Response to Reviewer #2

We want to thank the reviewer for using their precious time to check our paper and for giving useful comments to improve our work. Please find below our replies to your comments.

General Comments

Q1. The paper presents an operational model for the fast simulation of the generation, propagation and inundation of tsunamis in wide areas by exploiting modern multiGPU hardware. The model is tested and compared under a real tsunami scenario, obtaining a nice performance results from the operational point of view. The implementation of this operational model on a cluster of multiGPU computers involves the suitable integration of numerical schemes (MOC with dimensional splitting to solve spherical SWE for the Tsunami propagation, Surface Gradient Method to solve the cartesian SWE for inundation areas, ...) and computing techniques (quadtree-based mesh refinement to save resources, Hilbert Space-filling curves to preserve locality in the parallel partitioning, CUDA for GPU programming and MPI for remote communication, overlapping the computation in GPU and the generation of output files and rendering in CPU, etc.) to obtain an efficient complete CPU-multi-GPU operational model for Tsunami forecasting. This model would make it possible a very fast simulation which can help in the early identification of the tsunami consequences. In my opinion, the techniques which are presented and the scientific data which are included are coherent and relevant and can be useful to scientists working in this area because all the approaches and techniques are devised in conjunction to perform very quickly realistic simulations. Although the paper is well organized and written, the reading of several pieces of the sections which explain the multiGPU implementation is not easy to understand and several implementation decisions which are presented are not clear. Moreover, in Section 5, I think that the use of technical and English language should be checked (several corrections are included in the Section of Technical Corrections).

A1. We appreciate your kind description of our work. We considered your suggestion to check the use of technical English in section 5 and made the section about multi-GPU implementation more clear where possible. We appreciate your technical corrections in this matter as well.

It should be noted that even though one of the key elements of our work is GPU computing, the scope of the journal is not this area. For this reason, we tried to find a balance giving an appropriate description of our implementation without getting into extensive details. We tried to focus on the model and on the simulation results.

Q2. On the other hand, to intend the validation of the operational model with a real tsunami scenario when the input data are not sufficiently accurate is very ambitious.

A2. In order to have more validation data we have now included two sections in the manuscript to show results of several standard benchmark problems. Not having accurate enough initial input data is always an issue for all tsunami simulation models. Currently, the best approach to validate models consists on comparing results with existing analytical solutions and experimental data. For this reason we follow the NOAA Technical Memorandum OAR PMEL-135 (Synolakis et al., 2007) where several standard benchmark problems (BP) are given. We also thank Reviewer #1 for this suggestion.

Section 3 presents the results for the analytical 2D Parabolic Bowl benchmark. A new section was added to the manuscript to include results for benchmark problem 9 (Section 6) while the rest of the benchmarks are located in the appendix. The benchmark problems added are:

- BP4, Solitary wave on a simple beach
- BP6, Solitary wave on a conical island
- BP7, The tsunami run-up onto a complex 3D beach
- BP9, Okushiri Island Tsunami

Specific Comments

Q3. In Section 1, it would be interesting to include a comparison with previous works related with the multiGPU simulation of tsunamis to obtain faster-than-real-time results.

A3. We appreciate the suggestion; however, we consider that this kind of comparison would be unfair to do. For instance, each model utilizes different numerical schemes; the machines used might have different specifications; the domain used in each case might be different and in our case, we used an AMR-like technique for mesh refinement when grid nesting is more common.

However, we have included as references in section 1, other works that use GPU in their simulations. Two examples are:

- Vazhenin, A., Lavrentiev, M., Romanenko, A. and Marchuk, A.: Acceleration of tsunami wave propagation modeling based on re-engineering of computational components, International Journal of Computer Science and Network Security, 13, 32–70, 2013.
- 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 activitie, Pure and Applied Geophysics, 174, 3147–3183, 2017.

Additionally, we have included new benchmark results in the manuscript (Section 6 and Appendix) that are considered standard in the tsunami field. Using the data presented in the National Tsunami Hazard Mitigation workshop (NTHMP, 2012), error comparison with results of other models was included in our discussion and figures when available.

Q4. In Section 5.1., the description of the configuration of the main CUDA kernels (second paragraph of the section 5.1.) is not easy to understand. A graphical description of the configuration and a description of the calculations assigned to each CUDA thread (relating this section with section 3 and 4) would be very useful to understand it.

A4. A graphical description of the CUDA kernel has been added to the manuscript (Fig. A).

The grid is composed of 16 CUDA blocks in *y*-direction, each with 4 threads for 64 threads in total. In *x*-direction, the grid has one CUDA block with 64 threads.

