Heinrich C. Mayr, Martin Pinzger (Hrsg.): INFORMATIK 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 2053

A Parallel Non-Hydrostatic Shallow Water Model on Adaptive Triangular Meshes in sam(oa)^2

Philipp Samfaß[1] and Raphael Schaller[2]

Abstract: Even with current extreme scale systems, the accurate simulation of tsunamis continues to be a challenging problem. One commonly used model for this task are the *hydrostatic* shallow water equations which, however, are not able to represent all relevant physical effects of tsunamis. In this paper, we therefore show how to solve the *non-hydrostatic* shallow water equations in parallel within the partial differential equation framework sam(oa)^2 by extending the existing finite volume method for solving the hydrostatic shallow water equations. We present an element-oriented dual grid discretization of the equation for the non-hydrostatic pressure on triangular adaptive meshes which allows for a matrix-free conjugate gradient solver for the corresponding system of linear equations. In addition, the validity of the new model is shown based on two common numerical experiments for non-hydrostatic shallow water models. Because we aim at solving large-scale problems on high performance computing architectures, we demonstrate our distributed memory parallelization which resulted in a parallel efficiency of 90.1% from 1 to 8192 cores for weak scaling and 96.8% from 16 to 512 cores for strong scaling.

Keywords: Adaptive triangular mesh, conservation law, dual grid, element-oriented method, finite volume method, high performance computing, matrix-free solver, non-hydrostatic shallow water equations, parallelization, tsunami

1 Introduction

During the past few decades, computation resources on extreme scale systems have significantly increased. This has resulted in an increasing interest of scientists in using such high performance machines in order to simulate important problems stemming from natural sciences and engineering. One of these problems is the modelling and prediction of tsunamis. Considering the devastating consequences of a tsunami such as the one caused by the Tōhoku earthquake in Japan in 2011, the critical need for accurate computational models becomes evident. Valid mathematical and physical models should be efficiently implemented on current high performance systems. This is a requirement to obtain meaningful insights at scale into the development and propagation of tsunamis.

Even with current supercomputers having their peak performance in the petaflop regime, a full three-dimensional model for the simulation of a tsunami might still not be computationally feasible. Instead, the so-called shallow water equations provide a reasonable two-dimensional approximation by essentially depth-averaging the full three-dimensional governing equations. A model that is commonly used—the hydrostatic shallow water model—neglects the impact of the vertical velocity and assumes that the pressure is given by the

[1] Technical University of Munich, Department of Informatics, samfass@in.tum.de
[2] Technical University of Munich, Department of Informatics, schaller@in.tum.de

hydrostatic pressure (the result of gravity). This applies to (long) shallow water waves that are characterized by $h/\lambda \ll 1$, where h is the height of the water column and λ the wave length [LN08]. However, the corresponding hydrostatic shallow water equations do not model the dispersion of deeper water waves that might e.g. occur near the coast [LN08].

Therefore, efforts have been made to extend existing hydrostatic solvers towards a modified non-hydrostatic shallow water model (e.g. in [Cu13], [Fu13], [SZ03] or [Wa05]). Here, the general approach is to correct the solution produced by the hydrostatic solver in each time step while still retaining a 2D depth-averaged model. As will be discussed later, this correction requires to solve a system of linear equations in each time step.

In this paper, we first show how to adapt this scheme to solving the non-hydrostatic shallow water equations using a conservation-law-based finite volume method for solving the hydrostatic shallow water equations (Sec. 2.1). In Sec. 2.2, we present an element-oriented approach towards solving the system of linear equations for the non-hydrostatic pressure using a dual-grid discretization. Our extension has been integrated into sam(oa)^2 [MRB11] which is a parallel framework for solving partial differential equations (PDE) on adaptive triangular meshes using space-filling curves for cache and memory efficient grid traversals. Based on some common experiments for non-hydrostatic models, we show the validity of the new model (Sec. 4.1) and investigate the feasibility of using a conjugate gradient solver for the system of linear equations (Sec. 4.2).

Since we aim at solving large-scale problems on high performance computing (HPC) systems, we furthermore discuss our parallelization strategy for the model which is based on sam(oa)^2's parallelization interface (Sec. 3). Finally, strong and weak scaling experiments verify a successful parallelization (Sec. 4.3).

2 Numerical Approach

In this section, we briefly introduce the non-hydrostatic shallow water equations, the assumptions on which they are based and the general approach towards solving them. Next, we present our element-oriented spatial discretization on triangular adaptive meshes in sam(oa)^2 that allows for a matrix-free linear solver.

