SYCL and Block-Structured Grids: Performance Impact on Simulations of Complex Costal Ocean Domains

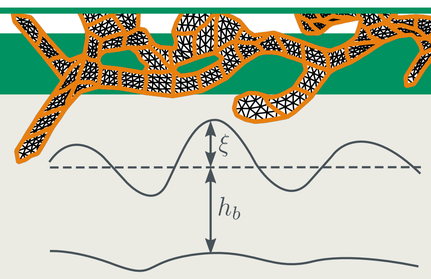
Jonathan Schmalfuß [a,*], Daniel Zint [b], Sara Faghih-Naini [c], Julian Stahl [d], Markus Büttner [a], Roberto Grosso [d], Vadym Aizinger [a]

[a] University of Bayreuth, Scientific Computing; * jonathan.schmalfuss@uni-bayreuth.de

[b] New York University, Courant Institute of Mathematical Sciences; [c] European Centre for Medium-Range Weather Forecasts; [d] Friedrich-Alexander-Universität Erlangen-Nürnberg, Department of Computer Science

Mathematical Model: 2D shallow water equations

ξ: elevation of the free water surface, h_b: bathymetric depth, $H = h_b + \xi$: total fluid depth, f_c: Coriolis parameter, $(U, V)^T$: depth-integrated horizontal velocity field, $c = (\xi, U, V)^T$: local unknowns, F: forcing term, g: gravitational acceleration, τ_{bf}: bottom friction coefficient



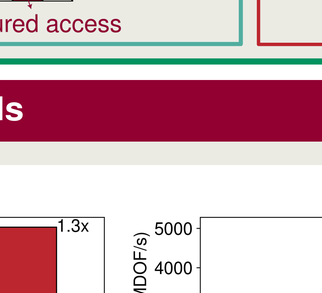
$$\begin{pmatrix} \frac{\partial \xi}{\partial t} \\ \frac{\partial U}{\partial t} \\ \frac{\partial V}{\partial t} \end{pmatrix} + \nabla \cdot \begin{pmatrix} U & V \\ \frac{U^2}{H} + \frac{g\xi(H+h_b)}{2} & \frac{UV}{H} \\ \frac{UV}{H} & \frac{V^2}{H} + \frac{g\xi(H+h_b)}{2} \end{pmatrix} = \begin{pmatrix} 0 \\ -\tau_{bf}U + f_c V + g\xi\partial_x h_b + F_x \\ -\tau_{bf}V - f_c U + g\xi\partial_y h_b + F_y \end{pmatrix}$$

Numerical Scheme: Discontinous Galerkin (DG) Discretization[1]

$$(\partial_t c_\Delta, \phi_\Delta)_{\Omega_e} \underbrace{- (A(c_\Delta), \nabla\phi_\Delta)_{\Omega_e} - (r(c_\Delta), \phi_\Delta)_{\Omega_e}}_{\text{Element integrals}} + \underbrace{\langle \hat{A}(c_\Delta, c_\Delta^+, n), \phi_\Delta \rangle_{\partial\Omega_e}}_{\text{Edge integrals}} = 0$$

- Domain partitioned using triangular mesh
- Fully explicit scheme, no iterative solvers
- Modal Discontinuous Galerkin (DG) bases, low order (piecewise constant/ linear/quadratic)
- Easy to parallelize, only one global barrier after each Runge-Kutta stage

BSGs[3]: Tidal flow scenario in The Bahamas - Bight of Abaco



- topologically block-structured grid (BSG): an unstructured collection of blocks, each containing a structured grid
- 10 BSGs with each 1.4 mio elements, block sizes = {32, 50, 98, 128, 200, 242, 392, 450, 800, 882}

SYCL Code[2]: Support for CPUs, GPUs, and FPGAs

UTBEST - SYCL

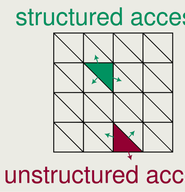
- parallization over elements → work group size of 256 items
- edge integrals are computed redundantly, only local solution is updated
- struct of arrays layout for memory efficiency
- unstructured memory accesses cost mitigated by hardware caches

```
// Pseudocode - not actual C++
for (int step = 0 ... nsteps)
    for (int rk = 0 ... rk_stages)
        for (int element = 0 ... num_elements)
            integration::process_element(/* omitted */);
```

SYCL Code: Support for Block-Structured Grids on CPUs and GPUs

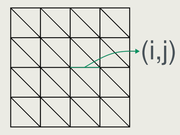
base: local memory only

- work_group_size = block_size
- implemented via sycl::local_accessor
- interior block elements solution is from local memory



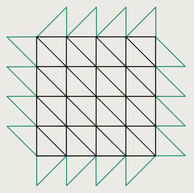
+ minimize bytes per element

- deduplicate information e.g. normals, measure only stored once per edge per block
- different treatment of block boundary edges
- explicit indexing via mappings
- introduces additional algorithmic complexity