One CUDA block processes a portion equal in size of the mesh block. One CUDA thread computes one mesh block cell. The specific calculation varies depending on the block type (*Wet*, *Dry*); however, the configuration remains the same. In both cases, threads compute the governing equations described in section 3.1. The main difference occurs in the case of a *Dry* block; in this case, cells that represent land or coastline compute a reflective wall boundary.

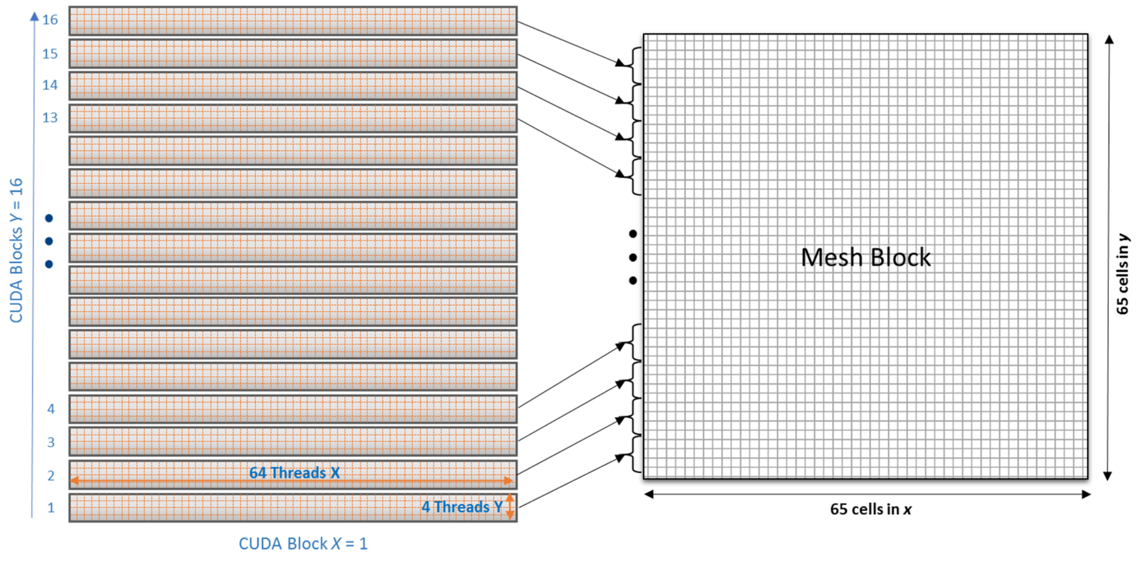

Fig. A Mesh block computation using CUDA kernels. Each CUDA block is made of 64×4 threads and computes a portion of the mesh block. One CUDA thread computes one mesh block cell

Q5. The specialized kernel types presented in section 5.1.2. could affect the load balancing between GPUs because the computational execution cost of each kernel type on a mesh block would be possibly different. I do not know if this fact is taken into account for the considerations included in section 5.2.

A5. Thank you for pointing this detail. This fact is taken into account during the load balancing, by assigning a different weight to the space-filling curve (SFC) based on the block type. This was not mentioned explicitly in the manuscript. A mention to this has been added in section 5.1.3 where SFC weights are discussed with sub-cycling.

Q6. I think it would be interesting to report graphically execution times for each particular GPU in order to evaluate the effectiveness of the domain partitioning and even to rethink the approach by designing a dynamic load balancer.

A6. Without being a detailed dynamic load balancer, our model includes this feature. During the mesh generation, blocks are assigned a different weight based on its type and the sub-cycling number. This weight is used in the space-filling curve to find a good domain partition. Not being a static process, this means that if a new domain mesh is required, the program will balance the new load.

