the amount of transistors ⇒ doubling the amount of *cores*
2. *multi-core*-architectures have been established in the mass market
3. *many-core*-architectures too (GPUs) or: are about to come (Larrabee)
4. ever more *heterogeneities*: Cell BE

The challenge

Application programmers have to deal with...

...both, implementing specific functionality and the paradigm shift of the underlying hardware:

- high-level optimisation on the application level
- hardware-oriented implementation to exploit parallel hardware



today: focus on the method with respect to *multi-level* parallelism

Application programmers have to deal with...

- increase in data being acquired for fast ad-hoc processing or future analysis
- applications hunger for ever larger amounts of processing and memory resources
- application design is highly interdisciplinary

The impact on CFD simulation software

- CFD becomes ubiquitous: applications in many different fields
- fast access to result data is an economic issue, software solutions must be:
  - reliable, portable, designed for testability
  - *extendable*, flexible
- real-time constraints become more important

Model Problem

incompressible, laminar fluids with *free surfaces*

Render fluid volumes at *interactive rates* for use in a 3D environment (interactive demonstrations, computer games, tsunami forecast,...)

Quality criteria

- real-time constraint
- embedded in a virtual environment
- qualitative correctness: correct behaviour
- quantitative correctness: numerical quality less important
- stability (!)

Design goals

- implementation within a building blocks library for maximum reusability
- exploit multi-level parallelism

Overview

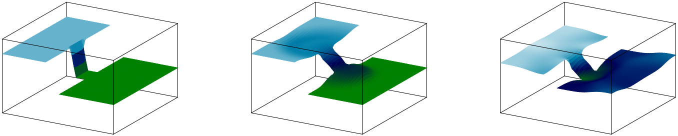
technische universität dortmund

Goal

quality-down for *speed-up*: approximate fluid by its surface

2D Shallow Water Equations

$$\frac{\partial h}{\partial t} + \frac{\partial (hu_j)}{\partial x_j} = 0 \quad \text{and} \quad \frac{\partial hu_i}{\partial t} + \frac{\partial (hu_i u_j)}{\partial x_j} + g\frac{\partial}{\partial x_i}\left(\frac{h^2}{2}\right) = 0,$$



Goal

enable interaction with the scene geometry: *slope* and *friction*

Inhomogeneous SWE

$$\frac{\partial h}{\partial t} + \frac{\partial(hu_j)}{\partial x_j} = 0 \quad \text{and} \quad \frac{\partial hu_i}{\partial t} + \frac{\partial(hu_iu_j)}{\partial x_j} + g\frac{\partial}{\partial x_i}(\frac{h^2}{2}) = S_i,$$

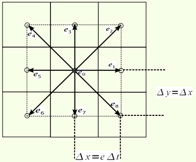
$$S_i = -g\left(h\frac{\partial b}{\partial x_i} + n_b^2 h^{-\frac{1}{3}} u_i \sqrt{u_j u_j}\right)$$

Problem: *dry-states* numerically difficult

Numerical method: LBM

technische universität
dortmund

Discrete phase space: D2Q9 ; equilibrium distributions for SWE



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

LBE with LBGK collision operator

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

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

technische universität dortmund

Extraction of physical quantities

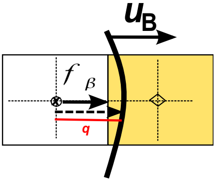
$$h(\mathbf{x},t) = \sum_\alpha f_\alpha(\mathbf{x},t) \quad \text{and} \quad u_i(\mathbf{x},t) = \frac{1}{h(\mathbf{x},t)} \sum_\alpha e_{\alpha i} f_\alpha$$

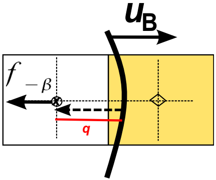
Coping with dry-states

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

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

$$u_i(\mathbf{x},t) = \begin{cases} \frac{1}{h(\mathbf{x},t)} \sum_\alpha e_{\alpha i} f_\alpha(\mathbf{x},t), & h(\mathbf{x},t) > \epsilon, \\ \phi_U\left(\frac{1}{h(\mathbf{x},t)}\right) \sum_\alpha e_{\alpha i} f_\alpha(\mathbf{x},t), & \text{otherwise.} \end{cases}$$





