# On the Performance of the Parallel Implementation of the Shallow Water Model on Distributed Memory Architectures

K.Ganeshamoorthy, D.N.Ranasinghe, K.P.M.K.Silva and R.Wait

***Abstract*— This paper presents a study of the impact of memory architectures, distributed memory (DM) and virtual shared memory (VSM), in the solution of parallel numerical algorithms on a multi-processor cluster. A parallel implementation of the shallow water equations to model a Tsunami is chosen as the case study. Data is partitioned into sub-domains, namely a four by three grid scheme and an eight by six grid scheme which are used for the parallel implementation of this model. There are four versions of the parallel algorithm for each grid scheme: distributed memory without threads, distributed memory with threads, virtual shared memory without threads, and virtual shared memory with threads. These four parallel versions have been implemented on a high performance cluster, connected to the "Nordugrid". Experiments are realized using the Message Passing Interface (MPI) library, the C/Linda, and the Linux pthreads. Subject to the availability of memory, the virtual shared memory version without threads performs best, and as the task is scaled up, the threaded version becomes efficient in both DM and VSM implementations.**

***Index Terms*—MPI, Linda, multi processors, Shallow water equations, tsunami model.**

## I. Introduction

A Tsunami is a series of waves generated in a body of water by an impulsive disturbance that vertically displaces the water column [1], [2]. Earthquakes, landslides, volcanic eruptions, explosions can generate a tsunami. A tsunami can savagely attack coastlines, causing devastating property damage and loss of life.

Manuscript received March 6, 2009. Accepted July 7th, 2009. This research was funded by the National Science Foundation, Sri Lanka (Grant No. RG/2005/FR/07) and by SPIDER, the Swedish programme of ICT for developing Regions.

K.Ganeshamoorthy is with the Department of Computation & Intelligent Systems, University of Colombo School of Computing, 35, Reid Avenue, Colombo-7, Sri Lanka. (e-mail: ganesh@webmail.cmb.ac.lk).

D.N.Ranasinghe and K.P.M.K.Silva are also with the Department of Computation & Intelligent Systems, University of Colombo School of Computing, 35, Reid Avenue, Colombo-7, Sri Lanka. (email: dnr@ucsc.cmb.ac.lk, mks@ucsc.cmb.ac.lk)

R.Wait is with the International Science Programme and Department of Information Technology, Uppsala University, Uppsala, Sweden. (richard.wait@isp.uu.se)

A Tsunami is characterized as a shallow water wave. Shallow water waves are different from wind generated waves, the waves many of us have seen at the beach. Wind generated waves usually have a period of five to twenty seconds and a wavelength of about one hundred to two hundred meters. A tsunami can have a period in the range of ten minutes to two hours and a wavelength in excess of 500 km. It is because of their long wavelengths that a tsunami behaves as shallow water waves. A wave is characterized as a shallow water wave when the ratio between the water depth and its wavelength gets very small [1].

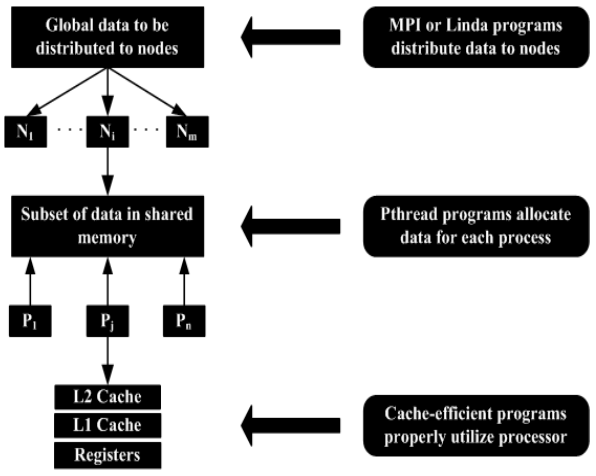
Global data to be distributed to nodes
MPI or Linda programs distribute data to nodes
$N_1$ · · · $N_i$ · · · $N_m$
Subset of data in shared memory
Pthread programs allocate data for each process
$P_1$
$P_j$
$P_n$
L2 Cache
L1 Cache
Registers
Cache-efficient programs properly utilize processor

Fig.1. Typical mixed mode programming model [9].

The shallow water equations on a rotating sphere serve as a primary test problem for numerical methods used in modeling global atmospheric flows [3]. They describe the behaviour of a shallow homogeneous incompressible and inviscid fluid layer. They present the major difficulties found in the horizontal aspects of three-dimensional global atmospheric modeling. Thus, they provide a first test to weed out potentially non-competitive schemes without the effort of building a complete model. However, because they do not represent the complete atmospheric system, the shallow water equations are only a first test. Ultimately schemes which look attractive based on these tests must be applied to the complete baroclinic problem. The existence of a standard test set for the shallow water equations will encourage the continued exploration of alternative numerical methods and provide the community with a mechanism for judging the merits of numerical schemes and parallel computers for atmospheric flow calculations [3].

