Acceleration of Tsunami Wave Propagation Modeling based on Re-engineering of Computational Components

Alexander Vazhenin†, Mikhail Lavrentiev††, Alexey Romanenko†††, and Andrey Marchuk††††,

†Graduate School Department, University of Aizu, Aizu-Wakamatsu, 965-8580, Japan
††Sobolev Institute of Mathematics of Russian Academy of Science, Novosibirsk, 630090, Russia
†††Department of Information Technology, Novosibirsk State University, Novosibirsk, 630090, Russia
††††Institute of Computational Mathematics and Mathematical Geophysics SD RAS, Novosibirsk, 630090, Russia

Summary
The paper is devoted to creating effective and flexible Tsunami Modeling Environment based on a Service-Oriented Architecture (SOA) allowing high-level of operability and reusability of system components. Accordingly, we use the original Virtual MVC-design pattern (VMVC-pattern) approach that is demarcating a Functional (View) and an Implementation (Model) task by inducing an Integrator (Controller). This allows integrating a big variety of methods and services for Tsunami Modeling with respect to the various development platforms and architectures. The Model is organized on a set of layers in the form of Application Engines each of which is a subset of endpoint services that holds specific logic associated to a business process. Hence, an Engine can encompass functionalities of an API and realize processing that is specific to an application. The paper describes basic features of the MOST (Method of Splitting Tsunami) software package that was used as an initial Propagation Software Engine. This package was accepted by the USA National Ocean and Atmosphere Administration as the basic tool to calculate tsunami wave propagation and to create inundation maps. We describe a set of engines that was designed for several programming platforms including OpenMP, CELL architecture, and GPU's allowing the flexible usage of available computational resources. Paper also includes an analysis the initial and output tsunami data, code design techniques as well as results of some numerical experiments and validation procedures.

Key words:
Tsunami Wave Propagation Modeling, Method of Splitting Tsunami, Service-oriented Architecture, Virtual MVC-design pattern, Fine-grained Parallelization.

1. Introduction

The complexity and importance of the Tsunami Modelling Problem for many practical reasons require embedding modern approaches in the software design. These technological advances are introducing new types of social interactions including collaboration among communities, through tools such as Wikis and Social Networking, which have already found its roots among all types of generations. They facilitate exchange of tools, functionalities and semantics. The end user based customization is possible through integrating a wide range of underlying services. Another challenge is creating scalable technologies, which support an arbitrary number of users while offering them with a personalized and customizable working environment. Therefore, it is obvious that disparate applications and services are being developed over heterogeneous technologies and platforms. Thereby, reusability and interoperability are the important aspects for developing and exposing of computational services [1,2].

As mentioned in [3], “lessons learned from the Japan tsunami will provide direction to research and emergency management communities on how to develop tools, models and methods for mitigating impact of such devastating event both locally and globally.” Shallow water approximations (both linear and nonlinear) are considered worldwide as the basic propagation models for tsunami waves. These models describe reasonably well waves parameters (both amplitudes and traveling times between all recorded sources and available measurement stations) even for rather rough digital bathymetry, provided that the initial seabed displacement at source is given. Several software packages have been proposed to simulate wave propagation over the ocean and to calculate inundation zones. Accordingly, we can distinguish several approaches related to the practical realization of those packages.

The method described in [4] is oriented to create a parallel hybrid tsunami simulator that can mix different models, methods and meshes, maybe even incorporate ``alien software''. This goal is achieved by combining overlapping domain decomposition and object--oriented programming. Actually, the computing performance is not a main goal of this approach.

In paper [5], eight different parallel implementations were used to simulate tsunami propagation with the help of the shallow water equations. Each of these implementations has used a mixed—mode programming model from thread based shared memory to distributed memory and, finally, to a virtual shared memory. As shown in this paper, scalability issues become paramount, and threading

Manuscript received March 5, 2013
Manuscript revised March 20, 2013

becomes a significant bottleneck if sufficient node memory is not available.

TsunamiClaw is a package based on a finite volume numerical method, which means that the solution is rep-resented as a piecewise constant, with numerical values approximating the average solution value in each discrete computational grid cell [6]. There is no specific reference to the shoreline—grid cells may simply be wet or dry depending on their location, and may fill-up or drain-out of water as waves inundate or draw-down at the shoreline. This means that dry land is part of the computational domain, and a single grid is a simple rectangle that may overlap the shoreline. TsunamiClaw solves the shallow water equations in their physically relevant conservative form, therefore, the solution is represented as water depth and momentum.

TUNAMI-N2 software [7] is a tsunami numerical simulation program with the linear theory in deep sea and with the shallow water theory in coastal areas and on dry land with constant grid size in the entire calculation domain. TUNAMI was originally authored by Imamura in 1993 for the Tsunami Inundation Modeling Exchange (TIME) program, and has been applied to several tsunami events.

MOST (Method of Splitting Tsunami) [8,9], developed at Pacific Marine Environmental Laboratory (NOAA, Seattle, USA), allows for real time tsunami inundation forecasting by incorporating real--time data from detection buoys. The model MOST is also used in the United States for developing inundation maps as well as for Tsunami Inundation Modeling [10]. The system has also the web enabled interface named ComMIT.

