The VOLNA-OP2 Tsunami Code (Version 1.0)
Response to referees

...

September 13, 2018

We would like to thank the referees for their valuable comments and appraisal. We are revising the manuscript in line with their comments. We have made significant additions to both the simulation software, as well as this paper, therefore we will be updating the software version number to 1.5, and we would also like to add two co-authors who have been instrumental in developing this new version.

1   Referee 1

Main points:

Point 1 – This is presumably the first model implementation of its kind, and I find that the implementation should be of interest for the tsunami community, as well as other scientific communities with interest of solving shallow water wave equations or related problems without in depth knowledge of different types of hardware architecture. This part is highly regarded.
Reply – We give a better overview of the model in the paper, however, the numerical implementation has been described in detail in previous work (which we now highlight better).
Changes in manuscript – Section 4.2, Page 5 line 28 - Page 6 line 26.

Point 2 – The study of the speedup on different types of hardware are also new, and the findings are interesting in their own right. However, if possible, I encourage the authors to see if it is possible to compare the model speedup also with other models (such as HySEA) for inter model comparison.
Reply – Direct comparison is very difficult, because the open source models and implementation available all differ from ours in various, but significant ways - e.g. Tsunami-HySEA only supports structured meshes (vs. our unstructured meshes) and a single GPU - support for nested meshes and multi-GPU are not available in the public version. Instead, we pull results from their published papers to discuss the performance and the speedups they achieve.
Changes in manuscript – Section 2, Page 3 line 28 - 33.

Point 3 – The validation of the model is entirely missing. I know that the previous VOLNA codes have been benchmarked towards NTHMP tests previously, but this is a new implementation. While one may expect a similar accuracy for this code as well, validation needs to be demonstrated. To emphasise this, the novelty of this paper actually hinges on some kind of proof; i.e. that the model can produce results consistent with previous versions. Moreover, no explicit tsunami results are shown, only results showing the speedup. As a minimum, some results showing that the tsunami code gives a reasonable output needs to be included. I would propose that the authors include one or two of the standard tsunami inundation benchmark tests. I'm sure the authors have some such tests available.
Reply We are adding results of two NTHMP benchmarks to demonstrate the numerical accuracy of the code.
Changes in manuscript – Section 4.3, Page 6 line 28 - Page 9 line 9.

Point 4 – The text and reference list is a bit imbalanced with respect to the authors own work. It would be beneficial if some more external references are added, reference to external work is

moved upfront, or alternatively, discussion of the authors own work that are not strictly relevant for this paper are omitted (some parts seems not strictly necessary, see below). In the line-by-line review below some examples are listed. The references in the related work section should be moved upfront (section 2 seems unnecessary).

Reply – We have included a considerable number of additional references to related work, but for better structure and readability we kept the Related Works section of the paper - as many publications in GMD do..

Changes in manuscript – Section 1, Page 1 line 14 - Page 2 line 2, Section 2, Page 3 line 23 - Page 3 line 33.

Line-by-line comments for edits to the text and references:

Comment on Page 1, line 14 – More references to external work should preferably be placed up front (e.g. here). It makes sense to pay attention to the general literature first, and use this to put the authors own work into a general context thereafter.

Reply – We are adding a number of citations, particularly to work related to tsunami simulation, to this first paragraph.

Changes in manuscript – Section 1, Page 1 line 14 - Page 2 line 2

Comment on Page 1, line 15 – The statement "there are only a handful of codes that are suitable for integration into a workflow" is unsubstantiated, please remove.

Reply – We have removed this sentence

Changes in manuscript – Section 1, Page 1 line 14

Comment on Page 2, line 1 – The science perspective is missing here, but is obvious, for instance the need for running sensitivity analysis (such as varibable slip or uncertainty assessments, e.g. Goda et al., 2014), probabilistic tsunami hazard assessments (e.g. Geist and Parsons, 2006; Davies et al., 2018; Grezio et al., 2017), or for more efficient and informed tsunami early warning (e.g. Oishi et al., 2015, Castro et al., 2015). I think it would strengthen the paper to mention and discuss such examples.

Reply – We have placed additional references in the introduction to emphasize the science perspective as the reviewer suggested.

Changes in manuscript – Section 1, Page 1 line 14 - Page 2 line 2

Comment on Page 2, lines 19-28 – Reading this paragraph, you get the impression that the Volna code is unique with respect to workflow integration, which is not the case (see comment to Page 1, line 15). There are probably more than 10 codes worldwide that can do much of the same analysis. Granted, the new development presented in this paper provide new opportunities wrt hardware independence. This should be the main message.

Reply – We clarify that it is Volna-OP2's performance and portability that prompted its use in the cited papers and its integration into workflows.

Changes in manuscript – Section 1, Page 2 line 24

Comment on Page 3 – References and discussions in Related Work section should preferably be moved upfront.

Reply – We have included a considerable number of additional references to related work, but for better structure and readability we kept the Related Works section of the paper - as many publications in GMD do.

Changes in manuscript – Section 2, Page 3 line 23 - Page 3 line 33.

Comment on Page 3, line 10 – Unsubstantiated statement: "Since there is no consensus as to their advantage. . .". What do the authors mean here? Clarify, or remove statement. Simply, Boussinesq models are needed wherever tsunami dispersion is needed (see e.g. Glimsdal et al., 2013), otherwise the shallow water approximation is sufficient.

Reply – We have changed the text to say these models are primarily needed for dispersion, and added the citation.

Changes in manuscript – Section 2, Page 3 line 19

Comment on Page 3, line 17 – The authors should provide a literature search here and add more references, as GPU implementation of shallow water is under rapid development. As a minimum, the authors needs to add reference to the GPU SWE code by Brodtkorp et al. (2010) and the Boussinesq GPU code Celeris (Tavakkol and Lynett, 2017).
Reply – We have added text to reference the work by Brodtkorp et al., Tabakkol and Lynett, Acuna and Aoiki, Liang et. al.
Changes in manuscript – Section 2, Page 3 line 28 - 33

Comment on Page 4 – I could not find anywhere from the discussion whether related OP2 applications have been performed for other (similar) applications of hyperbolic equation. If such implementations exists, it would be of interest to discuss their performance.
Reply – Other applications implemented in OP2 are either elliptic or not PDE solvers. We place a comment referencing three key papers describing performance of other OP2 applications.
Changes in manuscript – Section 3, Page 4 line 15, Section 3.1 Page 5 line 13

Comment on Page 5, line 9 – Remove "very".
Reply – We have removed the word
Changes in manuscript – Section 4.1, Page 5 line 22

Comment on Page 5, lines 7-12 – Treatments of shocks and breaking waves are probably the main reason FV are used, so this needs explicit mentioning.
Reply – This is now mentioned in the paragraph
Changes in manuscript – Section 4.1, Page 5 line 21

Comment on Page 5, line 14 – Replace "megatsunami" with either "large tsunami" or "transoceanic tsunami".
Reply – The paragraph was reworded
Changes in manuscript – Section 4.1, Page 5 line 28

Comment on Page 5, line 15 – Again, we refer to Glimsdal et al. (2013). Because frequency dispersion is a time dependent property, important of dispersion increases with time for a given initial condition, so it is not sufficient to refer to dispersion as weak just based on the properties at a given snapshot. The discussion here seems to merge the effect of dispersion on deep water waves and inundation, which are very different. Either the authors needs to clarify better, however, it would probably be better to omit this discussion here, and rather state that the present implementation is based on the non-linear shallow water model (you do not have to justify that dispersion is not included).
Reply – As suggested, we omit this discussion
Changes in manuscript – Section 4.1, Page 5 line 28

Comment on Page 6, first paragraph – This repeated reference to applications of the code seems awkward as it is not needed in this context, beside, this is already discussed in the introduction.
Reply – We reorganise this paragraph, removing most references.
Changes in manuscript – Section 4.2, Page 6 line 10

Comment on Page 9, lines 14-15 – See previous comment.
Reply – We have removed these references
Changes in manuscript – Section 4.4, Page 11 line 16

Comment on Page 14, lines 29-31 – I cant see that this is more relevant than other and more general applications such as PTHA and tsunami early warning. As said, a more general discussion with references from a broader literature is needed.
Reply – We have removed this paragraph as it was indeed too specific and it is not a discussion/conclusion really
Changes in manuscript – Section 6, Page 18 line 6

2 Referee 2

Comment 1 – Use of uniform triangular meshes as opposed to non-uniform ones. Am I right if I say that the most typical use case of VOLNA-OP2 is with non-uniform meshes? If so, why using uniform ones? The mesh can have a drastic impact on the achieved performance, due to different MPI partitioning, load balancing, effectiveness of mesh renumbering etc. Is this a weakness of the analysis? if not, why?
Reply – we are re-evaluating scaling performance using non-uniform meshes of the same area
Changes in manuscript – Section 5.2, Page 12 line 30 - Section 5.5 Page 16 line 33

Comment 2 – I know that with other OP2-based applications, traditionally, MPI+OpenMP has never really outperformed pure MPI (due to issues that have been extensively described in prior work), or at least not by a significant factor. Is the situation different with VOLNA? If so, can you say why? Speculation: is it because VOLNA is "more compute-bound" than other codes used in the past? I'm asking because I see that all numbers reported derive from MPI+OpenMP. By the way, I assume that you have taken care of pinning etc – can this be confirmed?
Reply – This is a good point, we added further performance figures to the paper and a brief discussion on the difference. MPI performs better at low node counts, but scales worse the MPI+OpenMP on the KNL. Pinning was done, through the built-in mechanisms in the MPI distribution.
Changes in manuscript – Sections 5.3 and 5.4, Page 13 line 15 - Page 16 line 14, Figure 9.

Comment 3 – Was data alignment enforced for directly accessed datasets ? I understand the same dataset can be accessed directly in one loop and indirectly in another loop, but perhaps there's hope to exploit some alignment in at least some of the memory-bounded loops?
Reply – Data alignment is enforced and the appropriate annotations are supplied in the generated code, although in practice it made no difference to the resulting performance
Changes in manuscript –

Comment 4 – Can you be more precise as to why it was not possible to vectorise 'applyFluxes'? Can you share more details about how the vectorisation of the other loops has been achieved – is it via auto-vectorisation or something else?
Reply – The details of the vectorised code generation are discussed in a previous paper - this is now appropriately cited. It was not working on the loop in question due to a compiler bug - this was reported to Intel, but not yet fixed.
Changes in manuscript – Section 5.3, Page 14, line 1 - line 3

Comment 5 – Which loops are compute-bound and which are memory-bound?
Reply – In this paper we did not really want to go into too much detail on the optimisations, as that may not really be of too much interest to the reader of this journal. We place short notes on which loops are bandwidth/compute/control limited. Detailed performance analysis of these loops individually is available in prior work, which we now cite: I. Z. Reguly, E. László, G. R. Mudalige, and M. B. Giles. 2014. Vectorizing Unstructured Mesh Computations for Many-core Architectures. In Proceedings of Programming Models and Applications on Multicores and Manycores (PMAM'14). ACM, New York, NY, USA, , Pages 39 , 12 pages. DOI=http://dx.doi.org/10.1145/2560683.2560686
Changes in manuscript – Section 5.1, Page 12, line 28

Comment 6 – Can you report the max memory bandwidth of *all* platforms? I think some are currently missing. Also, how was the memory bandwidth limit determined ? specs, STREAM, or...?
Reply – We report on the max bandwidth achieved on different systems, as measured by STREAM Triad.
Changes in manuscript – Section 5.3, Page 16, line 8, Section 5.4, Page 16, line 2

Comment 7 – Can you state the model of XeonPhi used? Some have 68 cores, not 64 like the one used in the experimentation.
Reply – We added a note to the text on using the 64 core 7210 model.
Changes in manuscript – Section 5.3, Page 15, line 8

Comment 8 – I think comparing the VOLNA-OP2 performance in different architectures is a bit unfair, at least for two reasons: 1- The number of "degrees-of-freedom" (or simply triangles) per core is generally different, and so is the proportion of time spent is computation and communication. Even data locality may have been impacted. 2- These architectures are profoundly different among them – in theoretical attainable peak performance, price, release date, etc. I don't like sentences such as "The GPU system with X1 nodes was Y times faster than the CPU system with X2 nodes"; I don't think they add much to the paper, and in fact they may be misleading. Perhaps you can normalise over a common metric, but again that's quite tough. Also, the Phi suffers a lot from the lack of the vectorisation in 'applyFluxes', and it's unclear whether this issue can be worked around or not.

Reply – While for computer science benchmarking purposes we entirely agree that such a comparison is not fair, this paper is aimed at people who use and run tsunami modeling software (most of the authors are not computer scientists). We believe that for this audience such speedup numbers are relevant, as they will have to choose a platform to run on, and their basis for decision is only the cost of access and the relative performance between them. We dedicate a section to this (Running Costs and Power Consumption), and now point out that this kind of comparison is indeed unfair.