The current state-of-the-art in tsunami modeling still requires considerable quality control, judgment, and iterative, exploratory computations before a scenario is assumed reliable. This is why the efficient computation of many scenarios for the creation of a database of pre-computed scenarios that have been carefully analyzed and interpreted by a knowledgeable and experience tsunami modeler is an essential first step in the development of a reliable and robust tsunami forecasting and hazard assessment capability. Using more advanced parallel algorithms, it may become technically feasible to execute real-time model runs for guidance as an actual event unfolds. However, this is not currently justified on scientific grounds; an operational real-time model forecasting capability must await improved and more detailed characterization of earthquakes in real-time, and verification that the real-time tsunami model computations are sufficiently robust to be used in an operational, real-time model [4].

Shallow water equations have been widely used to study the Tsunami phenomenon [5] [3], as a model of the basic fluid dynamics of the ocean. Solving partial differential equations numerically for real-life problems are computationally demanding, therefore, utilizing super-computers/clusters efficiently is important in order to achieve computational efficiency [6]. Strategies to improve the accuracy and overall quality of model predictions have been and continue to be of great interest to numerical model developers but in addition to accuracy, the utility of a numerical model is greatly affected by the algorithm's efficiency [7].

Parallel computing provides a feasible and efficient approach to solve very large-scale prediction problems but any redistribution of the data is a potentially time-consuming task for parallel architectures [8]. Fig.1. shows the memory hierarchy that exists in most nodes of a modern cluster environment [9], where many nodes are linked together by a high-speed network; and inside each node there may be two or more processors; along with each processor, memory access is either to a high speed memory unit "cache" or the low speed "main memory".

The aim of this paper is to study the impact of memory architectures associated with distributed memory and virtual shared memory in the extraction of multiple levels of parallelism, for the solution of numerical algorithms. The objective here is to evaluate the effects of the programming model on the scalability of this shallow water model. Computing the wave propagation in the tsunami model, where the entire ocean is the solution domain, is challenging, both due to the huge amount of computation needed and due to the fact that different physics applies in different regions [10].

The Message Passing Interface, MPI [11], and C/Linda [12] are alternative paradigms for communication between global nodes in the distributed memory environment and in the virtual shared memory environment, respectively. In the mixed mode programming model, both MPI and C/Linda are used for communication between global nodes while inside each MPI process and within each C/Linda process POSIX threads [13] are used in order to extract further parallelism.

This paper is organized as follows: In the section, Related Work, we present the related work for our study of research. The section on Linear Long Wave Theory describes the linear long wave theory used to simulate the tsunami model. In Algorithm Design, we describe our design principles. Implementation is presented next. Experimental results from both memory architectures on high performance cluster are presented and are discussed in Evaluation. Finally, in Conclusion we make the concluding remarks.

## II. Related Work

### A. *Parallel simulators for Tsunami*

Hybrid tsunami simulators that allow different sub-domains to use different mathematical models, spatial discretizations, local meshes, and serial codes have been proposed by Xing Cai [10]. Boussinesq water wave equations given below are used for this purpose.

$$\frac{\partial\eta}{\partial t}+\nabla.(H+\alpha\eta)\nabla\phi+\varepsilon H\left(\frac{1}{6}\frac{\partial\eta}{\partial t}-\frac{1}{3}\nabla H.\nabla\phi\right)\nabla H=0 \tag{1}$$

$$\frac{\partial\phi}{\partial t}+\frac{\alpha}{2}\nabla\phi.\nabla\phi+\eta-\frac{\varepsilon}{2}H\nabla.\left(H\nabla\frac{\partial\phi}{\partial t}\right)+\frac{\varepsilon}{6}H^{2}\nabla^{2}\frac{\partial\phi}{\partial t}=0 \tag{2}$$

where $\eta$ and $\phi$ are primary unknowns denoting, respectively, the water surface elevation and velocity potential. The water depth $H$ is assumed to be a function of the spatial coordinates $x$ and $y$ . In equations (1) and (2) the weak effect of dispersion and nonlinearity is controlled by the two dimensionless constants $\varepsilon$ and $\alpha$ respectively. The widely used linear shallow water equations can be derived by choosing $\varepsilon = \alpha = 0$.