Moving Boundaries: BFL rule reduced to no-slip condition

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

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

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

Extrapolation of physical quantities: Equilibrium refill

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

$$\tilde{\mathbf{u}}(\mathbf{x}, t+\Delta t) = 2\Delta x \frac{\mathbf{u}_B(\mathbf{b}_\beta, t+\Delta t)}{q^2+3q+2} + 2q\frac{\mathbf{u}(\mathbf{x}+\mathbf{e}_{-\beta}\Delta t, t+\Delta t)}{q+1} - 2q\frac{\mathbf{u}(\mathbf{x}+2(\mathbf{e}_{-\beta}\Delta t), t+\Delta t)}{q+2}$$

Again, we substitute $q = \frac{1}{2}$ and we obtain

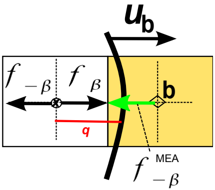
$$\begin{aligned}\tilde{\mathbf{u}}(\mathbf{x}, t+\Delta t) &= 8/15\Delta x \mathbf{u}_B(\mathbf{b}_\beta, t+\Delta t) + 2/3\mathbf{u}(\mathbf{x}+\mathbf{e}_{-\beta}\Delta t, t+\Delta t) \\ &- 2/5\mathbf{u}(\mathbf{x}+2(\mathbf{e}_{-\beta}\Delta t), t+\Delta t)\end{aligned}$$

Solid reaction: Momentum Exchange Algorithm

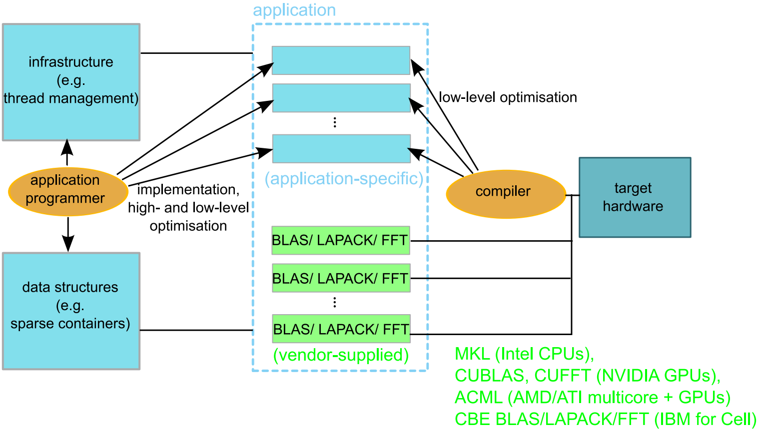
$$f_{-\beta}^{\mathrm{MEA}}(\mathbf{b},t) = e_{\beta i}(f_{\beta}^{\mathrm{temp}}(\mathbf{x},t) + f_{-\beta}^{\mathrm{temp}}(\mathbf{x},t+\Delta t)).$$

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

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



Implementation

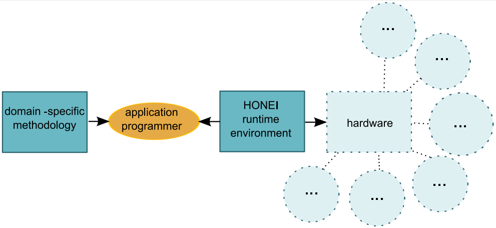


but: Relying upon compilers? Dealing with hardware details?

Framework:HONEI libraries

Primary design-goal:

Abstract from target-hardware!



