The Shallow Water Equations and CUDA

Daniel Butnaru, Christoph Kowitz

December 5
Improvement of the MM multiplication

- coalesced memory access
- loop unrolling
- thread granularity

Performance Measures

- runtime
- occupancy
- warp serialize
- instruction throughput

What can still be done?

binary fan in for reduction
New Playground: Shallow Water Equations

• used for the simulation of layers of fluid
• applicable when horizontal scales are much larger than vertical scales
• derived from Navier-Stokes equations

Applications

• tsunami simulations
  • the ocean is not shallow, but wide
  • whole water column is influenced
• atmospheric simulations
  • for whole planets
Your Exercise

- use CUDA to efficiently calculate the SWE’s
- framework will be given
Shallow Water Equations

\[ \frac{\partial h}{\partial t} + \frac{\partial (v_x h)}{\partial x} + \frac{\partial (v_y h)}{\partial y} = 0 \]

\[ \frac{\partial (hv_x)}{\partial t} + \frac{\partial (hv_x v_x)}{\partial x} + \frac{\partial (hv_y v_x)}{\partial y} + \frac{1}{2}g \frac{\partial (h^2)}{\partial x} = -gh \frac{\partial b}{\partial x}, \quad (1) \]

\[ \frac{\partial (hv_y)}{\partial t} + \frac{\partial (hv_x v_y)}{\partial x} + \frac{\partial (hv_y v_y)}{\partial y} + \frac{1}{2}g \frac{\partial (h^2)}{\partial y} = -gh \frac{\partial b}{\partial y}, \]

with

\[ h \quad \text{water height over ground} \]

\[ v_x, v_y \quad \text{velocity in x and y direction} \]

\[ g \quad \text{earths gravitation acceleration} \]

\[ b \quad \text{bathymetry (height of the sea ground)} \]

\[ hv_i = p, q \quad \text{linear momentum of the fluid} \]
Spatial Discretization

- cartesian equidistant grid
- all data a centered in the nodes \((p, q, h)\)

\[
\frac{h_{ij}^{(n+1)} - h_{ij}^{(n)}}{\tau} + \frac{\partial p}{\partial x}\bigg|_{ij}^{(n)} + \frac{\partial q}{\partial y}\bigg|_{ij}^{(n)} = 0
\]

\[
\frac{p_{ij}^{(n+1)} - p_{ij}^{(n)}}{\tau} + \frac{\partial (v_x p)}{\partial x}\bigg|_{ij}^{(n)} + \frac{\partial (v_y p)}{\partial y}\bigg|_{ij}^{(n)} + \frac{1}{2} g \frac{\partial (h^2)}{\partial x}\bigg|_{ij}^{(n)} = -gh \frac{\partial b}{\partial x}\bigg|_{ij}
\]

\[
\frac{q_{ij}^{(n+1)} - q_{ij}^{(n)}}{\tau} + \frac{\partial (v_x q)}{\partial x}\bigg|_{ij}^{(n)} + \frac{\partial (v_y q)}{\partial y}\bigg|_{ij}^{(n)} + \frac{1}{2} g \frac{\partial (h^2)}{\partial y}\bigg|_{ij}^{(n)} = -gh \frac{\partial b}{\partial y}\bigg|_{ij}
\]

Time Discretization

- forward euler
- variable timesteps
Computing Spatial Derivatives

Finite Differences

\[
\frac{\partial p}{\partial x} \approx \frac{p_{i+1} - p_{i-1}}{2 \Delta x} \tag{2}
\]

Finite Volume

- computation by evaluation of the fluxes over cell edge
- change of water height \( h \) can depend on the influx and outflux \( h v \)

\[
\frac{\partial p}{\partial x} \bigg|_{\text{n}} \approx \frac{p_{i+1/2,j} - p_{i-1/2,j}}{\Delta x}
\]

\[
\frac{\partial q}{\partial y} \bigg|_{\text{n}} \approx \frac{q_{i,j+1/2} - q_{i,j-1/2}}{\Delta y}
\]
Computing Spatial Derivatives

Finite Differences

- compute derivative from cell neighbours

\[
\frac{\partial p}{\partial x} \approx \frac{p_{i+1} - p_{i-1}}{2\Delta x}
\]  

(2)
Computing Spatial Derivatives

Finite Differences

- compute derivative from cell neighbours

\[
\frac{\partial p}{\partial x} \approx \frac{p_{i+1} - p_{i-1}}{2\Delta x}
\]  

(2)

Finite Volume

- computation by evaluation of the fluxes over cell edge
- change of waterheight \((h)\) can depends on the influx and outflux \((hv)\)

\[
\frac{\partial p}{\partial x}\bigg|_{ij}^{(n)} \approx \frac{p_{i+\frac{1}{2},j}^{(n)} - p_{i-\frac{1}{2},j}^{(n)}}{\Delta x} \quad \text{and} \quad \frac{\partial q}{\partial y}\bigg|_{ij}^{(n)} \approx \frac{q_{i,j+\frac{1}{2}}^{(n)} - q_{i,j-\frac{1}{2}}^{(n)}}{\Delta y},
\]
Evaluation of the Flux on the Edge