Changes in manuscript – Section 5.6, Page 17, line 2 - line 13

Comment 9 – In the conclusions, I disagree with the sentence: "Through performance scaling and analysis of the code on (...), we have demonstrated that VOLNA-OP2 indeed delivers near-optimal performance on a variety of platforms (...)" . My point is that I don't think that you have demonstrated that you're achieving near-optimal performance – it is true that some loops are relatively close to the architecture memory bandwidth limit, but others are not. We don't know the reason – in fact, we don't even know whether these loops are compute- or memory-bound (see point above). So I suggest to either drop that sentence, or rephrase it, or add a roofline plot.

Reply – We have dropped this statement, especially because for unstructured mesh computations this is difficult to quantify. Also because fair comparison to other codes was not possible.

Changes in manuscript – Section 6, Page 18, line 1 - line 2

Comment 10 – Of course, I assume that comparing the performance of VOLNA-OP2 to that of other codes is way too difficult, if not impossible

Reply – Unfortunately we found no open-source codes that would allow direct comparison. We have extracted some performance figures from related papers

Changes in manuscript – Section 2, Page 3 line 28 - 33.

Comment 11 – I see that version 1.0 has been "Zenodoized" – don't forget to add the DOI to the paper. Also, I suggest to move the code to its own organisation on GitHub so that permissions can be more easily set and contributing to VOLNA-OP2 gets easier. I also suggest to link VOLNA-OP2 from the 'apps' section of OP2.

Reply – We have added DOIs to the paper

Changes in manuscript – Section 6, Page 18 line 17.

Comment 12 – It might be just me, but it feels like that different sections of the paper have been written by different people. That is, the transition from some paragraphs to others is not as smooth as it might be. Also, there are some typos (e.g., outline -> outlined) and/or missing words ("the physical across") in various sections. All this should be improved prior to publication.

Reply – We are making changes to the text to make the transitions smoother, and clearing up typos.

Changes in manuscript – Throughout Sections 1-6

Comment 13 – Section 3.1 might benefit from some figures. I know this is not the main point of the article, but I'm not sure that readers who are unfamiliar with OP2 will be able to understand how, for example, GPU parallelisation works. Maybe just cite some prior paper?

Reply – In the interest of brevity, we added citations to previous work detailing the parallelisation approaches.

Changes in manuscript – Section 3.1, Page 5 line 13

Comment 14 – In the performance section, Table 1 gives a name to three meshes: M1, M2, M3.

These could be used systematically throughout the whole section, and in the plots as well, instead of referring to "the 1.4M mesh".

Reply – We have made the suggested changes - with the new non-uniform meshes, there are labelled as NU_3...NU_0

Changes in manuscript – Table 1, Page 13, Figures 9-10, Page 15 and 17, Sections 5.2-5.5.

The VOLNA-OP2 Tsunami Code (Version ~~1.0~~1.5)

Istvan Z Reguly[1], Daniel Giles[5], Devaraj Gopinathan[2], Laure Quivy[6], Joakim H Beck[3], Michael B Giles[4], Serge Guillas[2], and Frederic Dias[5]

[1]Pázmány Péter Catholic University, Faculty of Information Technology and Bionics, Prater u 50/a, 1088 Budapest, Hungary
[2]Department of Statistical Science, University College London, London, UK
[3]Computer, Electrical and Mathematical Science and Engineering Division (CEMSE), King Abdullah University of Science and Technology (KAUST), Thuwal, 23955-6900, Saudi Arabia
[4]Math Institute, University of Oxford, Oxford, UK
[5]School of Mathematics and Statistics, University College Dublin, Dublin, Ireland
[6]Centre de Mathématiques et de Leurs Applications (CMLA), Ecole Normale Supérieure, Paris-Saclay, Centre National de la Recherche Scientifique, Université Paris-Saclay, 94235 Cachan, France

Correspondence: Istvan Z Reguly (reguly.istvan@itk.ppke.hu)

Abstract. In this paper, we present the VOLNA-OP2 tsunami model and implementation; a finite volume non-linear shallow water equations (NSWE) solver built on the OP2 domain specific language (DSL) for unstructured mesh computations. VOLNA-OP2 is unique among tsunami solvers in its support for several high performance computing platforms: CPUs, the Intel Xeon Phi, and GPUs. This is achieved in a way that the scientific code is kept separate from various parallel implementations, enabling easy maintainability. It has already been used in production for several years, here we discuss how it can be integrated into various workflows, such as a statistical emulator. The scalability of the code is demonstrated on three supercomputers, built with classical Xeon CPUs, the Intel Xeon Phi, and NVIDIA P100 GPUs. VOLNA-OP2 shows an ability to deliver productivity to its users, as well as performance and portability ~~on~~ across a number of platforms.

Copyright statement. TEXT

1 Introduction

After the Indian Ocean tsunami of 26 December 2004, Bernard et al. (2006) emphasized that one of the greatest contributions of science to society is to serve it purposefully, as when providing forecasts to allow communities to respond before a disaster strikes. In the last twelve years, the numerical modelling of ~~tsunami~~ tsunamis has experienced great progress (Behrens and Dias (2015)). There is a variety of mathematical models, such as the shallow-water equations (Titov and Gonzalez (1997); Liu et al. (1998); Gail), the Boussinesq equations (Kennedy et al. (2000); Lynett et al. (2002)), or the 3D ~~Navier- Stokes equations , and there exist a huge~~ Navier-Stokes equations (Abadie et al. (2012); Gisler et al. (2006)), and a large number of implementations, primarily for individual target computer architectures. The use cases of such models are wide ranging, and most rely on high numerical accuracy as well as high computational performance to deliver results - ~~yet there are only a handful codes that are suitable for integration into a workflow, aimed at forecasting or statistical exploration~~examples include sensitivity analysis by Goda et al. (2014)

1

, probabilistic tsunami hazard assessments by Geist and Parsons (2006); Davies et al. (2017); Anita et al., and more efficient and informed tsunami early warning by Yusuke et al.; Castro et al. (2015).

For widespread use three key ingredients are needed; first, the stability and robustness of the numerical approach, that gives a confidence in the results produced, second, the computational performance of the code, which allows for getting the right results quickly, efficiently utilising the available computational resources, and third, the ability to integrate into a workflow, allowing for simple pre- and post-processing, efficiently supporting the kinds of use cases that come up - for example large numbers of different initial conditions.

In the Related Work section we discuss a number of codes currently being used in production, and as such are trusted and reliable codes, already being used as part of a workflow. Yet, the computational performance of most of these codes is "good enough"; they were written by domain scientists, and may have been tuned to one architecture or an other, but for example, GPU support is almost non-existent. In today's and tomorrow's quickly changing hardware landscape however, "future-proofing" numerical codes is of exceptional importance for continued scientific delivery. Domain scientists can not be expected to keep up with architectural advances, and spend a significant amount of time re-factoring code to new hardware. What to compute must be separated from how it is computed - indeed in a recent paper by Lawrence et al. (2017), leaders in the weather community chart the ways forward, and point to Domain Specific Languages (DSLs) as a potential way to address this issue.

OP2, by Mudalige et al. (2012), is such a DSL, embedded in C/C++ and Fortran; it has been in development since 2009: it provides an abstraction for expressing unstructured mesh computations at a high-level, and then provides automated tools to translate scientific code written once, into a range of high-performance implementations targeting multi-core CPUs, GPUs, and large heterogeneous supercomputers. The original VOLNA model (Dutykh et al. (2011)) was already discussed and validated in detail - it was used in production for small-scale experiments and modelling, but was inadequate for targeting large-scale scenarios and statistical analysis, therefore it was re-implemented on top of OP2; this paper describes the process, challenges and results from that work.

As VOLNA-OP2 delivered a qualitative leap in terms of possible uses due to the high performance it can deliver on a variety of hardware architectures, its users have started integrating it into a wide variety of workflows; one of the key uses is for uncertainty quantification; for the stochastic inversion problem of the 2004 Sumatra tsunami in Gopinathan et al. (2017), for developing Gaussian process emulators which help reduce the number of simulation runs (Beck and Guillas (2016); Liu and Guillas (2017)), applications of stochastic emulators to a submarine slide at the Rockall Bank (Salmanidou et al. (2017)), a study of run-up behind islands (Stefanakis et al. (2014)), the durability of oscillating wave surge converters when hit by tsunamis (O'Brien et al. (2015)), tsunamis in the St. Lawrence estuary (Poncet et al. (2010)), a study of the generation and inundation phases of tsunamis (Dias et al. (2014)), and others.

The time-dependency in the deformation enables the tsunami to be actively generated (Dutykh and Dias (2009)). This is a step-forward from the common passive mode of tsunamigenesis that utilises an instantaneous rupture. The active mode is particularly important for tsunamigenic earthquakes with long and slow ruptures, *e.g.* the 2004 Sumatra-Andaman event

(Lay et al. (2005); Gopinathan et al. (2017)) and submerged landslides (Løvholt et al. (2015)), *e.g.* the Rockall Bank event (Salmanidou et al. (2017)).

These applications present a number of challenges in integration into the workflow, as well as scalable performance: the need for extracting snapshots of state variables on the full mesh, or at a number of specified locations, capturing the maximum wave elevation or inundation - all in the context of distributed memory execution.

As the above references indicate, VOLNA-OP2 has already been key in delivering scientific results in a range of scenarios, and through the collaboration of the authors, it is now capable of efficiently supporting a number of use cases, making it a versatile tool to the community, therefore we have now publicly released it: it is freely available at ~~github. com/reguly/volna.~~ github.com/reguly/volna.

The rest of the paper is organised as follows: Section ~~2~~ 2 discusses related work, Section ~~3~~ 3 presents the OP2 library, upon which VOLNA-OP2 is built, Section ~~4~~ 4 discusses the VOLNA simulator itself, its structure and features, Section ~~5~~ 5 discusses performance and scalability results on CPUs and GPUs, and finally Section ~~6~~ 6 draws conclusions.

2 Related Work

Tsunamis have long been a key target for scientific simulations. Behrens and Dias (2015) give a detailed look at various mathematical, numerical, and implementational approaches to past and current tsunami simulations. The most common set of equations solved are the shallow water equations, and most codes use structured and nested meshes. A popular discretisation is finite differences, such codes include: NOAA's MOST (Titov and Gonzalez (1997)), COMCOT (Liu et al. (1998)), CENALT (Gailler et al. (2013)). On more flexible meshes many use the finite element discretisation, such as SELFE (Zhang and Baptista (2008)) and TsunAWI (Harig et al. (2008)), ASCETE (Vater and Behrens (2014)), Firedrake-Fluids (Jacobs and Piggott (2015)) or the finite volume discretisation, such as the VOLNA code (Dutykh et al. (2011)), GeoClaw (George and LeVeque (2006)) or HySEA (Macías et al. (2017)). Another model is described by the Boussinesq equations - these equations and the solver are more complex than shallow-water solvers. Since ~~there is no consensus as to their advantage over it, they are~~ they are primarily needed only for dispersion (Glimsdal et al. (2013)); they are used less commonly, examples include FUNWAVE Kennedy et al. (2000) and COULWAVE (Lynett et al. (2002)). Finally, the 3D Navier-Storkes equations provide the most complete description, but they are significantly more complex than other models - examples include SAGE (Gisler et al. (2006)) and the work of Abadie et al. (2012).

Most of these codes described above work on CPUs, and while there has been some work on GPU implementations by ~~Satria et al. (2012); Liang et al. (2009a)~~ Satria et al. (2012); Liang et al. (2009a); Brodtkorb et al. (2010); Acuña and Aoki (2009); Liang et al., which use structured meshes and finite differences or finite volumes, it is unclear whether these are used in production~~.~~, and they are not open source. Celeris Tavakkol and Lynett (2017) is a Boussinesq solver that uses finite volumes and a structured mesh - it is hand-coded for GPUs using graphics shaders, and its source code is available, however it can only use a single GPU.

