Distributed nested streamed models of tsunami waves

Kensaku Hayashi and Alexander Vazhenin*

Department of Information Systems,
University of Aizu,
Aizuwakamatsu, Japan
Email: m5161111@gmail.com
Email: vazhenin@u-aizu.ac.jp
*Corresponding author

Andrey Marchuk

Tsunami Laboratory,
ICMMG SB RAS,
Novosibirsk, Russia
Email: mag@omzg.sscc.ru

Abstract: This research focuses on designing a high-speed scheme for tsunami modelling using nested computing. Computations are carried out on a sequence of grids composed of geographical areas with resolutions where each is embedded within another. This decreases the total number of calculations by excluding unimportant coastal areas from the process. The paper describes the distributed streaming computational scheme allowing for flexible reconfiguration of heterogeneous computing resources with a variable set of modelling zones. Computations are implemented by distributing these areas over modelling components and by synchronising the transitions of boundary data between them. Results of numerical modelling experiments are also presented.

Keywords: Tsunami modelling; nested grids; distributed systems; coarse-grained parallelisation; streaming computing; communicating processes; process synchronisation; task parallelism; programming model; component-based software engineering.

Reference to this paper should be made as follows: Hayashi, K., Vazhenin, A. and Marchuk, A. (xxxx) 'Distributed nested streamed tsunami waves', *Int. J. Computational Science and Engineering*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Kensaku Hayashi received his MS in Computer Science and Engineering from the University of Aizu (Japan) in 2014. His research interests include parallel architectures, algorithms and tsunami modelling. Currently, he is a Doctoral student in the Active Knowledge Engineering Laboratory at the University of Aizu.

Alexander Vazhenin received his PhD in Computer Science from the Institute of Informatics Systems of the Siberian Division of the Russian Academy of Sciences in 1993. His research and educational interests include parallel architectures, algorithms and programming tools, tsunami modelling, visual and multimedia and Internet technology. Currently, he is a Professor at the University of Aizu, Japan.

Andrey Marchuk received his PhD in Numerical Mathematics from the Institute of Pure and Applied Mechanics SB RAS in 1981. In 2000, he was awarded Doctor of Physics – Mathematical Sciences in Numerical Modelling from the Institute of Computational Mathematics and Mathematical Geophysics SB RAS. His research interests include numerical modelling, tsunami waves kinematics and dynamics, inverse problems for tsunamis, and geographic information systems. Currently, he is a leading researcher at the Institute of Computational Mathematics and Mathematical Geophysics SB RAS and Novosibirsk State University, Russia.

This paper is a revised and expanded version of a paper entitled 'Cloud-based pipelined nested tsunami modeling' presented at the 28th International Ocean and Polar Engineering Conference, Sapporo, Japan, 10–15 June 2018.

1 Introduction

The threat of tsunami (tidal wave) requires a study of the impact of such events at different space scales. There are many problems related to tsunamis and these need to be solved to reduce their disastrous effects. The two main issues to be addressed are:

1 real-time tsunami warnings

2 long-term hazard assessment.

Real-time tsunami warning systems are designed to predict the basic parameters of tsunami waves based on time (Wei et al., 2008; Titov et al., 2016a; Meenakshi and Rodrigues, 2014). Long-term hazard assessment and tsunami inundation mapping is required to support preparedness measures and effective disaster reduction (Titov et al., 2016b; Kaistrenko, 2011). Both approaches to the numerical modelling of tsunami wave propagation consider it a computationally intensive problem requiring the acceleration of calculations through parallel processing.

More reliable results from tsunami propagation can be obtained by improving the resolution of a computational grid (tens of metres) especially for near-coast areas. Taking account of stability conditions, the modelling process should also be implemented with a small time step. Doing so will result in a significant increase in the numerical calculation time that is inadmissible in real-time calculations. Therefore, it is necessary to conduct such calculations with the use of computational grids whose spatial step decreases when approaching the coast.

The usage of nested grids allows improvements and solutions to this problem. This means that the first stage of simulation is made on a coarse calculation grid (with a cell size of several hundred metres). Then, as the coastline is approached, a more refined calculation grid is used. In this case the new grid covers only a small subarea of the initial simulation domain. Specifically, the numerical method MOST was designed for the splitting of difference schemes in coordinate directions with a possibility of using adaptive grids that become more refined in domains of decreasing depth (Titov, 1990).

The main goal of the presented work is to design a distributed tsunami modelling infrastructure to support high-speed tsunami modelling on systems with limited computational resources. Usually, these systems consist of several personal computers supported by standard accelerators like GPU and FPGA boards. Indeed, this situation is typical for laboratory level numerical experiments. The system should be capable of obtaining a detailed distribution of the expected tsunami heights simultaneously along the coast of several bays or ports.

An important intention is to provide high reusability of experimental data so that modelling results obtained for deep ocean areas can be applied to coastal areas, for example, for investigation of the protection properties of the artificial underwater object's (see Hayashi et al., 2017). The computational scheme is realised by distributing bathymetries over computational resources and synchronising the processing of each area using data buffering at boundaries between areas. The paper also describes applying this scheme to server-based computations to create a flexible and reconfigurable computational scheme for a changeable set of modelling areas.

The rest of the paper is organised as follows. In Section 2, the relevant work progress is analysed with reference to high-performance tsunami simulations using new and modern computing systems based on heterogeneous computing paradigms. Section 3 explains the mathematical model for simulating wave propagation and data source sets needed for numerical modelling, as well as the nested multi-grid algorithm for tsunami propagation computation from the initial source to the coastline involving scale switching. In Section 4, we discuss the main elements of the designed computational architecture and the synchronisation protocols between the system components. The results of the numerical experiments are presented in Section 5 and discussed in Section 6. Finally, we conclude with remarks and comments about future work.

2 Related works

Many different works exist related to different aspects of tsunami modelling. The presented review is oriented to analyse the current state of parallel implementations of modelling algorithms. Accordingly, we distinguish the fine-grained parallelisation providing calculations on the whole computational area, and coarse-grained (nested) parallelisation providing calculations on the set of these areas. We also focus on the GPU and FPGA-based parallelisation that can be implemented as an acceleration board to the standard PC architecture. These modules can be used as constructive elements for the coarse-grained computational schemes.

A review of the coarse-grained algorithms focuses on nested computing in which computations are carried out on a sequence of gridded geographical areas with various resolutions, where one is embedded into another. Special attention is paid to approaches for transition of boundaries between modelling zones.

2.1 Fine-grained parallelisation

Cai and Langtangen (2008) suggested that complex mathematical models and high mesh resolution should only be used when necessary. They have developed a parallel hybrid tsunami simulator based on mixing different models, methods and meshes. This simulator was implemented using object-oriented techniques, allowing the easy reuse of existing codes. Here, high performance was not the main goal of this research, but to focus on combining various approaches to develop high-quality hybrid tsunami models.

Sottile et al. (2013) presented preliminary work examining a programming methodology that provides

Fortran programmers with access to GPU architectures. The transformations in ForOpenCL are based on a simple mapping from Fortran to OpenCL. Using a stencil code solving the shallow-water fluid equations, it was shown that the performance of the ForOpenCL compiler-generated transformations is comparable with that of hand-optimised OpenCL code.