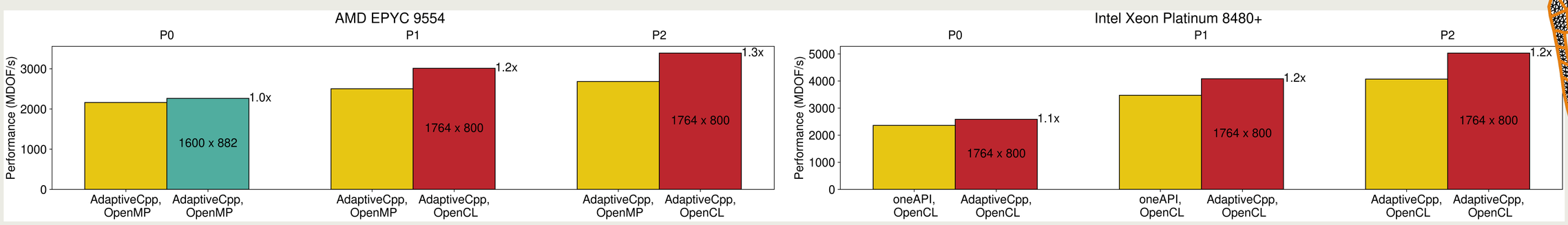


+ block wise halo layer

- reformulate boundary conditions (edge to element)
- enables equal treatment of all edges of block
- increases memory requirements

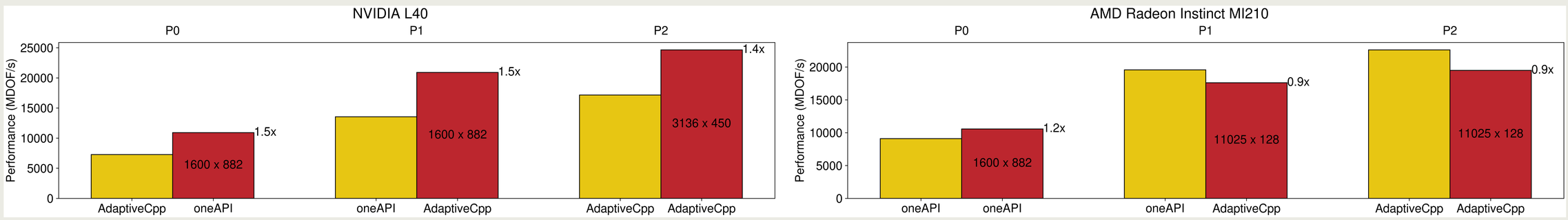


Results & Discussion: Performance Impact of Block-Structured Grids



- Figure: BSG Impact on CPUs, best performing compiler and backend, per order left bar: unstructured performance, per order right bar: block-structured grid, color according to different algorithmic strategies

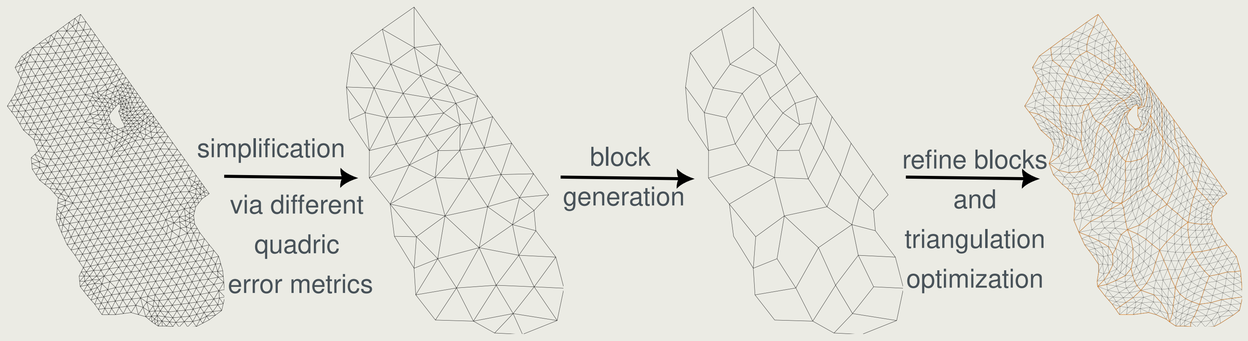
- largest speedup visible for higher order
- lower order converging towards best unstructured performance
- similar behaviour for AMD EPYC 7742
- almost no speedup visible for lower orders, larger up to 1.2 visible for higher orders
- with OpenCL on Intel Xeon architecture vectorizes only for block sizes divisible by 16



- Figure: BSG Impact on GPUs, best performing compiler and backend, per order left bar: unstructured performance, per order right bar: block-structured grid, color according to different algorithmic strategies

- equal speedup visible for all orders, outperformaing unstructured up to factor 1.5
- performance almost identically between compilers
- similar behaviour for NVIDIA Quadro RTX 6000
- small speedup visible for lower orders, degregation for higher orders
- similar challenges for NVIDIA H100

Block-Structured Grid Generation: HPMeshGen[3]



Support BSGs Generation:

- standard BSGs = structured blocks
- masked BSGs = structured blocks + masking
- hybrid BSGs = structured blocks + unstructured blocks

Outlook

- Support of unstructured blocks
- BSGs for FPGAs
- MPI + BSGs
- Improving BSGs generation
- ...

References and Acknowledgements

[1] V. Aizinger and C. Dawson. "A discontinuous Galerkin method for two-dimensional flow and transport in shallow water". In: Advances in Water Resources 25.1 (2002)
[2] M. Büttner, C. Alt, T. Kenter, H. Köstler, C. Plessl, and V. Aizinger. “Analysing performance portability for a SYCL implementation of the 2D shallow water equations”. In: The Journal of Supercomputing (2025)
[3] D. Zint, R. Grosso, V. Aizinger, S. Faghih-Naini, S. Kuckuk, and H. Köstler. "Automatic Generation of Load-Balancing-Aware Block-Structured Grids for Complex Ocean Domains". 2022 SIAM International Meshing Roundtable (IMR).

System used for performance measurements: Festus (Bayreuth)
Parts of this Research are funded by Deutsche Forschungsgemeinschaft under grants AI 117/7-1 and KE 2844/1-1 and by KONWIHR project "Performance Optimization for the SYCL implementation of UTBEST