3
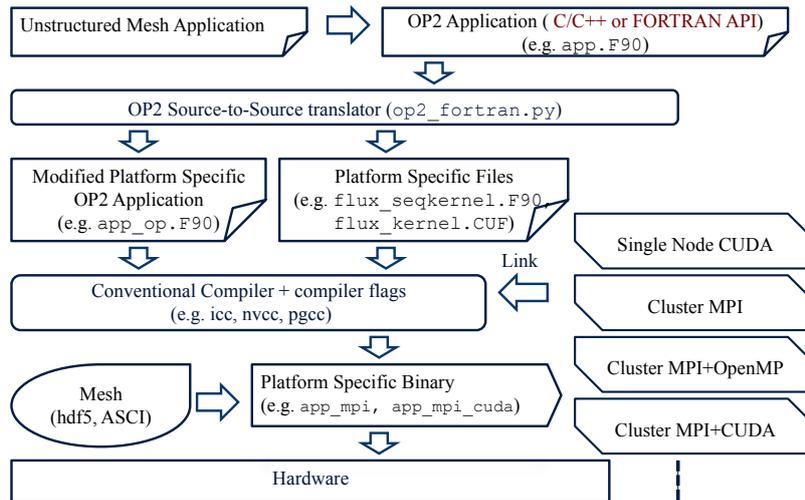
Figure 1. Build system with OP2

As far as we are aware, only Tsunami-HySEA (Macías et al. (2017)), also using finite volumes, is using GPU clusters in production - that code however only supports GPUs, and is hand-written in CUDA. Performance reported by Castro et al. (2015) on a 10M point testcase shows a strong scaling efficiency going from 1 GPU to 12 GPUs between 88% and 73% (overall 12 GPUs are 5.88 faster than 1 GPU), and a $25\times$ speedup with 1 GPU over an unspecified (likely single core) CPU implementation. Direct comparison to VOLNA-OP2 is not possible since Tsunami-HySEA uses (nested) structured meshes, and the multi-GPU version is not open source.

3 The OP2 Domain Specific Language

The OP2 library (Mudalige et al. (2012)) is a domain specific language embedded in C and Fortran that allows unstructured mesh algorithms to be expressed at a high level, and provides automatic parallelisation and a number of other features. It provides an abstraction that lets the domain scientist describe a mesh using a number of sets (such as quadrilaterals or vertices), connections between these sets (such as edges-to-nodes), and data defined on sets (such as ~~x, y~~ x, y coordinates on vertices). Once the mesh is defined, an algorithm can be implemented as a sequence of parallel loops, each over all elements of a given set applying different ~~"kernel functions"~~ "kernel functions", accessing data either directly on the iteration set, or indirectly through at most one level ~~25~~ of indirection. This abstraction enables the implementation of a wide range of algorithms, such as the finite volume algorithms that VOLNA uses, but it does require that for any given parallel loop, the order of execution must not affect the end result (within machine precision) - this precludes the implementation of e.g. Gauss-Seidel iterations.

OP2 enables its users to write an application only once using its API, which is then automatically parallelised to utilise multi-core CPUs, GPUs, and large supercomputers through the use of MPI, OpenMP and CUDA. This is done in part through ~~30~~ a code generator that parses the parallel loop expressions and generates ~~boilerplate~~ code around the computational kernel to facilitate parallelism and data movement, and in part through different back-end libraries that manage data, including MPI

4

halo exchanges, or GPU memory management, as shown in Figure ~~1. Fore~~ 1. For more details see Giles et al. (2011); Mudalige et al. (2012).

3.1 Parallelisation Approaches in OP2

OP2 takes full responsibility for orchestrating parallelism and data movement - from the user perspective, the code written looks and feels like sequential C code that makes calls to an external library. To utilise clusters and supercomputers, OP2 uses the Message Passing Interface (MPI) to parallelise in a distributed memory environment; once the mesh is defined by the user, OP2 automatically partitions and distributes it among the available resources. It uses the standard owner-compute approach with halo exchanges, and overlaps computations with communications. In conjunction with MPI, OP2 uses a number of shared-memory parallelisation approaches, such as CUDA and OpenMP.

A key challenge in the fine-grained parallelisation of unstructured mesh algorithms is the avoidance of race conditions when data is indirectly modified. For example, in a parallel loop over edges, when indirectly incrementing data on vertices, ~~10~~ multiple edges may try to increment the same vertex, leading to race conditions. OP2 uses a colouring approach to resolve this; elements of the iteration set are grouped into mini-partitions, and each element within these mini-partitions is coloured, so no two elements of the same colour access the same value indirectly. Subsequently mini-partitions are coloured as well. For CUDA, we assign mini-partitions of the same colour to different CUDA thread blocks, and for OpenMP to different threads. There is then a global synchronisation between different mini-partition colours. In case of CUDA, threads processing elements within each thread block use the first level of colouring to apply increments in a safe way, with block-level synchronisation in-between. Code generation that is suitable for auto-vectorisation by the compilers is also supported; it carries out the packing and unpacking of vector registers. Previous work describes further details and performance comparisons on various architectures, these are available in Mudalige et al. (2012); Reguly et al. (2007).

3.2 Input and Output

OP2 supports parallel file I/O through the HDF5 library (The HDF Group (2000-2010)), which is critically important to its integration into VOLNA's workflow: reading in the input problem and writing out data required for analysis simultaneously on multiple processes.

4 The VOLNA simulator

4.1 Model, numerics, previous validation

The finite volume (FV) framework is the most natural numerical method to solve the non-linear shallow water equations (NSWE), in part because of their ability to treat shocks and breaking waves. It belongs to a class of discretisation schemes that are highly efficient in the numerical solution of systems of conservation laws, which are ~~very~~ common in compressible and incompressible fluid dynamics. Finite Volume methods are preferred over finite differences and often over finite elements

because they intrinsically address conservation issues, improving their robustness: total energy, momentum and mass quantities are conserved exactly, assuming no source terms, and appropriate boundary conditions. The code was validated against the classical benchmarks in the tsunami community as described below.

4.2 Numerical model

~~In a typical megatsunami (say, with characteristic wavelength ~100km, average ocean depth ~4km and tsunami wave amplitude of ~0.5 m), the non-linearity and dispersion are found to be weak (Dutykh et al. (2011)). Although frequency dispersion as the tsunami approaches the shore (or shallow depths) may precipitate local effects, they are overwhelmed by the inundation/run-up caused by the longer waves. Accordingly, the~~ Following the needs of the target applications, the following non-dispersive NSWEs (in Cartesian coordinates) form the physical model of VOLNA:

$$H_t + \nabla \cdot (H\boldsymbol{v}) = 0 \tag{1}$$

$$(H\boldsymbol{v})_t + \nabla \cdot \left(H\boldsymbol{v} \otimes \boldsymbol{v} + \frac{g}{2}H^2 \boldsymbol{I}_2 \right) = \underline{-gH}\underset{\sim}{gH}\nabla \underline{b}\underline{d} \tag{2}$$

Here, $\underline{b(\boldsymbol{x},t)}$ $\underline{d(\boldsymbol{x},t)}$ is the time-dependent bathymetry, $\boldsymbol{v}(\boldsymbol{x},t)$ is the horizontal component of the depth-averaged velocity, g is the acceleration due to gravity and $H(\boldsymbol{x},t)$ is the total water depth. Further, $\boldsymbol{I}_2$ is the identity matrix of order 2. The tsunami wave height or elevation of free surface $\eta(\boldsymbol{x},t)$, is computed as,

$$\eta(\boldsymbol{x},t) = H(\boldsymbol{x},t) - \underline{b}\underline{d}(\boldsymbol{x},t) \tag{3}$$

where the sum of static bathymetry $\underline{b_s(\boldsymbol{x})}$ $\underline{d_s(\boldsymbol{x})}$ and the dynamic seabed uplift $u_z(\boldsymbol{x},t)$ constitute the dynamic bathymetry,

$$\underline{b}\underline{d}(\boldsymbol{x},t) = \underline{b}\underline{d}_s(\boldsymbol{x}) + u_z(\boldsymbol{x},t) \tag{4}$$

$\underline{b_s}$ $\underline{d_s}$ is usually sourced from bathymetry datasets pertaining to the region of interest (say, global datasets like ETOPO1/GEBCO or regional bathymetries). The vertical component $u_z(\boldsymbol{x},t)$ of the seabed deformation is calculated depending on the physics of tsunami generation, *e.g.* via co-seismic displacement for finite fault segmentations by ~~Gopinathan et al. (2017)~~ Gopinathan et al. (2017), submarine sliding by ~~Salmanidou et al. (2017, 2018) etc. The time-dependency in uz (x,t) enables the tsunami to be actively generated (Dutykh and Dias (2009)). This is a step-forward from the common passive mode of tsunamigenesis that utilises an instanta- neous rupture. The active mode is particularly important for tsunamigenic earthquakes with long and slow ruptures, e.g. the 2004 Sumatra-Andaman event (Lay et al. (2005); Gopinathan et al. (2017)) and submerged landslides (Løvholt et al. (2015)), e.g. the Rockall Bank event (Salmanidou et al. (2017)).~~ Salmanidou et al. (2017, 2018) *etc.*.

In addition to the capabilities of employing active generation and consequent tsunami propagation, VOLNA also models the run-up/run-down (~~i.e.~~ *i.e.* the final inundation stage of the tsunami). These three functionalities qualify VOLNA to simulate the entire tsunami life-cycle. The ability of the NSWEs (~~1-2~~ 1-2) to model both propagation, as well as run-up and run-down processes was validated in Kervella et al. (2007) and Dutykh et al. (2011), respectively. Thus, the use of uniform model for

the entire life-cycle obviates many technical issues such as the coupling between the sea bed deformation and the sea surface deformation and the use of nested grids.

VOLNA uses the cell-centered approach for control volume tesselation, meaning that degrees of freedom are associated with cell barycenters. ~~A~~ However, in order to improve the spatial accuracy a second order extension is employed. A local gradient of the physical variables over each cell is calculated, then a limited linear projection of the variables at the cell interfaces is used within the numerical flux solver. The limiter used is a restrictive version of the scheme purposed by **?**, the minimum calculated limiter of the physical variables within a cell is used in the reconstruction, this limiter ensures that numerical oscillations are constrained in realistic cases. A Harten-Lax-van Leer (~~HLL~~HLLC) numerical flux ~~was~~ which incorporates the contact discontinuity is used to ensure that the standard conservation and consistency properties are satisfied: the fluxes from adjacent triangles that share an edge exactly cancel when summed and the numerical flux with identical state arguments reduces to the true flux of the same state. Details of the numerical implementation can be found in Dutykh et al. (2011).

4.3 Validation

The original version of VOLNA was thoroughly validated against the National Tsunami Hazard Mitigation Program (NTHMP) benchmark problems Dutykh et al. (2011). A brief look at how the new implementation, which utilizes the more restrictive limiter performs with regards to two benchmark problems is given below. The reader is referred to the original paper Dutykh et al. (2011) or the NTHMP website for further details on the set-up of the benchmark problems.

4.3.1 Benchmark Problem 1 - Solitary Wave on a Simple Beach

The analytical solution to the run up of a solitary wave on a sloping beach was derived by Synolakis (1987). Thus, in this benchmark problem one compares the simulated results with the derived analytical solution.

Set Up

The beach bathymetry comprises of a constant depth (d) followed by a sloping plane beach of angle $\beta = \mathrm{arccot}(19.85)$. The initial water level is defined as a solitary wave of height η centered at a distance L from the toe of the beach and the initial wave-particle velocity is proportional to the initial water level.

$$H(x,0) = \eta \mathrm{sech}^2(\gamma(x - X_1)/d) \tag{5}$$

$$u(x,0) = -\sqrt{\frac{g}{d}} H(x,0) \tag{6}$$

where $x = X_0 = d\cot(\beta)$, $L = \mathrm{arccosh}(\sqrt{20})/\gamma$, $X_1 = X_0 + L$, and $\gamma = \sqrt{3\eta/4d}$. For this benchmark problem the following ratio must also hold: $\eta t/d = 0.019$.