Vazhenin et al. (2013) described a set of tsunami wave modelling engines designed for several programming platforms, including OpenMP, CELL architecture and GPUs, allowing the flexible usage of available computational resources. This paper also included an analysis of the initial and output tsunami data, code design techniques, as well as the results of some numerical experiments and validation procedures. In the presented work, the code of these engines is used to create a reconfigurable and scalable structure for the nested tsunami modelling.

Imamura et al. (2006) developed a tsunami software package (TUNAMI-N1) with the staggered leapfrog scheme. Gidra et al. (2011) evaluated parallelised TUNAMI-N1 code by CUDA on NVIDIA QUADRO FX 1700. They reported results on various sizes of ocean bathymetry data sets for 7,200 time-steps. For $1{,}040 \times 668$ knots, they obtained a 5.86x speed-up in comparison with sequential computation on a single processor.

Acuna and Aoki (2014) used Tesla M2050 GPU to numerically solve the shallow water equations for tsunami simulation. They used a solution based on the CIP-CSL2 semi-Lagrangian scheme and simulated the tsunami over a large grid covering the entire Pacific ocean using a Tsubame 2.0 system with multiple GPUs. Fujita (2015) also reported his accelerated tsunami simulation on FPGA. He manually extracted large data flow graphs from the program and compiled it into FPGA circuits. The size of the computation grid is $1{,}040 \times 668$ and the simulation is conducted in 7,200 steps defining one time-step as 1 second. It was shown that an FPGA tsunami simulation is 46 times faster than an Intel core i7 processor at 2.93 GHz.

Kono et al. (2018) developed tsunami propagation code based on the MOST algorithm, and implemented different parallel optimisations for GPU and FPGA. They obtained good performance from the OpenCL kernel, on which was implemented a tsunami simulation on an AMD Radeon 280X GPU and on FPGA using OpenCL.

2.2 Computations on nested grids

Initially, nested grids were used for the same purpose in the calculation domain regions where the propagating wave becomes shorter and higher. First, the method of nested grids employed was as follows: at some moment in time all parameters of the flow were 'frozen' in a subdomain. Then an interpolation to a more refined calculation grid in this subdomain was made. Calculations were continued in the nested subdomain on the new grid, and possibly with a new time step. One can continue the calculation on the same computer (terminating the calculation in the 'large' domain), or transfer the data to another calculation module (Hasan et al., 2015). Nested grids have been used by many authors to increase the resolution of calculations in some subdomains to estimate the wave parameters (Shigihara and Fujima, 2012; Son et al., 2011; Karim et al., 2014; Gusyakov et al., 2008).

These methods differ only in the ways of transferring some of the parameters of a propagating wave (water surface height and water flow velocity components) from the entire calculation domain to a subdomain with a more detailed calculation grid. In some cases a different mathematical model is used in the subdomain and adaptive grids with triangular cells are used by some authors (Bader et al., 2008; Harig et al., 2008). This approach complicates the calculation algorithm, but it does not speed up the calculations since a step-by-step calculation is made in the entire calculation domain that includes the subdomain (subdomains). However, some authors do not specify how the data are transferred from the 'large' domain to a subdomain.

Köstler et al. (2017) described a prototype implementation in Scala for a framework that enables abstract descriptions of partial differential equations (PDEs), their discretisation and their numerical solution via multigrid algorithms. Two test problems illustrate the potential of this approach for both CPU and GPU target platforms. Indeed, this approach seems to be attractive but it needs to be adopted to calculations with real data.

Baba et al. (2016) improved the tsunami simulation code JAGURS for a large-scale, high-speed tsunami prediction in the Nankai trough off the coast of Japan. They optimised the code for velocity update and communications using a three-dimensional torus network. Linear scaling was obtained for the full system capability of the K computer (82,944 nodes) in a strong test that used 100 billion finite-difference grid points. The performance on the K computer reached up to 1.2 petaflops (11.5% of peak speed). Communications were optimised for a three-nested-grid model consisting of 0.68 billion-grid points. Grid resolution of about 5m was obtained in the area 180 km $\times$ 120 km. They successfully implemented a large-scale tsunami simulation using this model that ran in about 30 percent of real time.

Hayashi et al. (2018) proposed a computational pipelined scheme of tsunamimodelling implemented on a single computer. The possibility of storing intermediate data on cloud-based computers is also shown. The presented paper extends this approach by adopting it to distributed multicomputer systems with the data-driven streamed communication mechanism.

3 Tsunami modelling scheme

3.1 Basic tsunami wave propagation algorithm

The best known and widely used tsunami modelling tools are TUNAMI and MOST. They calculate the long wave

propagation in the ocean using the so-called shallow-water differential equations:

$$\begin{cases} H_t + (uH)_x + (vH)_y = 0 \\ u_t + uu_x + vu_y + gH_x = gD_x \\ v_t + uv_x + vv_y + gH_y = gD_y \end{cases} \tag{1}$$

where $H(x,y,t) = \eta(x,y,t) + D(x,y,t)$, η is the water surface displacement, D is depth, $u(x,y,t)$ and $v(x,y,t)$ are the horizontal flow velocity components along the axis x and y, and g is acceleration of gravity. As it follows from the shallow-water model, the tsunami propagation velocity does not depend on its length and is expressed by the so-called Lagrange formula $c = \sqrt{g(D+\eta)}$ that plays a key role in long wave (tsunami) kinematics. The horizontal flow velocity is calculated using wave amplitude and water depth in the following formula

$$|\vec{u}| = \sqrt{u^2 + v^2} = \eta\sqrt{\frac{g}{D}} \tag{2}$$

The numerical algorithm is based on splitting the difference scheme, which approximates equation (1) with spatial directions as well as permits setting boundary conditions for a finite difference boundary value problem using a characteristic line method. Tsunami (MOST) is used for shallow wave equation (1) and comprises a consecutive numerical solution of two one-dimensional systems of equations:

$$\begin{cases} V_t + UV_x = 0 \\ U_t + UU_x + gH_x = gD_x \\ H_t + (UH)_x = 0 \end{cases} \quad \begin{cases} V_t + VV_x + gH_x = gD_x \\ U_t + VU_x = 0 \\ H_t + (VH)_x = 0 \end{cases} \tag{3}$$