In this paper, we are focusing on transforming the Tsunami Wave Propagation Modeling Software to a Service-Oriented Architecture (SOA). This transformation was based on the original Virtual MVC-design pattern (VMVC-pattern) that was developed in University of Aizu. It is in demarcating a Functional (View) and an Implementation (Model) task by inducing an Integrator (Controller). Encapsulating certain Non-Functional activities such as security, reliability, scalability, and routing can enrich the Controller. This enables the separation of Integration Logic from that of Functional Logic (Client Application) and Implementation Logic (Service). With this approach, the View and Model become loosely couple. Such abstract layer of controller further enables the View to produce high-level queries that are parsed and analyzed by the controller in order to compose the required services.

According to this approach, the Model is represented as a service inventory on a set of layers in the form of Engines. An Engine is a subset of endpoint services that holds specific logic associated to a business process. Hence, an Engine can encompass functionalities of an API and realize processing that is specific to an application. Particularly, the presented paper is focused on design and comparative analysis of parallel algorithms for the OpenMP platform, IBM Cell BE architecture, and the most attractive today CUDA for GPU use. Here we present results of investigations of a performance gain, obtained with the help of parallel technologies and devoted to fine--grained parallelization of a part of the MOST software that is used for calculating the tsunami wave propagation over the deep ocean.

The rest of this paper is organized as follows. Section 2 explains the mathematical model for simulating wave propagation used in the MOST software including description of computational process and analysis of numerical data needed for numerical modelling. In Sections 3, we describe a set of computational engines that was designed for several programming platforms. Section 4 contains results of some numerical experiments and validation procedures. Finally, conclusion includes some concluding remarks and comments about future work.

2. Mathematical Model and Calculation Process

2.1 Mathematical Model

The approximations of shallow-water theory (both lineal and non-lineal) are used as the basic models for describing wave propagation throughout the ocean. These models rather accurately reflect basic wave parameters (propagation time period from the source to the recording device and wave amplitudes) even for a fairly rough numerical bathymetry in the assumption that initial displacement in the source is unknown.

The MOST software package uses numerical model of calculating wave propagation through deep water zone applying decomposition method for spatial variables. This method was initially developed in the Tsunami Laboratory, Computer Center of the Siberian Division, Academy of Sciences of USSR in Novosibirsk. Then the method was updated in the Pacific Marine Environmental Laboratory (NOAA, Seattle, USA) and was adapted to the models and standards of data accepted by tsunami watch services in the United States as well as other countries and used in tsunami research works in most countries. MOST is used to numerically simulate three processes of tsunami evolution: the estimation of residual displacement area resulting from an earthquake and tsunami production, transoceanic propagation through deep water zones, and contact with land (run-up and inundation). The given research work is concerned with the second stage – deep water wave propagation.

Nonlinear approximation of shallow water system is used for numerical calculation of tsunami wave propagation as follows [6]:

$$\begin{aligned} &H_t + (uH)_x + (vH)_y = 0, \\ &u_t + uu_x + vu_y + gH_x = gD_x, \\ &v_t + uv_x + vv_y + gH_y = gD_y, \end{aligned} \tag{1.1}$$

Where H(x, y, t) = h(x, y, t) + D(x, y, t), h - stands for the height of the wave calculated from unperturbed level, D – the function delineating bottom configuration (digital bathymetry), u(x, y, t), v(x, y, t) – speed vector components along x and y respectively, and g – acceleration of gravity.

The adduced shallow water model soundly describes the process of tsunami waves transoceanic propagation providing that the horizontal dimension of ocean floor surge by an order exceeds the ocean depth at that point.

System (1.1) could be presented in symmetric form:

$$\frac{\partial z}{\partial t} + A\frac{\partial z}{\partial x} + B\frac{\partial z}{\partial y} = F \,, \tag{1.2}$$

where

$$z = \begin{pmatrix} u \\ v \\ H \end{pmatrix}, \quad A = \begin{pmatrix} u & 0 & g \\ 0 & u & 0 \\ H & 0 & u \end{pmatrix},$$

$$B = \begin{pmatrix} v & 0 & 0 \\ 0 & u & g \\ 0 & H & u \end{pmatrix}, \quad F = \begin{pmatrix} gD_x \\ gD_y \\ 0 \end{pmatrix}.$$

The rectangular zone $\Omega = \{x, y : 0 \le x \le X, 0 \le y \le Y\}$ with its sides parallel to coordinate axes will be viewed as the variation domain of spatial variables.

The numerical algorithm to solve the system (1,2) is based on spatial decomposition along axis directions. For this purpose, let us study two backup systems, each of which depends on one spatial variable only.

$$\frac{\partial \varphi}{\partial t} + A\frac{\partial \varphi}{\partial x} = F_1, \quad 0 \le x \le X \;; \tag{1.3}$$

$$\frac{\partial \psi}{\partial t} + B\frac{\partial \psi}{\partial y} = F_2, \quad 0 \le y \le Y \,, \tag{1.4}$$

where $F_1 = \begin{pmatrix} gD_x \\ 0 \\ 0 \end{pmatrix}, \quad F_2 = \begin{pmatrix} 0 \\ gD_y \\ 0 \end{pmatrix}.$

In order to find numerical solution of the system (1.2), it will suffice to make numerically stable solution for systems (1.3) and (1.4). Let us make the difference scheme for the system (1.3). The equations for this system are put down as follows:

$$\begin{aligned} &v_t + uv_x = 0, \\ &u_t + uu_x + gH_x = gD_x, \\ &H_t + (uH)_x = 0. \end{aligned} \tag{1.5}$$