Tasks
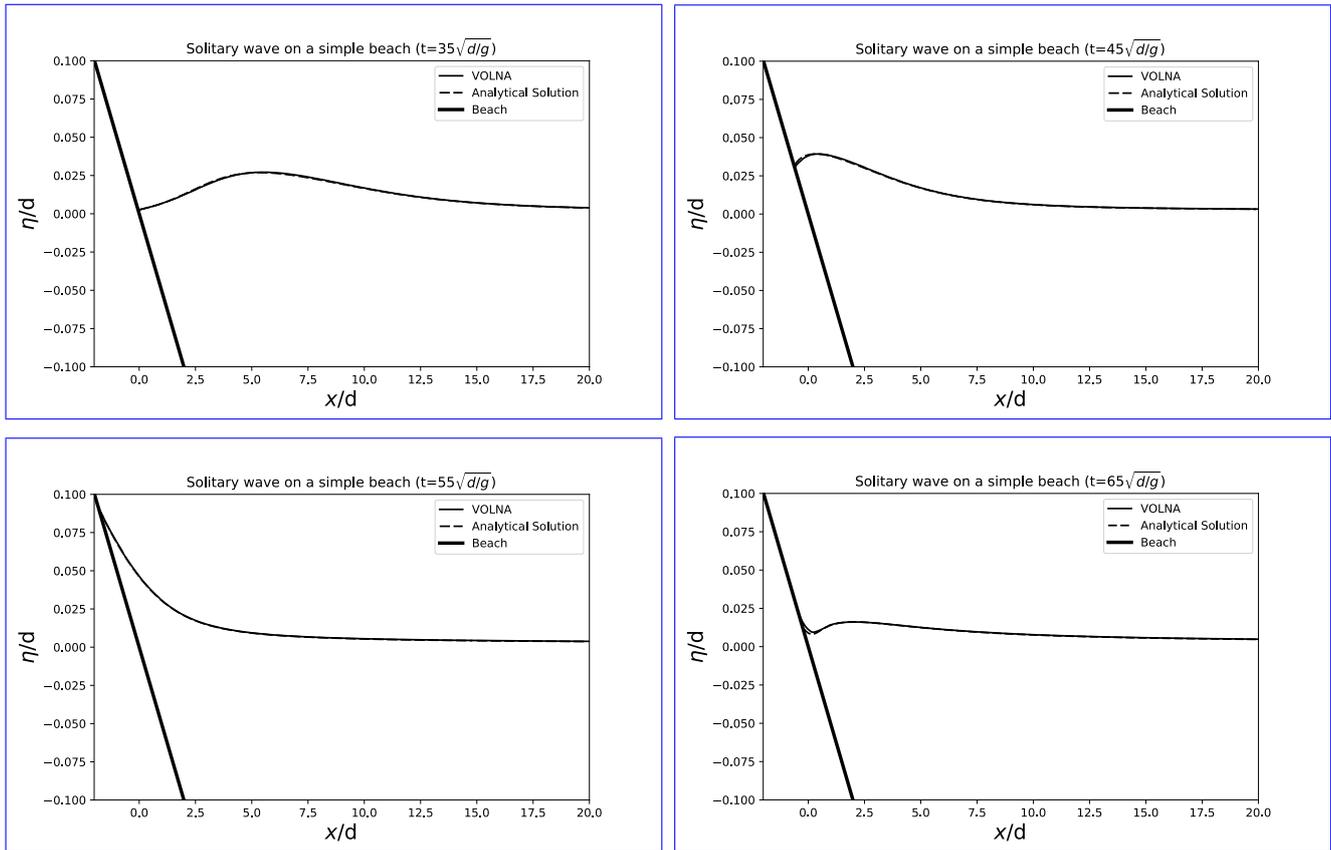
Figure 2. Solitary wave on a simple beach - Comparison between the simulated run-up and analytical solution at the shoreline (Time = 35, 45, 55, 65 $\sqrt{d/g}$). Solid line - VOLNA, Dashed line - Analytical Solution, Thick line - Beach

In order to verify the model, the wave run up at various time steps (Figure 2) and the wave height at two locations ($x/d = 0.25$ and $x/d = 9.95$) (Figure 3) are compared to the analytical solution.

It can be seen from the plots above that the agreement between numerical results and the analytical solutions is very good. So therefore, the new implementation of the model is able to accurately simulate the run-up of the solitary wave.

4.3.2   Benchmark Problem 2 - Wave Run-Up onto a Complex 3D Beach

This benchmark problem involves the comparison of laboratory results for a tsunami run up onto a complex 3D beach with simulated results. The laboratory experiment reproduces the 1993 Hokkaido-Nansei-Oki tsunami which struck the Island of Okushiri, Japan. The experiment is a 1:400 scale model of the bathymetry and topography around a narrow gully and the tsunami is an incident wave fed in as a boundary condition.

Set Up

The computational and laboratory domain corresponds to a 5.49m by 3.40m wave tank and the bathymetry for the domain is given for 0.014m by 0.014m grid cells. The incoming wave is incident on the x=0m boundary and is defined for the first 22.5s

8
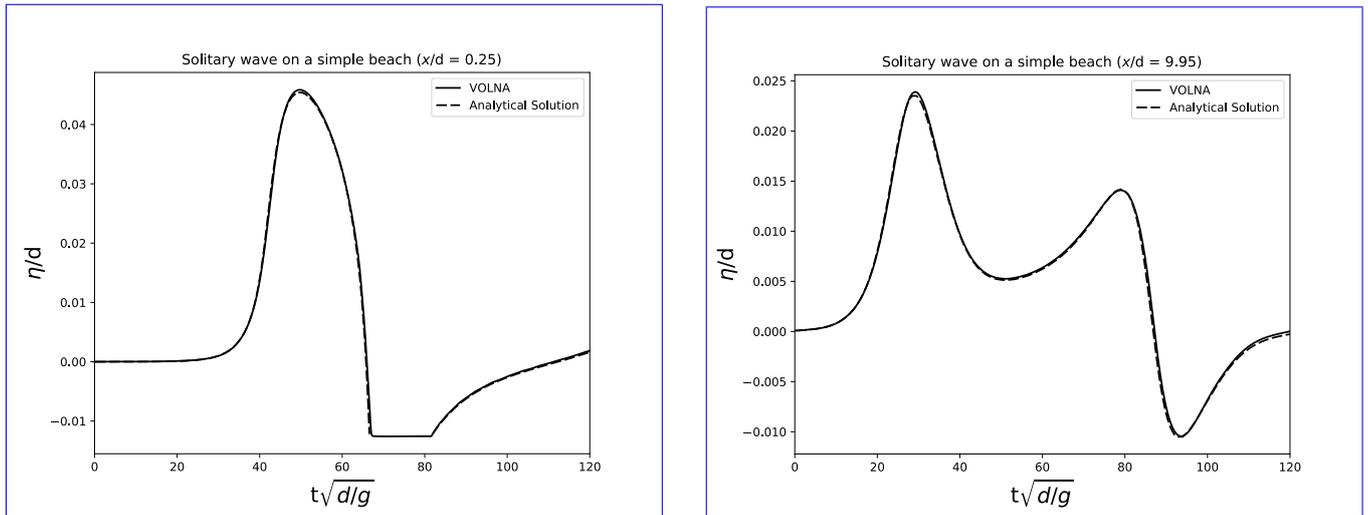
Figure 3. Solitary wave on a simple beach - Comparison between VOLNA and solution at different locations: (a) $x/d = 0.25$: Notice that the location becomes 'dry' for t $\approx (67\sqrt{d/g} - 82\sqrt{d/g}))$, (b) $x/d = 9.95$.

(Figure 4(a)), after which it is recommended that a non-reflective boundary condition be set. At $y=0$, $y= 3.4$ and $x=5.5$m fully reflective boundaries are to be defined.

Tasks

5   The validation of the model involves comparing the temporal variation of the moving shoreline, the water height at fixed gauges and the maximum run up. For the basis of this brief validation, we compared the water height at three gauges installed in the tank, located at (4.521, 1.196), (4.521, 1.696) and (4.521, 2.196).

It can be seen from the gauge plots on Figure 4(b-d) that the first elevation wave arrives between 15 and 25s. The overall dynamics of this elevation wave is accurately captured by the model at all the gauges, particularly the arrival time and initial 10   amplitude. Considering the results of the two benchmark tests and the full validation of the original VOLNA code, one can see that the new implementation which implements a more restrictive limiter still preforms satisfactorily and is consistent with the previous version.

4.4   Code structure

The structure of the code is ~~outline in Algorithm 1~~ outlined in Algorithm 1; the user inputs a configuration file (.vln), which 15   specifies the mesh to be read in from *gmsh* files, as well as initial/boundary conditions of state variables, such as the bathymetry deformation starting the tsunami, which can be defined in various ways (mathematical expressions or files, or a mix of both). We use a variable timestep ~~second-order~~ third-order (four stage) Runge-Kutta method for evolving the solution in time. In each iteration, events may be triggered; e.g. further bathymetry deformations, displaying the current simulation time, or outputting simulation data to VTK files for visualisation.
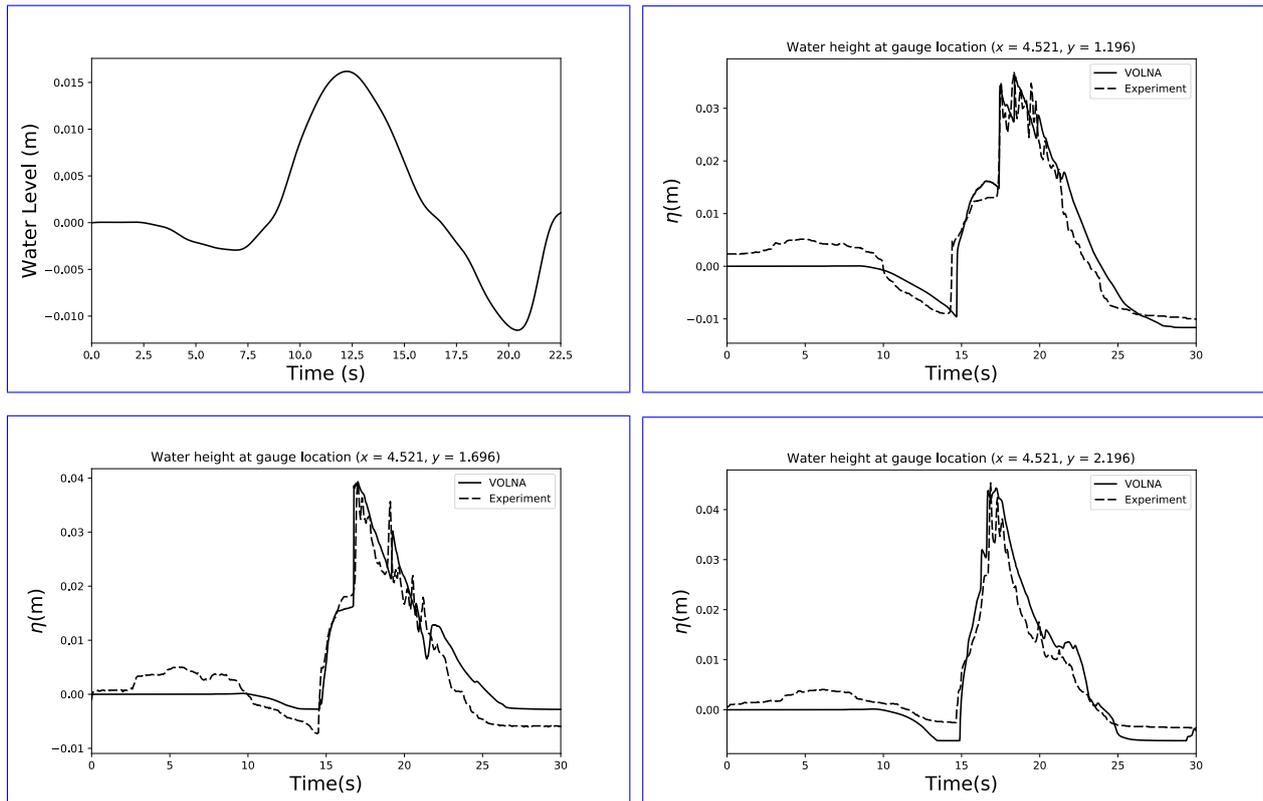
Figure 4. Benchmark Problem 2: (a) The incoming water level incident on the x=0m boundary, Comparison between VOLNA and laboratory results at different locations: (b) x=4.521, y=1.196, (c) x=4.521, y=1.696, (d) x=4.521, y=2.196

Algorithm 1 Code structure of VOLNA

Initalise mesh from gmsh file

Initialise state variables

while $t < t_{final}$ **do**

    Perform pre-iteration events

    Third-order Runge-Kutta time stepper

        Determine local gradients of state variables on each cell

        Compute a local limiter on each cell

        Reconstruct state variables, compute boundary conditions and determine fluxes across cell faces

