SFC-based Communication Metadata Encoding for Adaptive Mesh Refinement

Martin Schreiber, Tobias Weinzierl, Hans-Joachim Bungartz

September 10, 2013
Sierpiński for 2D triangular grids

(Created in collaboration with Alexander Breuer. Based on GeoClaw solver.)
Peano for nD Cartesian grids

(Created in collaboration with Kristof Unterweger. Based on PyClaw solver.)
Motivation

Study **dynamic, parallel adaptive** mesh refinement (AMR) PDE solvers based upon spacetrees:

- **Domain decomposition** in combination with **space-filling curves** (SFCs) is state-of-the-art as **SFC** partitions serialize grid into well-suited **subdomains**.
- Handling of connectivity, exchange cardinality, communication partners non-trivial: This is what we refer to as **meta data**.
- **Parallelization approaches** depend on the decomposition and communication: **shared** or **replicated** data scheme.

We study this issues with two parallel AMR frameworks (Peano, Sierpiński) and exploit the underlying SFC and spacetree.
Outline
Domain decomposition and parallelization

**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.
Outline
SFC induces partitioning

- SFC serializes cells into a single stream.
- Domain decomposition of SFC induces decomposition of stream.
RLE encoded input stream

- Store per leaf how many records are read by cell for first time.
- Restrict information on chunks along tree for entire sub-trees.
Parallelization strategy

- Fork tasks top-down along tree.
- Per fork: Split input stream into separately read streams (stack views).
- It is uniquely determined which task reads (and owns) which data from input stream.
- Realize streaming to output stream analogously.
Case study: Load and store process

- Poisson equation access scheme
- 100 Iterationen, 1.5 Mio. cells, 1.5 Mio. vertices.
- Chunks are setup by regular 2D sub-trees.
- No computations, only data load/store.
Outline
SFC based clustering

- **Grid meta- and payload-data** is stored independent to other subdomains.
- **Replicated communication data + multiple independent subdomains in program context** $\approx$ clusters.
- **Cluster traversal is based on recursive spacetree-traversal**

M. Bader, K. Rahnema and C. A. Vigh: Memory-Efficient Sierpinski-Order Traversals on Dynamically Adaptive, Recursively Structured Triangular Grids

Figure: Left: Example of grid. Right: Corresponding space-tree.
RLE communication example (edge)

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

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

Clustering

Edge communication graph
RLE communication examples (nodes)

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

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

RLE meta information
- Communication data example
- Right vertex comm. stack
- Left vertex comm. stack
RLE encoded communication graph
Adaptivity traversal:

- **Refine and coarsen cells dynamically.**
- **Close to wave-front**, refine cells using newest vertex bisection, otherwise coarsen cells.
- **Update RLE communication** information based on refine and coarsening information consecutively stored on edges shared by clusters.
Results: MPI scalability on regular grid

- RLE based edge communication
- Regular grid
Results: DG simulation with RLE based communication

- Edge-based communication used for DG simulation
- Node-based comm. used for visualization of closed surface
Model: Memory storage savings with RLE

- **Cell-wise storage** demand vs. **cluster RLE based** demand
- Regular grid.

**Figure**: Varying amount of payload per cell and cells in each cluster
Summary & Outlook

We presented two different parallelization approaches:

- **Shared data** with single input/output stream
- **Replicated data** with duplicated (distributed) meta and communication data

Outlook/next steps:

- **MPI load balancing & hybrid** parallelization (Sierpiński, published soon)
- **Augment meta data** by machine properties such as latency to guide load balancing (Peano)
- Large scale case studies (both codes)
Thank you for your attention

This work was supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre “Invasive Computing” (SFB/TR 89). It is partially based on work supported by Award No. UK-c0020, made by the King Abdullah University of Science and Technology (KAUST). All software is freely available at http://www5.in.tum.de/sierpinski/ or http://www.peano-framework.org/.