Eigenvalues of equation (3) are real and different, and the system can be rewritten as

$$\begin{cases} V'_t + \lambda_1 V'_x = gD_x \\ P_t + \lambda_2 P_x = gD_x \\ \varrho_t + \lambda_3 \varrho_x = 0 \end{cases} \tag{4}$$

where $\lambda_1 = U$, $\lambda_{2,3} = U \pm \sqrt{gH}$ are eigenvalues, $V = V'$, $P = U + 2\sqrt{gH}$, $\rho = U - 2\sqrt{gH}$ are the Riemann invariants. The characteristic line method has been used to set the simple boundary conditions for the system. The open boundary conditions that provide wave exit from the computational domain without reflection can be written as $V' = 0, R = \pm 2\sqrt{gD}, R = U \pm 2\sqrt{gH}$ (i.e., $R = P$ or Q). And the wave reflection boundary conditions (on a coastline) are expressed as $V' = 0, P = -Q$. For the numerical solution of the system, the following finite difference scheme is used:

$$\begin{aligned} &\frac{\overset{1}{W}{}_i^{n+1} - \overset{1}{W}{}_i^{n}}{\Delta t} + A\frac{\overset{1}{W}{}_{i+1}^{n} - \overset{1}{W}{}_{i-1}^{n}}{2\Delta x} \\ &\quad - A\Delta t\frac{A(\overset{1}{W}{}_{i+1}^{n} - \overset{1}{W}{}_i^{n}) - A(\overset{1}{W}{}_i^{n} - \overset{1}{W}{}_{i-1}^{n})}{2\Delta x^2} \\ &\quad = \frac{\overset{1}{F}{}_{i+1} - \overset{1}{F}{}_{i-1}}{2\Delta x} - A\Delta t\frac{\overset{1}{F}{}_{i+1} - 2\overset{1}{F}{}_i - \overset{1}{F}{}_{i-1}}{2\Delta x^2} \end{aligned} \tag{5}$$

where

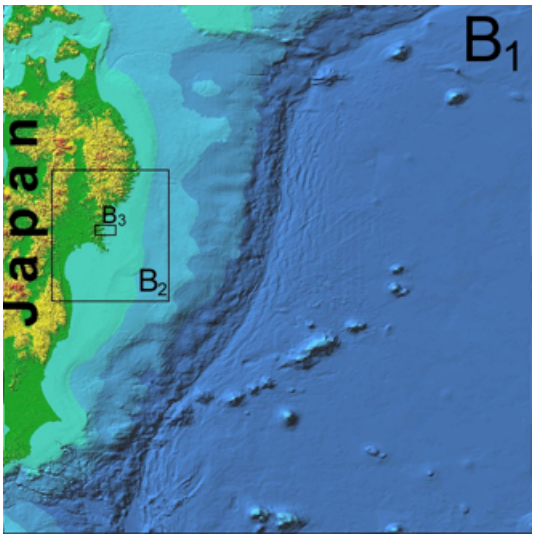
$$A = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}, \quad \overset{1}{W} = (V', P, \varrho), \ \overset{1}{F} = (0, gD_x, gD_x). \tag{6}$$

The stability condition of this scheme is $\Delta t \leq \Delta x/\sqrt{gH}$ requiring assignment of a smaller time step if a computational domain contains deep-water areas. As shown above, the digital bathymetry and initial water elevation (tsunami source) are important parameters for defining the behaviour of tsunami waves. In order to obtain more reliable results of the tsunami propagation (distribution of tsunami wave heights in a shelf zone), a rather small step in the computational grid (about tens of metres) is necessary. If we simulate the tsunami propagation within the whole area including both a source zone and sites of the coast, then we need to use this small spatial grid step. Furthermore, we will be compelled to calculate this with a small time step to improve stability. This will bring about a significant increase in duration of numerical calculations that is inadmissible in real-time calculations. The problem can be solved using the nested grid computation method described in the next subsection.

3.2 Scale switching nested computing

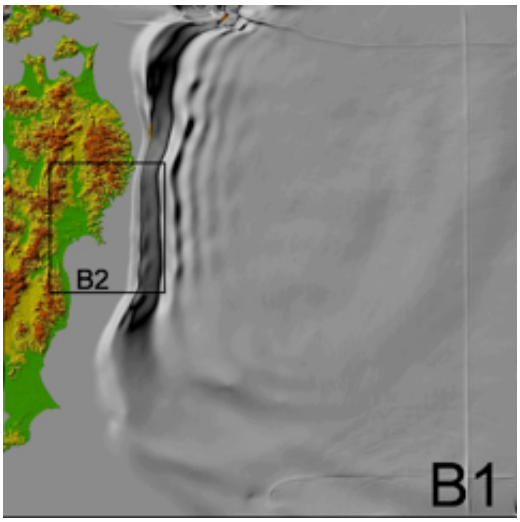
The presented algorithm provides consecutive calculations of the tsunami wave propagation in several computational domains, where each subsequent computational area is a subarea to the previous one, but with a smaller spatial step. Accordingly, we consider that calculations commence with obtaining so-called tsunami source data in which the initial vertical water surface displacement is reflected. Wave parameters are transferred to each subsequent subarea via boundary conditions while interpolating these data along the boundary on a smaller computational grid. Digital bathymetry sets were taken or developed using different sources. The scale-switching computations provide the tsunami wave propagation modelling from the initial source to the coastline. As shown in Figure 1, computations are carried out on a sequence of grids with various resolutions where one is embedded into another.

Figure 1 Nested grids (see online version for colours)



Consider computations in areas B1 and B2 (Figure 2). In the first stage of the numerical experiment, wave propagation was simulated in the 'large' computational domain, B1. In the whole process of computation, the wave parameters (amplitude, horizontal velocity components and geographical coordinates), buffering occurs at all grid points along the B2 boundary areas. The data output starts from the very first time step and continues with each time step. Data recording can be stopped when a tsunami wave has passed the right boundary of subarea B2 (at least the whole wave period). The gridded bathymetries of computational areas B1 and B2 may not be well correlated. This means that there are almost no grid points having the same geographic location. As already noted, the grid-step length in these two areas may be significantly different from one another.

Figure 2 Scale switching for B1 and B2 areas (see online version for colours)



Let us take one column of area B1 grid points, which are most closely situated to the right vertical boundary of B2. The latitudes of these B1 domain grid points are saved in 1D array $lat1i, i = 1, N$. Using linear interpolation the tsunami wave parameters at B2 boundary grid-points with the latitudes $lat2j, j = 1$,M are calculated.

The formulas for recalculating the wave parameters along the B2 domain boundary are as follows:

$$\begin{aligned} \eta 2_j &= \frac{\eta_{i+1}(lat2_j - lat1_i) + \eta 1_i(lat1_{i+1} - lat2_j)}{lat1_{i+1} - lat1_i}, \\ u2_j &= \frac{u_{i+1}(lat2_j - lat1_i) + u1_i(lat1_{i+1} - lat2_j)}{lat1_{i+1} - lat1_i} \\ &\quad \cdot \sqrt{\frac{D1_j}{D2_j}} \end{aligned} \tag{7}$$

Here the parameters $\eta 2i, u2j$ are related to the B2 computational area, and $\eta 1j, u1i$ are related to area B1. If time steps are different then wave parameters must also be interpolated with respect to time. Thus, after recalculating the flow parameters along B2 boundaries the calculation for this area can be provided. A more detailed description of this method can be found in Hayashi et al. (2015).

