Technische Universität Braunschweig

High performance computing on GPUs

Finite Volumes & Shallow water simulation

Thorsten Grahs, 06.01.2014

Overview

- **Finite Volumes**
- **Shallow water equations**
- **Discretisation**

Motivation

High Performance Computing

- not practiced as an end in itself
- used to solve real-world problems
- real world probems are transformed into the language of science by

Mathematical models

Mathematical model

a representation of the essential aspects of an existing system which presents knowledge of that system in usable form.
(Eykhoff 1974)

Mathematical modeling

Real world

Model

- (Navier-Stokes equation)

$$
\begin{aligned}
\nabla \cdot u &= 0 \\
\frac{\partial u}{\partial t} &= -(u \cdot \nabla) u - \frac{1}{\rho} \nabla p \\
&\quad + \nu \nabla^2 u + f
\end{aligned}
$$

- Boundary Conditions (B.C.)
- Initial Values (I.V.)

Mathematical modeling

Typically, one is interested in models that are

- Dynamic
  i.e. account for changes in time
- Heterogeneous
  i.e. account for heterogeneous systems

Typically represented with

Partial Differential Equations (PDEs)

Approximation techniques

In order to derive a numerical solution create a

- discretisation of the equation
  (i.e. map the continous description to a finite discrete subset)
- this formulation can be handled by a computer/cluster/GPU

Common approaches:

- Finite Differences
- Finite Elements
- Finite Volumes

We already handled on example/approach:

- 2D heat conduction via Finite Differences

Hyperbolic conservation laws

Describes systems where quantaties are conserved, f.i.

- mass
- momentum
- energy
- heat
- ...

Hyperbolic PDEs describe wave phanomena

- mostly time-dependent problems

Hyperbolic conservation laws

Application areas

- Acoustics
- Electromagnetics
- Seismic problems
- Optics
- Fluid mechanics
- ...

Simplest form of a conservation law

$$\frac{\partial}{\partial t}u(x,t) + \frac{\partial}{\partial x}f[u(x,t)] = 0$$

- $u(x,t)$ vector of conserved quantaties
- $f(u)$ Flux function

Finite Volume approach

- Simulation domain is divided into *grid cells* (finite volumes [FV])
- regarding the solution as the approximation of an integral over those finite volumes (instead of pointwise approximation at grid points as in finite differences)
- Spatial integration over finite Volume FV
- and temporal integration over a small time intervall Δt:

$$\int_{t}^{t+\Delta t} \partial_t \int_{FV} [u + \mathrm{div} f(u)]\, dV\, dt = \int_{\Delta t}^{t+\Delta t} \int_{FV} \partial_t u\, dV\, dt + \int_{t}^{t+\Delta t} \int_{FV} \mathrm{div} f(u) dV\, dt$$

Application of Gauß-Green theoreme

$$\int\limits_{t}^{t+\Delta t} \partial_t \int\limits_{FV} [u + \mathrm{div} f(u)]\, dV\, dt = \int\limits_{\Delta t}^{t+\Delta t} \int_{FV} \partial_t u \, dV\, dt$$

$$+ \int\limits_{t}^{t+\Delta t} \int_{\partial FV} \mathbf{n} \cdot f(u) dA\, dt$$

- Transforms Flux from volume into area integral
- Advantage: Compute Fluxes through cell faces
- Question: How to approximae Flux integral

Finite Volumes

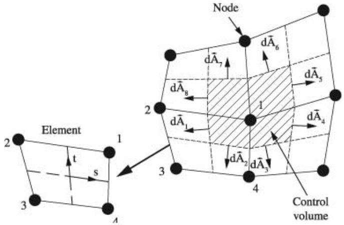
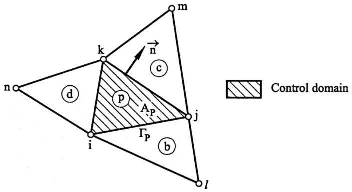


Figure 4. Control volume formation.



Averaged quantaties

- Regard averaged quantaties on Finite Volume element Ω_{ij}

$$U_{ij}(t) := \frac{1}{V_{KV}} \int\limits_{KV} u dV$$

- This the first integral becomes

$$\frac{\partial}{\partial t} \int\limits_{t^n}^{t^{n+1}} \int\limits_{\Omega_{ij}} [u] \, dV \, dt \; = \; (U_{i,j}^{n+1} - U_{i,j}^{n})|\Omega_{ij}|$$

- What about the flux integral?