$$\frac{\partial\eta}{\partial t}+\nabla.(H\nabla\phi)=0 \qquad (3)$$

$$\frac{\partial\phi}{\partial t}+\eta=0 \qquad (4)$$

The equations from (1) to (4) are used to model the tsunami. The equations (1) and (2) are resolved by unstructured meshes and finite element discretization, whereas structured meshes and finite differences are commonly used for equations (3) and (4). Such a parallelization strategy is most easily realized by using sub-domains, such that the entire spatial domain Ω is decomposed into a set of overlapping sub-domains $\{\Omega_s\}^{p}_{s=1.}$ In generic setting, where a partial differential equation (PDE) is expressed as $L_\Omega(u)=f_\Omega$, the Schwarz algorithm consists of an iterative processes generating $u^0$, $u^1$, $u^2$, ..........., $u^k$ as a series of approximate solutions. During Schwarz iteration *k*, each sub-domain first independenly updates its local solution through

$$L_{\Omega_s}\left(U_s^k\right)=f_{\Omega_s}^{k-1} \qquad (5)$$

where $f_{\Omega_s}^{k-1}$ refers to a right-hand side which is restricted within $\Omega_s$ and depends on the latest global approximation $u^{k-1}$ . Then, the new global solution $u^k$ is composed by sewing together the sub-domain local solutions $u^k_1$, $u^k_2$, $u^k_3$, ..........., $u^k_p$

Equation (5) thus opens for the possibility of using different local solvers in different sub-domains. Taking the idea of additive Schwarz one step further, different mathematical models in different sub-domains can be applied. Therefore, different serial codes may be deployed region wise. The proposed hybrid parallel tsunami simulator is implemented using object-oriented techniques and is based on an existing advanced C++ finite element solver named class Boussinesq applicable for unstructured meshes, and a legacy F77 finite difference code applicable for uniform meshes. The resulting hybrid parallel tsunami simulator thus has full flexibility and extensibility [10].

## *B. Parallel computation of a highly nonlinear Boussinesq equation model through domain decomposition*

Applications of the Boussinesq equations cover a broad spectrum of ocean and coastal problems of interest, from wind wave propagation in intermediate and shallow water depths to the study of tsunami wave propagation across large ocean basins. In general, implementations of the Boussinesq wave model to calculate free surface wave evolution in large basins are computationally intensive, requiring large amount of CPU time and memory. To facilitate such extensive computations, a parallel Boussinesq model has been developed by the Khairil et al. [2], using the domain decomposition technique in conjunction with MPI. The parallel Boussinesq model developed is based on its serial counterpart. The governing equations consist of the two-dimensional depth-integrated continuity equation:

$$\frac{\partial \mathrm{H}}{\partial \mathrm{t}}+\nabla.(\mathrm{H}\mathbf{u}_{\mathrm{x}})-\text{ì}^{2}\nabla.\left\{\mathrm{H}\left[\left(\frac{1}{6}\left(\text{ç}^{2}-\text{ç}\mathrm{h}+\mathrm{h}^{2}\right)-\frac{1}{2}\mathrm{z}_{\mathrm{x}}^{2}\right)\nabla\mathrm{S}+\left(\frac{1}{2}(\text{ç}-\mathrm{h})-\mathrm{z}_{\mathrm{x}}\right)\nabla\mathrm{T}\right]\right\}=0 \qquad (6)$$

and the horizontal momentum equation:

$$\frac{\partial\mathbf{u}_x}{\partial t}+\frac{1}{2}\nabla(\mathbf{u}_x.\mathbf{u}_x)+g\nabla\eta+\frac{\partial}{\partial t}\left\{\frac{1}{2}z_x^2\nabla S+z_x\nabla T-\nabla\left(\frac{1}{2}\eta^2 S+\eta T\right)\right\}$$
$$+\nabla\left\{\frac{\partial\eta}{\partial t}(T+\eta S)+(z_x-\eta)(\mathbf{u}_x.\nabla)T+\frac{1}{2}\left(z_x^2-\eta^2\right)(\mathbf{u}_x.\nabla)S+\frac{1}{2}(T+\eta S)^2\right\}=0 \qquad (7)$$

where $S=\nabla.u_x$, $T=\nabla.(hu_x)+\partial h/\partial t$ , *h* is depth, η is free surface elevation.

Equations (6) and (7) differ from the equations given by Wei *et al.* [14] in the inclusion of the time derivatives of the depth *($h_1$, $h_2$)*to account for temporal bottom profile changes that occur during landslide/earthquake, which is one of several possible sources of tsunami.

The parallel approach has had three important aspects, domain decomposition, communication, and parallel solver of the tridiagonal system of the simultaneous linear equations. Three different domain sizes have been considered:(500 x 500), (1000 x 1000), and (2000 x 2000). The overall performance of the model has been very good. The efficiency of the model decreases as the number of (500 x 500)and (1000 x 1000) processors increases which is apparent in the case of domains. The rate of the efficiency decrease is faster for smaller domain. This is due to the ratio of arithmetic operation time to communication time decreasing faster for domains with smaller number of nodes. The performance of the model improves as the number of grids increases; a favourable feature of a parallel model which is intended for simulation on ever-increasing domain sizes. Thus, this parallel model provides a future opportunity for large wave-resolving simulations in the near shore, with global domains of many millions of grid points, covering $O(100km^2)$ and greater basins. Further, real-time simulation with Boussinesq equations becomes a possibility.

