Parallelization of dynamically changing grids with a cluster-based approach and invasion

Martin Schreiber
November 11th, 2013
Sendai Tsunami Simulation

Created in collaboration with Alexander Breuer.
Based on GeoClaw solver and GEBCO datasets
Why dynamical adaptive mesh refinement?
Invest computations only in *feature rich areas*.
Outline

Simulations on dynamic adaptive grids
   Simulation and visualization requirements
   Introduction to clustering
   Issue 1) Cache aware memory access
   Issue 2) Parallelization
   Issue 3) Communication
   Software design
   Results

Invasive Computing
   Motivation
   Invasive Computing
   Results

Summary & Outlook
Simulations

Wave-dominated problems demanding for dynamically changing grids:

- We focus on hyperbolic PDEs given by the continuity equation

\[ \hat{U}_t(x, y, t) + \nabla \cdot F(\hat{U}(x, y, t)) = S(\hat{U}(x, y, t)). \]

\[ \hat{U}_t(x, y, t) + G_x(\hat{U}(x, y, t)) + H_y(\hat{U}(x, y, t)) = S(\hat{U}(x, y, t)). \]

- \( \hat{U} \): Conserved quantities (mass, momentum, energy)
- \( G, H \): flux functions
- \( S \): Source term
Example simulations with wave-propagations

**Shallow water equations:**

\[
G(U) := \begin{pmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{pmatrix}
\]

\[
H(U) := \begin{pmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{pmatrix}
\]

**Euler equations:**

\[
G(U) := \begin{pmatrix} \rho u \\ p + \rho u^2 \\ \rho uv \\ u(E + p) \end{pmatrix}
\]

\[
H(U) := \begin{pmatrix} \rho v \\ \rho uv \\ p + \rho v^2 \\ v(E + p) \end{pmatrix}
\]
Discretization in space and time

- Continuity equation in 2D:
  \[ U_t + G_x(U) + H_y(U) = S(U) \]

- Discretization in space:
  - Discontinuous Galerkin
  - Triangle grids due to well-established community
  - Applying Gaussian divergence theorem and discretization using basis functions \( \varphi_i \):
    \[
    \int_T \frac{dU}{dt} \varphi_i dT - \int_T F(U) \cdot \nabla \varphi_i dT + \oint_T F(U) \varphi_i \cdot ds = 0
    \]
    - mass-term
    - stiffness-term
    - flux-term

- Discretization in Time: Runge-Kutta-\( n \)
  For RK1 on \( \frac{dU}{dt} \), this yields:
  \[
  \tilde{U}_i(t + \Delta t) = \tilde{U}_i(t) + \Delta t M^{-1} \cdot (S(\tilde{U}(t)) + F(\tilde{U}(t), \tilde{U}^+(t)))
  \]
  - U for next time-step
  - U of previous time-step
  - cell-local
  - edge-communication
Simulation backends

- Offline backend: VTK
- Online backend: OpenGL (for educational purpose, steering, debugging, ...)

What interactive visualization would you prefer?

- Gaps at cell boundaries with direct visualization of surface of DG simulations.
- Creating a closed surface with visualization data (vertices, normals) in \( \approx O(n) \) requires computation of vertices and normals at nodes.
Summary of interface and data exchange requirements

Data storage and ”stencils”:

- **Cell data** to store DOFs.
- Exchange of data via **edges** to compute fluxes.
- Exchange of data via **edges** to forward adaptivity information.
- Exchange of data via **nodes** to generate closed surface.
Outline

Simulations on dynamic adaptive grids
- Simulation and visualization requirements
- Introduction to clustering
- Issue 1) Cache aware memory access
- Issue 2) Parallelization
- Issue 3) Communication
- Software design
- Results

Invasive Computing
- Motivation
- Invasive Computing
- Results

Summary & Outlook
Cluster

C. E. Castro, M. Käser and E. F. Toro: Space-time adaptive numerical methods for geophysical applications

\[ \Omega := C^{(1)} \cup C^{(2)} \cup C^{(3)} \]

- Non-overlapping $D$-dimensional domain decomposition into bulks of connected cells
Clustering with triangular grid

C. E. Castro, M. Käser and E. F. Toro: Space-time adaptive numerical methods for geophysical applications

- The initial reason for clustering was advancing different timestep sizes in a cluster ⇒ independence to other cluster with “well defined” interfaces.
- We require communication via cluster’s shared edges and vertices.
Desired cluster and grid properties

Wishlist for a **HPC simulation** on dynamically changing grid:

1) **Cache aware memory access and reduced memory footprint**: Maintain compact data storage and access for cells within each cluster even with dynamically changing grid.