Flux integral

- Flux integral on Cartesian grids (2d):

$$\int_{t^n}^{t^{n+1}} \int_{\partial\Omega} \mathbf{F}(u) \cdot \mathbf{n}\, ds\, dt = \int_{t^n}^{t^{n+1}} \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \left[F(u(x_{i+\frac{1}{2}}, y, t) - F(u(x_{i-\frac{1}{2}}, y, t) \right]$$

$$+ \int_{t^n}^{t^{n+1}} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \left[G(u(x, y_{j+\frac{1}{2}}, t) - G(u(x, y_{j-\frac{1}{2}}, t) \right]$$

Explicit time stepping scheme

$$
\begin{aligned}
U_{i,j}^{n+1} - U_{i,j}^{n} = & \quad \frac{\Delta t}{\Delta y}\left[F(u(x_{i+\frac{1}{2}}, y, t) - F(u(x_{i-\frac{1}{2}}, y, t)\right] \\
+ & \quad \frac{\Delta t}{\Delta x}\left[G(u(x, y_{j+\frac{1}{2}}, t) - G(u(x, y_{j-\frac{1}{2}}, t)\right]
\end{aligned}
$$

Question:

- How to compute fluxes
$$F_{i+\frac{1}{2},j}^{n} := F(u(x_{i+\frac{1}{2}}, y, t)$$
- Several possible numerical flux functions

Central and Upwind fluxes

- Define fluxes $F^n_{i+\frac{1}{2},j}$, $G^n_{i,j+\frac{1}{2}}$ via 1D numerical flux function $\mathcal{F}$, $\mathcal{G}$:

$$F^n_{i+\frac{1}{2},j} = \mathcal{F}(U^n_{i,j}, U^n_{i+1,j}), \quad G^n_{i,j+\frac{1}{2}} = \mathcal{G}(U^n_{i,j}, U^n_{i,j+1})$$

- **Central flux**:

$$F^n_{i+\frac{1}{2},j} = \mathcal{F}(U^n_{i,j}, U^n_{i+1,j}) := \frac{1}{2}\left[F(U^n_{i,j}) + F(U^n_{i+1,j})\right]$$

  Could be unstable for convective transport

- **Upwind flux** (here, for flux $F(u) = hu$):

$$F^n_{i+\frac{1}{2},j} = \mathcal{F}(U^n_{i,j}, U^n_{i+1,j}) := \begin{cases} hu|_i & \textit{if } u|_{i+\frac{1}{2}} > 0 \\ hu|_{i+1} & \textit{if } u|_{i+\frac{1}{2}} < 0 \end{cases}$$

  stable, but more diffusive

Lax-Friedrichs Flux

- classical **Lax-Friedrichs** numerical flux function:

$$F^n_{i+\frac{1}{2},j} = \mathcal{F}(U^n_{i,j}, U^n_{i+1,j}) := \frac{1}{2}\left[F(U^n_{i,j}) + F(U^n_{i+1,j})\right] - \frac{h}{2\Delta t}\left(U^n_{i+1,j} - U^n_{i,j}\right)$$

- can be interpreted as central flux plus diffusion flux:

$$\frac{h}{2\Delta t}\left(U^n_{i+1,j} - U^n_{i,j}\right) = \frac{h^2}{2\Delta t}\frac{U^n_{i+1,j} - U^n_{i,j}}{h}$$

with diffusion coefficient $\frac{h^2}{2\Delta t}$, where $c := \frac{h}{\Delta t}$ is a velocity

local Lax-Friedrichs Flux

- **local Lax-Friedrichs** numerical flux function:

$$F^n_{i+\frac{1}{2},j} = \mathcal{F}(U^n_{i,j}, U^n_{i+1,j}) := \frac{1}{2}\left[F(U^n_{i,j}) + F(U^n_{i+1,j})\right] - \frac{a_{i+\frac{1}{2}}}{2}\left(U^n_{i+1,j} - U^n_{i,j}\right)$$

- Idea: Use local wave speed $a_{i+\frac{1}{2}}$ which is an approximation of the form

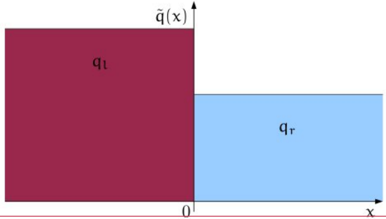
$$a_{i+\frac{1}{2}} \approx f'(U_{i+\frac{1}{2}})$$