Backends:

- x86 SIMD (handcrafted SSE4 using intrinsics)
- x86 multi-core (pthreads)
- distributed memory clusters (MPI)
- NVIDIA GPUs (CUDA 2.2) + multiple GPUs
- Cell BE (libspe2)

HONEI features

technische universität dortmund

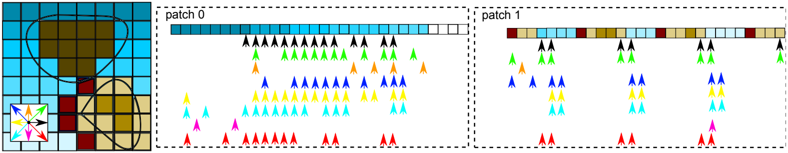
Building applications upon the provided operations/solvers

- generic programming $\Rightarrow$ clean interfaces:
  `ScaledSum<tags::CPU::SSE>::value(a, x, y)`
  $\Rightarrow$ implementational details kept away from the programmer
- solvers/operations can be plugged together:
  `ConjugateGradients<tags::CPU, methods::JAC,...>::value(...)`
- fixed + mixed precision implementations
- applications run automatically with all the supported backends
- backend combination (example):
  `ScaledSum<tags::Multicore::SSE>::value(a, x, y)`
- multiple device support e.g. `tags::CPU::SSE` + `tags::GPU::CUDA`
- straight-forward CPU implementation for numerical comparison (`tags::CPU`)

Step 1: Data structures

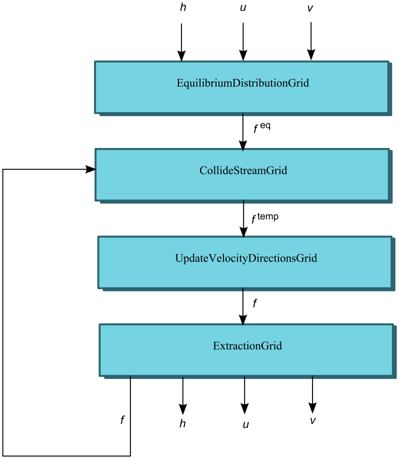
Packed lattice, domain decomposition

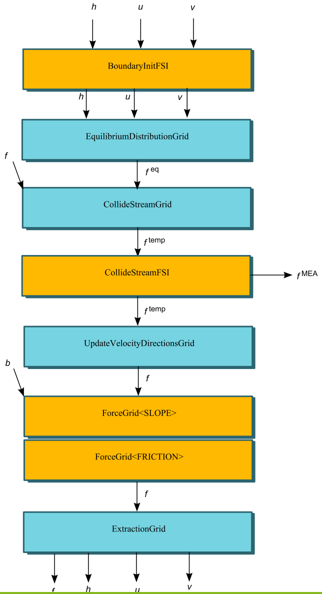
- compress lattice: stationary obstacles
- store contiguously in memory
- instationary obstacles (moving solids): colouring
- cut domain in one dimension



Step 2: Define building blocks

technische universität dortmund





Step 3: App.-specific kernels

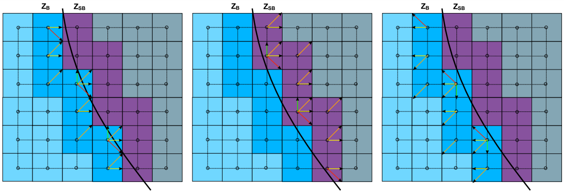
technische universität dortmund

Task

fuse functionality, preserve parallel efficiency and reusability.

Example: Advanced streaming operator - functionality

- use standard LBGK for most of the domain
- perform *backward streaming* for moving solids only: avoid branch divergence
- compute BFL values
- perform MEA