## *C. Implicit Parallel FEM Analysis of Shallow Water Equations*

Jiang Chunbo et al. [15] have solved the shallow water equations (SWEs) as the governing equation to model a river flow. SWEs are implemented on clustered workstations. For the parallel computation, the mesh is automatically partitioned using the geometric mesh partitioning method. The governing equations are then discretized implicitly to form a large sparse linear system, which is solved using a direct parallel generalize minimum residual algorithm (GMRES). The shallow water equations (8) and (9) used here are obtained by integrating the conservation of mass and momentum equations assuming a hydrostatic pressure distribution in the vertical direction.

$$\frac{\partial \eta}{\partial t} + \frac{\partial q_j}{\partial x_j} = 0 \tag{8}$$

$$\frac{\partial q_i}{\partial t} + q_j \frac{\partial}{\partial x_j}\left(\frac{q_i}{H}\right) = -gH\frac{\partial \eta}{\partial x_i} + \frac{1}{\rho}\left(\tau_i^s - \tau_i^b\right) + \frac{\partial}{\partial x_j}\left[v\left(\frac{\partial q_i}{\partial x_j} + \frac{\partial q_j}{\partial x_i}\right)\right], \quad i, j = 1,2 \tag{9}$$

where η is the water elevation, $H$ is the water depth, $q_i = uH_i$, $u_i$ is the mean horizontal velocity, $g$ is the gravitational acceleration, $v$ is the eddy viscosity, ρ is the water density, $\tau^s_i$ is the surface shear stress, and $\tau^b_i$ is the bottom shear stress.

The finite element method for triangular elements is used for the spatial discretization. Three kinds of finite element meshes are used to simulate the velocity field in the two numerical examples, flow around a circular cylinder and flow in the Yangtze River. MPI has been used for the communication between nodes. The computed results agree well with the observed results. The good speedup and efficiency for the parallel computation show that the parallel computing technique is a good method to solve large-scale problems [15].

## III. Linear Long Wave Theory

There are many different numerical methods for computing shallow water equations on a sphere. Therefore, a standard test suite of seven problems for evaluating numerical methods for the shallow water equations in spherical geometry was proposed by Williamson et al [3] and accepted by the modeling community in order to compare newly proposed methods. The shallow water equations are widely used as a prototype to study phenomena like wave-vortex interactions that occur in more complicated models of large scale atmosphere/ ocean dynamics [5].

Consider the sea to be a volume of incompressible water on a rotating sphere, with Coriolis force, *fu*. The horizontal coordinates are *x* and, *y* the vertical coordinate *z*, which is zero at the mean sea surface and positive upwards. The sea bed is located at *z* = - *H,* and the surface is located at *z* = *h.* The linear shallow water equations [16], [17], [18] consists of the continuity equation

$$\frac{\partial h}{\partial t} + \frac{\partial}{\partial x}[u(h+H)] + \frac{\partial}{\partial y}[v(h+H)] = 0 \tag{10}$$

and the conservation of horizontal momentum

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} - fv = -g\frac{\partial h}{\partial x} \tag{11}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + fu = -g\frac{\partial h}{\partial y} \tag{12}$$

*u* and *v* are the velocity components in *x*- and *y*- direction, is the surface elevation and *g* is the acceleration due to gravity.

Defining the equations in terms of the discharge fluxes *U* = *uH, V* = *vH,* leads to discretization that always satisfy the conservation of mass. Then the conservation of horizontal momentum can be written as

$$\frac{\partial U}{\partial t} + \frac{\partial}{\partial x}\left(\frac{U^2}{h+H}\right) + \frac{\partial}{\partial y}\left(\frac{UV}{h+H}\right) - fV = -g(h+H)\frac{\partial h}{\partial x} \tag{13}$$

$$\frac{\partial v}{\partial t} + \frac{\partial}{\partial x}\left(\frac{UV}{h+H}\right) + \frac{\partial}{\partial y}\left(\frac{V^2}{h+H}\right) + fU = -g(h+H)\frac{\partial h}{\partial y} \tag{14}$$

The finite difference approximations using centered differences in space and a leap-frog time discretization, are based on a staggered grid corresponding to an Arakawa C-grid [19], [20], with the continuity equation centered on the point $\left(x_i, y_j, t_{k+1/2}\right)$ and the equations of motion centered on the points $\left(x_{i+1/2}, y_j, t_k\right)$ and $\left(x_i, y_{j+1/2}, t_k\right)$ respectively.

