Shallow Water Simulations on Graphics Hardware
Ph.D. Thesis Presentation
2014-06-27

Martin Lilleeng Sætra
Outline

• Introduction
  – Parallel Computing and the GPU
  – Simulating Shallow Water Flow

• Topics of Thesis
  – Shallow Water Simulations on the GPU
  – Utilizing Multiple GPUs
  – Algorithms for Sparse Domains on GPUs
  – Adaptive Mesh Refinement on the GPU

• Summary
Introduction

• **Aim of thesis**: Utilize modern many-core architectures for efficient simulation of real-world shallow water problems

• The *graphics processing unit* (GPU) is the architecture investigated

• GPUs deliver high performance, but are challenging to utilize efficiently

• Shallow water phenomena have a big impact – affects many people

• Need fast and physically correct simulations
PARALLEL COMPUTING AND THE GPU
Why Do We Need Parallel Computing?

- Moore’s Law still valid
- CPUs were reaching power densities equivalent to that of a nuclear reactor core
  
  \[
  \text{Power density proportional to frequency cubed}
  \]
- Shifting from frequency scaling to concurrency scaling
- Multi-core and many-core uses lower frequencies, but more cores

More performance for less power

C. Moore, “Data Processing in Exascale-Class Computer Systems”
Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
GPUs are massively data parallel.
GPU – Supercomputer on a Chip?

1996: ASCI Red
First system 1 TF sustained, 72 Cabinets
850 kW (not including A/C)

2013: Nvidia Tesla K20X
1.31 TF theoretical peak, 1 PCI express slot
235 W

2008: ATI Radeon HD4850
First GPU 1 TF theoretical peak (single-precision), 1 PCI express slot
110 W
Widespread Use

Commodity Level

• Every modern laptop and desktop computer comes with a GPU

Supercomputing

• Today, 10% of the TOP 500 systems use accelerators and they constitute 35% of the performance
• About half of these systems use GPUs
• A supercomputer can have thousands of GPUs (Titan: 18,688)
SIMULATING
SHALLOW WATER FLOW
Simulating Physical Phenomena

Real-World Phenomena

Mathematical Model

\[ Q_t + F(Q)_x + G(Q)_y = H(Q, \nabla B) \]

Computer Model

```c
while(t < t_max) {
    q_next += dt * f(q_prev);
    t += dt;
    ...
}
```

Use data in decision making

Visualize

Simulate Scenario

Initial Conditions

2014-06-27 Shallow Water Simulations on Graphics Hardware
The Shallow Water Equations

- Conservation Law
- First formulated by Saint-Venant in 1871 (in 1D)
- Will not represent breaking waves
- Little vertical motion
- Wavelength >> depth
Tsunamis

Storm Surges

Flooding

Dam Breaks
SHALLOW WATER SIMULATIONS ON THE GPU
Research Timeline

2001
• First GPU computing paper

2005
• First SINTEF GPU computing paper

2007
• CUDA released
• Master’s thesis: Solving PDEs on multiple GPUs

Now
• Efficient shallow water simulations for real-world problems

2014-06-27
Shallow Water Simulations on Graphics Hardware
Shallow Water Simulations on the GPU

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

Vector of Conserved variables

Flux Functions

Bed slope source term

Bed friction source term

Continuous equation

Discrete spatial grid
Shallow Water Simulations on the GPU

- Explicit shallow water schemes can be written as stencils

- Stencil computations are embarrassingly parallel
Shallow Water Simulations on the GPU

- Discontinuities (shocks)
- Wetting and drying
- Lake at rest
- Complex bathymetry
Shallow Water Simulations on the GPU

• Hyperbolic PDEs – enables use of explicit schemes
• Stencil computations – good match with GPU execution model

• Second order accurate fluxes
• Total Variation Diminishing – avoid spurious oscillations
• Well-balanced – captures lake-at-rest
• No negative water depths – support dry states

A. Kurganov and G. Petrova,
”A Second-Order Well-Balanced Positivity Preserving Central-Upwind Scheme for the Saint-Venant System”,
Communications in Mathematical Sciences, 5 (2007), 133-160
Shallow Water Simulations on the GPU

- Continuous variables
- Discrete variables / Cell averages
- Slope reconstruction
- Flux calculation
- Evaluate integration points
- Dry states fix