This is a quasi-linear hyperbolic system. All eigenvalues of matrix A are real and diverse:

$$\lambda_1 = u, \quad \lambda_{2,3} = u \pm \sqrt{gH}.$$

We will use the canonical form of this system for the numerical solution. This allows actualizing boundary conditions for the finite-difference analogue of the boundary value problem. The canonical form is put down as follows:

$$\begin{aligned} &v'_t + \lambda_1 v'_x = 0, \\ &p_t + \lambda_2 p_x = gD_x, \\ &q_t + \lambda_3 q_x = gD_x. \end{aligned} \tag{1.6}$$

where *v'*, *p*, *q* are Riemannian invariants of the system (1.5) which have the following form:

$$\begin{aligned} &v' = v, \\ &p = u + 2\sqrt{gH}, \\ &q = u - 2\sqrt{gH}. \end{aligned} \tag{1.7}$$

In order to find numerical solution of the system (1.6), it was suggested to make an explicit difference scheme on a four-point stencil with quadric approximation order regarding spatial variables and first order approximation with respect to time.

2.2 Calculation Process

Numerical calculation of tsunami wave propagation process beginning from its spatial source usually starts with stating initial conditions that represent water surface vertical displacement zone (having summed it up with depth we get the thickness of the water layer) and initial wave stream velocity components. As a rule, in case of submarine earthquakes this velocity is minor and can be neglected, that is, it may be considered that water is quiescent, though it is no longer in equilibrium condition because of some zone's vertical displacement. Then only boundary conditions are actualized for every calculation step. In practice, as a rule, only wave amplitude has a practical meaning. That is why outbound parameters of the algorithm are represented by the values of water surface elevation at all cross-points of computational grid at certain given instants and sequences of ocean level values at certain area points. Figure 1 shows the block-diagram illustrating the main loop of calculating process.

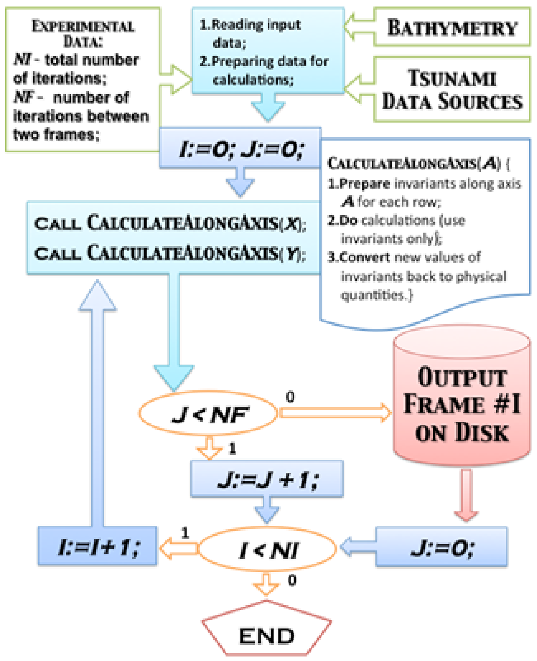

EXPERIMENTAL DATA:
NI - total number of iterations;
NF - number of iterations between two frames;
1.Reading input data;
2.Preparing data for calculations;
BATHYMETRY
TSUNAMI DATA SOURCES
I:=0; J:=0;
CALCULATEALONGAXIS(A) {
1.Prepare invariants along axis A for each row;
2.Do calculations (use invariants only);
3.Convert new values of invariants back to physical quantities.}
CALL CALCULATEALONGAXIS(X);
CALL CALCULATEALONGAXIS(Y);
OUTPUT FRAME #I ON DISK
J < NF
0
1
J:=J + 1;
I:=I+1;
1
I < NI
J:=0;
0
END

Fig. 1 Tsunami wave propagation calculation process

To run a program, it is necessary to specify information on

- Bottom topography or bathymetry data;
- Initial and boundary conditions;
- Modeling parameters such as time-steps and length of model run.

After launching, the main modelling program implements calculations and stores results as a series of frames with defined interval NF in the NetCDF format [11] that is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. There are some programs allowing analyzing and visualizing the NetCDF data.

3. SOA and Propagation Engines for Different Programming Platforms

3.1. SOA and Virtual MVC-design Patterns

Currently, the Service Oriented Architectures or SOA is effective software design platform based on the service virtualization that focuses on building new functionalities and services without bothering about how the services will be exposed, consumed and maintained. The service virtualization has abstracted peer services for transparent service lookup allowing flexible integration of heterogeneous software components.

The Model-View-Controller (MVC) is another effective way to avoid an expensive process of reinventing, rediscovering and revalidating agnostic software artifacts. As shown in [12], we are imitating the MVC design patterns in order to explore the other composite patterns for an efficient integration of the applications and services. The demarcation of a Functional (View) and an Implementation (Model) task can be achieved deliberately by inducing an Integrator (Controller). Encapsulating certain Non-Functional activities such as security, reliability, scalability, and routing can enrich the Controller. This enables the separation of Integration Logic from that of Functional Logic (Client Application) and Implementation Logic (Service).