2.1 Finite Volume Formulation of the Non-Hydrostatic Shallow Water Equations

The subsequently described numerical approach for the non-hydrostatic extension closely follows the one presented in [Cu13] and [Fu13]. Special attention is payed to the modifications required to combine this approach with our conservation-law-based finite volume solver (cf. [Le02]) for the hydrostatic shallow water equations.

As a basis for the extension, we use the non-hydrostatic shallow water equations in conservation law form (see [Sa14] for a derivation based on [Ma]):

$$\begin{bmatrix} h \\ hU \\ hV \\ hW \end{bmatrix}_t + \begin{bmatrix} hU \\ hU^2 + \frac{1}{2}gh^2 \\ hUV \\ hUW \end{bmatrix}_x + \begin{bmatrix} hV \\ hUV \\ hV^2 + \frac{1}{2}gh^2 \\ hVW \end{bmatrix}_y = \begin{bmatrix} 0 \\ -ghb_x - \left([\frac{h\hat{q}}{2}]_x + \hat{q}b_x \right) \\ -ghb_y - \left([\frac{h\hat{q}}{2}]_y + \hat{q}b_y \right) \\ \hat{q} \end{bmatrix}. \quad (1)$$

Here, $h(x,y) = \eta(x,y) - b(x,y)$ denotes the height of the water column, $\eta(x,y)$ the free surface elevation above the mean sea level, $b(x,y)$ the bathymetry (elevation of the sea floor), (U,V,W) the depth-averaged velocity vector, g the gravity of earth and a subscript the partial derivative with respect to the given coordinate. Further, $q(x,y,z)$ represents the non-hydrostatic portion of the pressure and $\hat{q} = [q]_{z=b}$ the non-hydrostatic pressure at the sea bottom. The hydrostatic shallow water equations are obtained as a special case by setting $q = 0$ and neglecting the vertical momentum equation.

The above equations include the assumption that the pressure can be decomposed into a hydrostatic part p_H and a non-hydrostatic portion q as introduced in [CS98]. We further assume a linear vertical distribution of the non-hydrostatic pressure (decreases to zero at the surface, maximum at the bottom) and the vertical velocity component (maximum at the water surface) [Wa05]. Since bathymetry is represented as a step function in our model, the vertical velocity w is zero at the sea bottom [Cu13]. Under these assumptions, $W = 0.5[w]_{z=\eta}$ holds.

We employ a fractional step scheme towards extending the previously existing hydrostatic solver similar to the one used in [Cu13]: in each time step, the quantities of the hydrostatic solution will be computed and then corrected with the numerical non-hydrostatic pressure parameter $\hat{q}$ at the sea bottom—the new unknown in the non-hydrostatic model. This gives rise to the need of solving a system of discretized Poisson-like equations for $\hat{q}$. The system is obtained using the pressure projection method ([Ch68]) that is e.g. commonly used to solve the Navier-Stokes equations [GDN95]. Our correction formulas for the discharges hU, hV computed by the hydrostatic solver follow from an explicit Euler time discretization of the non-hydrostatic equations (1). For correcting the vertical velocity field, we follow [Cu13] and neglect the non-linear terms in the vertical momentum equation. In summary, the main steps in a non-hydrostatic timestep are (in this order): compute a timestep with the hydrostatic solver (resulting in an intermediate solution), solve a linear system of equations for $\hat{q}$ and correct the intermediate solution with $\hat{q}$.

2.2 Dual Grid Discretization on Triangular Meshes in sam(oa)^2

Sam(oa)^2's particular setup allows efficient access only to element-local data in each grid traversal step such that an element-oriented and matrix-free assembly of the system of linear equations for $\hat{q}$ is required. Instead of collocating the control volumes for computing the non-hydrostatic pressure with the finite volume cells, we therefore propose a dual grid arrangement as shown in Fig. 1.