        Compute timestep

        Apply fluxes and bathymetric source terms to state variables on cells

    Perform post-iteration events

end while

The original VOLNA source code was implemented in C++, utilising libraries such as Boost (Schling (2011)). This gives a very clear structure, abstracting data management, event handling and low level array operations for the higher level algorithm - an example is shown in Figure ~~2.~~ 5. While this coding style was good for readability, it had its limitations in terms of performance: there was excessive amounts of data movement and certain operations could not be parallelised - indirect increments with ~~30~~ potential race conditions in particular. Some features - such as describing the bathymetry lift with a mathematical formula - were implemented with functionality and simplicity, not performance, in mind.

```
outConservative.H  *= dt;
outConservative.U  *= dt;
outConservative.V  *= dt;

outConservative.H  += MidPointConservative.H;
outConservative.U  += MidPointConservative.U;
outConservative.V  += MidPointConservative.V;

outConservative.H  += inConservative.H;
outConservative.U  += inConservative.U;
outConservative.V  += inConservative.V;

outConservative.H  *= .5;
outConservative.U  *= .5;
outConservative.V  *= .5;

outConservative.H  =
     ( outConservative.H.cwise()  <= EPS )
     .select( EPS, outConservative.H );
outConservative.Zb  = inConservative.Zb;
ToPhysicalVariables( outConservative, out );
//Implementation of ToPhysicalVariables:
ScalarValue TruncatedH =
     ( outConservative.H.cwise()  < EPS )
     .select( EPS, outConservative.H );
out.H = outConservative.H;
out.U = outConservative.U.cwise()  / TruncatedH;
out.V = outConservative.V.cwise()  / TruncatedH;
out.Zb = outConservative.Zb;
```

```
inline void EvolveValuesRK2_2(const  float *dT,
              float *outConservative,
              const float *inConservative,
              const float *midPointConservative,
              float *out)
{
  outConservative[0]  *= (*dT);
  outConservative[1]  *= (*dT);
  outConservative[2]  *= (*dT);

  outConservative[0]  += midPointConservative[0];
  outConservative[1]  += midPointConservative[1];
  outConservative[2]  += midPointConservative[2];

  outConservative[0]  += inConservative[0];
  outConservative[1]  += inConservative[1];
  outConservative[2]  += inConservative[2];

  outConservative[0]  *= 0.5f;
  outConservative[1]  *= 0.5f;
  outConservative[2]  *= 0.5f;

  outConservative[0]  = MAX(outConservative[0],EPS);
  outConservative[3]  = inConservative[3];

  //call to ToPhysicalVariables inlined
  float TruncatedH = outConservative[0];
  out[0] = outConservative[0];
  out[1] = outConservative[1] / TruncatedH;
  out[2] = outConservative[2] / TruncatedH;
  out[3] = outConservative[3];
}
...
op_par_loop(EvolveValuesRK2_2, "EvolveValuesRK2_2", cells,
  op_arg_gbl(&dT,1,"float",  OP_READ),
  op_arg_dat(outConservative,-1,OP_ID,4,"float",OP_RW),
  op_arg_dat(inConservative,-1,OP_ID,4,"float",OP_READ),
  op_arg_dat(midPointConservative,-1, OP_ID,4,"float",OP_READ),
  op_arg_dat(values_new,-1,OP_ID,4,"float",  OP_WRITE));
```

Figure 5. Code snippets from the original and OP2 versions

To better support performance and scalability, and thus allow for large-scale simulations, we have re-engineered the VOLNA code to use OP2 - the overall code structure is kept similar, but matters of data management and parallelism are now entrusted to OP2. To support parallel execution we separated the pre-processing step from the main body of the simulation: first the mesh

~~Code structure of VOLNA Initalise mesh from gmsh file Initialise state variables Perform pre-iteration events Second-order Runge-Kutta time stepper   Update state variables on cell faces, compute boundary conditions   Determine fluxes across cell faces and compute timestep   Apply fluxes and bathymetric source terms to state variables on cells Perform post-iteration events~~

~~Code snippet from the original VOLNA~~

~~Corresponding code in OP2-VOLNA~~

11

and simulation parameters are parsed into a HDF5 data file, which can then be read in parallel by the main simulation, which also uses HDF5~~'~~'s parallel file I/O to write results to disk.

Performance-critical parts of the code, essentially any operations on the computational mesh, are re-implemented using OP2: they are written with an element-centric approach and grouped for maximal data reuse. Calculations that were previously a ~~5~~ sequence of operations, each calculating all partial results for the entire mesh, now apply only to single elements (such as cells or edges), and OP2 automatically applies these computations to each element - this avoids the use of several temporaries and improves computational density. This process involves outlining the computational ~~"kernel"~~ "kernel" to be applied at each set element (cell or edge) to a separate function, and writing a call to the OP2 library - a matching code snippet is shown in Figure ~~3.~~ 5.

The workflow of VOLNA is made of a few sources of information being created and given as inputs to the code. The first is the merged bathymetry and topography over the whole computational domain, i.e. the seafloor and land elevations, over which the flow will propagate. This is given through an unstructured triangular mesh. This is then transformed into a usable input to VOLNA via the ~~volna2hdf5~~ _volna2hdf5_ code to generate compact HDF5 files. The mesh is also renumbered with the Gibbs-Poole-Stokmeyer algorithm to improve locality.

The second is the dynamic source of the tsunami. It can be an earthquake or a landslide. To describe the temporal evolution of seabed deformation, either a function can be used, or a series of files. When a series of files is used (typically when another numerical model provides the spatio-temporal information of a complex deformation), there is a need to define the frequency of these updates in the so-called ~~vln~~ _vln_ generic input file to VOLNA. A recent improvement has been the ability to define these series of files for a sub-region of the computational domain, and at possibly lower resolution. Performance is better when using a function for the seabed deformation, since I/O requirements for files can generate large overheads - VOLNA-OP2 allows for describing the initial bathymetry with an input file, and then specifying relative deformations using arbitrary code that is a function of spatial coordinates and time. ~~In a similar manner~~Similarly, one can also define initial conditions for wave elevation and velocity.

The generic input file of VOLNA, includes information about the frequency of the updates in the seabed deformation, the virtual gauges where time series of outputs will be produced and possibly some options to output time series of outputs over the whole computational domain in order to create movies for instance. These I/O requirements obviously affect performance: the more data to output and the slower the file system, the larger the effect.

To simulate tsunami hazard for a large number of scenarios is computationally expensive, so VOLNA has been replaced in past studies by a statistical emulator, i.e. a cheap surrogate model of the simulator~~(Sarri et al. (2012); Beck and Guillas (2016); Liu and Guillas (2017); Salmanidou et al. (2017); Guillas et al. (2018)).~~. To build the emulator, input parameters are varied in a design of experiments, and the runs are submitted with these inputs to collect input-output relationships. The output of interest could for example be the waveforms, free surface elevation, and velocity, among others. The increase in flexibility in the definition of the region over which the earthquake source of the tsunami is defined reduces the size of the series of files used as inputs: this is really helpful when a set of simulations needs to be run. Similarly, the ability to specify the relative deformation

using an arbitrary code that is a function of spatial coordinates and time also reduces the computational and memory overheads when running a set of simulations.

5 Results

5.1 Running VOLNA

A key goal of this paper is to demonstrate that by utilising the OP2 library, VOLNA delivers scalable high performance on a number of common systems. Therefore we take a testcase simulating tsunami propagation in the Indian Ocean, and run it on three different machines: NVIDIA P100 Graphical Processing Units (~~specifically a DGX-1 system in the UK's JADE supercomputer~~The Wilkes2 machine in Cambridge's CSD3), a classical CPU architecture in the ~~UK's Archer supercomputer~~ Peta5-Skylake part of CSD3 (specifically dual-socket Intel Xeon ~~12-core Ivy Bridge~~ Gold 6142 16-core Skylake CPUs), and Intel~~'~~'s Xeon Phi platform in Peta5-KNL (64-core Knights Landing-generation chips, configured in cache mode). ~~There are four~~

There are five key computational stages that make up 90% of the total runtime: a stage evolving time using the ~~second-order~~ third-order Runge-Kutta scheme (~~RK),~~ *RK), gradients* computes gradients between cells a stage that computes the ~~physical~~ fluxes across the edges of the mesh (~~fluxes~~*fluxes*), a stage that computes the minimum timestep (~~dT~~*dT*), and a stage that applies the fluxes to the cell-centered state variables (~~applyFluxes~~*applyFluxes*). Each of these stages consist of multiple steps, but for performance analysis we study them in groups.

The *RK* stage is computationally fairly simple, no indirect accesses are made, cell-centered state variables are updated using other cell-centered state variables, and therefore parallelism is easy to exploit, and the limiting factor to performance will be the speed at which we can move data; achieved bandwidth. ~~The~~ Both the *gradients* and the *fluxes* ~~stage is~~ stages are computationally complex, and ~~the results are edge-centered, therefore~~ involve accessing large amounts of data ~~accessed indirectly through an~~ indirectly through cells-to-cells and edges-to-cells ~~mapping~~mappings. The *dT* stage moves significant amounts of data to compute the appropriate timestep for each cell, triggering an MPI halo exchange as well, and then carries out a global reduction to calculate the minimum - particularly over MPI this can be an expensive operation, but overall it is limited by bandwidth. The *applyFluxes* stage, while computationally simple, is complex due to its indirect increment access patterns; per-edge values have to be added onto cell-centered values, and in parallelising this operation, OP2 needs to make sure to avoid race conditions.

The performance of this loop is limited by the irregular accesses and control throughout the hardware. For an in-depth study of individual computational loops and their performance we refer the reader to our previous work in Reguly et al. (2007).

5.2 Tsunami demonstration case

For performance and scaling analysis, we employ the Makran subduction zone as the tsunamigenic source for the numerical simulations. Our region of interest extends from ~~$55 \circ E$ to $79 \circ E$ and from $6 \circ N$ to $30 \circ N$~~$55°E$ to $79°E$ and from $6°N$ to $30°N$. The bathymetry (Fig. ~~4a)~~6(a)) is obtained from GEBCO (~~www.gebco.net~~www.gebco.net). The region of interest

Table 1. Details of the ~~uniform~~ non-uniform (NU) triangular meshes

Mesh	Name	~~Size~~ Vertices	Edges	Triangles	Source λ	~~($h \sim 0.9\,km$)~~ Mesh size at coast
		n_V	n_E	n_T	λ_0	h_{min}
~~M_1~~ NU_0	~~23.1~~ 53.7M	~~h~~ 26863692	~~11545882~~ 80564925	~~34627495~~ 53701234	~~23081614~~ 12.5 km	125 m
~~M_2~~ NU_1	~~5.7~~ 13.8M	~~2h~~ 6931758	~~2897099~~ 20771822	~~8681146~~ 13840065	~~578404~~ 25 km	250 m
~~M_3~~ NU_2	~~1.4~~ 3.6M	~~4h~~ 1812073	~~732710~~ 5414155	~~2187979~~ 3602083	~~145527~~ 50 km	500 m
NU_3	0.95M	485453	1435017	949565	100 km	1000 m

Table 2. Finite fault parameters of the 4-segment tsunamigenic earthquake source

Segment	Length (~~L~~ l)	Down-dip width (~~W~~ w)	Longitude	Latitude	Depth	Strike	Dip	Rake
i	(km)	(km)	(°)	(°)	(km)	(°)	(°)	(°)
1	220	150	65.23	24.50	10	263	6	90
2	188	150	63.08	24.23	10	263	7	90
3	199	150	61.25	24.00	5	281	8	90
4	209	150	59.32	24.32	5	286	9	90

