Development of Hierarchical Domain Decomposition Explicit MPS Method and Application to Large-scale Tsunami Analysis with Floating Objects

Kohei Murotani¹*, Seiichi Koshizuka¹, Tasuku Tamai¹, Kazuya Shibata¹, Naoto Mitsume¹, Shinobu Yoshimura¹, Satoshi Tanaka², Kyoko Hasegawa², Eiichi Nagai³ and Toshimitsu Fujisawa³

¹Department of Systems Innovation, School of Engineering, the University of Tokyo, Japan
²Department of Media Technology, Ritsumeikan University
³Prometech Software, Inc., Japan

* muro@sys.t.u-tokyo.ac.jp

Received: February 28, 2014; Accepted: June 9, 2014; Published: October 31, 2014

Abstract. In this research, a novel distributed memory parallel algorithm for the explicit moving particle simulation (MPS) method is presented. In this method, the analysis area is divided for distributed memory parallel computation using ParMETIS. The development offers two improved points from the distributed parallel explicit particle method of single bucket-based domain decomposition [1]. The first improved point is the hierarchical domain decomposition of two levels. The second improved point is that particle data are managed by buckets. Based on the improvements, we are efficiently able to run a large-scale run-up tsunami analysis.

Keywords: MPS, distributed memory parallel computing, explicit algorithm, ParMETIS, tsunami

Figure 1: Illustration showing two 10-m diameter tanks being carried along by the tsunami that inundated the Ishinomaki urban area.
1. Introduction

The moving particle simulation (MPS) method [2,3] is a popular particle method. The particle method regards a continuum as a set of particles, discretizes the physical laws governed by differential equations using interactions between the particles, and calculates the states and motions of the particles. Since the particles, as the calculation points, move on the time-marching processes, the particle method is superior to grid methods in terms of solving dynamic physical phenomena such as free surfaces and large deformations. However, the motion of particles makes it difficult to parallelize the particle method in distributed memory parallel computers.

Iribe et al. [4] developed the distributed memory parallelization of the semi-implicit MPS method using a PC cluster. They adopted a slice grid method for the domain decomposition and concentrated their efforts on improving the renumbering of particle numbers in order to reduce communication volumes in solving the Poisson equation. Though Iribe et al. [4] could increase the communication speed to a certain degree, their results were not enough with more than 16 processing elements. Iribe et al. [4] performed a tsunami simulation of 60 seconds with 6.3 million particles using 4 processing elements in 237 hours.

Ferrari et al. [1] developed the distributed memory parallelization of the smoothed particle hydrodynamics (SPH) method using ParMETIS [5,6,7]. In their method [1], virtual buckets are arranged across the whole analysis domain, all particles are assigned to the buckets, and bucket-based domain decomposition is performed using ParMETIS. Since ParMETIS computes minimal-cut partitions of graphs and meshes in parallel, the domain decomposition can be performed with an equal number of particles in subdomains and minimum communication volumes between subdomains. When Ferrari et al. solved the dam-break problem of 500,000 particles, the strong scaling from 1 CPU to 128 CPUs was 0.46. This result was not enough to use supercomputers with more than 1000 processing elements.

Marrone et al. [8] and Leffe et al. [9] also developed the distributed memory parallelization of SPH. Marrone et al. [8] targeted a simulation of hundreds of millions particles and performed a simulation of the wave pattern generated by a ship. Leffe et al. [9] performed a simulation of the lifeboat impact in still water using 100 millions of particles. Marrone et al. [8] and Leffe et al. [9] could also get the high parallel efficiency using SPH of an explicit algorithm. In researches of the discrete element method (DEM) by Horner et al. [10] and Tuszuki et al. [11,12] and the non-smooth contact dynamics (NSCD) method by Balzer et al. [13], the distributed memory parallelization was developed successfully. However, these all distributed memory parallel particle methods have shown relative values like parallel scalabilities not but achieved FLOPS.

In this research, we improve upon the distributed parallel explicit particle algorithm of single domain decomposition [1] and describe two improved points in section 3. The first improved point is the hierarchical domain decomposition of two levels. The second improved point is that particle data are managed by buckets. The parallel scalabilities and the achieved FLOPS of our method are shown in section 4. In section 5, we perform an analysis of two 10-m diameter tanks carried along by the tsunami that inundated the Ishinomaki urban area as shown in Fig. 1. Transparency point sprites are used to represent fluid particles while opacity surfaces are used for structural objects such as tanks and buildings in the figure.