Writing $D \equiv h + H$, and using upwind differences for the convection terms to maintain stability it follows that

$$\frac{\partial}{\partial x}\left(\frac{U^2}{D}\right) \approx \frac{1}{\Delta x}\left(\lambda_{1,1}\frac{\left(U^{k-\frac{1}{2}}_{i,j+\frac{3}{2}}\right)^2}{D^{k-\frac{1}{2}}_{i,j+\frac{3}{2}}} + \lambda_{2,1}\frac{\left(U^{k-\frac{1}{2}}_{i,j+\frac{1}{2}}\right)^2}{D^{k-\frac{1}{2}}_{i,j+\frac{1}{2}}} + \lambda_{3,1}\frac{\left(U^{k-\frac{1}{2}}_{i,j-\frac{1}{2}}\right)^2}{D^{k-\frac{1}{2}}_{i,j-\frac{1}{2}}}\right) \tag{15}$$

where with up-winding,

$$U^{k-\frac{1}{2}}_{i,j+\frac{1}{2}} \begin{cases} \geq 0 & \lambda_{1,1} = 0, \quad \lambda_{2,1} = 1, \quad \lambda_{3,1} = -1 \\ < 0 & \lambda_{1,1} = 1, \quad \lambda_{2,1} = -1, \quad \lambda_{3,1} = 0 \end{cases}$$

and similarly for the other terms. The difference equations are defined on a rectangular grid in terms of spherical polar coordinates. An accurate representation of tsunami running up the shore implies a grid spacing of no more that 100 meters in a region of about 4 km out from the shore. As this fine grid is not reasonable over the whole ocean a succession of overlapping grids is necessary near the coast. A data decomposition scheme is applied for the parallel solution of the shallow water equations. In data decomposition, we keep the sequential formulation of the problem, but distribute the data and operations among the processors. The scalability of several data decomposition algorithms for finite difference atmospheric and ocean models have been analyzed by several authors [21], [22].

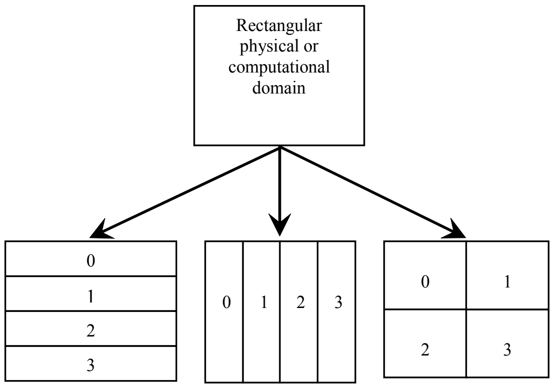

Fig. 2. Three possible ways of decomposing a rectangular domain. The numbers in the sub-domains represent the processor number.

Several strategies exist within the data decomposition paradigm for dividing domains into sub-domains. In the two-dimensional grid, the computational domain is decomposed both in x and y coordinate directions. In many cases, the computation is proportional to the volume of a sub-domain and the communication is proportional to the surface area. In such cases, a logical strategy is to partition the domain in such a way that it minimizes the surface area of each sub-domain relative to its volume. This keeps the computation-to-communication ratio high. In this study, two-dimensional decomposition is chosen. This involves assigning each sub-domain to a processor and solving the equations for that sub-domain on the respective processor. With the two dimensional decomposition, no global information is required at any particular grid point and inter-processor communications are required only at the boundaries of the sub-domains. The inner-border of a sub-domain requires the outer-border of the adjacent sub-domain during a time-step because of the spatial discretization [16], [17], [18].

## IV. Algorithm Design

In our work, the domain decomposition method is used to parallelize the tsunami model. In this method, the parallel algorithm is very similar to the serial algorithm with some additional routines added to facilitate the communication between processors. Using this method, all the processors involved in the parallel calculations basically perform the same computational operations. The only difference is in the data being processed in each processor.

The physical or computational domain used in this paper is rectangular in shape. In the domain decomposition method, the rectangular domain is divided into several smaller rectangular sub-domains, where the number of sub-domains is equal to the number of processors used. With four processors, for example, there are three possible ways of decomposing the domain into equal-area parts as depicted in Fig. 2. The best decomposition depends not only on the architecture of the system being used, but more importantly on the memory limitations on each node, especially in commodity clusters, as such, an assignment like two by two is recommended.

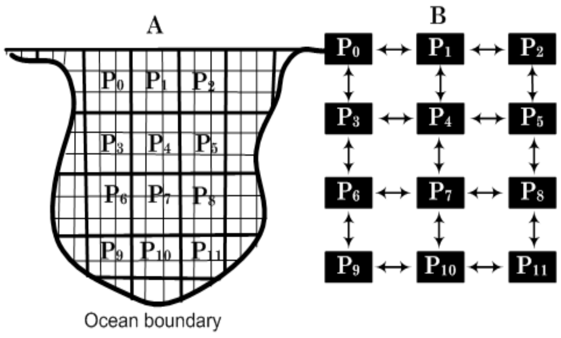

Fig. 3. (A) Overview of the unstructured grid and assignment of worker nodes to sub-domains. (B) Arrows indicate the inner-worker communication.