In the original MVC design pattern, the View updates itself from the Model, via the Observer pattern. Therefore, the original MVC pattern works like a closed loop wherein the View talks to the Controller, this as well connects to the Model, which in turns talks to the View. As shown in Figure 2, the direct link between the Model and the View can be removed. With this modification, a complete decoupling of the view from the model can be achieved. In the redefined virtual MVC design pattern, Controller acts as a single point of contact for the View layer and the Model layer [13]. This implies higher privacy of the business logic in Model from the View, and higher reusability of application components.

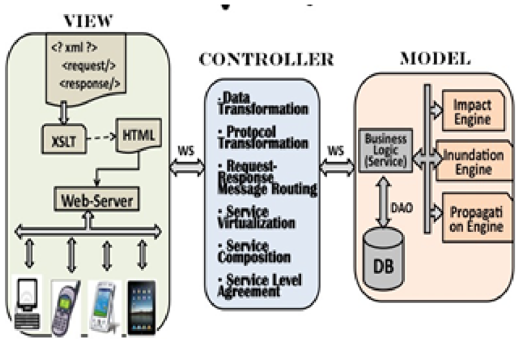


Fig. 2 SOA for Tsunami Modeling

Accordingly, the Controller can be considered as the compound design pattern of the Enterprise Service Bus (ESB) allowing integration of the software components by using the Dependency Injection (DI) pattern. We are proposing a design pattern based approach for the seamless integration among disparate applications and services based on the DI pattern that meets the most of the requirements. We are exploring the DI to find a solution for integrating and decoupling the services and applications by their transformation from a stand-alone program to a service-oriented tool.

The Model is organized on a set of layers in the form of Application Engines each of which is a subset of endpoint services that holds specific logic associated to a business process. Hence, an Engine can encompass functionalities of an API and realize processing that is specific to an application [9]. The Propagation and Inundation Engines can be constructed directly from the MOST software package. In the next sections, we are demonstrating a set of Propagation Engines developed for different programming platforms.

3.2. Sequential Engine

The original code of the MOST software was implemented on Fortran 90. It takes 3.31 seconds for one time step on 4 dual-core CPU server, based on Intel Xeon, 2.8GHz. After porting this program onto a C/C++ language, it takes about 3.00 seconds to process the one time step. This program was a basis for designing other propagation engines allowing efficient usage of a variety of available computational resources. This program was also adopted to a Java environment with the calculation time about 18.5 seconds for one time step. This allows also embedding this program into a Virtual MVC-architecture as well as combining it with visualization tools.

3.3. OpenMP Engine

The OpenMP paradigm [7] is mainly focused on performing loop iterations in parallel. Actually, a programmer should only point loops in a sequential program that can be processed in parallel.
As algorithm allows all rows and columns of input data to be processed independently, main loop of the program could be marked by the '#pragma' terms in the following form:

```
while(I++ < NI){
   #pragma omp parallel for
   for(x=0;x<NX; x++)
      processColumn(x);
// transpose 2D arrays
   …
   #pragma omp parallel for
      for(y=0;y<NY; y++)
         processRow(y);
      // transpose 2D arrays back
      …
}
```

To obtain a regularity of matrix scanning and effective usage of the CPU cash for both computational steps, the intermediate matrix transposition was introduced for both computational steps. It is necessary to implement forward and back transpositions of four matrices. Importantly, matrix transpositions were implemented in parallel using OpenMP operations based on a block-wise matrix transposition. To optimize the block size, several experiments have been carried out. This allowed achieving the maximum of performance with block size of 64x64 elements. The performance optimization was also realized using embedded SIMD-stream co-processors. From the programmer's point of view, they can be programmed using special built-in CPU stream operations known as SSE instructions. This part of executable code has been rewritten using SSE instructions for calculations of invariants, height of wave and its speed along axis. The more detailed information about parallel OpenMP algorithms for Tsunami modelling can be found in [].

3.4. IBM Cell BE Programming Environment

IBM Cell BE processor has the complex multicore architecture allowing implementation of intensive calculations in comparison with the ordinary CPU [16]. The Cell BE computational resources can be divided into two parts. The Power Processor Element (PPE) is a conventional processor the main responsibility of which to set up tasks for the Synergistic Processor Elements (SPE) that compute intensive Cell OS parts and applications. In a Cell based operating system, the PPE runs the OS kernel, service program and the most of the user's applications. The design of the Cell Engine was mainly focused on distributing iterations along axis among SPEs.
Actually, it was necessary to provide several steps in order to develop effective Cell Tsunami modeling algorithms. The first step is in porting a sequential program and running it on PPE. The next step was in reorganizing a program in order to support the PPE to assign tasks for SPEs. To increase the SPE performance, it was necessary to overlap I/O operations with calculations and processing operations by using two data buffers in SPE. One buffer is for data processing, and the other one is to prepare next portion of the data. The last re-engineering was done for the SSE code optimization. It includes a vectorization technique decreasing the number of operations in four times as well as a quad word exchange for loading and

storing data. Traditionally, the optimization was also oriented to reach the high level of usage of operational SSE registers.