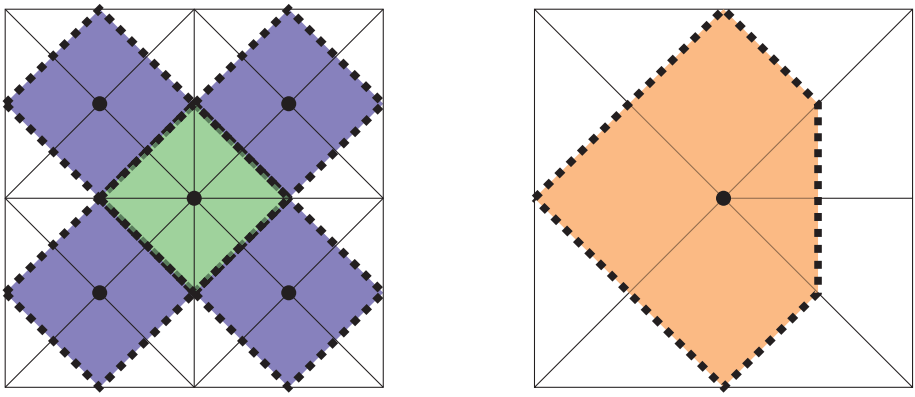

Fig. 1: Control volumes/dual grid cells (dashed) for the non-hydrostatic pressure: the left picture shows the two kinds (green and blue) of dual grid cells that might occur in a grid of uniform refinement depth. On the right, a control volume for the non-uniform case is depicted. Note that an element will contribute to three different control volumes. The points denote the nodes on which the unknowns $\hat{q}_d$ and the vertical velocity w_d for a dual grid cell d are located. For each $\hat{q}_d$, there is exactly one corresponding control volume/dual cell d.

Our finite volume method for solving the hydrostatic shallow water equations is basically a Galerkin method with piecewise constant ansatz and test functions that determines for each primary cell an average $Q_i^n = (h_i^n, [hU]_i^n, [hV]_i^n)^T$. For compatibility reasons, the same choice of basis and ansatz functions has also been made for discretizing $\hat{q}$ and w.

With these remarks, the basis for deriving the system of linear equations for the non-hydrostatic pressure will be the following weak form of the continuity equation given for a dual grid control volume:

$$\int_{\Gamma_d} n_x hU \, d\Gamma + \int_{\Gamma_d} n_y hV \, d\Gamma + \int_{\Omega_d} w \, d\Omega = 0 \tag{2}$$

Here, Γ_d denotes the boundary, $\mathbf{n} = (n_x, n_y)$ its outward pointing normal unit vector and Ω_d the region of the dual cell d. Note that some terms occurring in the derivation of this formula had to be neglected since they cannot be directly obtained from the hydrostatic quantities computed in our model[3].

Let N_d denote the set of primary elements that contribute to the dual cell d, $\Gamma_{e,d}$ the part of the boundary of d that lies in the primary element e and $\Omega_{e,d}$ the region of the dual cell that lies within e (compare Fig. 1). Then, an element-wise assembly of the equation as a sum of element contributions is reflected in the following equivalent form of (2):

$$\sum_{e \in N_d} \left(\int_{\Gamma_{e,d}} n_x hU \, d\Gamma + \int_{\Gamma} n_y hV \, d\Gamma + \int_{\Omega_{e,d}} w \, d\Omega \right) = 0. \tag{3}$$

[3] For instance, $\int_{V_d} \frac{\partial u}{\partial x} dV = \int_{\Gamma_d} n_x hU d\Gamma + \int_{\Omega_d} u_{z=b} \frac{\partial b}{\partial x} - u_{z=\eta} \frac{\partial \eta}{\partial x} d\Omega$ holds and the second and third summands had to be neglected.

Plugging the correction formulas for the discharges hU, hV and the vertical velocity w into the summand of (3) gives the following dual cell equation contribution:

$$\int_{\Gamma_{e,d}} \mathbf{n} \cdot \left[\begin{pmatrix} \widetilde{hU}^{(n+1)} \\ \widetilde{hV}^{(n+1)} \end{pmatrix} - \Delta t \frac{h^{(n)}}{2} \nabla \hat{q} \right] d\Gamma + \int_{\Omega_{e,d}} \widetilde{w}^{(n+1)} + 2\Delta t \frac{\hat{q}}{h^{(n+1)}} \, d\Omega = 0. \qquad (4)$$

In contrast to a collocated scheme, our dual grid arrangement avoids having to evaluate derivatives of hydrostatic quantities at primary cell boundaries (the location of a discontinuity between two cell averages) which would be particularly problematic in the adaptive case. The gradient of the non-hydrostatic pressure term $\nabla \hat{q}$ at the dual cell boundaries is approximated using central finite differences. Both the normal vectors and the non-hydrostatic pressure gradient have to be rotated appropriately from a canonical standard reference element orientation into the actual orientation of the considered element.