2) **Parallelization & domain decomposition**: Shared- and distributed memory.

3) **Efficient communication** for edges and vertices shared by two cluster.

C. E. Castro, et al.: Space-time adaptive numerical methods for geophysical applications

M. Schreiber: Parallelization of dynamically changing grids with a cluster-based approach and invasion
Parallelization of dynamically changing grids with a cluster-based approach and invasion, November 11th, 2013
Our solution: Heuristic based on Sierpiński SFC

Issue 1) **Cache aware memory access and reduced memory footprint:**
- We use the Sierpiński SFC for compact data storage and cache-oblivious computations.

  M. Bader, C. Zenger, Efficient Storage and Processing of Adaptive Triangular Grids Using Sierpiński Curves

Issue 2) **Parallelization:**
- Use **replication** of edges and vertices shared with adjacent cluster and **reduce** operations.

Issue 3) **En Bloc Communication:**
- Use run length encoding to store **information about adjacent cluster**.
  → More information on next slide
Outline

Simulations on dynamic adaptive grids
   Simulation and visualization requirements
   Introduction to clustering
   Issue 1) Cache aware memory access
   Issue 2) Parallelization
   Issue 3) Communication
   Software design
   Results

Invasive Computing
   Motivation
   Invasive Computing
   Results

Summary & Outlook
Issue 1) Cache aware memory access

Grid structure

- Create grid using **top-down** approach and **recursive bisection**.

- **Structure stack**: Grid structure is stored on a structure-stack which is read top-down during a traversal. 0: leaf element reached, 1: follow recursive definition

- Example:
Edge-based communication

- We still need to **access the element-data of adjacent elements**.
- Solution: **Edge-based communication via stacks**.
- We annotate the edges of the recursive definition using ’old’, ’new’ and ’boundary’ types:

\[ V' \]

\[ H' \]

\[ K' \]
Parallelization of dynamically changing grids with a cluster-based approach and invasion, November 11th, 2013
Parallelization of dynamically changing grids with a cluster-based approach and invasion, November 11th, 2013
M. Schreiber: Parallelization of dynamically changing grids with a cluster-based approach and invasion
Parallelization of dynamically changing grids with a cluster-based approach and invasion, November 11th, 2013
Parallelization of dynamically changing grids with a cluster-based approach and invasion

M. Schreiber
EdgeComm Stack
Left/Right

EdgeBuffer Stack
Node-based communication

- For visualization of a continuous surface, we require data exchange via nodes.
- Solution: **Node-based communication via stacks.**
Parallelization of dynamically changing grids with a cluster-based approach and invasion

M. Schreiber: Parallelization of dynamically changing grids with a cluster-based approach and invasion, November 11th, 2013
A
B C E
F
G
a
b
cd
e
fg
h
ij
k
l
m
n
D
o p
q
r
st
u
VertexComm Stack
Left/Right
cd a
be
f
VertexBuffer
Stack
Parallelization of dynamically changing grids with a cluster-based approach and invasion, November 11th, 2013
M. Schreiber: Parallelization of dynamically changing grids with a cluster-based approach and invasion
Parallelization of dynamically changing grids with a cluster-based approach and invasion, November 11th, 2013
VertexComm Stack
Left/Right
VertexBuffer Stack

M. Schreiber: Parallelization of dynamically changing grids with a cluster-based approach and invasion
Parallelization of dynamically changing grids with a cluster-based approach and invasion, November 11th, 2013
M. Schreiber: Parallelization of dynamically changing grids with a cluster-based approach and invasion
Parallelization of dynamically changing grids with a cluster-based approach and invasion, November 11th, 2013
Variance management and parameter unrolling

- High variety of different kernel interface requirements (normals, vertex coordinates, etc.)
- **branch mispredictions** - e.g. when testing edge type
- Solution: Use code generator and **parameter unrolling**
- $$K_{\text{traversal}}(\text{edge}\_\text{type}\_\text{hyp}, \text{edge}\_\text{type}\_\text{right}, \text{edge}\_\text{type}\_\text{left}, \text{int depth}, ...)$$
- $$\Rightarrow$$
  - $$K_{\text{nnn}\_\text{traversal}}(\text{int depth}, ...)$$
  - $$K_{\text{nno}\_\text{traversal}}(\text{int depth}, ...)$$
  - $$K_{\text{non}\_\text{traversal}}(\text{int depth}, ...)$$
  - ...
- Results for parameter unrolling:
  - **Reduced branch misprediction** from 62M to 13.1M
  - Severely **increased instruction cache misses** from 2.29M to 77.2M!
  - However: **29% faster** using average kernel operation
Outline