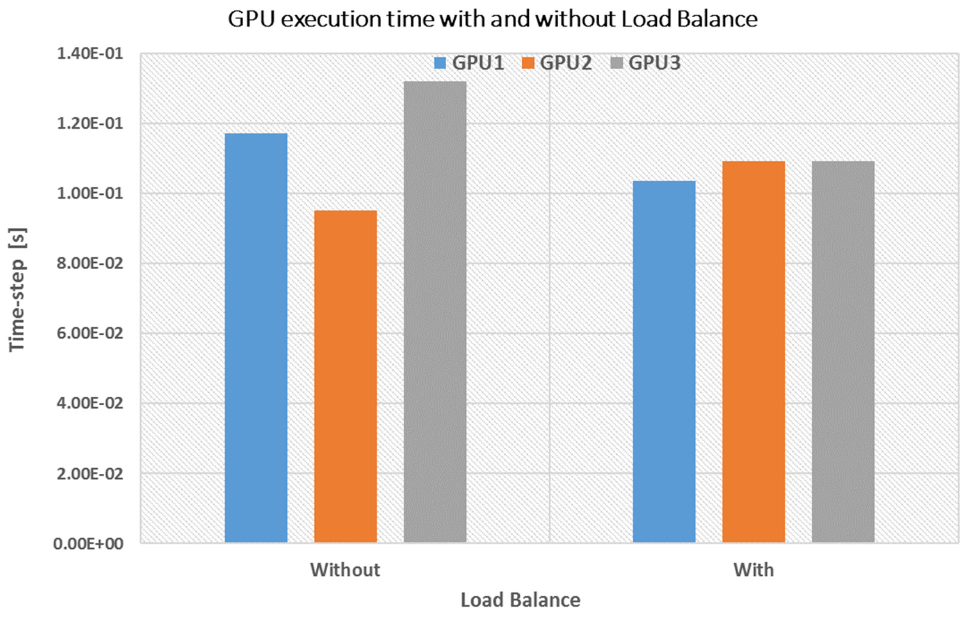

Fig. B GPU execution time with and without load balance

Evidently, the load balance is problem dependent. However, we include a chart (Fig. B) with the balancing results for our Indian Ocean domain using 4 focal areas. The effect of including the load balance can be seen on the right side of the chart. All GPUs spend almost the same time to execute a time-step. This avoids large overheads created by one GPU idling waiting for another to complete the tasks.

Q7. In Section 5.3.2., the configuration for the network which interconnects the TeslaP100- based nodes and the Tesla K80 nodes should be included to analyze Fig. 15.

A7. In the case of *Tsubame 3.0,* there are four Tesla P100 GPUs per node and the network is Intel Omni-Path HFI 100Gbps. In the case of the K80 machine used, there are four cards in one node (eight GPU in total), connected through PCI-Express 3.0. These network configurations have been added to the manuscript.

Q8. In Section 5.3.2. and in the Conclusions, authors underline evidences about the wall clock time and the speedup which are obtained with 3 GPUs. However, the particular performance results for 3 GPUs are not reported and they are not included in Figure 15.

A8. Thank you for noticing this detail. The runtime for 3 GPUs with K80 cards is 39.96 min and 12.1 min with P100 cards. These values have been now reported in the manuscript in section 5.3.2. Additionally, they have been included in Figure 15; the modified figure can be seen in Fig. C.

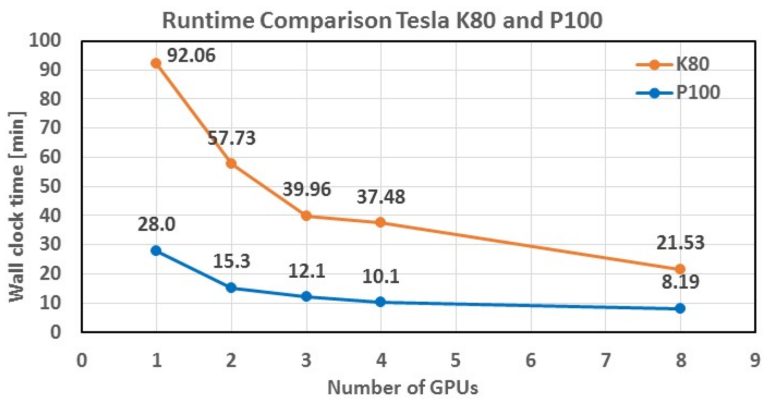

Fig. C Wall clock comparison of 10-hour simulation on Tesla K80 and Tesla P100

Q9. In Section 5.3.2., absolute performance measures on one GPU for the main kernels (it can be obtained by using the Nvidia CUDA profiler) could be useful to evaluate the efficiency of the CUDA implementation.

A9. We measured the FLOP/s performance of the main kernels for one GPU. The results obtained are shown in Table A, where *Inund* stands for Inundation kernel, *Wall* stands for the wall kernel, *Wet* for the *Wet* kernel and *X* and *Y* for the direction of the computation equivalent to longitude and latitude respectively.

Kernel	GFLOP/S
WallX	549.57
WallY	549.56
WetX	706.98
WetY	712.51
Inund	87.12

Table A. Kernel performance for one GPU in Giga FLOP per second

Q10. In section 6.1.2., the figure 18 Top which presents the RIMES results for Hambantota Inundation is not introduced textually in page 31, line 18. On the other hand, it is difficult to compare visually the RIMES and TRITON G inundation maps if the colours are used in a completely different manner for each map.