4 Nested modelling infrastructure and computational schemes

4.1 Common remarks

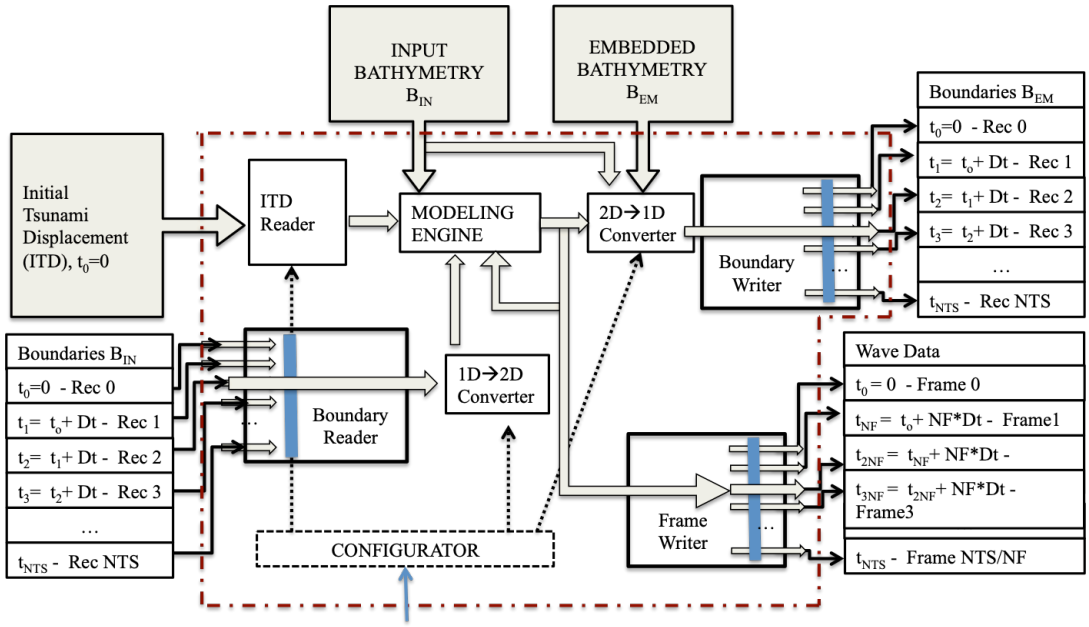
Component-based development embodies good software engineering practice. It is concerned with developing standardised components based on a component model, and composing these into application systems. Based on this approach and the scaling algorithm presented above, we developed a reconfigurable and scalable structure providing coarse-grained parallelisation of the nested tsunami modelling. Here we describe the system components and algorithms of functioning as well as show how to assemble the computational schemes.

4.2 Wave engine

The wave engine (WE-engine) is a component that calculates the tsunami wave parameters as well as forms the boundary data used for modelling on an embedded area. Figure 3 shows the structure of the wave engine. As shown in Figure 3, the input data are:

- the gridded bathymetry of modelling area $\boldsymbol{B_{IN}}$ with geographical coordinates are represented as a two-dimensional array, and each element of which is a value of the ocean depth
- initial tsunami displacement (ITD) is represented as two-dimensional arrays of size $\boldsymbol{B_{IN}}$ containing values of amplitude and horizontal flow velocity components

Figure 3 Structure of a wave engine (see online version for colours)



- the gridded bathymetry of embedded area $\boldsymbol{B_{EM}}$
- modelling parameters such as a time-step Δt, the model run length defined as a number of time steps (**NTS**) and number of time steps between two frames **NF**
- boundary values for $\boldsymbol{B_{IN}}$ represented as one-dimensional arrays as described in Section 3.

The output data of the WE-engine are:

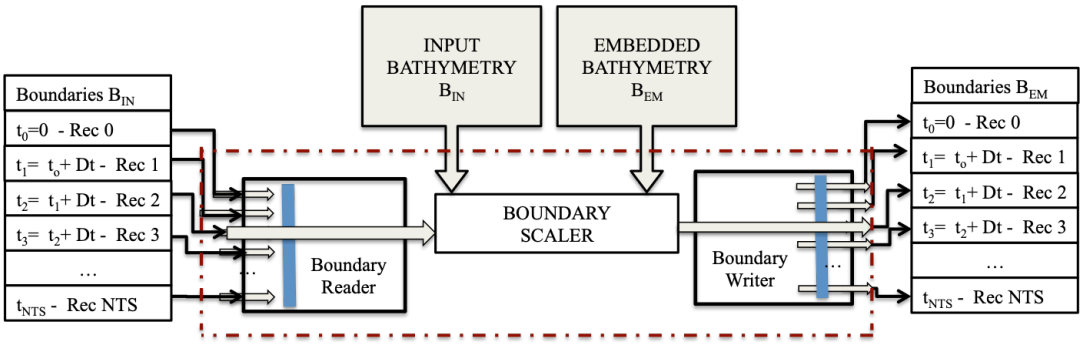
- wave data for all grid points of $\boldsymbol{B_{IN}}$ defining amplitude, horizontal velocity components and geographical coordinates;
- boundary values for $\boldsymbol{B_{EM}}$ extracted from the wave data.

The kernel of WE-engine is the modelling engine (ME-engine) calculating tsunami wave parameters. As pointed out above, the ME-engine can use different mathematical models of tsunami wave propagation as well as be realised based on parallel architectures, including CUDA GPU accelerators and FPGA arrays. Here we use the standard MOST algorithm providing calculations by equations (1)–(6). According to the modelling area (Figure 1), it is possible to distinguish types of WE-engines. In the initial stages of all modelling areas there is no tsunami source (the initial vertical water surface displacement). This appears in the deep-water area B1, and a tsunami wave is propagated to the coast over all nested areas. Therefore, the first type of WE-engine requires the tsunami source data identified in Figure 3 as the ITD. The outputs are tsunami wave parameters (wave data) and boundary data used for calculating the embedded area B_{EM}. For intermediate area B2, it is necessary to use boundaries from the external area B1, and it is not necessary to input ITD. That is the second type of WE-engine. The last area of this chain is the coastal area that does not need to output boundary data. This is considered to be the third type of WE-engine. Figure 4 indicates the main operations reflecting the iterative modelling process in the wave engine.

Figure 4 Modelling process in the WE-engine

```
/*------------------------------------*/
/*------ Procedure for Wave Engine ------*/
1. Read Number of Time Steps – NTS;
2. Read Number of Frames – NF;
3. Read WE; /*------ Define WE type ------*/
4. Read Bathymetry B_IN;
5. Read Bathymetry B_EM;
6. if (WE==0) Read Tsunami Sources for B_IN;
7. StF = NF; FC=0;
8. for (StepN=0; StepN < NTS; StepN++) {
9.    if (WE >= 1) {
10.      Wait Record_STEPN for Boundaries B_IN;
11.      Read Record_STEPN for Boundaries B_IN;}
12.   Run Modeling Engine for Bathymetry B_IN;
13.   if (StF==NF) {
14.      Write Frame_FC in the Wave Data File;
15.      FC = FC+1; StF = 0;}
16.   if (WE < 2) {
17.      Form Boundaries B_EM from Frame_STEPN;
18.      Write Record_STEPN for Boundaries B_EM;
19.       Open for reading Record_STEPN;}
20.   StF = StF+1;
      }
/*------------------------------------*/
```

Figure 5 Structure of a boundary engine (see online version for colours)



As shown in Figure 4, operations 1–6 are designed to obtain parameters defining the modelling process including the number of time steps NTS, number of frames NF, type of ME-engine WE, bathymetry B_{IN} parameters of the embedded bathymetry B_{EM} and initial vertical water surface displacement data ITD. Operation 12 calculates tsunami wave parameters for the current time step according to equations (1)–(6). As pointed out above, the modelling Engine can be realised using different architecture that includes CUDA GPU accelerators and FPGA arrays. Here we use our approach described in more detail in Vazhenin et al. (2013). During operations 13–15, the frame writer saves wave parameters for all grid points of the B_{IN}-area into the wave data. To reduce the volume of wave data saves only selected frames.