Riemann Problem

- solve **Riemann problem** to obtain solution $q(x_{i-\frac{1}{2}}, t^n)$
- 1D treatment: solve shallow water equations with initial conditions

$$q(x_{i-\frac{1}{2}}, t^n) = \begin{cases} q_l = U^n_{i-1} & if x < x_{i-\frac{1}{2}} \\ q_r = U^n_i & if x > x_{i-\frac{1}{2}} \end{cases}$$

- solution:
  two (left or right) outgoing waves (shock or rarefaction)

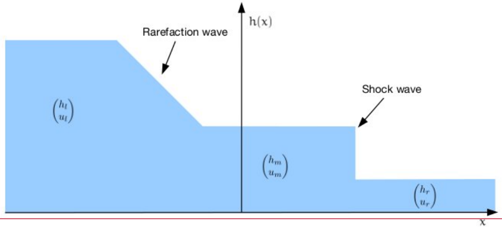


Riemann Problem

- solve **Riemann problem** to obtain solution $q(x_{i-\frac{1}{2}}, t^n)$
- 1D treatment: solve shallow water equations with initial conditions

$$q(x_{i-\frac{1}{2}}, t^n) = \begin{cases} q_l = U^n_{i-1} & if x < x_{i-\frac{1}{2}} \\ q_r = U^n_i & if x > x_{i-\frac{1}{2}} \end{cases}$$

- solution:
  two (left or right) outgoing waves (shock or rarefaction)

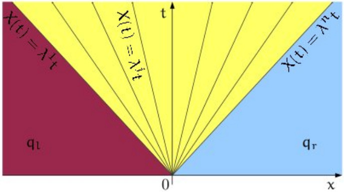


Riemann Problem

- wave propagation approach: split the jump into fluxes

$$F(Q_i) - F(Q_{i-1}) - \Delta x \psi_{i-\frac{1}{2}} = \sum_p \alpha_p r_p \equiv \sum_p Z_p \qquad \alpha_p \in \mathbb{R}.$$

r_p the eigenvector of the linearised problem,
$\psi_{i-\frac{1}{2}}$ a fix for the source term (bathymetry)



- implementation will compute **net updates**:

$$\mathcal{A}^+ \Delta Q_{i-1/2,j} = \sum_{p:\ \lambda_p > 0} Z_p \qquad \mathcal{A}^- \Delta Q_{i-1/2,j} = \sum_{p:\ \lambda_p < 0} Z_p$$

F-Wave solver

- use Roe eigenvalues $\lambda^{\text{Roe}}_{1/2}$ to approximate the wave speeds:

$$\lambda^{\text{Roe}}_{1/2}(q_l, q_r) = u^{\text{Roe}}(q_l, q_r) \pm \sqrt{g h^{\text{Roe}}(q_l, q_r)}$$

- with $h^{\text{Roe}}(q_l, q_r) = \frac{1}{2}(h_l + h_r)$ and $u^{\text{Roe}}(q_l, q_r) = \frac{u_l\sqrt{h_l} + u_r\sqrt{h_r}}{\sqrt{h_l} + \sqrt{h_r}}$
- eigenvectors $r^{\text{Roe}}_{1/2}$ for wave decomposition defined as

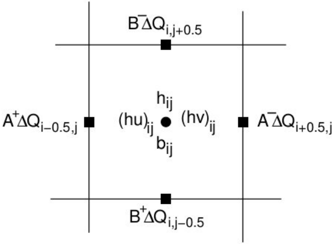
$$r^{\text{Roe}}_1 = \begin{pmatrix} 1 \\ \lambda^{\text{Roe}}_1 \end{pmatrix} \qquad r^{\text{Roe}}_2 = \begin{pmatrix} 1 \\ \lambda^{\text{Roe}}_2 \end{pmatrix}$$

- leads to net updates (source terms still missing):

$$A^-\Delta Q := \sum_{p:\{\lambda^{\text{Roe}}_p < 0\}} \alpha_p r_p \qquad A^+\Delta Q := \sum_{p:\{\lambda^{\text{Roe}}_p > 0\}} \alpha_p r_p$$

- with $\alpha_{1/2}$ computed from $\begin{pmatrix} 1 & 1 \\ \lambda^{\text{Roe}}_1 & \lambda^{\text{Roe}}_2 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} = F(Q_i) - F(Q_{i-1})$

Discretistaion



Unknowns and Numerical Fluxes:

- (averaged) unknowns *h*, *hu*, *hv* , and *b* located in cell centers
- two sets of "net updates" or "numerical fluxes" per edge; here: $A^{+}\Delta U_{i-\frac{1}{2},j}$, $B^{-}\Delta U_{i,j+\frac{1}{2},j}$ ("wave propagation form")