Simulations on dynamic adaptive grids
Simulation and visualization requirements
Introduction to clustering
Issue 1) Cache aware memory access
Issue 2) Parallelization
Issue 3) Communication
Software design
Results

Invasive Computing
Motivation
Invasive Computing
Results

Summary & Outlook
Issue 2) Parallelization & domain decomposition

Domain decomposition

Shared data scheme (SDS):
- Interfaces are shared by cell-chunks.
- No memory overhead due to shared communication buffer.
- Avoid race conditions on shared interfaces with proper synchronization.

Replicated data scheme (RDS):
- Data on shared interfaces are replicated.
- No race conditions possible.
- Reduce operations synchronize data on shared interfaces.

(M. Schreiber, T. Weinzierl and H.-J. Bungartz: SFC-based Communication Metadata Encoding for Adaptive Mesh, ParCo 2013, Garching)
Dynamic cluster generation

- Load balancing: SFC cuts


- Massive tree splits/joins:

- Hybrid methods: Load balancing via tree split/joins
Outline

Simulations on dynamic adaptive grids
  Simulation and visualization requirements
  Introduction to clustering
  Issue 1) Cache aware memory access
  Issue 2) Parallelization
  Issue 3) Communication
  Software design
  Results

Invasive Computing
  Motivation
  Invasive Computing
  Results

Summary & Outlook
Issue 3) Communication with RLE (edge)

Edge based communication via RLE encoding of replicated cluster boundary data.

Possible applications:
- Flux exchange
- Adaptivity communication information
- Flux limiter

(M. Schreiber, H.-J. Bungartz and M. Bader: Shared Memory Parallelization of Fully-Adaptive Simulations Using a Dynamic Tree-Split and -Join Approach, HiPC 2012, India)
RLE communication graph

Clustering

Edge communication graph
Updating RLE edge communication information

Stored initial adjacency information about sub-partitions A to D

Adaptive edge-communication-stacks after backward traversal for local sub-partition

Shared edge communication information stack and shared edge communication size counter
Issue 3) Communication with RLE (nodes)

Node based communication via RLE encoding of replicated cluster boundary data.

Possible applications:
- Visualization
- Node based flux limiter
- FEM, ...

(M. Schreiber, T. Weinzierl and H.-J. Bungartz: SFC-based Communication Metadata Encoding for Adaptive Mesh, ParCo 2013, Garching)
RLE encoded communication graph

Clustering

Node communication graph
Outline

Simulations on dynamic adaptive grids
  Simulation and visualization requirements
  Introduction to clustering
  Issue 1) Cache aware memory access
  Issue 2) Parallelization
  Issue 3) Communication
  Software design
  Results

Invasive Computing
  Motivation
  Invasive Computing
  Results

Summary & Outlook
Single cluster

- **Stacks**: Simulation data, communication, ...
- **Grid traversals**
- **Kernel**
- **Adaptivity**
- **Backend(s)**

**Simulation driver**: Setup, timesteps, sampling of domain, ...

**Data**

**Description**

**Grid- and cluster traversal**
Software design for parallel approach

Stacks
Sim., comm., ... data
Simulation driver
Setup, timesteps, sampling of domain, ... on cluster
user-data user-data
meta-data meta-data
Data Description Grid- and cluster traversal

Kernel Travs.
Kernel Travs.
Kernel Travs.
Kernel Travs.
Kernel Travs.
Kernel Travs.
Kernel Travs.
Kernel Travs.
Timest. Adapt. IO Timest. Adapt. IO Timest. Adapt. IO

Stack access
Trigger execution of grid traversal
Trigger execution of functions on cluster

M. Schreiber: Parallelization of dynamically changing grids with a cluster-based approach and invasion
Parallelization of dynamically changing grids with a cluster-based approach and invasion, November 11th, 2013
Cluster-based parallelization solves several issues

- Efficient hybrid parallelization
- Shared memory parallelization
- Cluster based local timestepping
- MPI data migration
- Work-stealing with massive splitting
- Sampling of simulation data
- Skipping of adaptivity traversals
- Field of view aware rendering

M. Schreiber: Parallelization of dynamically changing grids with a cluster-based approach and invasion, November 11th, 2013
Outline