Boundary reader and writer are implemented to synchronise communications and data exchange between engines. Operations 18 and 19 provide outputs of boundaries to be shared with WE-engines. The stability of the data exchange is realised by blocking access to this record until all boundary data have been correctly saved. Boundary data are represented as one-dimensional arrays. Therefore, the 2D→1D adaptation extracts the boundaries for the embedded area B_{EM} from the 2D wave B_{IN}-data. This is the main part of operation 18.

Operations 10 and 11 are active for WE-engines of the second or third type. They provide input from the boundary data obtained from the outer area. Operations 10 and 11 support correct reading boundaries from the outer area. This is implemented by waiting for a record of boundaries corresponding to the current time step of modelling. The 1D→2D converter transforms the one-dimensional array of boundaries to the two-dimensional representation.

4.3 Boundary engine

Boundary engines (BE-engine) provide interpolation of the boundary transition between grids as well as a synchronisation protocol for data exchange between these components. As shown in Figure 5, the boundary interpolator is the main component implementing calculations according to equation (7). The other components are the boundary reader and writer implementing the same functions as in ME-engine.

Figure 6 Modelling process in the BE-engine

```
/*-------------------------------*/
/*-- Procedure for Boundary Engine --*/
1. Read Number of Time Steps - NTS;
2. Read Bathymetry B_IN;
3. Read Bathymetry B_EM;
4. for (StepN=0; StepN < NTS; StepN++)
     {
5.     Wait Record_STEPN for Boundaries B_IN;
6.     Read Record_STEPN for Boundaries B_IN;}
7.     Scale Boundaries B_IN for StepN;
8.     Write Record_STEPN for Boundaries B_EM;
9.     Open for reading Record_STEPN;
     }
/*-------------------------------*/
```

Figure 6 shows the main operations realising the scale-switching process in the boundary engine. Operations 1–3 are to obtain components needed to implement the scale-switching procedure including the number of time steps NTS, and the current and embedded bathymetries B_{IN} and B_{EM}. The boundary interpolator works during operation 7. The synchronised communication and data exchange between engines is organised via the protocol implemented in operations 5, 6, 9 and 10. They are the same as in the ME-engine. Operations 5 and 6 support correct reading boundaries from the upper area B_{IN}. Operations 7–9 provide outputs of scaled boundaries to be shared with WE-engines.

4.4 Distributed modelling schemes

The approach presented here allows for the creation of flexible and reconfigurable computational schemes with a variable set of modelling zones by sharing these data with other chains implementing modelling for other embedded coastal areas. The first scheme can be made using so-called horizontal direction in order to increase a number of embedded grids as shown, for example, in Figure 7. Importantly, all communications and data exchanges are implemented for each time step of modelling. Therefore, the processing of streamed data is implemented via WE and BE engines connected in series where the output of one element is the input of the next one.

Figure 7 Three-nested computational scheme

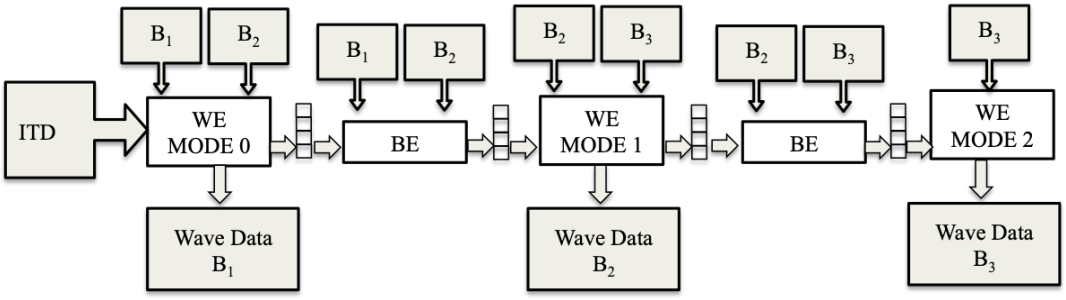
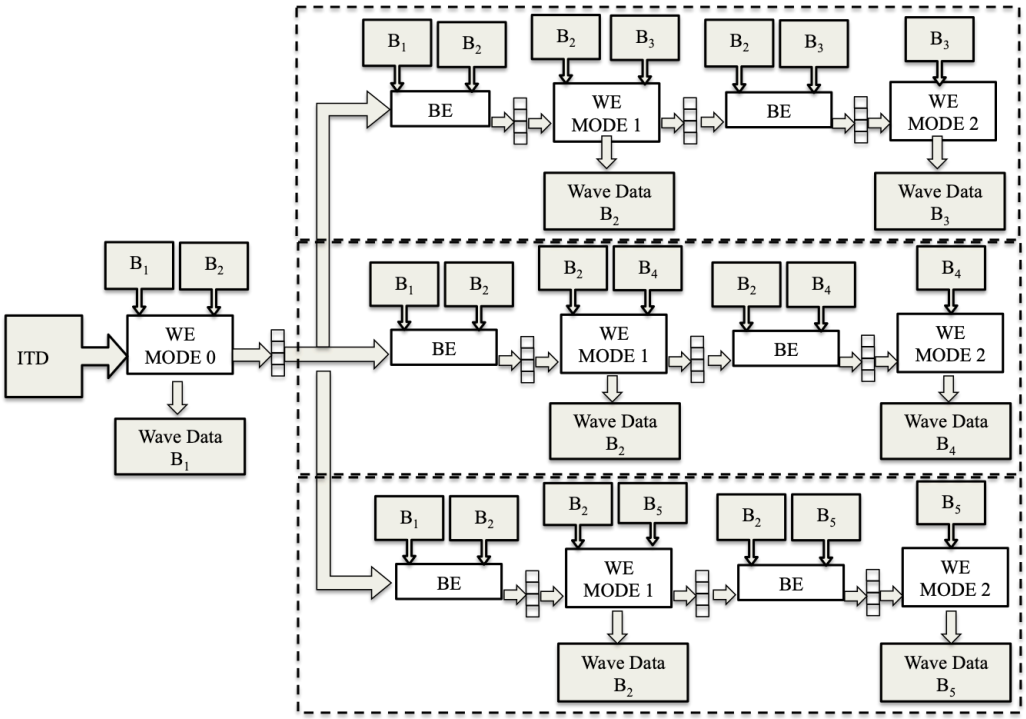


Figure 8 Calculations with three coastal areas



The extension of a number of coastal areas having high-resolution can also be organised in a so-called vertical direction. Figure 8 shows a type of calculation and resource distribution for parallel modelling in three areas that are embedded in area B2. Accordingly, WE-engines implement the same calculations of wave parameters, but form different boundaries defined by corresponding bathymetries.