3.5. CUDA Platform

Currently, the processing power of the Graphics Processing Units (GPUs) has increased tremendously allowing reaching a raw computing power of close to one TFLOPS with a relatively small cost. That is why researchers are looking at ways to utilize this efficiently for non-graphics based applications also in the framework of so-called GPGPU (General Purpose Graphics Processing Units) architecture. Nvidia now supports a C like programming language called CUDA [17,18] (Compute Unified Device Architecture) that allows a programmer to explicitly request that certain portions of the code be run on the GPU. It allows implementing mathematical algorithms on GPU knowing nothing about shaders, vertexes and other stuff referred to graphics.
The initial tsunami modelling code was re-engineered in order to adopt it to the GPU platform as follows:

```
for(int i=0; i<cfg.steps; i++){
// calculate along X
Invariants_X<<<dimGrid,
  dimBlock>>>(... , x_size, y_size);
SWater_<<<dimGrid,
  dimBlock>>>(... , x_size, y_size);
RInvariants_X<<<dimGrid,
  dimBlock>>>(... , x_size, y_size);


// calculate along Y
transpose<<<dimGrid,
dimBlock>>>(d_qwdata, d_q1data,  x_size, y_size);
transpose<<<dimGrid,
dimBlock>>>(d_uwdata,  d_u1data,        x_size,
y_size);
transpose<<<dimGrid,
dimBlock>>>(d_vwdata, d_v1data, x_size, y_size);
Invariants_Y<<<dimGrid,
  dimBlock>>>(... , y_size, x_size);
SWater_<<<dimGrid_, dimBlock>>>
  (...,  y_size, x_size);
RInvariants_Y<<<dimGrid,
  dimBlock>>>(..., y_size, x_size);
transpose<<<dimGrid_,
  dimBlock>>>(d_qwdata, d_q1data,
  y_size, x_size);
transpose<<<dimGrid_,
  dimBlock>>>(d_uwdata, d_u1data,
  y_size, x_size);
transpose<<<dimGrid_,
dimBlock>>>(d_vwdata, d_v1data, y_size, x_size);
// Save results
   ...
}
```

GPU has a SIMD-architecture that is oriented to perform same operations on different data with some significant advances. Therefore, matrix transposition was avoided because of features of the texture memory allowing two-dimensional data addressing. The good calculation speedup was achieved by using multiplications instead division operations. Discrete representation of modelling data causes some inaccuracy in invariant computation (kernels for Invariants_X and Invariants_Y) especially for the square root computation function that brings up an error in 1-2 least significant bits. This eventually leads to spontaneous swaying of the ocean surface already with 50 iterations. Several solutions to be considered were: to calculate invariants in double precision, to represent double through 2 floats, or, alternatively, to eliminate squared root computation within the mail loop. The first two variants did not incur considerable failure in productive capacity. Each stream multiprocessor currently contains 8 modules for processing single-precision floats and 2 for double-precision floats. The final variant – the elimination of square root computation – turned out to be the simplest and the most suitable in terms of its implementation. Other optimization techniques are described in [17].

4. Numerical Experiments and Validation

4.1. Common Remarks

As pointed in [3,13,19], only through parallel testing of models under identical conditions, can the community determine the relative merits of different computational formulations, an important step to further improvements in speed, accuracy, and reliability. It is important to note that modern parallel architectures are effective in getting very high performance. However, they are actually inefficient in minimizing the rounding errors for operations of single or double precision format. That is why a special testing was provided in order to optimize inundation engines codes as well as evaluate them on the equal data sets and conditions. Let us estimate the data volume needed for successful modelling and verification. For the typical digital bathymetry it is enough to have the distance between the mesh nodes about 3,6 km. Therefore, the computational domain should have a size of 2048x2048 points to cover the Pacific Ocean. Accordingly, the NOAA uses 2500x1800 mesh size. Taking into account this mesh size, it is possible to estimate the time complexity of the MOST method. The complete wave propagation modelling requires implementing calculations for multiple time steps. The required number of such time steps is about 1440 to cover for 24 hours time period.

We developed the original bathymetry covering the area of the Pacific Ocean adjacent to the northwest of the island of Honshu (Japan). The gridded digital bathymetry for numerical modelling was prepared using 500m resolution bathymetry around Japan [20], and 1 arc sec ASTER Global digital elevation model [21]. A computational rectangular grid of 2413x2405 points includes knots of pre-setup values of depth. Length of a spatial step in both directions made 0.0024844 geographical degrees that is about 277m in a North-South direction and about 221m in the West-East direction. The bottom relief of this computational domain A1 is stretching from 34 to 40 degrees of North Latitude and from 140 to 146 degrees of East Longitude is shown in Fig. 3.

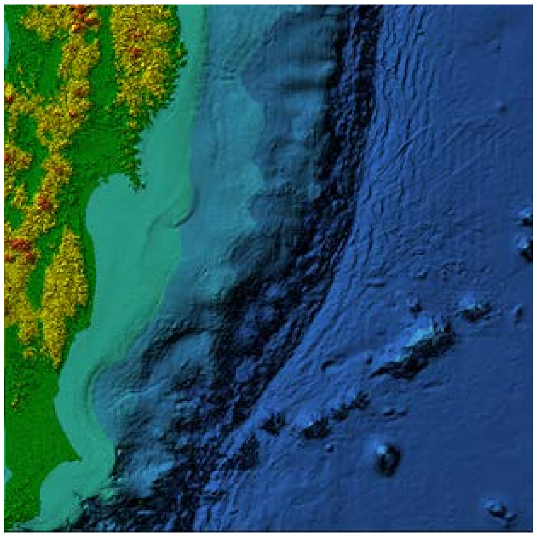


Fig. 3. Visualization of the 2413x2405 gridded relief around NE coast of Honshu island. Mesh size: 0.00248x0.00248 arc degrees (221x277 m).