(Elements of the diagrams include symbols and arrows indicating the flow of processes.)
Shallow Water Simulations on the GPU

1. Calculate fluxes
2. Calculate $\Delta t$
3. Halfstep
4. Calculate fluxes
5. Evolve in time
6. Apply boundary conditions

One cycle evolves the solution one time step ($\Delta t$) forward.
Verification – Parabolic Basin

- Compare with analytical solution for 2D parabolic basin (Thacker)
  - Planar water surface oscillates
  - $100^2$ cells
  - Horizontal scale: 8 km
  - Vertical scale: 3.3 m
- Simulation matches well with analytical solution

(a) $t \approx \frac{1\pi}{8\omega}$ (1392.85 s)
(b) $t \approx \frac{2\pi}{8\omega}$ (2787.95 s)
(c) $t \approx \frac{3\pi}{8\omega}$ (4198.64 s)
(d) $t \approx \frac{4\pi}{8\omega}$ (5559.27 s)
Validation – Malpasset Dam Break

- Near Fréjus in France
- A 66.5 meter high dam with a crest length of over 220 meters that impounded 55 million cubic meters of water in the reservoir
- Collapsed after heavy rainfall in early December 1959 creating a 40 meter high wall of water travelling at 70 km/h
Utilizing Multiple GPUs

- **Idea**: Utilize several GPUs for faster and larger simulations

  - Do row decomposition and solve *subdomains* on different GPUs
  - Synchronize time step size between GPUs

Overlapping cells must be exchanged between the subdomains before each time step.
Utilizing Multiple GPUs

- By increasing the overlap, more than one time step may be taken before exchanging overlapping cells – *ghost cell expansion*
Utilizing Multiple GPUs

- Ghost cell expansion increases performance – most visible when communication overhead is large compared to compute time.

- Global time step synchronization between subdomains/GPUs had a negligible effect on performance.

- Near perfect linear (weak and strong) scaling.

- The graphs are normalized wrt. the fastest 1-GPU run.

![Graph showing performance vs. domain size for different numbers of GPUs.](image)
Algorithms for Sparse Domains

- **Idea**: Do not compute the dry areas of a domain
  - Usually, all cells in the domain are read, computed, and stored, even cells without water

1910 Great Flood of Paris

2005 Hurricane Katrina
Algorithms for Sparse Domains

- Two algorithms: *Sparse Compute* and *Sparse Memory*
- Block-based algorithms

- Sparse Compute does not read or compute «dry» cells
- Sparse Memory does not read, compute, or store «dry» cells
Algorithms for Sparse Domains

- Both Sparse Compute and Sparse Memory shows speed-up
- Sparse Memory saves memory at the cost of an additional look-up
- Average number of wet cells: 26%
- The graphs are normalized wrt. the single fastest run
Adaptive Mesh Refinement

• **Idea:** Use a hierarchy of successively finer grids to adaptively get high resolution in important parts of the domain.
Adaptive Mesh Refinement

Each child grid is only dependent on its parent

All other grids may be computed in parallel
Adaptive Mesh Refinement

• Time must be synchronized between parent and child

• To achieve this, the last time step in a «series» is limited
Adaptive Mesh Refinement

- Special boundary conditions for child grids:
  - Use standard reconstruction at parent’s current and previous time step
  - Evaluate at child cell centers
  - Linear interpolate in time

- Average the solution on child grids onto the parent grid
Resolving subgrid features

Without AMR

With AMR

Shock Tracking

Adaptive Refinement
Adaptive Mesh Refinement

- Block-based AMR fully implemented on the GPU
- Verified with synthetic case and tested with real-world case
- Accuracy close to what can be achieved with global refinement
- Minimal grid effects when padding new child grids
- Easy to set custom refinement criteria
Summary

• GPU Computing
  – GPU development strategies
  – Guidelines for stencil-based explicit schemes on the GPU

• Physically correct shallow water simulations
  – Building a shallow water simulator on the GPU using a well-balanced high-resolution scheme, supporting dry states
  – Verified and validated

• Methods for more efficient & accurate shallow water simulations
  – Using multiple GPUs
  – Sparse Domain algorithms on the GPU
  – Adaptive Mesh Refinement fully implemented on the GPU
References