Simulations on dynamic adaptive grids
- Simulation and visualization requirements
- Introduction to clustering
- Issue 1) Cache aware memory access
- Issue 2) Parallelization
- Issue 3) Communication
- Software design
- Results

Invasive Computing
- Motivation
- Invasive Computing
- Results

Summary & Outlook
Benchmark results for skipping of conforming cluster

Based on higher-order DG SWE simulation.
(M. Schreiber, T. Weinzierl and H.-J. Bungartz: Cluster Optimization and Parallelization of Simulations with Dynamically Adaptive Grids, Euro-Par 2013, Aachen)
Shallow water simulation with finite volumes

Strong- and weak-scalability on SuperMUC with finite-volume simulation
Outline

Simulations on dynamic adaptive grids
Simulation and visualization requirements
Introduction to clustering
Issue 1) Cache aware memory access
Issue 2) Parallelization
Issue 3) Communication
Software design
Results

Invasive Computing
Motivation
Invasive Computing
Results

Summary & Outlook
Motivation

Trend for many scientific algorithms:
- Runtime adaptivity: Applications with a \textit{dynamically changing workload}.
- This leads to different \textit{scalability behavior} during run-time.

Supercomputing centers:
- Concurrent execution of applications to overcome “low” scalability.
- Resources are \textit{statically allocated} once an application was started.

Invasive approach:
- \textit{Dynamic resource allocation} for applications with \textit{dynamic resource requirements}.
Origins of Invasive Computing

- Current trend towards thousands of processing elements on System on Chip, but
  - how to manage resources?
  - how to program such systems?
  - how to schedule resource-competing applications?
- Invasive Computing was suggested to tackle these issues.
  (J. Teich, J. Henkel, A. Herkersdorf, et al., Invasive computing: An Overview, Multiprocessor System-on-Chip)
Invasive Computing on HPC shared memory

Shared memory view:

- **Dynamically changing number of cores** for application’s code executed in parallel.
- **Optimization of resource distribution** by resource manager.
- **Pinning** of application to computing resources claimed during invade call.
- **reinvade()**: Optimize resource utilization for constraints already stored in resource manager.
Invasive Computing (IC) programming paradigm

- **claim = Invade(constraints):** Request resources that satisfy certain constraints. E.g. min/max number of cores which are requested by the application.

- **claim.infect(iLet):** Execute iLet-code (kernel-code for the invasive commands) on previously invaded resources.

- **claim.retreat():** Release reserved resources (allocated by invade).

Resource manager (RM)

RM schedules cores targeting at optimization of throughput based on hints provided by applications.

- Scalability graph
- Workload
- ...

Changing scalability for SWE radial dam break simulation.

Optimization of throughput based on two different scalability graphs.
Benchmark scenarios

E.g. parameter studies:

- Five Tohoku tsunami simulations, different refinement depths.
- No perfect scalability possible (loading of bathymetry data, output of buoy data)
- Based on FV Riemann solvers

George, David L., and Randall J. LeVeque, "Finite volume methods and adaptive refinement for global tsunami propagation and local inundation"

- Invasive hints for resource scheduling: cell-workload
  More workload \( \Rightarrow \) more resources.

Dynamic resource distribution
Results

- Robust decrease of overall execution time compared to OMP sequential execution
- Results for directly started OMP parallelized applications have been omitted: cache-thrashing leads to executed runtime of several orders of magnitude.

![Graph showing overall execution time for different scenarios](image)

Y-Axis given in log-scale
Results

- Larger workload
  ⇒ improved scalability
  ⇒ less possible benefits with Invasive Computing?
- However, still a runtime improvement of > 40% for larger simulations scenarios.
Outline

Simulations on dynamic adaptive grids
   Simulation and visualization requirements
   Introduction to clustering
   Issue 1) Cache aware memory access
   Issue 2) Parallelization
   Issue 3) Communication
   Software design
   Results

Invasive Computing
   Motivation
   Invasive Computing
   Results

Summary & Outlook
Summary & Outlook

Summary:

- Scalable simulations on dynamic adaptive triangular grids.
- Cluster-based parallelization and optimization.
- Potential of Invasive Computing presented on shared-memory systems.

Outlook:

- Earth-scale: Oceanic, weather & climate.
- Multi-layer simulations.
- Invasive Computing on super-computing centers (distributed-memory).
Thank you for your attention.
Any questions?

martin.schreiber@in.tum.de

More infos & videos:
http://www5.in.tum.de/sierpinski/