Placing boundaries on the cloud-like storage allows for access to the data at any time as well as dynamically connecting important areas to a modelling process. Importantly, the high communication speed required cannot be reached by traditional cloud server organisation. It should be made in order to maintain high-speed communication channels.

5 Numerical experiments

To verify our approach, we implemented a set of numerical experiments in a multicomputer environment. The following four bathymetries were used in our experiment:

- The computational area B1 is the gridded 3,000 × 3,200 knots with the resolution of around 205 * 276 m. The B1 grid covers the geographic area from 140.0025° to 147.4539° E and from 34.02200° N up to 41.97305° N.
- The computational area B2 is 2,161 × 3,201 knots with the resolution of around 55 * 69 m covering a part of the Tohoku Area.
- The computational area B3 is the gridded 1,392 × 854 knots with the resolution of around 6.9 * 8.6 m. It covers the coast area of Onahama Bay in Fukushima Prefecture.
- The computational area B3′ is the gridded 2,148 × 1,074 knots with the resolution of around 13.5 * 17.2m. It covers the coast area of Oppa Bay in Miyagi Prefecture.

The information for each model is as follows:

- total number of steps $\boldsymbol{NTS}$: 15,000
- time corresponding to one step(sec): 0.25
- initial displacement of tsunami (m): 2.0
- interval of model output frame output (step) $\boldsymbol{NF}$: 200
- area corresponding to model 1: B1
- area corresponding to model 2: B2
- area corresponding to model 3: B3
- area corresponding to model 3′: B3′.

Firstly, we evaluated the computational scheme presented in Figure 3. As pointed out above, we used the MOST package as the basic code in our investigations. Originally, it was written in the FORTRAN language. Using its tsunami wave modelling code as the modelling Engine, we re-engineered it in order to create the WE-engine. To create the other version of the WE-engine, we also used the CUDA-based modification of the MOST code described in Vazhenin et al. (2013). The BE-engine was realised using the FORTRAN code. Experiments were implemented using one personal computer with multicore architecture. Its standard architecture was extended by two CUDA-boards. Characteristics of this PC are presented in Table 1.

Table 1 Calculation environment PC1

CPU	Intel(R) Core(TM) i7-7700K
Number of cores	4
Number of threads	8
Clock frequency	4.20 GHz
Memory	64 GB
GPU	GeForce GTX 1050 Ti (*2)
Number of GPU Cores	768
Core clock	1,290 MHz
GPU memory	4 GB

Table 2 Sequential tsunami modelling time using PC1

Name	*T1*	*T1′*	*T2*	*T2′*
MODEL1	FORTRAN	FORTRAN	CUDA	CUDA
TIME (min)	242.85	242.32	27.25	27.21
MODEL2	FORTRAN	FORTRAN	CUDA	CUDA
TIME (min)	157.28	158.37	32.18	32.45
MODEL3	FORTRAN	-	CUDA	-
TIME (min)	24.23	-	4.55	-
MODEL3′	-	FORTRAN	-	CUDA
TIME (min)	-	54.43	-	9.21
TOTAL	424.36	455.12	63.98	68.87
SPEED UP	1(T1)	1(T1′)	6.63(T1)	6.61(T1′)

Table 2 shows the results of numerical experiments with so-called sequential processing of areas B1, B2 and B3. This means that calculations in area B2 start after finishing a whole modelling process in area B1, and area B3 is processed after the modelling for area B2. This demonstrates an effect of the spatial acceleration of the modelling process using fine-grained GPU-parallelisation by CUDA. Speedups are calculated as the relation between the total execution time of sequential processing and the time obtained in the current experiment. This means, for example, that speedup is calculated as S = T1/T4 = 6.63 for column T4. Importantly, total time is measured for a whole process including all input/output operations.

Table 3 Streamed tsunami modelling time using PC1

Name	*T3*	*T3′*	*T4*	*T4′*
MODEL1	FORTRAN	FORTRAN	CUDA	CUDA
TIME (min)	244.58	244.65	40.47	40.16
MODEL2	FORTRAN	FORTRAN	CUDA	CUDA
TIME (min)	244.73	245.07	46.00	47.22
MODEL3	FORTRAN	-	CUDA	-
TIME (min)	244.85	-	46.35	-
MODEL3′	-	FORTRAN	-	CUDA
TIME (min)	-	245.43	-	47.5
TOTAL	244.85	245.43	46.35	47.5
SPEED UP	1.73(T1)	1.85(T1′)	9.16(T1)	9.58(T1′)

Table 4 Heterogeneous tsunami modelling using PC1

Name	*T5*	*T5′*	*T6*	*T6′*
MODEL1	FORTRAN	FORTRAN	CUDA	CUDA
TIME (min)	242.43	242.11	27.26	27.27
MODEL2	CUDA	CUDA	FORTRAN	FORTRAN
TIME (min)	242.7	243.28	156.55	161.64
MODEL3	CUDA	-	FORTRAN	-
TIME (min)	242.87	-	156.73	-
MODEL3′	-	CUDA	-	FORTRAN
TIME (min)	-	243.51	-	161.77
TOTAL	242.87	243.51	156.73	161.77
SPEED UP	1.75(T1)	1.87(T1′)	2.71(T1)	2.81(T1′)

Table 5 Optimised tsunami modelling using PC1

Name	*T7*	*T7′*
MODEL1	CUDA	CUDA
TIME	38.32	37.83
MODEL2	CUDA	CUDA
TIME	43.83	43.98
MODEL3	FORTRAN	-
TIME	44.25	-
MODEL3′	-	FORTRAN
TIME	-	56.58
TOTAL	44.25	56.58
SPEED UP	9.59(T1)	8.04(T1′)

In the next group of experiments, all boundary and wave engines were launched simultaneously. In this case, all engines are working in parallel to provide synchronisation and boundary exchange as described in Section 4. This produces the streaming computations in which calculations in the current area can be executed when at least one boundary record is ready for reading. In fact, this is streamed with an amount of stages defined by a number of modelling areas. Results presented in Table 3 demonstrate

additional performance in comparison with sequential computing.

Table 6 Calculation environment PC2

CPU	Intel(R) Core(TM) i7-3820
Number of cores	4
Number of threads	8
Clock frequency	3.60 GHz
Memory	16 GB
GPU	GeForce GTX 1050 Ti
Number of GPU cores	768
Core clock	1,290 MHz
GPU memory	4 GB

Table 7 Calculation environment PC3

CPU	Intel(R) Xeon(R) CPU E5-2650
Number of cores	8
Number of threads	16
Clock frequency	2.00 GHz
Memory	64 GB
GPU	GeForce GT 610
Number of GPU cores	48
Core clock	810 MHz
GPU Memory	1 GB

The concurrent process execution and resource management is implemented under the control of operating systems including processes that are executed on the GPUs. This may decrease the efficiency of computations due to an imbalance in using resources. As shown in Table 3, the efficiency of pipelining is reduced for the GPU-based engines because of the necessity of managing three heavy-computational processes on two GPU-boards, that are not well designed for multitask processing.