Flux Form vs. Wave Propagation Form

- numerical scheme in flux form:

$$Q_{i,j}^{(n+1)} = Q_{i,j}^{(n)} - \frac{\Delta t}{\Delta x}\left(F_{i+\frac{1}{2},j}^{(n)} - F_{i-\frac{1}{2},j}^{(n)}\right) - \frac{\Delta t}{\Delta y}\left(G_{i,j+\frac{1}{2}}^{(n)} - G_{i,j-\frac{1}{2}}^{(n)}\right)$$

  where $F_{i+\frac{1}{2},j}^{(n)}$, $G_{i,j+\frac{1}{2}}^{(n)}, \ldots$ approximate the flux functions $F(q)$ and $G(q)$ at the grid cell boundaries

- **Wave propagation form:**

$$\begin{aligned} Q_{i,j}^{n+1} = Q_{i,j}^{n} \quad & - \tfrac{\Delta t}{\Delta x}\left(\mathcal{A}^{+}\Delta Q_{i-1/2,j} + \mathcal{A}^{-}\Delta Q_{i+1/2,j}^{n}\right) \\ & - \tfrac{\Delta t}{\Delta y}\left(\mathcal{B}^{+}\Delta Q_{i,j-1/2} + \mathcal{B}^{-}\Delta Q_{i,j+1/2}^{n}\right). \end{aligned}$$

  where $\mathcal{A}^{+}\Delta Q_{i-1/2,j}$, $\mathcal{B}^{-}\Delta Q_{i,j+1/2}^{n}$, etc. are **net updates**

- difference in implementation: compute one “flux term” or two “net updates” for each edge

References & Literature

- LeVeque: Finite Volume Methods for Hyperbolic Problems, Cambridge University Press, 2002
- Toro: Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction, Springer, 2009
- Bale, LeVeque, Mitran, Rossmanith: A wave propagation method for conservation laws and balance laws with spatially varying flux functions, SIAM Journal on Scientific Computing 24 (3), 2003
- George: Augmented Riemann solvers for the shallow water equations over variable topography with steady states and inundation, Journal of Computational Physics 227 (6), 2008
- Breuer, Bader: Teaching Parallel Programming Models on a Shallow-Water Code, Proc. of the ISPDC 2012

The Shallow Water Equations

- A hyperbolic partial differential equation
  - First described by de Saint-Venant (1797 - 1886)
  - Conservation of mass and momentum
  - Gravity waves in 2D free surface
- Gravity-induced fluid motion
  - Governing flow is horizontal
- Not only used to describe physics of water:
  - Simplification of atmospheric flow
  - Avalanches

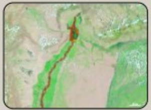
Water image from http://freephoto.com / Ian Britton

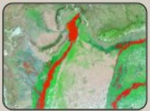
Target application areas

Tsunamis

2011: Japan (5321+)
2004: Indian Ocean (230 000)

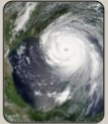
Floods





2010: Pakistan (2000+)
1931: China floods (2 500 000+)

Storm Surges



2005: Hurricane Katrina (1836)
1530: Netherlands (100 000+)

Dam breaks

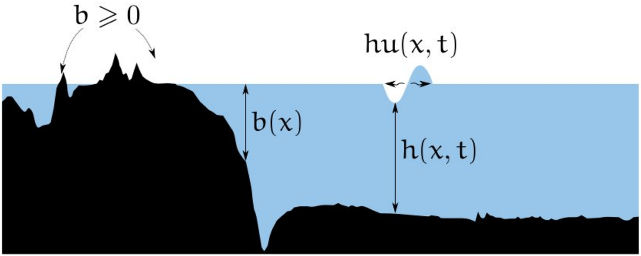
1975: Banqiao Dam (230 000+)
1959: Malpasset (423)

Equations

Simplified setting (no friction, no viscosity, no Coriolis forces, etc.):

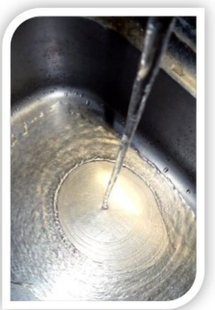
$$\frac{\partial}{\partial t}\begin{bmatrix} h \\ hu \\ hv \end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix} + \frac{\partial}{\partial y}\begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{\partial}{\partial x}(ghb) \\ -\frac{\partial}{\partial y}(ghb) \end{bmatrix}$$