The parallel computers used in this paper are the FX10 at the Information Technology
Center of the University of Tokyo, and the CX400 at the Research Institute for Information Technology of Kyushu University. The FX10 and the CX400 are both made by Fujitsu. The FX10 has 4800 processing elements with one SPARC64 IXfx 1.848-GHz CPU of 16 cores. The compiler of the FX10 is Fujitsu compiler provided for the FX10. Our compiler options of the FX10 are “-Kfast,openmp,parallel -SSL2BLAMP –Kocl”. For most inner loop, three directives of “swp”, “novrec” and “simd” are set. The software pipeline and SIMD vectorization become available by these compiler options and directives. FLOPS, L1 cache miss rate and execution time are measured with a Fujitsu profiler of the FX10. The CX400 has 1476 processing elements with two Intel Xeon E5-2680 2.7-GHz CPUs of 8 cores. The compiler of the CX400 of Kyushu University is Intel C compiler Version 12.1.5.339. Our compiler options of the CX400 are “-openmp -parallel -static-intel -O3 -ipo -xAVX –mkl”. Since the CX400 does not have a profiler, execution time is measured with the “gettimeofday” function.

2. Explicit MPS method

The explicit MPS method of Shakibaenia et al. [14] was proposed to analyze a weakly compressible flow with a free surface. The equation of state for pressure calculation is solved explicitly. Introduction of artificial weakly compressibility under a low Mach number (0.1 to 0.2) enables a high-speed explicit calculation of pressure and a good approximation of incompressibility. The governing equations are the equation of state and Navier-Stokes equations for an incompressible flow and are expressed as

\[
\frac{\partial P}{\partial \rho} = c^2, \quad (1)
\]

\[
\frac{D\bar{v}}{Dt} = -\frac{1}{\rho} \nabla P + v\Delta\bar{v} + \bar{g}, \quad (2)
\]

where \( P, \rho, c, \bar{v}, v \) and \( \bar{g} \) are pressure, density, sound speed, velocity, kinematic viscosity coefficient of fluid, and gravity, respectively. Let the explicit MPS algorithm discretized for the governing equations (1) and (2) be

\[
\bar{v}^* = \bar{v}^k + \Delta t (\bar{v}[\Delta\bar{v}]^k + \bar{g}), \quad (3)
\]

\[
\bar{r}^* = \bar{r}^k + \Delta t \bar{v}^*, \quad (4)
\]

\[
P_{i}^{k+1} = c^2 \rho \frac{n_{i}^* - n_{i}^{n}}{n_{i}^{n}}, \quad (5)
\]

\[
\bar{v}^* = \frac{\Delta t}{\rho} [\nabla P]^{k+1}, \quad (6)
\]

\[
\bar{v}^{k+1} = \bar{v}^* + \bar{v}^*, \quad (7)
\]

\[
\bar{r}^{k+1} = \bar{r}^* + \Delta t \bar{v}^*, \quad (8)
\]

where \( \bar{r} \) is position of a particle, \( \Delta t \) is time increment, superscript \( k \) is the time step num-
ber, superscript * is the temporal value at the time, subscript i is the particle number, n are the number of particles in a unit volume and $n^0$ is particularly the number of particles in a unit volume at an initial state. In the MPS method, the governing equations are discretized by replacing the differential operators with approximating discrete operators.

The particle interaction models of MPS assume that particles within a radius of an interaction domain are regularly located in a grid. However, since this assumption is not satisfied for the random distribution of particles including boundaries, the computational accuracy there is low, as pointed out by Tamai et al. [15]. A higher-order spatial derivative scheme such as the moving least squares particle [16,17] and the reproducing kernel particle methods [18] have been proposed to solve the problems. In this paper, spatial discretization schemes with arbitrary high order consistency based on a Taylor expansion proposed by Tamai et al. [15] is adopted.

3. Algorithm

3.1. Parallel algorithm

The explicit MPS algorithm of the hierarchical domain decomposition described in this section is a modification of the distributed parallel explicit particle method of single bucket-based domain decomposition [1]. The hierarchical domain decomposition of two levels and the data management by buckets are newly developed in this study.