In our implementation, during a non-hydrostatic grid traversal (after the hydrostatic step), all three dual cell equation contributions (4) will be computed yielding 3×3 local element matrices for the unknowns on the three nodes of a primary element. In another traversal, the linear solver determines the local residuals for the dual cell equations using these element matrices. Finally, the unknowns $\hat{q}$ on the nodes are updated according to the chosen iterative method for solving the system. In another traversal, the obtained solution will be used to correct the cell averages and the averaged vertical velocity.

3 Parallelization

In order to be able to solve large-scale problems, the non-hydrostatic extension is intended to being run on current HPC systems. While the linear solver and the hydrostatic component had already been parallelized in previous work, our non-hydrostatic extension required a solid parallelization strategy as presented subsequently.

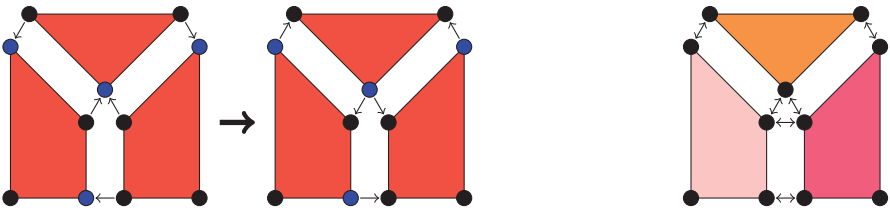

Fig. 2: Merging process. Left two pictures: Merging sections owned by one process. Rightmost picture: Merging sections owned by multiple processes. Color of sections denote their process affiliation. The blue marked nodes are master nodes. The figure is based on [MB15].

3.1 Parallelization in sam(oa)^2

Sam(oa)^2 provides solid groundwork for parallelization: by ordering elements along a space-filling curve and then cutting sequentialized elements into sections, the grid is automatically split into atomic work units which are distributed to the processes. In each time

step, sam(oa)^2 also transparently handles load balancing. When splitting the grid, border nodes belonging to multiple sections are duplicated (see Fig. 2). Therefore, only partial results are stored on the border nodes which requires a merging of the nodes. Merging is performed in different ways depending on whether the border nodes' sections belong to one process (shared memory) or to multiple processes (distributed memory), as depicted in Fig. 2. For distributed memory, a peer-to-peer merging is conducted, so each node is merged with each other node. For shared memory, sam(oa)^2 merges the nodes master-slave-wise: first, all slave nodes are merged into one master node, then the master's result is copied back to the slaves.

3.2 Merge Operators

Merge operators can be used to define how sam(oa)2 merges the nodes. A merge operator takes two nodes, the local node and the neighbor node, with the latter being read only. The merge operator then shall merge the neighbor node into the local node.

Here, two merge operators are required: one after setting up the system of equations and one for the correction traversal. In the former, the partial right hand sides of the system of linear equations are merged. In the latter, we merge two helper variables which are required for the correction. For all these cases, we can abstract the problem to a sum being computed on the node, which sums information stored on the node's adjacent cells:

$$\hat{p}_n = \sum_{\forall m \in M_n} p_{m;n}, \tag{5}$$

where $\hat{p}_n$ is the result to be stored on node n, M_n is the set of cells adjacent to this node and $p_{m;n}$ is the information on the cell m for this node. If node n is a border node, the adjacent cells are a subset of M_n, so only a partial sum is computed. Since $M_{n,1} \cup M_{n,2} \cup \ldots \cup M_{n,k} = M_n$, where $M_{n,i}$ denotes the set of adjacent cells of node n in section i, we get

$$\hat{p}_n = \sum_{\forall m \in M_{n,1}} p_{m;n} + \sum_{\forall m \in M_{n,2}} p_{m;n} + \ldots + \sum_{\forall m \in M_{n,k}} p_{m;n}. \tag{6}$$

Thus, the merge operators need to compute the sum of the partial results stored on the nodes. For merging the right hand side, it generally looks like this:

```
local_node.rhs := local_node.rhs + neighbor_node.rhs
```

4 Results

4.1 Physical Validation

In order to validate the new model, we conducted different common experiments for non-hydrostatic models. To reliably ensure convergence we used for these experiments a simple Jacobi solver with a maximum absolute local residual threshold of $\varepsilon = 0.1$. However, as we show in Sec. 4.2, the CG method is suitable for solving the system, too.