Quantities and unknowns:



Modelling aspects

- A hyperbolic partial differential equation
  - Enables explicit schemes
- Solutions form discontinuities / shocks
  - Require high accuracy in smooth parts without oscillations near discontinuities
- Solutions include dry areas
  - Negative water depths ruin simulations
- Often high requirements to accuracy
  - Order of spatial/temporal discretization
  - Floating point rounding errors
- Can be difficult to capture „lake at rest“



A standing wave or *shock*

Conserved quantaties

Simplified setting (no friction, no viscosity, no Coriolis forces, etc.):

$$\frac{\partial}{\partial t}\begin{bmatrix} h \\ hu \\ hv \end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix} + \frac{\partial}{\partial y}\begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{\partial}{\partial x}(ghb) \\ -\frac{\partial}{\partial y}(ghb) \end{bmatrix}$$

Derived from conservation laws

- *h* equation – conservation of mass
- equations for *hu* and *hv* – conservation of momentum
- $\frac{1}{2}gh^2$: averaged hydrostatic pressure due to water column *h*
- may also be derived by vertical averaging from the 3D incompressible Navier-Stokes equations

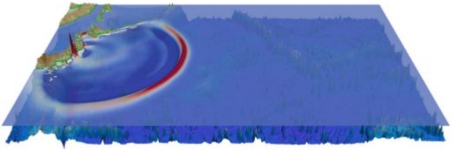
Conserved quantaties

Simplified setting (no friction, no viscosity, no Coriolis forces, etc.):

$$\frac{\partial}{\partial t}\begin{bmatrix} h \\ hu \\ hv \end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix} + \frac{\partial}{\partial y}\begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{\partial}{\partial x}(ghb) \\ -\frac{\partial}{\partial y}(ghb) \end{bmatrix}$$

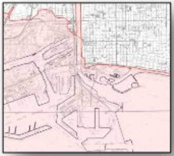
The ocean as “shallow water”??

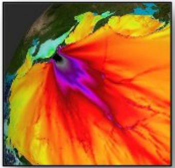
- compare horizontal ($\sim$ 1000 km) to vertical ($\sim$ 4 km) scale
- wave lengths large compared to water depth
- vertical flow may be neglected; movement of the “entire water column”



Why using GPUs for shallow water simulation

- In preparation for events: Evaluate possible scenarios
  - Simulation of many ensemble members
  - Creation of inundation maps and emergency action plans
- In response to ongoing events
  - Simulate possible scenarios in real-time
  - Simulate strategies for action (deployment of barriers, evacuation of affected areas, etc.)

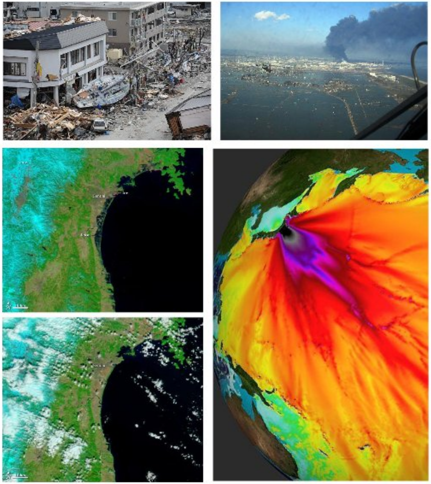




High requirements to performance ⇒ Use the GPU

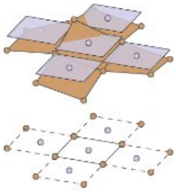
Japan Tsunami

- Tsunami warnings must be issued in real-time
  - Huge computational domains
  - Rapid wave propagation
  - Uncertainties wrt. Tsunami cause
- Warnings must be accurate
  - Wrongful warning are dangerous!
- GPUs can be used to increase quality of warnings



Discretisation

- Our grid consists of a set of cells or volumes
  - The bathymetry is a piecewise bilinear function
  - The physical variables
    (h, hu, hv)
    are piecewise constants per volume
- Physical quantities are transported across the cell interfaces
- Source term (bathymetry) are per cell