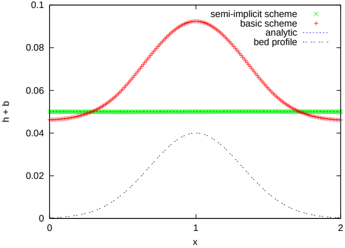
Step 4: Tune numerics

Task

Use most simple and fast schemes, match functionality constraints

Example: Source term - numeric scheme

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



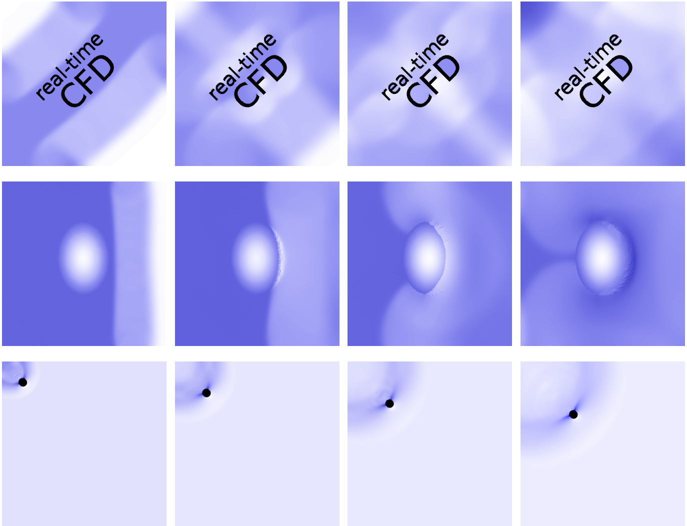
Use $S_i = S_i(\mathbf{x} + \frac{1}{2}\mathbf{e}_\alpha \Delta t, t)$ instead ([Zhou 2004])

Overview

technische universität dortmund

1. High performance CFD simulations
   - Hardware
   - Software
2. Shallow Water simulations with Fluid Structure Interaction
   - Composition of the method
   - Implementational issues
3. Results
4. Conclusion + Future Work

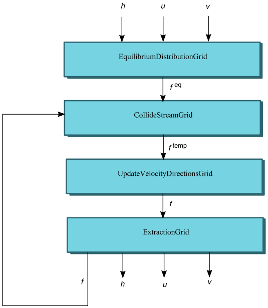
Results



technische universität dortmund

Hardware

Core i7 CPU, GTX 285 GPU, QS22 Cell

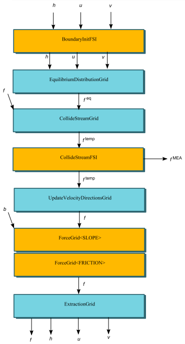


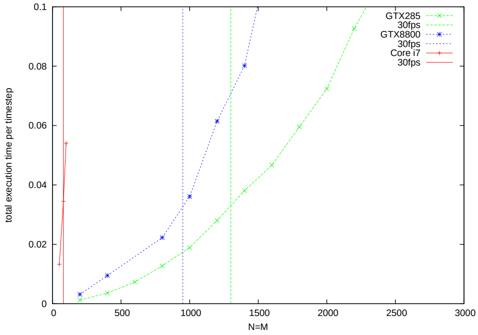
Type	LBM (1500^2 sites)	
	MFLUPS	Speedup
QS22 Blade	22.36	-
1 CPU core	15	-
4 CPU cores	40	2.7
1 GPU	193	12.9
2 GPUs	348	23
4 CPU cores + 1 GPU	217	14.5
4 CPU cores + 2 GPUs	362	24.1

technische universität dortmund

Real-time performance on the GPU - hardware

GTX 8800, GTX 285





Overview

technische universität
dortmund

1. High performance CFD simulations
   - Hardware
   - Software
2. Shallow Water simulations with Fluid Structure Interaction
   - Composition of the method
   - Implementational issues
3. Results
4. Conclusion + Future Work

Conclusion + Future Work

technische universität dortmund

Lessons learned

- Advanced CFD applications can be performed in real-time on a single node with reasonable resolution
- The LBM is well suited for this kind of application: it is
  - easy to extend on the discrete level
  - in its basics, well suited for spatial parallelism
- Drawbacks:
  - presented method only first order accurate
  - in general: hard to preserve locality
  - always: hard to stabilise

Perspectives

- More numerical and performance comparisons with state-of-the-art SWE solvers (FD, FV)
- Large scale faster-than-real-time simulations for tsunami forecast
- Use higher order numerics for real-time simulations (FEM)

Acknowledgements

Supported by DFG, projects TU 102/22-1, TU 102/22-2
and BMBF, *HPC Software für skalierbare Parallelrechner: SKALB project 01IH08003D*.
Thanks to IBM for access to Cell hardware.