is projected about the center latitude (~~i.e. 18∘N~~ i.e. $18°N$) to form the rectangular computational domain for VOLNA in Cartesian co-ordinates (Fig. ~~4b)~~ 6(b)). This translates to a region of approximately ~~2500km× 2700km~~ $2500\,km \times 2700\,km$ in area. The calculation of the sea-floor deformation or uplift (assumed instantaneous) is modeled via the Okada solution (~~Okada (1992))~~ Okada (1992)). This deformation is generated by the earthquake source which is modeled as a 4-segment finite fault model (Table ~~2~~ 2) with a uniform slip of ~~30m. Although~~ $30\,m$. The non-uniform meshes ~~are essential for resolution of certain effects like inundation, port vortices and velocities, for the purpose of ascertaining the speed-up we restrict ourselves to uniform meshes. These uniform triangular meshes for~~ for the simulation are generated using Gmsh (Geuzaine and Remacle (2009)). A simple strategy is used to generate these meshes. Using the dimensions of the finite fault earthquake sources ($l \times w$), an approximate source wavelength ($\lambda_0 < min(l,w)$) of the tsunami, and the ocean depth of the Makran trench ($d_0 \sim 3\,km$), we calculate the time period (T) of the ~~simulations are generated using Gmsh (Geuzaine and Remacle (2009)). The characteristic mesh sizes are h, $2h$, $4h$ (Table 1), where $h \sim 0.9\,km \sim$ GEBCO resolution of 30". Finally~~ wave as, $T = \dfrac{\lambda_0}{\sqrt{gd_0}}$. Next, assuming that the time period of the tsunami is same everywhere in the domain, we get for a depth d_n, ~~the total number of virtual gauges is around 10^4 distributed in a uniform 100×100 grid~~ $\dfrac{\lambda_n}{\sqrt{d_n}} = \dfrac{\lambda_0}{\sqrt{d_0}}$, which in turn relates the characteristic triangle (or element) length h_n for depth d_n as, $h_n = \dfrac{\lambda_0}{k}\sqrt{\dfrac{d_n}{d_0}}$, where $k = 10$. At the shore (i.e. $d = 0$), a minimum mesh size (h_{min}) is specified. Linear interpolation is carried out to further smoothen the mesh gradation. A combination of λ_0 and h_{min} is used to generate a series of non-uniform meshes (Table 1 and Figure 7). We also fix the triangle size as $25\,km$ for regions that are deep inland. Finally, Figure 8 shows the tsunami waveforms at two virtual gauge locations. Simulated time is $21660\,s$ for all mesh sizes, however, for timed runs at different scales on different platforms we restrict this to $2000\,s$ to conserve computer time.
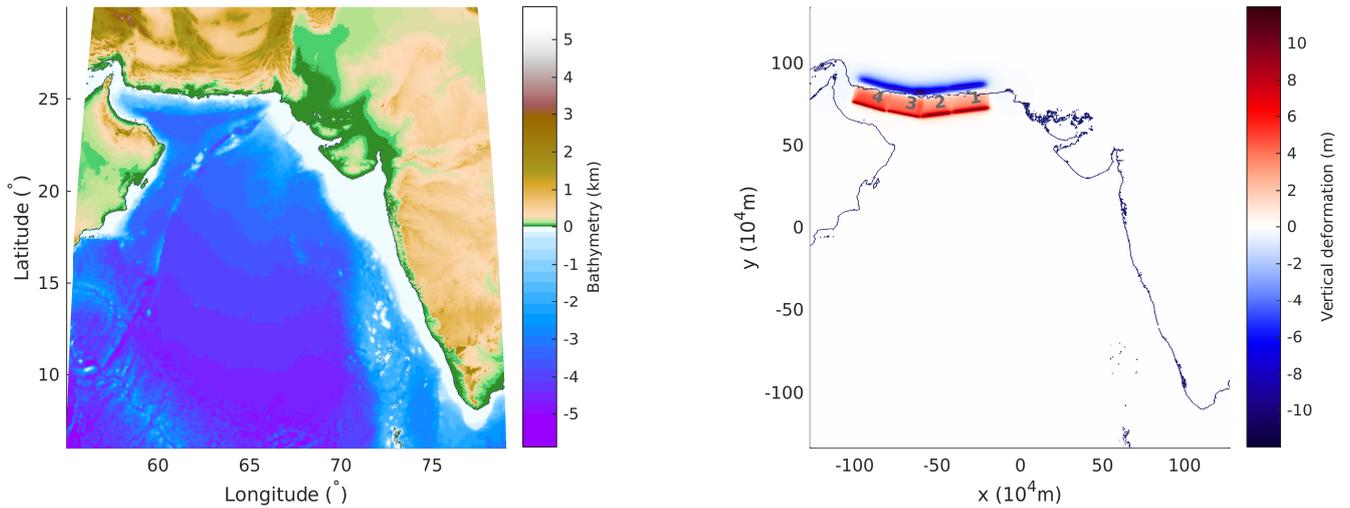
Figure 6. (a) Bathymetry from GEBCO's geodetic grid is mapped onto a Cartesian grid for use in VOLNA. (b) Uplift caused by a uniform slip of $30\,m$ in the 4 segment finite fault model (given in Table 2).
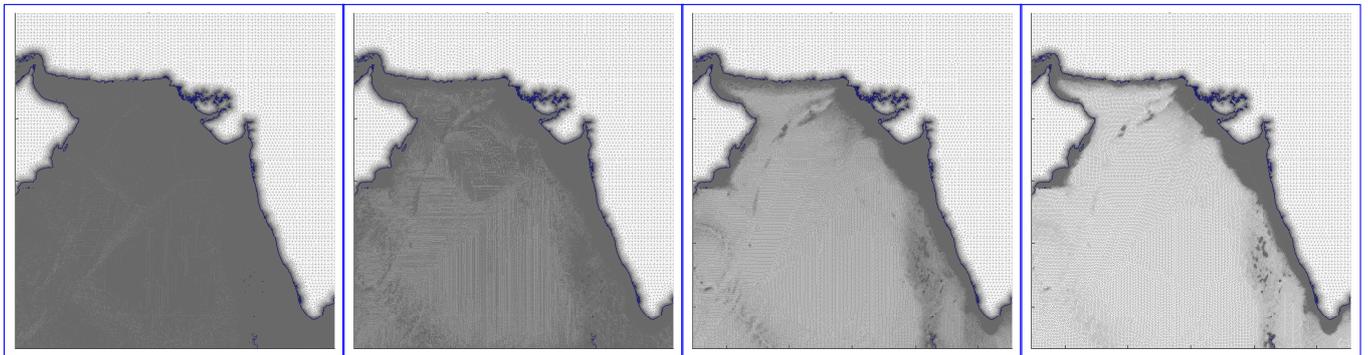
Figure 7. Non-uniform meshes corresponding to the test cases (see Table 1).

5.3 Performance and Scaling on classical CPUs

As the most commonly used architecture, we first evaluate performance on a classical CPUs in the ~~Archer~~ Cambridge CSD3 supercomputer: dual-socket Xeon ~~E5-2697 v2~~ Gold 6142 CPU, with ~~12~~ 16 cores each, supporting the ~~AVX~~ AVX512 instruction set. We ~~use a~~ test a plain MPI configuration (32 processes per node), as well as a hybrid MPI+OpenMP configuration, with 2 MPI processes per node (1 per socket), and ~~12~~ 16 OpenMP threads each, with process and thread binding enabled.

~~First, we compared two code variants generated by~~ We use OP2~~: a 'plain' version, and one with extra code generated to enable AVX vectorisation. The RK~~ 's vectorised code generation capabilities, as described in Mudalige et al. (2016). The *RK* stage performs the same in both variants, however the ~~fluxes and dT~~ *fluxes* and *dT* stages saw significant performance gains - the

15
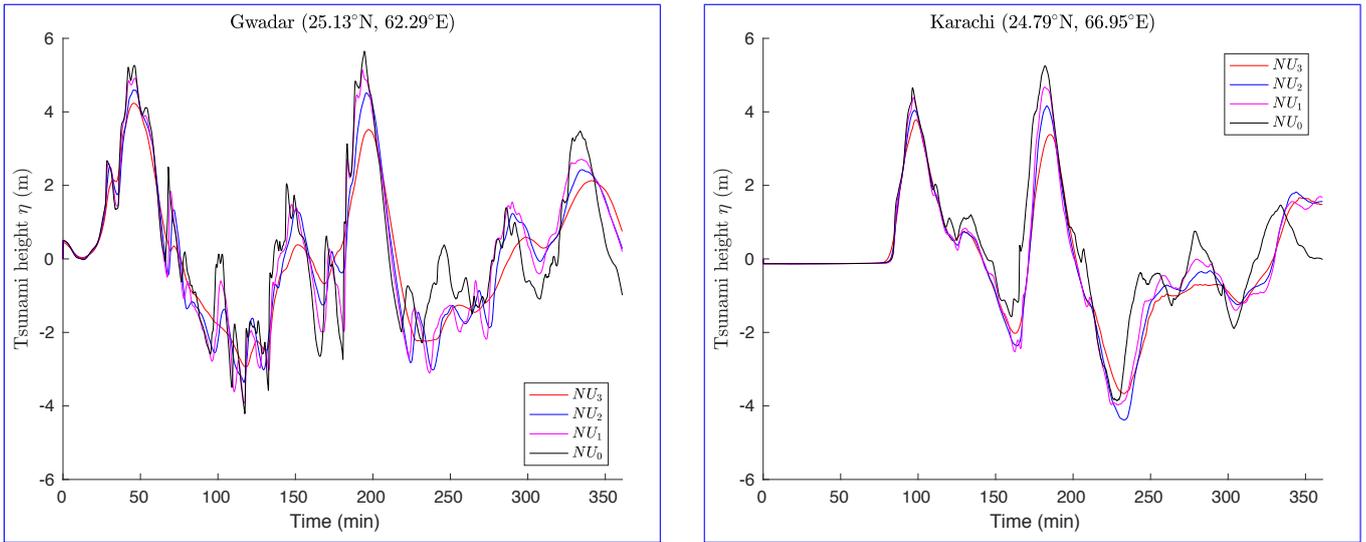
Figure 8. Tsunami waveforms at virtual gauges located at Gwadar and Karachi.

compiler did not automatically vectorise computations~~in these stages~~, it had to be forced to do so. The ~~applyFluxes~~ *applyFluxes* stage could not be vectorised due to ~~race conditions~~a compiler issue.

On a single node with pure MPI, running the largest mesh, ~~26~~9% of time was spent in the ~~RK~~ *RK* stage, achieving ~~96~~ 182 GB/s throughput on average, ~~29~~40% of time was spent in the ~~fluxes~~ *gradients* stage, achieving ~~68~~ 108 GB/s, ~~13~~25% of time was spent in the ~~dT~~ *fluxes* stage, achieving 142 GB/s, 12% of time was spent in the *dT* phase, achieving ~~76~~ 65 GB/s, and ~~30~~12% of time was spent in the ~~applyFluxes~~ *applyFluxes* stage, achieving ~~76~~ 221 GB/s thanks to a high degree of data reuse. The maximum bandwidth on this platform is ~~120~~ 189 GB/s as measured by STREAM Triad. The time spent in MPI communications ranged from ~~7~~23% on the smallest mesh to ~~1.9~~10% on the largest mesh.
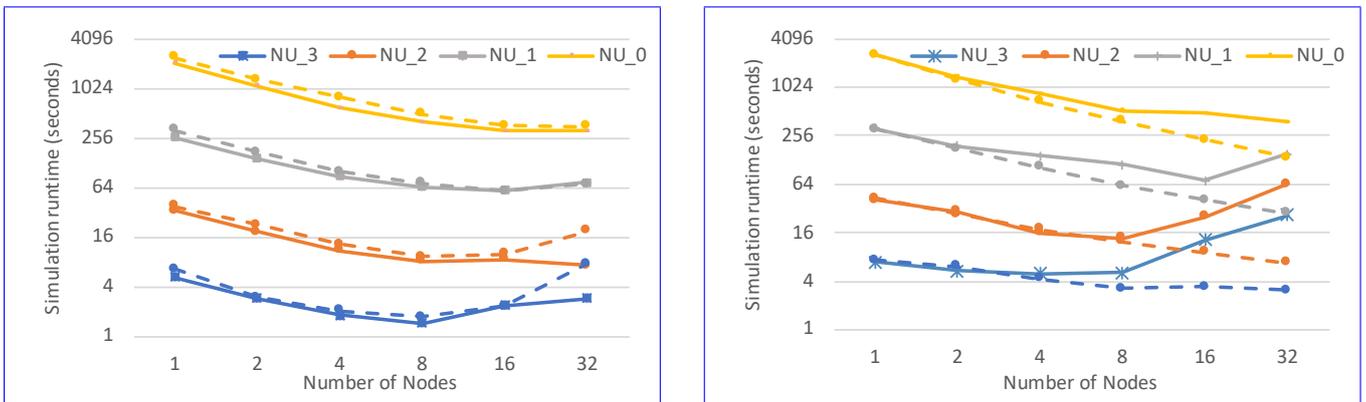
Figure 9. Performance scaling on (a) Peta5-CPU (Intel Xeon CPU) and (b) Peta5-KNL (Intel Xeon Phi) at different mesh sizes with pure MPI (solid) and MPI+OpenMP (dashed)

16

When scaling to multiple nodes ~~with pure MPI,~~ with pure MPI, as shown in Figure ~~5~~9(a), it is particularly evident on the smallest problem that the problem size per node needs to remain reasonable, otherwise MPI communications will dominate the runtime: for the ~~1.4M~~ NU_0 mesh, at 32 nodes ~~5.8~~ 251 seconds out of ~~9.36 total (62~~308 total (81%). This can be characterised by the ~~strong scaling efficiency~~ *strong scaling efficiency*; when doubling the number of computational resources (nodes), what percentage of the ideal 2~~x~~ $\times$ speedup is achieved. For ~~the smallest problem, scaling efficiency is above 90%up to 8 nodes, but at 16 and~~ small node counts these values remain above a reasonable 85%, but particularly for the smaller problems runtimes actually become worse. It is evident that on the Peta5-Skylake cluster the interconnect used for MPI communications becomes a bottleneck for scaling - this overhead is significantly lower on e.g. Archer on the largest mesh at 32 ~~it falls rapidly, to 69% and 58% respectively. On M2 efficiency is above 95%up to 16 nodes, and falls to 87% at 32 nodes, and for the largest mesh, it remains above 95%~~nodes it is only 32%.

~~Performance scaling on Archer (Intel Xeon CPU) at three different mesh sizes with 1.4, 5.7, and 23.1 million triangles~~We have also evaluated execution with a hybrid MPI+OpenMP approach, as shown with the dashed lines in Figure 9(a). However, on this platform it failed to outperform the pure MPI configuration.

~~Performance scaling on Archer-KNL (Intel Xeon Phi) at different mesh sizes~~

5.4 Performance and Scaling on the Intel Xeon Phi

Second, we evaluate Intel~~'~~'s latest many-core chip, the Xeon Phi x7210, which integrates 64 cores, each equipped with AVX-512 vector processing units and supporting 4 threads, and built with a 16GB on-chip high-bandwidth memory, here used as a cache for off-chip DDR4 memory. The chips were configured in the ~~"quad"~~ "quad" mode, and all 16GB as cache. We ~~use~~ evaluate a pure MPI approach (128 processes) as well as using 4 MPI processes, one per quadrant, and 32 OpenMP threads each. Bandwidth achieved as measured by STREAM Triad is 448 GB/s. Vectorisation on this platform is paramount for achieving high performance - every stage with the exception of ~~applyFluxes~~ *applyFluxes* was vectorised - the latter was not due to compiler issues.