We have also provided modelling according the tsunami data generated from the Great Japanese Earthquake (38.322°N, 142.369°E, Mw = 8.9 at 5:46:23 UTC) on March 11, 2011 [22]. The fault length and width are 400km × 150km (Fig. 4). Next subsections are demonstrating the results of evaluating these different engines based on the data described.

4.2. OpenMP Engine

Fig. 5 depicts results of speedup obtained by following to this simple strategy (graph "OpenMP") that was used for loops providing calculations along X and Y axes (Fig. 5). Graph named "+Transposition" shows that this strategy allowed obtaining stable speedup growing even that additional time has been spend to implement matrix transpositions. Results of the final parallel program implementation using embedded SSE coding are also shown in Fig. 5.

The best result is about ten times speedup for 5 cores in comparison with the sequential program. Performance decrease with more than 5 cores is because of using a smaller amount of data to be processed, and a prevalence of communications over the computations in CPUs.

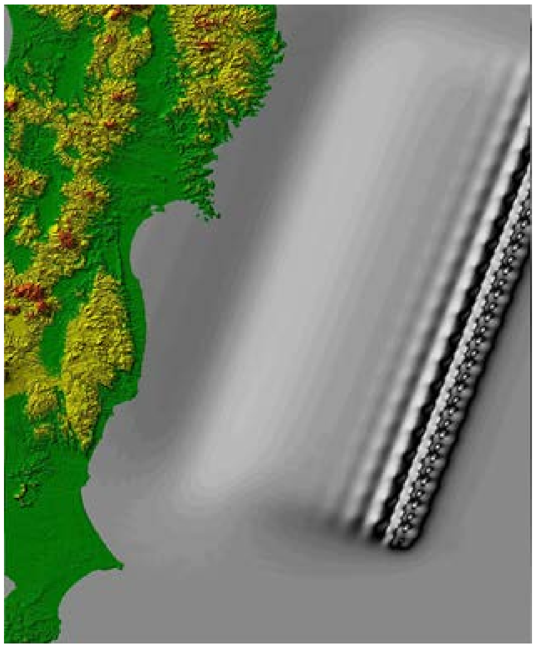


Fig. 4. Modeling according the Great Japanese Earthquake data

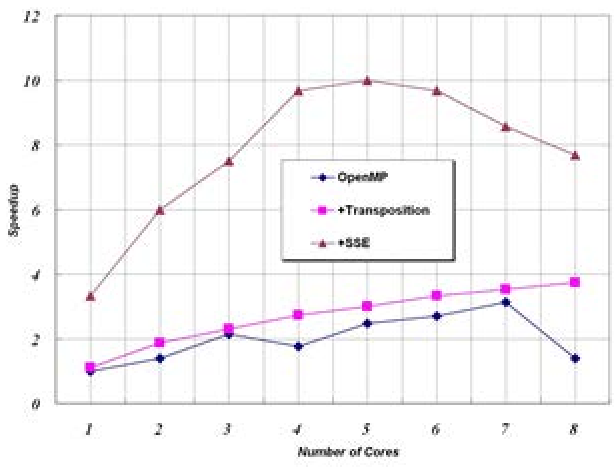


Fig. 5. OpenMP Engine Calculation Speedup

As was pointed above, we used the fixed mesh size required by acceptable result accuracy. Actually, here we have additional possibilities to increase performance by increasing a size of mesh. To analyse the accuracy of parallel algorithms, a set of numerical experiments are also provided with different types of data distributions among the processes. The final result fluctuates in the acceptable range of error measurements. The results demonstrate also good system scalability allowing calculating the one modelling step for 0.18 seconds at 16-cores system with shared memory access SMP 4 x Intel Xeon CPU X7350, 2.93GHz.

4.2. IBM Cell BE Engine

This engine was realized on the SONY PlayStation 3 (PS3), the architecture of which includes the Cell Broadband Engine (Cell BE) Processor in which six SPE nodes are actually used for work. Others two are used for resource reservation. Fig. 6 presents result of measurements time required for one modeling and corresponding speedup depending on number of SPE.