Standing Wave

As a first experiment, we simulated a standing wave in a closed (wall boundaries) cuboidal basin of quadratic shape with length 10 m. This experiment has been conducted in other related work such as [Cu13], [Fu13] and [SZ03]. For details on the experimental setup and the analytical solution, please see [Fu13]. The constant depth of the basin d is incrementally increased yielding a higher ratio d/λ such that a hydrostatic model fails to approximate the correct propagation speed as can be seen in Fig. 3.

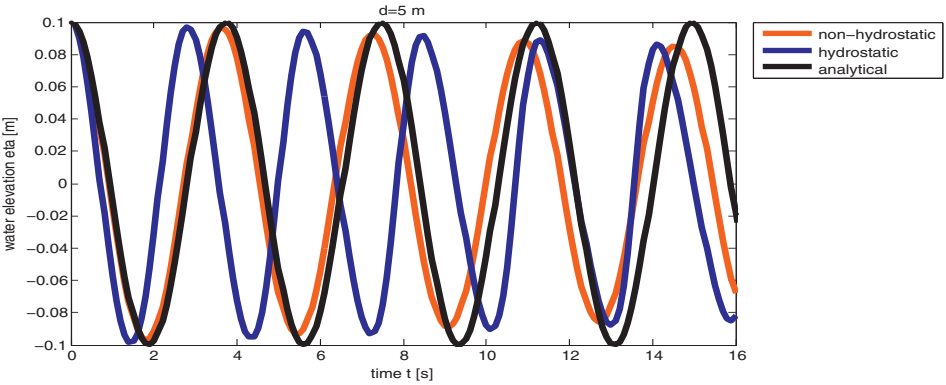

Fig. 3: Standing wave: plot of the water elevation at $(x,y) = (10\,\text{m}, 5\,\text{m})$ over time. 32 768 cells were used for the uniform grid.

Notice the numerical damping for both the hydrostatic and the non-hydrostatic model. This is a common phenomenon especially for low-order discretizations like the one we employ here.

Solitary Wave

In the next test case, the propagation of a solitary wave in a closed channel of width 2 m and depth $d = 10\,\text{m}$ is examined—another standard test case for non-hydrostatic models (cf. e.g. [Cu13], [SZ03], [Wa05] and [WJ00]). Since viscosity and friction are absent in our model, the solitary wave should not deform while propagating. As depicted in Fig. 4, the non-hydrostatic solution keeps the correct water level over time with some small trailing waves. These trailing waves have been observed in other non-hydrostatic models such as [SZ03]. They are likely due to the spatial discretization's inherent failure of resolving large eigenmodes of the problem correctly. Fig. 5 shows that the hydrostatic model produces an overall wrong sawtooth shaped water surface profile.

4.2 Solver

Even if the physical validation experiments were successfully performed by using the Jacobi method, we intended to apply conjugate gradient (CG) to benefit from faster con-

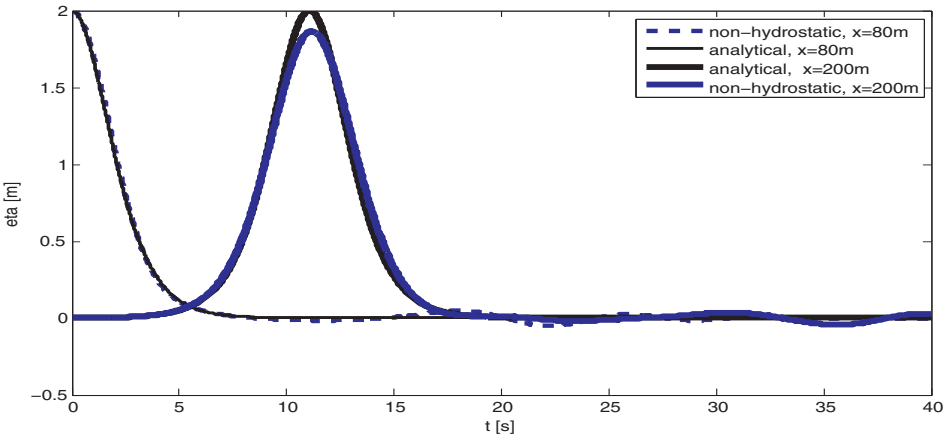

Fig. 4: Solitary wave (#cells = 38400): water elevation over time at the points $(x_1, y_1) = (80\,\mathrm{m}, 0.25\,\mathrm{m})$ and $(x_2, y_2) = (200\,\mathrm{m}, 0.25\,\mathrm{m})$. For the sake of clarity, the hydrostatic solution is omitted in this plot.