An important aspect in decomposing the domain is the load balancing, i.e. all processors should have equal or almost equal amount of data to be processed. If the number of grid points is divisible by the number of processors, the grid points in each processor is simply the ratio of the number of grid points to processors. If it is not, the remainder is distributed across the first m processors, where m is the remainder. The load should be balanced in both x and y directions.

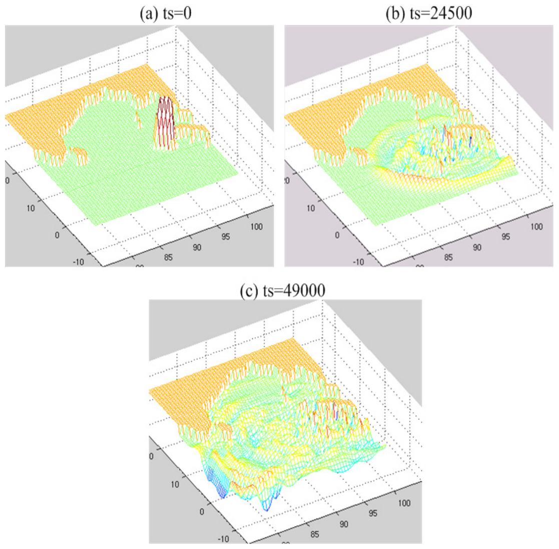

Fig. 4. Linear shallow water wave profiles at three different time-steps (ts) calculated using 48 processors.

Two grid schemes, a 12 node (4 x 3) grid system and a 48 node (8 x 6) grid system, are used to parallelize the tsunami model with a domain size of (900 x 1226). Portions of the domains are assigned to each of the worker nodes, as illustrated in Fig. 3 where each sub-domain is labeled with a processor (worker) number. Snapshots of the free surface evolution are shown in Fig. 4. Data for each sub-domain is stored with each processor including water depth, coordinates, and initial disturbance. This data is read by each of the workers in a pre-processing stage i.e., prior to initiating the time-integration loop. Since the model updates the solution explicitly using local data, each processor works independently of the others but requires data from neighbouring workers to update solution along sub-domain boundaries. The exchange of data between processors occurs several times per time-step. There are two key features to this exchange: First, only data along the boundaries between processors is exchanged. Second, each processor is only required to communicate with at most four other processors. The domain decomposition is performed with this second feature in mind to avoid communications between more than four other processors.

Communication occurs between two adjacent processors during message passing. In passing the data from one processor to another, an efficient and safe communication must be developed. To efficiently exchange the data between adjacent processors, the data is first stored in a contiguous memory prior to executing the sending processes. At the same time contiguous memories of the same size as used in the sending processes are created to receive the data from the sending processes. At this point, the data is ready for sending and receiving processes.

TABLE I
Run Time In Seconds (wot – without thread; wt – with thread)

| Nodes | MPI | | C/Linda | |
|---|---|---|---|---|
| | wot | wt | wot | wt |
| 12 | 9686.54 | 11053.54 | 2743.74 | 3297.78 |
| 48 | 11934.38 | 10273.73 | 3183.43 | 2845.34 |

In this model, message passing occurs four times per time-step and the MPI function, ``MPI_ Sendrecv", and the C/Linda operations "in" and "out" are used to perform the message passing between global nodes. This corresponds to eight messages per time- step per processor independent of the number of processors being used. Many more messages are sent during pre-processing, but these are ignored for run-time analysis purposes since time integration is by far the most time consuming element of the program. The parallel algorithm is outlined below.

1. Decompose rectangular domain into load balanced rectangular sub-domains where the width and length of a rectangular sub-domain are $W/N_w$ approximately, and $L/N_l$ respectively, where $W$ and $L$ are the width and length of domain, and $N = N_w$*$N_1$ is the number of processors to be used in the ($N_w$ x $N_1$) grid scheme.

2. Specify parallel language related parameters, such as locations, neighbours, sub-domain sizes, and file names, of processors. Location of $k^{th}$ processor is ($r_k$, $c_k$), where $r_k = k/gdm1 + 1$ and $c_k = k - (r_k - 1)* gdm1 + 1$ with *gdm1* and *gdm2* being the dimensions of the grid. In the four by three grid scheme, gdm1 = *3* and *gdm2 = 4*, and in the eight by six scheme, *gdm1=6* and *gdm2=8.* For $k$ = 0, 1, 2, 3, 4, . . . . . . . , ($gdm1*gdm2$ -1), define $north_k$, $south_k$, $east_k$, and $west_k$ to be the neighbours located in the side of north, south, east and west of the $k^{th}$ processor, then:

| | |
|---|---|
| $north_k = k - gdm1$ | if $r_k > 1$ |
| $south_k = k + gdm1$ | if $r_k < gdm2$ |
| $east_k = k + 1$ | if $c_k < gdm1$ |
| $west_k = k - 1$ | if $c_k > 1$ |

3. Input data and initial conditions:

   (a) Each processor, $P_k$ , reads the water depth, coordinates, and initial disturbance from the text file assigned in step 2.