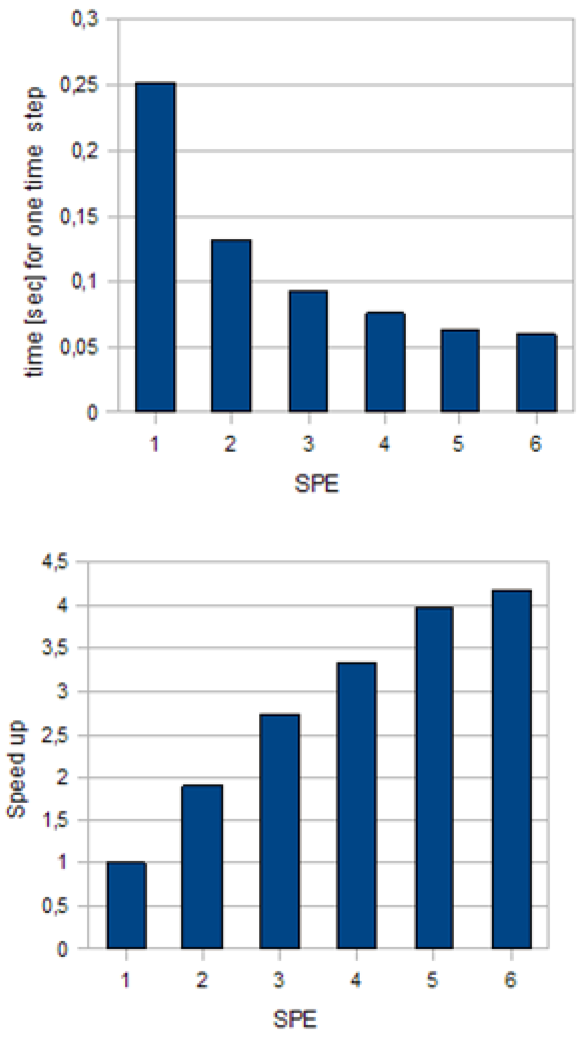


Fig. 6. Cell BE Engine calculation time and Speedup

To process all data, they needed to be distributed among SPEs. The way when SPEs take the sub-tasks is more preferable in contrast to the variant when PPE acts as a master sending tasks to SPEs. One PPE hardly could have enough time to manage all SPEs, because processing time is rather small (our case). We were limited by amount of local memory for each SPE.

Taking into account a problem complexity (about 109 operations), the maximum performance is about 17 GFLOPS for the one modelling step 6 SPE (Fig. 6). It is more than 5 times faster than a parallel version optimized for 4 cores for OpenMP Engine as well as more than 50 times faster than the original sequential version.

4.3. CUDA Engine and Validity Testing

The best results were obtained using Tesla C2050 GPU. The calculation speedup was about 150 times for a single modelling step. Table 1 shows results of the tsunami wave propagation modelling with the two types of bathymetry and data initial sources described in subsection 4.1. The modelling was implemented according to the scheme of Tsunami Wave Propagation Calculation Process showed in the Figure 1. The modelling parameters were as follows. The total number of iterations was NI=5000, and the number of iterations between two frames was NF=100. So, the output file contains fifty frames for all experiments. Results show that the CUDA Engine supports calculations with high level of performance even the presence of the input/output operations.

Table 1: Results of Testing of Tsunami Wave Propagation Calculation Process

		TSUNAMI SOURSE			
		TEST		***GREAT EARTHQUAKE***	
BATHIMETRY	***NOAA***	FORTRAN	128.8 min	FORTRAN	124.3 min
		CUDA	3.38 min	CUDA	2.69 min
		SPEEDUP	37.5	SPEEDUP	37.5
	TOHOKU	X		FORTRAN	97.36 min
				CUDA	2.12 min
				SPEEDUP	45.9

Special attention was paid to evaluate the accuracy of parallel engines. The evaluation was implemented by numerical and visual comparison of FORTRAN-based calculation results with the other engines. For example, is able to process trans-oceanic tsunami wave propagation in less than 15 minutes (4' mash). Maximum distortion is less that 0.001 cm comparing to the FORTRAN-based engine.

5. Conclusion

Modern applications and services are rather diverse and disparate with respect to the various development platforms and architectures. The communication gap and integrity are improving gradually. In order to achieve the high level of application and service reusability, it is more effective to integrate the applications and services rather than rebuilding because redevelopment is a costly affaire. The design patterns based approach enables us to reuse the proven and consolidated design knowledge, supporting the development of high-quality software systems.

We developed and tested a set of tsunami modelling engines supporting several parallel platforms including OpenMP, CELL architecture, and GPU's. The set computational services can be provided allowing the flexible usage of available computational resources and services. The numerical experiments and validation procedures confirm the reliability of proposed technique as well significant acceleration of modelling process. They allow also design a variety of applications based on component-wise design approach.