Table 4 demonstrates the possibility of heterogeneous processing by joining the CUDA- and FORTRAN-engines in a single computational scheme. Accordingly, the user can optimise the execution time by balancing the computational load between the engines. Assigning a CUDA-based engine for modelling the most heavy-computational area can do this, and useful when computational resources are limited. Usually, the area B_{IN} has a bigger size than the embedded area B_{EM} even if it has lower resolution. In our experiments, area B1 is the most difficult area. The results presented in Table 4 demonstrate the effect of using one CUDA-based engine for area B1 (cases T6 and T6′).

Table 5 shows the most balanced computational scheme in which each CUDA-based engine is used for modelling tsunami waves in areas B1 and B2 correspondingly. The calculations for the smaller area B3 are implemented using the FORTRAN WE-engine. It is possible to see that the execution time is comparable with the case when all CUDA-based engines are used (cases T4 and T4′). This situation is very useful when GPUs do not have enough memory to implement calculations for all areas.

The next group of experiments was implemented in a distributed environment according to the computational scheme shown in Figure 8. We used three computers connected locally via the standard TCP/IP protocol. The characteristics of two additional computers are described in Tables 6 and 7, correspondingly. For sharing boundary data, we developed a file server PC3 based on a network file system (NFS) that enables local users to access remote data and files in the same way as they are accessed locally (Tanenbaum and Bos, 2015).

Computer PC3 was used for this server creation as well as for the modelling for area B1. The client computer PC1 implemented calculations for areas B2 and B3, and computer PC2 was assigned to areas B2 and B3′. Calculations for all computers were initiated simultaneously.

The results of the experiments (Table 8) show that the proposed scheme allows for implementing tsunami modelling simultaneously for several coastal areas. It should also be noted that the calculator must be selected correctly to accelerate the calculation optimally. Therefore, a special resource management system will be useful in optimising computational load among disparate computers including predicting the efficiency of multi-core processing associated with a set of tasks with varied CPU and main memory, for example, as shown in Hasan et al. (2017).

6 Discussions

The described approach allows for joining bathymetries with non-proportional grid steps and inconsistent bottom reliefs in a whole computational scheme making it significantly different from traditional methods. Using this approach, higher model accuracy can be achieved, because boundaries are transferred between areas in all computational steps without any data loss during data transitions.

As shown in section 2.2, the calculations in the embedded (nested) areas are implemented after some selected computational steps by freezing the wave parameters, including surface displacement data as well as parameters defining horizontal water flow velocity. Based on the two-dimensional interpolation, calculations can be continued in embedded subareas having more detailed computational grids as well as different time steps. The novelty of the proposed approach is that calculations of hydrodynamic parameters can be started simultaneously in all computational domains and implemented with the whole time step. The one-dimensional interpolation simplifies calculations and makes it possible to reduce computational errors. Moreover, this approach allows the correction of values in water flow velocity in a case of mismatching bathymetric data between computational grids. The calculations can also be branched at the late stages of modelling. This allows one to also realise the old technology by starting the embedded process from the selected time step. Similar simultaneous calculations were realised in Baba et al. (2016) using the K supercomputer. The authors used the new JAGURS mathematical model that requires additional validation procedures.

Table 8 Streamed tsunami modelling time (min) using multi-computer with NFS

Name	*T8*	*T8′*	*T9*	*T9′*	*T9′**
MODEL1	CUDA	-	CUDA	-	CUDA
PC	3	-	3	-	3
TIME	33.73	-	29.6	-	29.38
MODEL2	CUDA	CUDA	FORTRAN	FORTRAN	FORTRAN
PC	1	2	1	2	1
TIME	41.33	44.15	165.07	548.58	165.7
MODEL3	CUDA	-	FORTRAN	-	-
PC	1	-	1	-	-
TIME	41.48	-	165.18	-	-
MODEL3′	-	CUDA	-	FORTRAN	FORTRAN
PC	-	2	-	2	1
TIME	-	44.57	-	548.90	166.0
TOTAL	41.48	44.57	165.18	548.90	166.0
SPEED UP	10.23(T1)	10.21(T1′)	2.57(T1)	0.83(T1′)	2.74(T1′)

The streamed data processing is implemented via modelling engines connected in a chain using data-driven synchronisation between them. These engines can be realised using different computational paradigm sequential (FORTRAN), GPU or FPGA-based architectures. The presented infrastructure has high flexibility and can be dynamically reconfigured according to a variable set of modelling zones. Further optimisation can be provided by taking into account the specifics of concrete problems such as a real-time tsunami warning or a long-term hazard assessment.

To create the wave engines we used the code developed by Vazhenin et al. (2013), which is based on the well-validated MOST package developed at Pacific Marine Environmental Laboratory (NOAA, Seattle, USA), and is also used in the USA for developing inundation maps as well as for Tsunami Inundation maps. Accordingly, the evaluation of parallel CUDA-engines was implemented by numerical and visual comparison with the FORTRAN-based calculation results. For example, this engine is able to process trans-oceanic tsunami wave propagation in less than 15 minutes (4' mash). Maximum distortion is less than 0.001 cm compared to the FORTRAN-based engine.

The results presented confirm the possibility of implementing high-speed computations concurrently at the laboratory level using distributed computing in combination with CUDA-accelerators. As shown in Section 2, modern accelerators are mostly oriented to speedup calculations in a whole computational area. They are much more powerful than the coprocessors used in our experiments. This makes it possible to significantly improve the total calculation speed compatible with calculations on supercomputers.

7 Summary and conclusions

The tsunami modelling on nested grids allows for decreasing the total amount of calculations by processing only selected coastal areas. In fact, nested computing is already used widely in numerical modelling such as weather forecasting and tsunami warning. These applications use highly consistent grids with proportional changing of grid resolution. However, in comparison with those approaches, our proposal integrates in a single modelling scheme the bathymetry grids designed by different developers that usually have non-proportional grid steps and differences in bottom relief.

Our proposal can also be considered a coarse-grained acceleration of the modelling process that is realised by joining heterogeneous components in a streamed computational scheme. These components can show different performance as well as different computational paradigms. Therefore, a special resource management system will be useful in optimising computational load among disparate computers including predicting the efficiency of multi-core processing associated with a set of tasks with varied CPU and main memory.

References

Acuna, A.M. and Aoki, T. (2014) 'AMR multi-gpu accelerated tsunami simulation', *International Conference on Computational Engineering and Science for Safety and Environmental Problems*, pp.708–710.

Baba, T., Ando, K., Matsuoka, D., Hyodo, M., Hori, T., Takahashi, N., Obayashi, R., Imato, Y., Kitamura, D., Uehara, H., Kato, T. and Saka, R. (2016) 'Large-scale, high-speed tsunami prediction for the Great Nankai Trough Earthquake on the K computer', *The International Journal of High Performance Computing Applications*, Vol. 30, No. 1, pp.71–84.