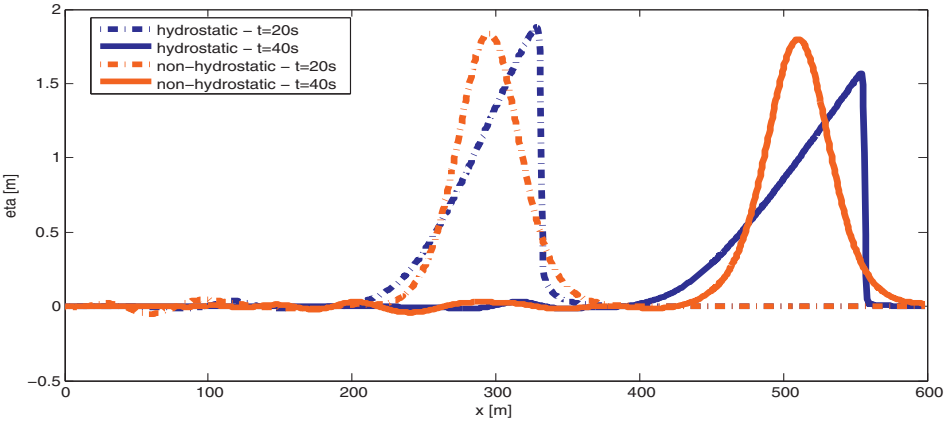

Fig. 5: Solitary wave (#cells = 38400): water level at two different points in time.

vergence. However, CG is only defined for symmetric positive definite system matrices [HS52]. Although we expected the matrix to be symmetric positive definite for constant bathymetry, we conducted tests by simulating several scenarios and checking the properties of the global matrix at every time step. We only found symmetric positive definite system matrices during these tests. Certainly, our tests did not cover all possible scenarios, so even if the matrix was symmetric positive definite in our case, these results do not prove that the matrix will also be symmetric positive definite for other scenarios.

The blue line in Fig. 6 displays the average number of CG iterations for the first 0.05s of simulation time of the standing wave scenario for different grid resolutions. Starting with 58 iterations for 32k cells, we ended at 812 iterations for 256M cells. The line is neatly fitted by the function

$$y = 1.2712x^{0.3258}, \tag{7}$$

where x is the number of cells and y the number of iterations. It is depicted in Fig. 6 as a dashed line. Thus, for this scenario, the number of iterations increases by a factor of about $2^{0.3258} = 1.2534$ if the number of cells is doubled. So even with perfect weak scaling (see Sec. 4.3), one has to expect at least an 25% increase in simulation run time each time the number of cells is doubled.

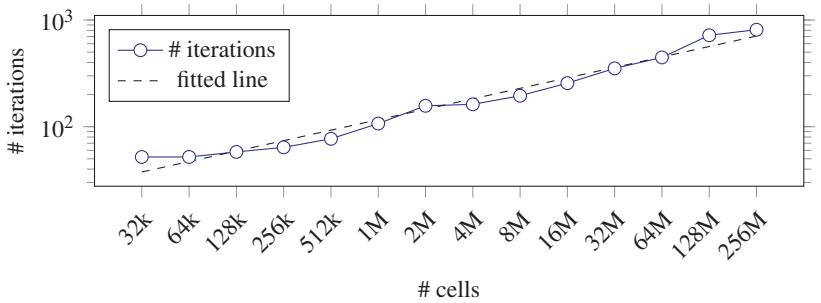

Fig. 6: Average number of CG iterations for the first 0.05s seconds of simulation time of the standing wave scenario for different grid resolutions. The dashed line shows the fitted function as denoted in Eq. 7.

4.3 Scaling

All measurements were performed on the SuperMUC[4] Thin Nodes, a supercomputer operated by the Leibniz Supercomputing Center near Munich. Each node is equipped with two eight-core Xeon E5-2680 processors. One island incorporates 512 nodes, resulting in 8192 cores per island. The nodes are connected via Infiniband FDR10. We used the standing wave scenario and always placed one MPI process and one section on each physical core with section splitting[5] turned on. We employed *PipeCG* as solver which is a specialized version of CG. While being equivalent to CG up to rounding errors, it reduces the global communication (see [GV14] for more information).

We measured performance in *element throughput*, which is defined as the number of elements which are handled per second per core. Thus, since this is a measure of parallel efficiency, a constant element throughput indicates perfect scaling.