Taking Averages on the edge

- average the momentum flux between to cells

\[
P_{i+\frac{1}{2},j}^{(n)} = \frac{1}{2} \left( p_{ij}^{(n)} + p_{i+1,j}^{(n)} \right) \quad \text{or} \quad q_{i,j-\frac{1}{2}}^{(n)} = \frac{1}{2} \left( q_{i,j-1}^{(n)} + q_{ij}^{(n)} \right)
\]
Lax-Friedrich Flux Computation

due to stability issues the computed flux has to be corrected to accurate terms by an averaging over the flux density.

\[
p_{i+\frac{1}{2},j}^{(n)} = \frac{1}{2} \left( p_{ij}^{(n)} + p_{i+1,j}^{(n)} \right) + \frac{\Delta x}{2\tau} \left( h_{ij}^{(n)} - h_{i+1,j}^{(n)} \right) \quad (3)
\]
How to compute the SWE’s so far:

1. compute required fluxes on the edges
   - for $h$, $p$ and $q$
   - for $p$, $pv_x$ and $pv_y$
   - for $q$, $qv_x$ and $qv_y$

2. compute euler time-step for $h$, $p$ and $q$
Source Terms

Pressure Induced Force by Watercolumn

\[ \frac{1}{2} g \left( \frac{\partial (h^2)}{\partial x} \right) \bigg|_{ij}^{(n)} \]

\( h \) (mass of the watercolumn) \( \times \frac{h}{2} \) height of the center of gravity = pressure

Pressure Difference due to changes of the bathymetry

\[ gh \left( \frac{\partial b}{\partial x} \right) \bigg|_{ij} \]

for the same depth over ground a change in the bathymetry induces a pressure difference
Balance of the forces

For a not moving watersurface, the two terms have to be in equilibrium

\[
\frac{1}{2} g \frac{\partial (h^2)}{\partial x} \bigg|_{ij}^{(n)} = -gh \frac{\partial b}{\partial x} \bigg|_{ij}
\]

Discretization

regular:

\[
h \frac{\partial b}{\partial x} \bigg|_{ij} = h_{ij} \frac{b_{i+1,j} - b_{i-1,j}}{2\Delta x}
\]
Balance of the forces

For a not moving watersurface, the two terms have to be in equilibrium

\[
\frac{1}{2} g \frac{\partial (h^2)}{\partial x} \bigg|_{ij}^{(n)} = -gh \frac{\partial b}{\partial x} \bigg|_{ij}
\]

Discretization

regular:

\[
h \frac{\partial b}{\partial x} \bigg|_{ij} = h_{ij} \frac{b_{i+1,j} - b_{i-1,j}}{2 \Delta x}
\]

stable:

\[
h \frac{\partial b}{\partial x} \bigg|_{ij} = \frac{h_{i+1,j} + h_{i-1,j}}{2} \cdot \frac{b_{i+1,j} - b_{i-1,j}}{2 \Delta x}
\]
Boundary Conditions

realized by ghost cells

Outflow

• ghost cells are set equal to neighbouring fluid cells

Wall

• normal velocity is set to zero on edge $u_{i-1} = -u_i$
• the other quantities are treated like in for the ghost cells
Optimal Timestep

Velocity of characteristic shallow water wave

\[ v_C = \sqrt{gh_{\text{max}}} + \max(u_{\text{max}}, v_{\text{max}}) \]

leads to a maximal allowed timestep

\[ \tau_{\text{max}} = \frac{\Delta x}{v_C} \]

which is computed after each iteration.
1. compute the maximum time-step
2. set ghost cells for boundary treatment
3. compute bathymetric sources
4. compute fluxes (there are more edges than cells)
5. do Euler time-step
Some Details for the CUDA Implementation

- no domain decomposition required (one big block with ghost cells)
- domain is departed in smaller threadblocks of TILE_SIZE
- each thread computes one cells unknowns
- the respective data structures have to be created on the device
  - \(h, u, v\) for each cell \(\rightarrow hd, hud, hvd\)
  - the fluxes for each edge (\(Fhd, Fud, Fvd\) for fluxes in \(x\)-direction and \(Ghd, Gud, Gvd\) for fluxes in \(y\)-direction)
  - the bathymetry \(bd\)
  - the bathymetric source \(Bxd, Byd\)
Structure of the Framework

TODO marks places where something has to be added

Program structure

compute Boundary on host, already implemented

bath. sources have to be computed on device and stored in local memory ($B_{xd}, B_{yd}$)

compute fluxes on device, remember that there are more edges than cells, c-code as comment

Euler time-step on device, c-code as comment

max. time-step on host, already implemented