   (b) Each processor, $P_k$, exchanges sub-domain boundary data from *$east_k$ to $west_k$, from $west_k$ to $east_k$, from $north_k$ to $south_k$* , and form south, to northk, in oder.

4. Set parameters and coefficients used at the open sea boundary.

5. Repeat the following steps for the pre-defined time-steps:

   (a) Each processor, $P_k$, exchanges sub-domain boundary data from $east_k$ to $west_k$.

   (b) Each processor, $P_k$, exchanges sub-domain boundary data from $north_k$ to $south_k$

   (c) Computation of the equation of continuity.

   (d) Setting of the open sea boundary condition.

(e) Each processor, $P_k$, exchanges sub-domain boundary data from *west*$_k$ to *east*$_k$.

(f) Each processor, $P_k$, exchanges sub-domain boundary data from *south*$_k$ to *north*$_k$.

6. Gather computed results among all processors.

7. Compute the output on processor, where both are equal to null.

## V. Implementation

Data is partitioned by sub-domains, where a four by three grid scheme and an eight by six grid scheme are used for the parallel implementation of this model. In each of the grid schemes, there are four parallel variations: (1) distributed memory without threads; (2) distributed memory with threads; (3) virtual shared memory without threads; (4) virtual shared memory with threads. Implementations use the Message Passing Interface (MPI) library [11], the C/Linda [12] and the pthread library [13].

The mixed-mode programming model which uses thread programming in the shared memory layer and message passing programming in the distributed memory layer is a method commonly used to utilize the hierarchy of memory resources efficiently [9]. In our mixed-mode programming model, MPI is used for the data communication between the global nodes, and within each MPI process, pthreads are used. In the virtual shared memory mixed-mode approach, C/Linda is used for the data communication between the global nodes, and within each C/Linda process, pthreads are used.

These four algorithms have been implemented on the Monolith cluster of Nordugrid [23] which consists of a high speed backbone interconnecting multi processor nodes. The Monolith cluster has 396 nodes, all of which are i686 architecture with 2.20GHz Intel(R) XEON(TM) processors, dual processor nodes and 2048MB per node main memory and 512KB cache. The operating system is Linux version 2.4.34-cap1-smp.

## VI. Evaluation

Table I shows the timing values for the eight scenarios arising from the four parallel variations of the algorithm mentioned above. Consider the four by three grid scheme, with both threading and non-threading. The parallel algorithms for the virtual shared memory exhibit better performance than the algorithms for distributed memory. Though the virtual shared memory implicitly passes messages, the replication management subsystem has been optimized in C/Linda compared to MPI [10], to yield better performance.

In the four by three grid scheme, in both scenarios for distributed and virtual shared memory, non-threading algorithms exhibit better performance than threading algorithms. One possible explanation for this is that each node keeps nine, two dimensional floating point type arrays of size equal to the sub domain size. Because of the memory limitations it is not possible to declare local two-dimensional arrays for threads, causing all threads in a node to concurrently use global arrays for their own computations.

Now consider the eight by six grid scheme. Here too, for both threading and non-threading, the parallel algorithms for the virtual shared memory exhibit better performance than the algorithms for distributed memory. In contrast to the previous instance, both algorithms for distributed and virtual shared memory, the threading algorithms exhibit better performance than non-threading algorithms. This is because that the sub domain size allocated to each node is half size of sub domain size of the smaller grid scheme of four by three size. The local two dimensional floating point type array allocation for threads in a node is now possible compared to the four by three grid system. Therefore, since all threads of a node work independently, the threading parallel algorithm shows better performance than non-threading parallel algorithm.

Among the parallel variations not using threads in both memory architectures, the four by three grid scheme shows better performance than eight by six grid scheme. This is due to the ratio of computation time to communication overhead decreasing faster for domains with smaller size. However, when the task is scaled up, say up to eight by six grid system, owing to the smaller sub-domain sizes aligning with the available memory in the node, the threaded versions become more efficient in both MPI and C/Linda implementations. In both threading and non-threading environments, the C/Linda version exhibits better performance than the MPI version.

## VII. Conclusion

This paper has presented eight different parallel implementations of a tsunami model based on the shallow water equations. Each of these implementations use a mixed-mode programming model from thread based shared memory, to distributed memory and finally to virtual shared memory. Owing to node memory limitations, scalability issues become paramount, and threading becomes a significant bottleneck if sufficient node memory is not available, offsetting the middleware advantages. With sufficient node memory however, C/Linda with threads outperforms MPI with threads, indicating the effectiveness of extracting parallelism over virtual shared memory, distributed memory and shared memory multiple levels for this class of problems.