Time Stepping

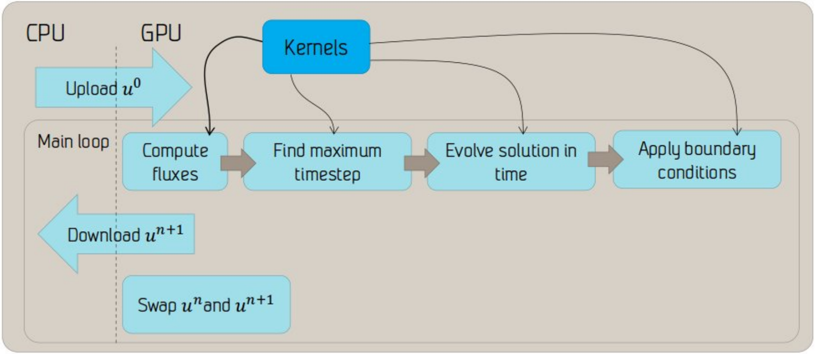
CFL Condition:

- we only consider neighbour cells for a time step
  $\Rightarrow$ information must not travel faster than one cell per timestep!
- timesteps need to consider characteristic wave speeds
- rule of thumb: wave speed depends on water depth,
  $\lambda = \sqrt{gh}$
  $\Rightarrow$ maximum-reduction necessary to find global time step

Sequential main loop f- adaptive time step control

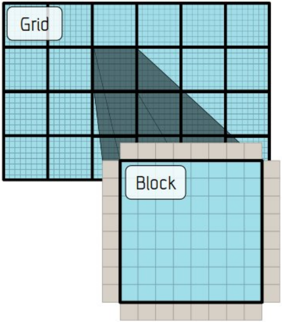
1. solve Riemann problems, compute wave speeds
2. compute maximum wave speed and infer global Δt
3. update unknowns

Possible simulation set-up



Flux kernel | Grids and blocks

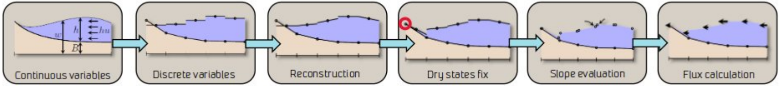
- Observations
  - Our shallow water problem is 2D
  - The GPU requires a parallel algorithm
  - The GPU has native support for 2D grids and blocks
- Proceeding:
  - Split up the computation into independent 2D blocks
  - Each block is similar to a node in an MPI cluster
  - Execute all blocks in parallel



Computing fluxes

$$\begin{bmatrix} h \\ hu \\ hv \end{bmatrix}_t + \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}_x + \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix}_y = \begin{bmatrix} 0 \\ -ghB_x \\ -ghB_y \end{bmatrix} + \begin{bmatrix} 0 \\ -gu\sqrt{u^2+v^2}/C_z^2 \\ -gv\sqrt{u^2+v^2}/C_z^2 \end{bmatrix}$$

$$\frac{dQ_{ij}}{dt} = H_f(Q_{ij}) + H_B(Q_{ij}, \nabla B) - \left[F(Q_{i+1/2,j}) - F(Q_{i-1/2,j})\right] - \left[G(Q_{i,j+1/2}) - G(Q_{i,j-1/2})\right]$$

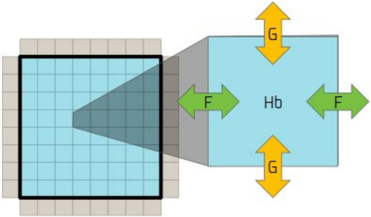


The flux calculation is a set of stencil operations:
1. slope reconstruction, 2. slope evaluation 3. flux calculation.

from: A.R. Brodkorb *Hyperbolic Conservation Laws on GPUs*, SINTEF Research Norway

Computing fluxes

$$\frac{dQ_{ij}}{dt} = H_f(Q_{ij}) + H_B(Q_{ij}, \nabla B) - \left[F(Q_{i+1/2,j}) - F(Q_{i-1/2,j})\right] - \left[G(Q_{i,j+1/2}) - G(Q_{i,j-1/2})\right]$$



- The fluxes, *F* and *G*, are computed for each cell interface
- The source term, *Hb* is computed for each cell
- Shared memory could be used to limit data traffic and reuse data

References

- A. R. Brodtkorb, M. L. Sætra, and M. Altinakar, Efficient Shallow Water Simulations on GPUs: Implementation, Visualization, Verification, and Validation, Computers & Fuids, 55, (2011)
- M. L. Sætra and A. R. Brodtkorb, Shallow Water Simulations on Multiple GPUs, Proceedings of the Para 2010 Conference Part II, Lecture Notes in Computer Science 7134 (2012), pp 56–66,
- Gene Golub SIAM Summer School 2012 Simulation and Supercomputing in the Geosciences http://www.mac.tum.de/g2s3/