On a single node with pure MPI, the straightforward computations of the ~~RK~~ *RK* stage can utilise the available high bandwidth very efficiently: only ~~15~~8.3% of time spent here, achieving ~~a 352~~194 GB/s. The ~~fluxes stage takes 2~~*gradients* stage takes 42% of time~~and achieves 156~~, achieving 82 GB/s, the *fluxes* stage takes 25% of time and achieves 104 GB/s, ~~dT 12.5~~*dT* 11.4% and achieves ~~160~~46 GB/s, and the ~~applyFluxes stage takes 47~~*applyFluxes* stage takes 11.6% and achieves ~~only 92~~165 GB/s~~largely due to the lack of vectorisation and irregular memory access patterns~~. On the largest mesh, it is ~~1.92× faster~~21% slower than a single node of the classical CPU system~~, but on the smallest it is only 8% faster - due to underutilisation~~.

Performance when scaling to multiple nodes with pure MPI is shown in Figure ~~6~~9(b): it is quite clear that scaling is worse than on the classical CPU architecture for smaller problem sizes - the Xeon Phi requires a considerably larger problem size per node to operate efficiently. Strong scaling efficiency is particularly poor on the smallest mesh~~at 58-61%~~, but even on the largest mesh it ~~'s only between 82-87%. At scale therefore the performance advantage over~~ is only between 63-92%. Similarly to the classical CPU system, the interconnect becomes a bottleneck to scaling. Running with a hybrid MPI+OpenMP configuration on the Xeon Phi does improve scaling significantly, as shown in Figure 9(b) - this is due to having to exchange much fewer
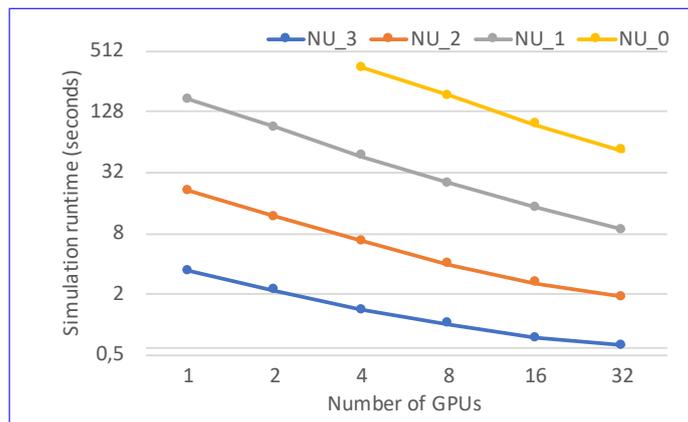
Figure 10. Performance scaling on ~~JADE~~ Wilkes2 (P100 GPU) at different mesh sizes

(but larger) messages. Strong scaling efficiency on the largest problem remains above 82%. At scale, at least on this cluster, the Xeon Phi can outperform the classical CPU system ~~becomes lower~~on a node-to-node basis of comparison.

5.5 Performance and Scaling on P100 GPUs

Third, we evaluate performance on GPUs - an architecture that has continually been increasing its market share in high ~~per-~~
5 ~~formance~~ performance computing thanks to its efficient parallel architecture. The P100 GPUs are hosted in ~~NVIDIA's DGX-1 system, they are interconnected by high-speed NVLink connections~~the Wilkes2 system, with 4 GPU per node connected via the PCI-e bus. Each chip contains 60 Scalar Multiprocessors, with 64 CUDA cores each, giving a total of 3840 cores. There is also 16 GB of high-bandwidth memory on-package, with a bandwidth of 497 GB/s. To utilise these devices, we use CUDA code generated by OP2, and compiled with CUDA ~~8.~~9. Similarly to Intel~~·~~'s Xeon Phi, high vector efficiency is required for
10 good performance on the GPU.

On a single GPU, running the ~~largest mesh , 25~~second-largest mesh NU_1 (Because NU_0 does not fit in memory), 8.3% of runtime is spent in the ~~RK~~ *RK* stage, achieving ~~281~~ 342 GB/s, ~~fluxes takes~~ *gradients* takes 50% if time, achieving only 136 GB/s due to its high complexity, *fluxes* takes 15%, achieving ~~528~~ 379 GB/s thanks to a high degree of data re-use in indirect accesses, ~~dT takes 11.4~~*dT* takes 4.4% and achieves ~~490~~ 382 GB/s, and finally ~~applyFluxes takes 49~~*applyFluxes* takes 20%
15 of time, achieving ~~223~~ 204 GB/s. Indeed, this last phase has the most irregular memory access patterns, which is commonly known to be degrading performance on GPUs. Nevertheless, even a single GPU outperforms a classical CPU node by a factor of ~~3.6~~1.5, and the Xeon Phi by ~~1.89×. On the smallest problem, this ratio is only 2.9 - more parallelism is required for efficient operation.~~1.85×.

Performance when scaling to multiple GPUs is shown in Figure ~~7.~~10; similarly to the Xeon Phi, GPUs are also sensitive
20 to the problem size and the overhead of MPI communications, ~~even more so than the Phi~~however given that there are 4 GPUs in one node, the overhead of communications is significantly lower. On the smallest problem efficiency drops from ~~64% to 48~~78% to 58%, and on the largest problem from 95% to ~~85%. Even so, at 8 GPUs it is still 2.3~~89%. Thanks to much better

18

scaling (due to lower MPI overhead), 32 GPUs are 6.9/~~3.3× faster than 8 nodes of the Phi or the CPU system~~ 2.6× faster than 32 nodes of Xeon CPUs and Xeon Phis respectively.

5.6 Running Costs and Power Consumption~~Considerations~~

~~When running large numbers of long simulations , the cost-to-solution is an important metric, and a primary driver of this is energy consumption~~ Ultimately, when one needs to decide what platform to run these simulations on, a key deciding factor aside from time-to-solution, is cost-to-solution. In the analysis above, aside from discussing absolute performance metrics, we have reported speedup numbers relative to other platforms - ~~admittedly~~ which from a performance benchmarking perspective is not strictly fair. However, such relative performance figures combined with the cost of access do help in the decision.

Admittedly the cost buying hardware, as well as the cost of core-hours or GPU-hours varies significantly, therefore here we do not look at specific prices. However, ~~in terms of energy efficiency, a~~ energy consumption is an indicator of pricing. A dual-socket CPU consumes up to 260 Watts, which is then roughly tripled when looking at the whole node, due to memory, disks, networking, etc. In comparison, the Intel Xeon Phi CPU has a TDP of ~~215W, roughly 750W~~ 215 W, roughly 750 W for the node. A P100 GPU has a TDP of ~~300W~~ 300 W, but has to be hosted in a CPU system - the more GPUs in a single machine, the better amortised this cost is: the TDP of a ~~DGX-1 system is 3.2 KW (8x300~~ GPU node in Wilkes2 is around 1.8 KW (4x250 for the GPUs, plus 800 for the rest of the system) - which averages to ~~400~~ 450 W per GPU. Thus in terms of power efficiency GPUs are by far the best choice for VOLNA. Nevertheless, a key benefit of VOLNA-OP2, is that it can efficiently utilise any high performance hardware commonly available.

6 Conclusions

In this paper we have introduced and described the VOLNA-OP2 code; a tsunami simulator built on the OP2 library, enabling execution on CPUs, GPUs, and heterogeneous supercomputers. By building on OP2, the science code of VOLNA itself is written only once using a high-level abstraction, capturing ~~what~~ *what* to compute, but not ~~how~~ *how* to compute it. This approach enables OP2 to take control of the data structures and parallel execution; VOLNA is then automatically translated to use sequential execution, OpenMP, or CUDA, and by linking with the appropriate OP2 back-end library, these are then combined with MPI. This approach also future-proofs the science code: as new architectures come along, the developers of OP2 will update the back-ends and the code generators, allowing VOLNA to make use of them without further effort. This kind of ease-of-use and portability makes VOLNA-OP2 unique between the tsunami simulation codes. Through performance scaling and analysis of the code on traditional CPU clusters, as well as GPUs and Intel's Xeon Phi, we have demonstrated that VOLNA-OP2 indeed delivers high performance on a variety of platforms and, depending on problem size, scales well to multiple nodes.

We have described the key features of VOLNA, the discretisation of the underlying physical model (~~i.e.~~ *i.e.* NSWE) in the finite volume context and the ~~second-order~~ third-order Runge-Kutta timestepper, as well as the input/output features that allow the integration of the simulation step into a larger workflow; initial conditions, and bathymetry in particular, can be specified in a number of ways to minimise I/O requirements, and ~~efficient~~ parallel output is used to write out simulation data on the full mesh or specified points.

~~We have discussed how VOLNA-OP2 is currently being used in different of ways, with a particular focus on its use within a statistical emulator; there are already published papers on this: Beck and Guillas (2016); Liu and Guillas (2017); Salmanidou et al. (2017).~~

~~Through performance scaling and analysis of the code on traditional CPU clusters, as well as GPUs and Intel's Xeon Phi, we have demonstrated that VOLNA-OP2 indeed delivers near-optimal performance on a variety of platforms and, depending on problem size, scales well to multiple nodes.~~

5

There is still a need for even more streamlined and efficient workflows. For instance, we could integrate within VOLNA, the finite fault source model for the slip with some assumptions on the rupture dynamics, we could also integrate the ~~bathymetry-based~~ bathymetry-based meshing (the mesh needs to be tailored to the depth and gradients of the bathymetry to optimally reduce computational time). Indeed, there would be even less exchanges of files and more efficient computations, especially in the context of uncertainty quantification tasks such as emulation or inversion.

10

In the end, the gain in computational efficiency will allow higher resolution modelling, such as using ~~2 m~~ $2m$ topography and bathymetry collected from LIDAR, i.e. a greater capability. It will allow greater capacity by enabling more simulations to be performed. Both of these enhancements will subsequently lead to better warnings more tailored to the actual impact on the coast as well as better urban planning since hazard maps will gain in precision geographically and probabilistically, due to the possibility of exploring a larger number of more realistic scenarios.

15

20

Competing interests. The authors declare that they have no conflict of interest.

References

Abadie, S. M., Harris, J. C., Grilli, S. T., and Fabre, R.: Numerical modeling of tsunami waves generated by the flank collapse of the Cumbre Vieja Volcano (La Palma, Canary Islands): Tsunami source and near field effects, Journal of Geophysical Research: Oceans, 117, n/a–n/a, https://doi.org/10.1029/2011JC007646, http://dx.doi.org/10.1029/2011JC007646, c05030, 2012.

5 Acuña, M. and Aoki, T.: Real-time tsunami simulation on multi-node GPU cluster, in: ACM/IEEE conference on supercomputing, 2009.

Anita, G., Andrey, B., Ana, B. M., Jörn, B., Antonio, C., Gareth, D., L., G. E., Sylfest, G., I., G. F., Jonathan, G., B., H. C., J., L. R., Stefano, L., Finn, L., Rachid, O., Christof, M., Raphaël, P., Tom, P., Jascha, P., William, P., Jacopo, S., B., S. M., and Kie, T. H.: Probabilistic Tsunami Hazard Analysis: Multiple Sources and Global Applications, Reviews of Geophysics, 55, 1158–1198, https://doi.org/10.1002/2017RG000579, https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2017RG000579.

10 Beck, J. and Guillas, S.: Sequential Design with Mutual Information for Computer Experiments (MICE): Emulation of a Tsunami Model, SIAM/ASA Journal on Uncertainty Quantification, 4, 739–766, https://doi.org/10.1137/140989613, https://doi.org/10.1137/140989613, 2016.

Behrens, J. and Dias, F.: New computational methods in tsunami science, Phil. Trans. R. Soc. A, 373, 20140 382, 2015.

Bernard, E., Mofjeld, H., Titov, V., Synolakis, C., and González, F.: Tsunami: scientific frontiers, mitigation, forecasting and policy implica-
15 tions, Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 364, 1989–2007, https://doi.org/10.1098/rsta.2006.1809, http://rsta.royalsocietypublishing.org/content/364/1845/1989, 2006.

Brodtkorb, A. R., Hagen, T. R., Lie, K.-A., and Natvig, J. R.: Simulation and visualization of the Saint-Venant system using GPUs, Comput-ing and Visualization in Science, 13, 341–353, https://doi.org/10.1007/s00791-010-0149-x, https://doi.org/10.1007/s00791-010-0149-x, 2010.

20 Castro, M., González-Vida, J., Macías, J., Ortega, S., and de la Asunción, M.: Tsunami-HySEA: a GPU-based model for tsunami early warning systems, in: Proc XXIV Congress on Differential Equations and Applications, Cádiz, June, pp. 8–12, 2015.