## Acknowledgment

This work was conducted at the Department of Computation and Intelligent Systems, University of Colombo School of Computing. This research was performed using computational resources at the National Supercomputer Centre, Linkoping University, Sweden, and of the computational resources of Research Laboratory, University of Colombo School of Computing.

## References

[1] Wikipedia. http://en.wikipedia.org/wiki/Tsunami, 2009.
[2] K. I. Sitanggang and P. Lynett, Parallel computation of a highly nonlinear Boussinesq equation model through domain decomposition, International journal for numerical methods in fluids, ISSN 0271-2091, vol. 49, no.1, 2005, pp. 57-74.
[3] D. L. Williamson, J. B. Drake, J. J. Hack, R. Jakob, and P. N. Swarztrauber, A standard Test Set for numerical Approximations to the Shallow Water Equations in Spherical Geometry, Journal of Computational Physics, 102(1), pp. 211-224, 1991.
[4] T. Vasily, G. Frank, and M. Hal, Tsunami Forecasting and Hazard Assessment Capabilities, Pacific Disaster Center (PDC) & Maui High Performance Computing Center (MHPCC) Tsunami Modeling, National Oceanic and Atmospheric Administration Center for Tsunami Research ( http://nctr.pmel.noaa.gov/, 2009)
[5] J. D. Paul, and S. Rick, Shallow Water Equations with a Complete Coriolis Force and Topography, Physics of Fluids, September 2005.
[6] B. Ragnhild, Nested Parallelism in OpenMP with Application to Adaptive Mesh Refinement, University of Bergen, February 2003.
[7] F. S. Brett and C. P. John, Parallel Implementation of an Explicit Finite-volume Shallow-water Model, 16th ASCE Engineering Mechanics Conference, July 16-18, 2003, University of Washinton, Seattle.
[8] S. L. Johnsson and C. Ho, Algorithms for Matrix Transposition on Boolean N-Cube Configured Ensemble Architectures, SIAM J. Matrix Analysis and Application, vol.9(3), July 1988, pp.419-454.
[9] W. Meng-Shiou, A. Srinivas, and A. K. Ricky, Mixed Mode Matrix Multiplication, Proceedings of the IEEE International Conference on Cluster Computing, IEEE Computer Society Washington, DC, USA, 2002.
[10] C. Xing and P. L. Hans, Making Hybrid Tsunami Simulators in a Parallel Software Framework, LNCS, Applied Parallel Computing. State of the Art in Scientific Computing, vol. 4699/2008, pp. 686-693.
[11] MPI Forum http://www.mpi-forum.org/http://www-unix.mcs.anl.gov/mpi/, 2009.
[12] Linda User Guide, Scientific Computing Associates IC., One Century Tower, 265 Church Street, New haven, CT 06510-7010 USA, September, 2005. (http://www.lindaspaces.com/about/index.html, 2009)
[13] Pthreads.http://www.math.arizona.edu/swig/pthreads/threads.html, 2009.
[14] G. Wei, J. T. Kirby, S. T. Grilli, R. Subramanya, A fully nonlinear Boussinesq model for surface waves, Part I, Highly nonlinear unsteady waves, Journal Fluid Mechanics1995; 294:71-92.
[15] J. Chunbo, L. Kai, L. Ning, and Z. Qinghai, Implicit Parallel FEM Analysis of Shallow Water Equations, Tsinghua Science & Technology, vol. 10, Issue 3, June 2005, pp. 364-371.
[16] http://www.gfdl.noaa.gov/fms/pubrel/m/atm_dycores/src/atmos_spectral_shallow/shallow.pdf, 2009.
[17] http://www.sea.ee/~elken/DO5.pdf, 2009.
[18] http://www.misu.su.se/~goran/shallow_water/, 2009.
[19] Arakawa A and Lamb V, Computational design of the basic dynamical processes of the UCLA general circulation model, Methods in Computational Physics, vol. 17, Academic Press, 1977, pp. 174-267.
[20] C. Goto and Y. Ogawa, Dept. of Civil Engineering, Tohoku University, Translated for the Tsunami Inumdation Modelling Exchange Project, by N. Shuto. Numerical method of tsunami simulation with the leap-frog scheme, UNESCO Intergovernmental Oceangraphic Commission, Manuals and Guides, 35, 1992.
[21] S. Roar, Scalability of Parallel Grid Point Limited Area Atmospheric Models I & II, Manuscript, Department of Mathematics Sciences, Norwegian University of Science and Technology, Trodheim, Norway, 1996.
[22] S. Thomas, J. Cote, A. Staniforth, I. Lie, and R. Skalin, A Semi-Implicit Semi-Lagrange Shallow-Water Model for Massively Parallel Processors, Proceedings of the 6th ECMWF Workshop on the Use of Parallel Processors in Meteorology, November 1994, pp. 407-423.
[23] The Grid middleware project in the Nordic countries. http://nordugrid.org