Bader, M., Schraufstetter, S., Vigh, C.A. and Behrens, J. (2008) 'Memory efficient adaptive mesh generation and implementation of multi-grid algorithms using Sierpinski curves', *International Journal of Computational Science and Engineering*, Vol. 4, No. 1, pp.12–21.

Cai, X. and Langtangen, H.P. (2008) 'Making hybrid tsunami simulators in a parallel software framework', *LNCS*, Vol. 4699, pp.686–693, Springer-Verlag.

Fujita, M. (2015) *Tsunami Simulation on FPGA/GPU and its analysis based on Statistical Model Checking* [online] http://cmacs.cs.cmu.edu/seminars/slides/fujita3.pdf (accessed 19 August 2020).

Gidra, H., Haque, I., Kumar, N.P., Sargurunathan, M., Gaur, M.S., Laxmi, V., Zwolinski, M. and Singh, V. (2011) 'Parallelizing TSUNAMI-N1 using GPGPU', *2011 IEEE International Conference on High Performance Computing and Communications (HPCC)*, pp.845–850, IEEE.

Gusyakov, V.K., Fedotova, Z.I., Khakimzyanov, G.S., Chubarov, L. and Shokin, Y. (2008) 'Some approaches to local modelling of tsunami wave runup on a coast', *Russ. J. Num. An. Math. Model*, Vol. 23, No. 6, pp.551–565.

Harig, S., Chaeroni, C., Pranowo, W.S. and Behrens, J. (2008) 'Tsunami simulations on several scales: comparison of approaches with unstructured meshes and nested grids', *Oc. Dyn.*, Vol. 58, No. 5, pp.429–440.

Hasan, K.S., Antonio, J.K. and Radhakrishnan, S. (2017) 'A model-driven approach for predicting and analysing the execution efficiency of multi-core processing' *International Journal of Computational Science and Engineering*, Vol. 14, No. 2, pp.105–125.

Hasan, M.M., Rahman, S.M.M. and Mahamud, U. (2015) 'Numerical modeling for the propagation of tsunami wave and corresponding inundation', *IOSR J. Mech. Civil Eng*, Vol. 12, No. 2, Ver. 4, pp.55–62.

Hayashi, K., Vazhenin, A. and Marchuk, A. (2015) 'Trans-boundary realization of the nested-grid method for tsunami propagation modeling', *Proc. of the 25th International Ocean and Polar Engineering Conference*, pp.741–746, International Society of Offshore and Polar Engineers.

Hayashi, K., Vazhenin, A. and Marchuk, A. (2017) 'Investigation of the artificial underwater object's protection properties using numerical modeling' *Proc. of the 27th International Ocean and Polar Engineering Conference*, pp.981–988, International Society of Offshore and Polar Engineers.

Hayashi, K., Vazhenin, A. and Marchuk, A. (2018) 'Cloud-based pipelined nested tsunami modeling', *Proc. of the 28th International Ocean and Polar Engineering Conference*, pp.714–719, International Society of Offshore and Polar Engineers.

Imamura, F., Yalciner, A. and Ozyurt, G. (2006) *TUNAMI N2: Tsunami Modeling Manual*, Tsunami Engineering Laboratory, International Research Institute of Disaster Science, Tohoku University [online] http://www.tsunami.civil.tohoku.ac.jp/hokusai3/J/projects/manual-ver-3.1.pdf (accessed 19 August 2020).

Kaistrenko, V. (2011) 'Tsunami recurrence versus tsunami height distribution along the coast', *Pure and Applied Geophysics.*, Vol. 168, No. 11, pp.2065–2069.

Karim, M.F., Ismail, A.I. and Meah, M.A. (2014) 'A boundary fitted nested grid model for tsunami computation along Penang Island in Peninsular Malaysia', *Int. J. Math. Comput. Phys.*, Vol. 8, No. 2, pp.277–284, El. Comp. Eng.

Kono, F., Nakasato, N., Hayashi, K., Vazhenin, A. and Sedukhin, S. (2018) 'Evaluations of OpenCL-written tsunami simulation on FPGA and comparison with GPU implementation', *The Journal of Supercomputing*, Vol. 74, No. 6, pp.2747–2775.

Köstler, H., Schmitt, C., Kuckuk., S., Kronawitter, S., Hannig, F., Teich, J., Rüde, U. and Lengauer, C. (2017) 'A scala prototype to generate multigrid solver implementations for different problems and target multi-core platforms', *International Journal of Computational Science and Engineering*, Vol. 14, No. 2, pp.150–163.

Meenakshi, N. and Rodrigues, P. (2014) 'Tsunami detection and forewarning system using wireless sensor network – a survey', *International Journal of Computational Science and Engineering*, Vol. 2, No. 3, pp.76–79.

Sottile, M.J., Rasmussen, C.E., Weseloh, W.N., Robey, R.W., Quinlan, D. and Overbey, J.(2013) 'ForOpenCL: transformations exploiting array syntax in Fortran for accelerator programming', *International Journal of Computational Science and Engineering*, Vol. 8, No. 1, pp.47–57.

Shigihara, Y. and Fujima, K. (2012) 'Development of tsunami model integrating several different grid systems', *Proc. Fifteenth World Conf. on Earthquake Engineering*, Lisbon, Portugal.

Son, S., Lynett, P.J. and Kim, D.H. (2011) 'Nested and multi-physics modeling of tsunami evolution from generation to inundation', *Ocean Modelling*, Vol. 38, Nos. 1–2, pp.96–113.

Tanenbaum, A.S. and Bos, H. (2015) *Modern Operating Systems*, 4th ed., Prentice Hall, USA.

Titov, V.V. (1990) 'Numerical modeling of tsunami propagation by using variable grids, tsunamis: their science and hazard mitigation', in Gusiakov, V.K. (Ed.): *Proc. Int. Tsunami Symposium*, 31 July–3 August, Computing Center, Siberian Branch, USSR Acad. Sci., Novosibirsk, pp.46–51.

Titov, V.V., Kanoglu, U. and Synolakis, C. (2016a) 'Development of MOST for real-time tsunami forecasting', *J. Waterw. Port Coast. Ocean Eng.*, Vol. 142, No. 6, 03116004, doi: 10.1061/(ASCE)WW.1943-5460.0000357, published online.

Titov, V., Moore, C., Spillane, M., Wei, Y., Gica, E. and Zhou, H. (2016b) 'Tsunami hazard assessment based on wave generation, propagation, and inundation modeling for the U.S. East Coast', *J. NUREG*, CR-7222, US Nuclear Regulatory Commission, US NRC Library, Washington, DC.

Vazhenin, A., Lavrentiev, M., Romanenko, A. and Marchuk, A. (2013) 'Acceleration of tsunami wave propagation modeling based on re-engineering of computational components', *International Journal of Computer Science and Network Security*, Vol. 13, No. 3, pp.24–31.

Wei, Y., Bernard, E., Tang, L., Weiss, R., Titov, V., Moore, C., Spillane, M., Hopkins, M. and Kanoglu, U. (2008) 'Real-time experimental forecast of the Peruvian tsunami of August 2007 for U.S. coastlines', *Geophys. Res. Lett.*, Vol. 35, L04609, doi: 10.1029/2007GL032250.