A10. We reordered and modified the paragraph in page 31 to introduce textually RIMES' figure.

Additionally, we modified TRITON-G figure's color scale to make it match better with RIMES' scale. The result is shown in Fig D.

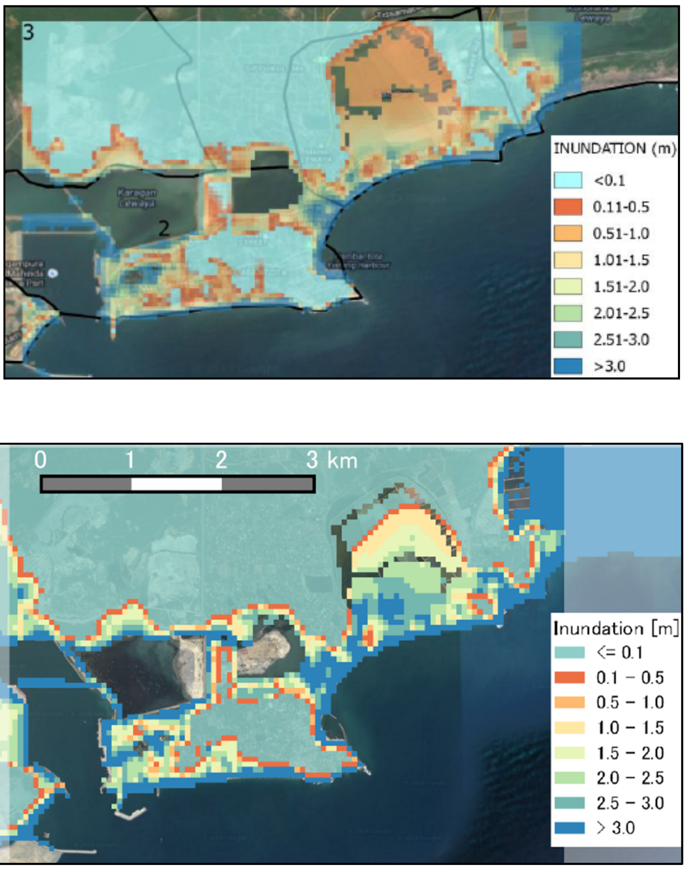

Fig. D Inundation comparison for Hambantota, Sri Lanka. Top: RIMES model. Bottom: TRITON-G model.

Technical corrections

Q11. - Page 13, Section 4.1, line 7: "blocks close to the coast until reaching a target ..."

A11. Changed sentence accordingly.

Q12. - Page 15, Line 7,8: The sentence is not clear.

A12. Sentences were rewritten to make them more clear.

> “Additionally, all dry blocks at Level 7 (highest resolution) that are inside a FA are considered inundation areas. This implies that run-up is computed on the coastlines instead of using a reflective boundary.”

Q13. - Page 17, Line 20; The sentence is not easy to understand.

A13. Sentence rewritten.

> “CUDA provides kernels as the way to define functions that are executed in parallel on GPU.”

Q14. - Page 17, Line 21: "organized in a grid of blocks of CUDA threads ..."

A14. Changed sentence accordingly.

Q15. - Page 18, Line 8: "branch diversion" is not the usual term. "branch divergence" is more frequent in this context.

A15. Thank you for the suggestion. Changed sentence accordingly.

Q16. - Page 18, Line 12: A comma in the sentence after "speed up" would help in order for it to make sense.

A16. Changed sentence accordingly.

Q17. - Page 18, Line 18: "... This way the kernels can be launched ...".

A17. Changed sentence accordingly.

Q18. - Page 19, Line 5: "... is illustrated in Fig. 7 ...".

A18. Changed sentence accordingly.

Q19. - Page 23, Line 5: "... was introduced in order ..."

A19. Changed sentence accordingly.

References

NTHMP: NTHMP, 2012. National Tsunami Hazard Mitigation Program (NTHMP). 2012. Proceedings and Results of the 2011 NTHMP Model Benchmarking Workshop, Department of Commerce/NOAA/NTHMP; NOAA Special Report., 2012.

Synolakis, C. E., Bernard, E. N., Titov, V. V., Kânolu, U. and González, F. I.: Standards, criteria, and procedures for NOAA evaluation of tsunami numerical models., NOAA., 2007.

Corollary

Author's comment: Additional to all the reviewers' suggestions, we decided to remove the paragraph about the circular shoal benchmark in page 26, from line 11 to 17. With the introduction of several new benchmark problems (Reviewer #1 Question #7) and the modifications to the original manuscript, it felt unnecessary to keep this reference since the new results covered far more than what this benchmark offered.