First, after a bounding box of a whole analysis domain is defined, the bounding box is filled with buckets of cubes as shown in Fig. 2. Since the influence radius of a particle is defined in the MPS method, the width of the bucket must be larger than the influence radius. All particles are assigned to buckets. In Figs. 2, 3, 4 and 5, particles and buckets are expressed as circles and squares, respectively. Next, the bucket-based domain decomposition is performed with an equal number of particles in each processing element (PE) as shown in Fig. 3. Each color of buckets in Fig. 3 represents a subdomain assigned to each processing element.

In this research, the domain decomposition is performed by METIS and ParMETIS [5,6,7]. METIS is a library for partitioning graphs developed by George Karypis at the University of Minnesota. Since METIS is fast and robust, it has been used in a wide variety of applications. ParMETIS is the version for parallel computers. Since a graph consisting of the vertices and edges provides the input data for METIS, if METIS is used for the buckets, the buckets are converted into a graph whose vertices correspond to the elements of the buckets and whose edges correspond to the faces of the buckets. The number of particles in a bucket is assigned to the weight of the vertex of the graph because we want to equalize the number of particles in the subdomains.

Each decomposed subdomain is expanded from the boundaries by one bucket width. The particles in the expanded domains are assigned to one processing element as shown in Fig. 4. The regions expanded by one bucket width (areas of faint color in Fig. 4) are called “halos”. The consistency of the particle data in a halo is maintained by a halo exchange of communication between neighboring processing elements. Arrows in Fig. 4 represent the appearance that particle data of the other processing elements is sent to halo of processing element 0. In the distributed parallel explicit MPS algorithm, halo exchange is performed three times in one time step after (4), (5), and (8). If an imbalance in the number of particles among subdomains
appears as the analysis progresses, domain decomposition by METIS is performed again to recover the balance of the number of particles.

Figure 2: Assigning particles into buckets

Figure 3: Domain decomposition for buckets

Figure 4: Halo exchange pattern

Figure 5: Hierarchical domain decomposition of two levels

In the present explicit MPS algorithm of the hierarchical domain decomposition, the analyzed domain is decomposed into two steps as shown in Fig. 5. The large decomposed unit
of the first hierarchy level is called a “Part”, and the smaller unit in the decomposed “Part” (the second hierarchy level) is called a “Subdomain”. In a parallel computer, one “Part” must be assigned to one processing element, as each “Part” is further partitioned into a number of “Subdomains”. If the dynamic scheduling of OpenMP is assigned to the loop of “Subdomains”, OpenMP automatically performs dynamic load balances among “Subdomains”.

3.2. Rearrangement in memories of particles by buckets

In the MPS method, the interaction of particles with each other within the radius of an interaction domain is calculated. When particles within a radius of an interaction domain are searched, only particles belonging to $3 \times 3 \times 3$ buckets with a center on the bucket to which the target particle belongs need be searched. This algorithm to search particles within a radius of an interaction domain is also called a “cell linked lists” [19,20,21,22] of an algorithm to search all atom pairs within a cut-off distance in molecular dynamics simulations.

When the bucket structure is used, all particles must be assigned to corresponding buckets. Here, each bucket must have a sequence of particle numbers assigned to each bucket in some way. When particles within the radius of an interaction domain are searched, the particle data are accessed through the particle number sequence of the bucket. In this case, the array of the particle data, such as position, velocity, and pressure, can be used without changing the order in the array from the beginning to the end of the analysis. For example, if 6 particles are assigned into 4 buckets using a standard one-way linked list as shown in Fig. 6 (a), two arrays of

\[
\text{FirstIdxInBucket}[4] = \{0, 4, -1, 1\}, \\
\text{NextIdx}[6] = \{2, 3, -1, 5, -1, -1\}
\]

must be prepared. In the array FirstIdxInBucket[], the first particle number in each bucket is written. If the component of FirstIdxInBucket[] is “-1”, the bucket is empty. In the array NextIdx[], the particle number linked from each particle is written. If the component of NextIdx[] is “-1”, more particles than that do not exist in the bucket.

If the particle data array is accessed using the linked list, read in memories become “Indirect access” in order to be accessed through NextIdx[]. As the analysis progress, neighboring particle data in the particle data array in memories often became far in the analysis area. As shown in Fig. 7 (a), even if a particle number increases by 1, the bucket to which a particle belongs is often far away from the previous bucket. Arrows in Fig. 7 represent the movement of the bucket to which a particle belongs when a particle number increases by 1. In this case, since all particle data in a bucket must be re-stored in an L1 cache, the calculation becomes less efficient.