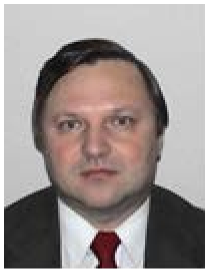
References

[1] Th. Erl, SOA Design Patterns, Prentice Hall, 2010.
[2] M. Kuniavsky, Smart Things: Ubiquitous Computing User Experience Design, Elsevier, 2009.
[3] V. Titov, “March 11, 2011 Tohoku-Japan Tsunami: Lessons from Forecast Assessment”, Proc. of the Joint International Conference on Human-Centered Computer Environments (HCCE '12), Aizu-Wakamatsu, Japan, ACM Publisher, pp. 99-100, 2012.
[4] X.Caiand, P.Langtangen, “Making Hybrid Tsunami Simulators in a Parallel Software Framework”, LNCS, vol. 4699, Springer-Verlag, pp. 686–693, 2008.
[5] K.Ganeshamoorthy, D. Ranasinghe, K.Silva, and R.Wait, “Performance of Shallow Water Equations Model on the Computational Grid with Overlay Memory Architectures”, Proc. of the Second International Conference on Industrial and Information Systems (ICIIS 2007), IEEE Press, Sri Lanka, pp. 415–420, 2007.
[6] D. George, TsunamiClaw User’s Guide, http://faculty.washington.edu/rjl/pubs/icm06/TsunamiClaw Doc.pdf.
[7] N.Shuto, F.Imamura, A.C.Yalciner, and G.Ozyurt, TUNAMI2: Tsunami Modeling Manual, http://tunamin2.ce.metu.edu.tr/
[8] V. Titov, “Numerical Modeling of Tsunami Propagation by using Variable Grid”, Proc. of the IUGG/IOC International Tsunami Symposium, Computing Center Siberian Division USSR Academy of Sciences, Novosibirsk, USSR, pp. 46–51, 1989.
[9] V. Titov and F. Gonzalez, “Implementation and Testing of the Method of Splitting Tsunami (MOST)”, Technical Memorandum ERL PMEL-112, National Oceanic and Atmospheric Administration, Washington DC, 1997.
[10] J.C. Borrero, K. Sieh, M. Chlieh, and C.E. Synolakis, “Tsunami Inundation Modeling for Western Sumatra”, Proc. of the National Academy of Sciences of the USA, Vol. 103, N 52, http://www.pnas.org/content/103/52/19673.full, 2006.
[11] NetCDF (Network Common Data Form) - http://www.unidata.ucar.edu/software/netcdf/
[12] S. Rajam, R. Cortez, A. Vazhenin, S. Bhalla. “Modified MVC-design Patterns for Service Oriented Applications”, Frontiers in Artificial Intelligence and Applications. IOS Press, Vol. 231, pp. 108-126, 2011.
[13] A. Vazhenin, K. Hayashi, Al. Romanenko, “Service-oriented tsunami wave propagation modeling tools”. Proc. of the Joint International Conference on Human-Centered Computer Environments (HCCE '12), Aizu-Wakamatsu, Japan, ACM Publisher, pp. 131-136, 2012.
[14] R.Chandra, Parallel Programming in OpenMP, Morgan Kaupmann Publishers, 2001.
[15] M. Lavrentiev-jr, A. Romanenko, V. Titov, A. Vazhenin, “High-Performance Tsunami Wave Propagation Modeling”. LNCS, vol. 5698, pp. 423-434, 2009.
[16] CellBroadband Engine, Programming Handbook, IBM 2007.
[17] N. Corporation, CUDA: Compute Unified Device Architecture Programming Guide. Tech. rep., 2007.
[18] CUDA C Programming Guide, http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html.
[19] C. Synolakis1, E. Bernard, V. Titov, U. Kanoglu, F. Gonzalez. “Standards, criteria, and procedures for NOAA evaluation of tsunami numerical models”. NOAA Technical Memorandum OAR PMEL-135, 2007.
[20] http://jdoss1.jodc.go.jp/cgi-bin/1997/depth500_file.
[21] http://www.gdem.aster.ersdac.or.jp/search.jsp.
[22] http://iisee.kenken.go.jp/staff/fujii/OffTohokuPacific2011/tsunami.html

Mikhail Lavrentiev graduated from the Novosibirsk State University (mathematics) in 1978. He received PhD in differential equations in 1982, associate professor since 1986, second doctoral degree in 1993, full professor since 2012. He works in the Novosibirsk State University and with the research institutes of the Russian Academy of Sciences (Sobolev Institute of Mathematics and Institute of Automation and Electrometry). Major interests are in the area of math and computer modeling in geo-sciences.

Alexey Romanenko received the B.S. and M.S. degree in Physics (speciality – Computer science) from Novosibirsk State University in 1999 and 2001, respectively. He received PhD in software engineering in 2004. During 1999-2004, he stayed in Supercomputer Software Department, Institute of Computational Mathematics and Mathematical Geophysics SB RAS to study modern compute architectures, algorithms, and techniques. From 2006 he is involved in adopting of geoscince models to GPUs. Now he is with Novosibirsk State University (Russia) and NVIDIA Ltd.

Andrey Marchuk received the M.S. degree in applied mathematics from Novosibirsk state university in 1976. During 1976-1983, he stayed in the Institute of pure and applied mechanics SD RAS. There in 1981 he has got PhD in numerical mathematics. Since 1983 he is working in the Institute of computational mathematics and mathematical geophysics SD RAS (Novosibirsk, Russia) where he was studying numerical modeling of tsunami waves. In 2000 he have got degree of Doctor of physics-mathematical sciences in numerical modeling. He also works in Novosibirsk state university as leading researcher. Andrey Marchuk received the M.S. degree in applied mathematics from Novosibirsk state university in 1976. During 1976-1983, he stayed in the Institute of pure and applied mechanics SD RAS. There in 1981 he have got PhD in numerical mathematics. Since 1983 he is working in the Institute of computational mathematics and mathematical geophysics SD RAS (Novosibirsk, Russia) where he was studying numerical modeling of tsunami waves. In 2000 he have got degree of Doctor of physics-mathematical sciences in numerical modeling. He also works in Novosibirsk state university as leading researcher.



Alexander Vazhenin received his M.S in Computer Engineering from the Novosibirsk State Technical University (Russia) in 1978. He received his PhD in Computer Science from the Institute of Informatics Systems of the Siberian Division of the Russian Academy of Sciences in 1993. He published about 100 refereed academic papers. His research and educational interests include parallel architectures, algorithms and programming tools, self-explanatory software high-accuracy computations, visual, and multimedia and Internet technology. He has been program and organizing committee member of many international conferences. He is currently senior associate professor at the University of Aizu, Japan.