Weak Scaling

The results of the weak scaling measurements are displayed in Fig. 7. Each measurement ran for 0.05 s of simulation time. The scaling was conducted for one up to 8192 cores with

[4] http://www.lrz.de/services/compute/supermuc/systemdescription/

[5] Normally, sam(oa)^2 treats sections as atomic work units (see Sec. 3). For load balancing, however, sections can be split and partially transfered to other processes.

a grid resolution of 32k cells up to 256M cells. Furthermore, a run on 16384 cores and a grid resolution of 512M cells was performed, but for 0.0002 s of simulation time due to limited CPU time budget. We achieved a parallel efficiency of 90.1% for 8192 cores.

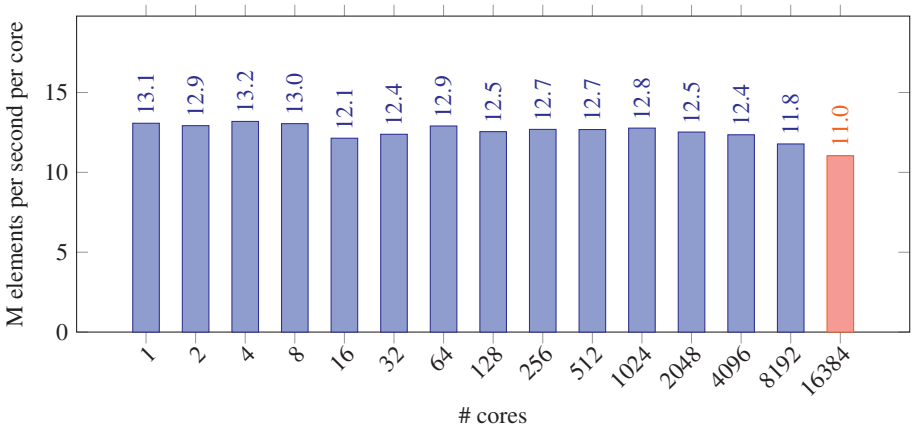

Fig. 7: Weak scaling of the standing wave scenario on 1 up to 16384 cores. While the blue bars ran for 0.05 s of simulation time, the red bar ran for 0.0002 s.

Strong Scaling

The strong scaling results are shown in Fig. 8. We performed measurements on 16 up to 512 cores, each for 100 time steps and with a grid resolution of 2M, 4M and 8M cells. On 512 cores with a grid resolution of 8M cells, we reached a parallel efficiency of 96.8%. On 128 cores with the same grid resolution, we even achieved super-linear speedup (101.0%). However, due to increased overhead, the lower the grid resolution is, the worse is the parallel efficiency. This effect appears for a grid resolution of 2M cells: on 512 cores, a parallel efficiency of only 46.8% is reached, since the number of cells per core (here 2048) is too low.

5 Conclusion & Outlook

Starting with a finite-volume conservation-law formulation of the non-hydrostatic shallow water equations, we have shown how the equation for the non-hydrostatic correction parameter can be discretized on triangular meshes. This allows for adaptivity and a matrix-free approach towards solving the corresponding system. This was attained by using a dual grid which resolved the drawbacks of a collocated discretization. Our results obtained with common test cases demonstrate the success of our numerical model to more accurately resolve the dispersion relation of a standing and a solitary wave compared to the previously existing hydrostatic solver. We furthermore have shown feasibility of conjugate gradient for our test cases. Our parallelization strategy yielded solid strong and weak scaling results which allows to solve large problems on HPC systems.

