Concepts for performance improvements of shallow water flow simulations

Lennart Steffen[1*], Finn Amann[1], Reinhard Hinkelmann[1]

[1]Technische Universität Berlin, Chair of Water Resource Management and Modeling of Hydrosystems, Berlin, Germany

*Correspondence YP: lennart.steffen@wahyd.tu-berlin.de

IAHR YPN

Hosted by Spain Water and IWHR, China

wahyd.tu-berlin.de

Abstract

Flow simulations with the depth-averaged shallow water equations are applicable to many naturally occurring surface water flows, such as open channel flow or floods. Improving the performance of simulation software allows larger simulations or lower time and energy requirements. To explore avenues for performance improvements, an open-source C++ framework for explicit finite-volume-method solvers was designed and a robust shallow water equations solver was implemented. First results show significant speedups from vectorisation of cell-based calculations, and preliminary tests show measurable speedups from reordering edge flux computations to reduce cache misses and locks between threads. Furthermore, a simple load rebalancing scheme for distributed computations is introduced and two load criteria are compared.

Keywords: shallow water equations, finite volume method, high performance computing.

Methods

Different approaches for performance improvements are investigated:

- **vectorisation** of cell-based computations
- **reordering flux calculations** (see Fig. 1) to improve data locality and to reduce locks between threads
- **dynamic load balancing**: simple two-stage algorithm, aggregate criteria (e.g. processor load), cell-based criteria (wet/dry flags)

Results

Vectorisation of cell-based computations:
- comparably simple to implement using free numerics libraries (here: eigen3)
- yields large speedups, time share of respective steps reduced from ~50% to 10-15%

Reordering flux calculations:
- yields measurable, but lower speedups
- interfaces between threads cause significant wait times when using atomic write accesses

Dynamic load balancing:
- similar partitioning quality between aggregate and highly resolved cell-based criteria
- aggregate criterion much cheaper

Conclusions

Vectorisation of cell-based computations: comparably simple and yields good results

Reordering flux calculations:
- main focus for optimisation going forward
- useful instrument to increase data locality
- yields promising first results
- removing locks between threads is avenue for further optimisation

Dynamic load balancing: usage of aggregate criterion superior to cell-based one, due to lower computational costs and similar results.

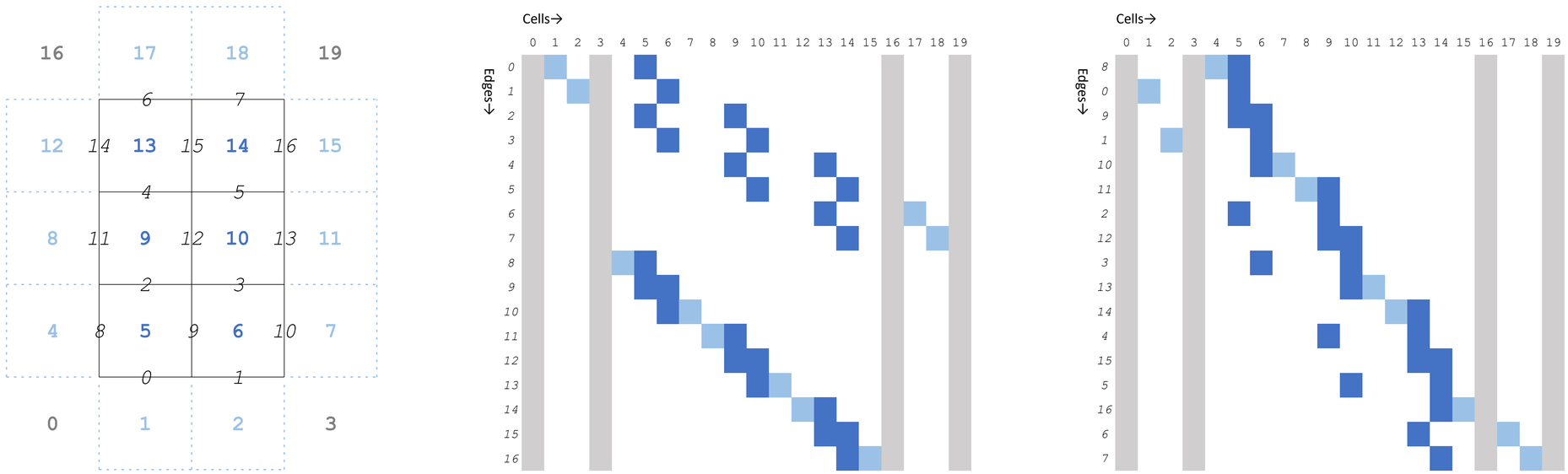

Figure 1: Edge traversal strategies. Left: Example mesh with cell (bold) and edge indices (italic). Middle: ordered traversal. Right: row-wise traversal. Memory access follows top-to-bottom axis. Row-wise traversal shows better locality of cell data.