Davies, G., Griffin, J., Løvholt, F., Glimsdal, S., Harbitz, C., Thio, H. K., Lorito, S., Basili, R., Selva, J., Geist, E., and Baptista, M. A.: A global probabilistic tsunami hazard assessment from earthquake sources, Geological Society, London, Special Publications, 456, https://doi.org/10.1144/SP456.5, http://sp.lyellcollection.org/content/early/2017/03/01/SP456.5, 2017.

25 Dias, F., Dutykh, D., O'Brien, L., Renzi, E., and Stefanakis, T.: On the Modelling of Tsunami Generation and Tsunami Inundation, Proce-dia IUTAM, 10, 338 – 355, https://doi.org/https://doi.org/10.1016/j.piutam.2014.01.029, http://www.sciencedirect.com/science/article/pii/S2210983814000303, mechanics for the World: Proceedings of the 23rd International Congress of Theoretical and Applied Mechanics, ICTAM2012, 2014.

Dutykh, D. and Dias, F.: Tsunami generation by dynamic displacement of sea bed due to dip-slip faulting, Mathematics and Computers in
30 Simulation, 80, 837 – 848, https://doi.org/http://dx.doi.org/10.1016/j.matcom.2009.08.036, http://www.sciencedirect.com/science/article/pii/S0378475409002821, 2009.

Dutykh, D., Poncet, R., and Dias, F.: The VOLNA code for the numerical modeling of tsunami waves: Generation, propagation and inunda-tion, European Journal of Mechanics-B/Fluids, 30, 598–615, 2011.

Gailler, A., Hébert, H., Loevenbruck, A., and Hernandez, B.: Simulation systems for tsunami wave propagation forecasting within the French
35 tsunami warning center, Natural Hazards and Earth System Sciences, 13, 2465, 2013.

Geist, E. L. and Parsons, T.: Probabilistic Analysis of Tsunami Hazards*, Natural Hazards, 37, 277–314, https://doi.org/10.1007/s11069-005-4646-z, https://doi.org/10.1007/s11069-005-4646-z, 2006.

George, D. L. and LeVeque, R. J.: Finite volume methods and adaptive refinement for global tsunami propagation and local inundation, 2006.

Geuzaine, C. and Remacle, J.-F.: Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering, 79, 1309–1331, https://doi.org/10.1002/nme.2579, http://dx.doi.org/10.1002/nme.2579, 2009.

5   Giles, M. B., Mudalige, G. R., Sharif, Z., Markall, G., and Kelly, P. H.: Performance analysis and optimization of the OP2 framework on many-core architectures, The Computer Journal, 55, 168–180, 2011.

Gisler, G., Weaver, R., and Gittings, M. L.: SAGE calculations of the tsunami threat from La Palma, Sci. Tsunami Hazards, 24, 288–312, 2006.

Glimsdal, S., Pedersen, G. K., Harbitz, C. B., and Løvholt, F.: Dispersion of tsunamis: does it really matter?, Natural Hazards and Earth System Sciences, 13, 1507–1526, https://doi.org/10.5194/nhess-13-1507-2013, https://www.nat-hazards-earth-syst-sci.net/13/1507/2013/, 10   2013.

Goda, K., Mai, P. M., Yasuda, T., and Mori, N.: Sensitivity of tsunami wave profiles and inundation simulations to earthquake slip and fault geometry for the 2011 Tohoku earthquake, Earth, Planets and Space, 66, 105, https://doi.org/10.1186/1880-5981-66-105, https://doi.org/10.1186/1880-5981-66-105, 2014.

15   Gopinathan, D., Venugopal, M., Roy, D., Rajendran, K., Guillas, S., and Dias, F.: Uncertainties in the 2004 Sumatra–Andaman source through nonlinear stochastic inversion of tsunami waves, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 473, https://doi.org/10.1098/rspa.2017.0353, http://rspa.royalsocietypublishing.org/content/473/2205/20170353, 2017.

Harig, S., Pranowo, W. S., and Behrens, J.: Tsunami simulations on several scales, Ocean Dynamics, 58, 429–440, 2008.

Jacobs, C. T. and Piggott, M. D.: Firedrake-Fluids v0.1: numerical modelling of shallow water flows using an automated solution framework, 20   Geoscientific Model Development, 8, 533–547, https://doi.org/10.5194/gmd-8-533-2015, https://www.geosci-model-dev.net/8/533/2015/, 2015.

Kennedy, A. B., Chen, Q., Kirby, J. T., and Dalrymple, R. A.: Boussinesq modeling of wave transformation, breaking, and runup. I: 1D, Journal of waterway, port, coastal, and ocean engineering, 126, 39–47, 2000.

Kervella, Y., Dutykh, D., and Dias, F.: Comparison between three-dimensional linear and nonlinear tsunami generation models, 25   Theoretical and Computational Fluid Dynamics, 21, 245–269, https://doi.org/10.1007/s00162-007-0047-0, https://doi.org/10.1007/s00162-007-0047-0, 2007.

Lawrence, B. N., Rezny, M., Budich, R., Bauer, P., Behrens, J., Carter, M., Deconinck, W., Ford, R., Maynard, C., Mullerworth, S., Osuna, C., Porter, A., Serradell, K., Valcke, S., Wedi, N., and Wilson, S.: Crossing the Chasm: How to develop weather and climate models for next generation computers?, Geoscientific Model Development Discussions, 2017, 1–36, https://doi.org/10.5194/gmd-2017-186, https: 30   //www.geosci-model-dev-discuss.net/gmd-2017-186/, 2017.

Lay, T., Kanamori, H., Ammon, C. J., Nettles, M., Ward, S. N., Aster, R. C., Beck, S. L., Bilek, S. L., Brudzinski, M. R., Butler, R., DeShon, H. R., Ekström, G., Satake, K., and Sipkin, S.: The Great Sumatra-Andaman Earthquake of 26 December 2004, Science, 308, 1127–1133, https://doi.org/10.1126/science.1112250, http://science.sciencemag.org/content/308/5725/1127, 2005.

Liang, W.-Y., Hsieh, T.-J., Satria, M. T., Chang, Y.-L., Fang, J.-P., Chen, C.-C., and Han, C.-C.: A GPU-Based Simulation of Tsunami 35   Propagation and Inundation, pp. 593–603, Springer Berlin Heidelberg, Berlin, Heidelberg, https://doi.org/10.1007/978-3-642-03095-6_56, https://doi.org/10.1007/978-3-642-03095-6_56, 2009a.

Liang, W.-Y., Hsieh, T.-J., Satria, M. T., Chang, Y.-L., Fang, J.-P., Chen, C.-C., and Han, C.-C.: A GPU-Based Simulation of Tsunami Propagation and Inundation, in: Algorithms and Architectures for Parallel Processing, edited by Hua, A. and Chang, S.-L., pp. 593–603, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009b.

Liu, P. L., Woo, S.-B., and Cho, Y.-S.: Computer programs for tsunami propagation and inundation, Cornell University, 1998.

5   Liu, X. and Guillas, S.: Dimension Reduction for Gaussian Process Emulation: An Application to the Influence of Bathymetry on Tsunami Heights, SIAM/ASA Journal on Uncertainty Quantification, 5, 787–812, https://doi.org/10.1137/16M1090648, https://doi.org/10.1137/16M1090648, 2017.

Løvholt, F., Pedersen, G., Harbitz, C. B., Glimsdal, S., and Kim, J.: On the characteristics of landslide tsunamis, Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 373, https://doi.org/10.1098/rsta.2014.0376, http://rsta.royalsocietypublishing.org/content/373/2053/20140376, 2015.

10 Lynett, P. J., Wu, T.-R., and Liu, P. L.-F.: Modeling wave runup with depth-integrated equations, Coastal Engineering, 46, 89–107, 2002.

Macías, J., Castro, M. J., Ortega, S., Escalante, C., and González-Vida, J. M.: Performance Benchmarking of Tsunami-HySEA Model for NTHMP's Inundation Mapping Activities, Pure and Applied Geophysics, 174, 3147–3183, https://doi.org/10.1007/s00024-017-1583-1, https://doi.org/10.1007/s00024-017-1583-1, 2017.

15 Mudalige, G., Giles, M., Reguly, I., Bertolli, C., and Kelly, P.: OP2: An active library framework for solving unstructured mesh-based applications on multi-core and many-core architectures, in: Innovative Parallel Computing (InPar), 2012, pp. 1–12, IEEE, 2012.

Mudalige, G. R., Reguly, I. Z., and Giles, M. B.: Auto-vectorizing a Large-scale Production Unstructured-mesh CFD Application, in: Proceedings of the 3rd Workshop on Programming Models for SIMD/Vector Processing, WPMVP '16, pp. 5:1–5:8, ACM, New York, NY, USA, https://doi.org/10.1145/2870650.2870651, http://doi.acm.org/10.1145/2870650.2870651, 2016.

20 Okada, Y.: Internal deformation due to shear and tensile faults in a half-space, Bulletin of the Seismological Society of America, 82, 1018–1040, http://www.bssaonline.org/content/82/2/1018.abstract, 1992.

O'Brien, L., Christodoulides, P., Renzi, E., Stefanakis, T., and Dias, F.: Will oscillating wave surge converters survive tsunamis?, Theoretical and Applied Mechanics Letters, 5, 160–166, 2015.

Poncet, R., Campbell, C., Dias, F., Locat, J., and Mosher, D.: A Study of the Tsunami Effects of Two Landslides in the St.
25 Lawrence Estuary, pp. 755–764, Springer Netherlands, Dordrecht, https://doi.org/10.1007/978-90-481-3071-9_61, https://doi.org/10.1007/978-90-481-3071-9_61, 2010.

Reguly, I. Z., László, E., Mudalige, G. R., and Giles, M. B.: Vectorizing Unstructured Mesh Computations for Many-core Architectures, in: Proceedings of Programming Models and Applications on Multicores and Manycores, PMAM'14, pp. 39:39–39:50, ACM, New York, NY, USA, https://doi.org/10.1145/2560683.2560686, http://doi.acm.org/10.1145/2560683.2560686, 2007.

30 Salmanidou, D. M., Guillas, S., Georgiopoulou, A., and Dias, F.: Statistical emulation of landslide-induced tsunamis at the Rockall Bank, NE Atlantic, Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 473, https://doi.org/10.1098/rspa.2017.0026, http://rspa.royalsocietypublishing.org/content/473/2200/20170026, 2017.

Salmanidou, D. M., Georgiopoulou, A., Guillas, S., and Dias, F.: Rheological considerations for the modelling of submarine sliding at Rockall Bank, NE Atlantic Ocean, Physics of Fluids, 2018.

35 Satria, M. T., Huang, B., Hsieh, T.-J., Chang, Y.-L., and Liang, W.-Y.: GPU acceleration of tsunami propagation model, IEEE Journal of Selected Topics in Applied Earth Observations and remote Sensing, 5, 1014–1023, 2012.

Schling, B.: The Boost C++ Libraries, XML Press, 2011.

Stefanakis, T. S., Contal, E., Vayatis, N., Dias, F., and Synolakis, C. E.: Can small islands protect nearby coasts from tsunamis? An active experimental design approach, in: Proc. R. Soc. A, vol. 470, p. 20140575, The Royal Society, 2014.

Synolakis, C. E.: The runup of solitary waves, Journal of Fluid Mechanics, 185, 523–545, https://doi.org/10.1017/S002211208700329X, 1987.

5 Tavakkol, S. and Lynett, P.: Celeris: A GPU-accelerated open source software with a Boussinesq-type wave solver for real-time interactive simulation and visualization, Computer Physics Communications, 217, 117 – 127, https://doi.org/https://doi.org/10.1016/j.cpc.2017.03.002, http://www.sciencedirect.com/science/article/pii/S0010465517300784, 2017.

The HDF Group: Hierarchical data format version 5, http://www.hdfgroup.org/HDF5, 2000-2010.

Titov, V. V. and Gonzalez, F. I.: Implementation and testing of the method of splitting tsunami (MOST) model, Tech. rep., NOAA Technical
10   Memorandum ERL PMEL-112, 11 pp UNIDATA, 1997.

Vater, S. and Behrens, J.: Well-balanced inundation modeling for shallow-water flows with discontinuous Galerkin schemes, in: Finite volumes for complex applications VII-elliptic, parabolic and hyperbolic problems, pp. 965–973, Springer, 2014.

Yusuke, O., Fumihiko, I., and Daisuke, S.: Near-field tsunami inundation forecast using the parallel TUNAMI-N2 model: Application to the 2011 Tohoku-Oki earthquake combined with source inversions, Geophysical Research Letters, 42, 1083–1091,
15   https://doi.org/10.1002/2014GL062577, https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2014GL062577.

Zhang, Y. J. and Baptista, A. M.: An Efficient and Robust Tsunami Model on Unstructured Grids. Part I: Inundation Benchmarks, Pure and Applied Geophysics, 165, 2229–2248, https://doi.org/10.1007/s00024-008-0424-7, https://doi.org/10.1007/s00024-008-0424-7, 2008.