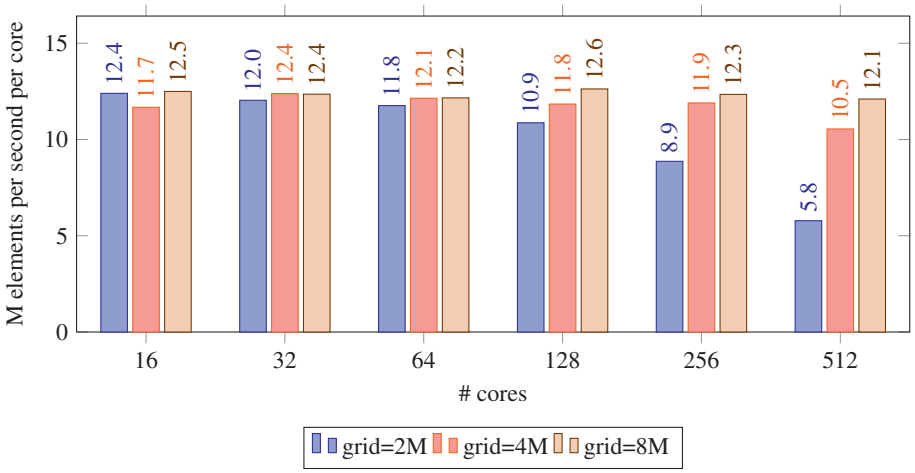

Fig. 8: Strong scaling of the standing wave scenario for 16 up to 512 cores for a grid resolution of 2M, 4M and 8M cells.

Future work will have to study the non-hydrostatic model's behavior on more complex and realistic test cases. Although the terms that had to be neglected in the continuity equation did not seem to adversely affect the results we obtained in the validation test cases, including them into the model certainly leaves room for improvement. Moreover, further research is required for applying faster solvers.

Acknowledgments

We would like to particularly thank Michael Bader, Oliver Meister and Kaveh Rahnema for giving very useful hints and directions on this work. Special thanks also go to Kyle Mandli (Columbia University) who provided an internal report that has been very helpful for deriving the non-hydrostatic shallow water equations in conservation-law form.

References

[Ch68] Chorin, Alexandre J.: Numerical Solution of the Navier-Stokes Equations. Mathematics of Computation, 22(104):745–762, 1968.

[CS98] Casulli, Vincenzo; Stelling, Guus S.: Numerical Simulation of 3D Quasi-Hydrostatic, Free-Surface Flows. Journal of Hydraulic Engineering, 124(7):678–686, 1998.

[Cu13] Cui, Haiyang: A New Numerical Model for Simulating the Propagation of and Inundation by Tsunami Waves. Dissertation, Delft University of Technology, 2013.

[Fu13] Fuchs, Annika: Effiziente parallele Verfahren zur Lösung verteilter, dünnbesetzer Gleichungssysteme eines nichthydrostatischen Tsunamimodells. PhD thesis, Staats-und Universitätsbibliothek Bremen, 2013.

[GDN95] Griebel, Michael; Dornseifer, Thomas; Neunhoeffer, Tilman: Numerical Simulation in Fluid Dynamics. Vieweg, 1995.

[GV14] Ghysels, Peter; Vanroose, Wim: Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm. Parallel Computing, 40(7):224 – 238, 2014.

[HS52] Hestenes, Magnus R.; Stiefel, Eduard: Methods of Conjugate Gradients for Solving Linear Systems. Journal of Research of the National Bureau of Standards, 49(6):409–436, 1952.

[Le02] LeVeque, Randall J.: Finite Volume Methods for Hyperbolic Problems. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002.

[LN08] Levin, Boris; Nosov, Mikhail: Physics of Tsunamis. Springer Science & Business Media, 2008.

[Ma] Mandli, Kyle T.: Derivation of Depth Averaged Flow. Private report.

[MB15] Meister, Oliver; Bader, Michael: 2D Adaptivity for 3D Problems: Parallel SPE10 Reservoir Simulation on Dynamically Adaptive Prism Grids. Journal of Computational Science, 9:101–106, May 2015.

[MRB11] Meister, Oliver; Rahnema, Kaveh; Bader, Michael: A Software Concept for Cache-Efficient Simulation on Dynamically Adaptive Structured Triangular Grids. In: PARCO. pp. 251–260, 2011.

[Sa14] Samfass, Philipp J.: Extension of the Finite Volume Solver SWE towards the Non-Hydrostatic Shallow Water Equations. Bachelor's thesis, Institut für Informatik, Technische Universität München, September 2014.

[SZ03] Stelling, Guus; Zijlema, Marcel: An accurate and efficient finite-difference algorithm for non-hydrostatic free-surface flow with application to wave propagation. International Journal for Numerical Methods in Fluids, 43(1):1–23, 2003.

[Wa05] Walters, Roy A.: A semi-implicit finite element model for non-hydrostatic (dispersive) surface waves. International Journal for Numerical Methods in Fluids, 49(7):721–737, 2005.

[WJ00] Weilbeer, H; Jankowski, JA: A Three-Dimensional Non-Hydrostatic Model for Free Surface Flows – Development, Verification and Limitations. In: Proceedings of the 6th International Conference on Estuarine and Coastal Modeling. pp. 162–177, 2000.