On the other hand, particle data can be rearranged or sorted by buckets after particle numbers are assigned to each bucket. The particle data need not be sorted in each bucket. If the indices of boundaries of particle numbers between buckets exist, the particle numbers assigned to each bucket need not be used. For example, as shown in Fig. 6 (b), if particle data are rearranged by buckets, the indices of boundaries of particle numbers between buckets can be prepared as an array of
The particle data in the bucket “n” exists from NumParticleInBucket[n] to NumParticleInBucket[n+1]-1, read in memories become “Direct access” in order to access directly particle data. Since neighboring particles in the analysis area are stored in neighboring position in memories, as shown in Fig. 7 (b), if the target particles remain in the same bucket, the same neighboring particles data can be used again (there is no need to re-store in L1 cache). Even if the target particles move to a neighboring bucket, two-thirds of the particles data remain stored in the L1 cache.

![Bucket 0 Bucket 1](image1)

(a) Particles belonged in buckets     (b) Particles rearranged by bucket after (a)

Figure 6: Particles belonging to buckets

![Bucket 0 Bucket 1](image2)

(a) Before rearranged by bucket     (b) After rearranged by bucket

Figure 7: Access pattern of buckets

Here, a comparison of before and after being rearranged by bucket is demonstrated by the dam break problem [2]. The analysis area is 1.0 m × 0.2 m × 0.6 m, the water column is 0.25 m × 0.2 m × 0.5 m, the time is 1 second and \( r_e = l_0 \times 2.1 \), where \( l_0 \) and \( r_e \) are the initial spacing between particles and the radius of the interaction domain, respectively. 1 processing element of 16 threads of the CX400 at Kyushu University was utilized for this analysis.

Figures 8 and 9 are wall-clock time per time step in the cases of \( l_0 = 0.01 \) and \( l_0 = 0.005 \), and the numbers of particles are 842,416 and 416,016, respectively. The green line represents the time for assigning particles to buckets and making a linked list for each bucket. The blue line shows the time for assigning particles to buckets and rearranging particle data by buckets. The purple line is the time for performing the total MPS calculation by linked list. This calculation is called “Indirect access” in this paper. The red line is the time required for performing the total MPS calculation by rearranging particle data after assigning particles to buckets. This calculation is called “Direct access” in this paper.
As shown in Figs. 8 and 9, when the purple and red lines are compared, the wall-clock time of the red lines is smaller than that of the purple lines, and the wall-clock time of the green and blue lines is quite a lot smaller than that of the red and purple lines. From these results, we can find two facts. First, the wall-clock time of the rearrangement of particle data is small for the total MPS calculation. Second, although the difference in wall-clock time between “Direct access” and “Indirect access” is not large initially, it becomes large as the analysis progress. The reason for the second fact comes from the fact that particles are arranged initially in order of the coordinates of the particles, and the particles become mixed as the analysis progress. The neighboring particles in memories become farther in the analysis.
Journal of Advanced Simulation in Science and Engineering

area as the analysis progresses. At the end of the analysis, the difference in wall-clock time between “Direct access” and “Indirect access” varies by a factor of 1.5 to 2.0.

The L1 cache miss rate was measured using the FX10 at the University of Tokyo with a Fujitsu profiler to search for the cause of the difference in the calculation time between “Direct access” and “Indirect access”. The SPARC64IXfx of the FX10 has a 32-KB/Core L1 cache. Table 1 shows a comparison between “Direct access” and “Indirect access” in the calculation part of particle interactions requiring the longest time. This calculation time is the generation time of the moment matrix in the high-order derivative scheme. The Laplacian operator in the original MPS is used in 0 of the polynomial order.

In general, the L1 cache miss rate of a low polynomial order is larger than the L1 cache miss rate of a high polynomial order. This is because the reuse ratio of particle data in a low polynomial order becomes low, whereas the reuse ratio of particle data in a high polynomial order becomes high. Comparing the calculation times, although the calculation time for “Indirect access” is three times greater than for “Direct access” in the case of polynomial order 0, the difference is decreased as the polynomial order is increased. From these facts, we can see that the difference in the calculation times between “Direct access” and “Indirect access” stems from the L1 cache miss rate.

Table 1: Comparison between direct and indirect access ($l_0 = 0.005$)

<table>
<thead>
<tr>
<th>Radius of interaction domain: $r_e$</th>
<th>Polynomial Order</th>
<th>Average of wall-clock time per time step (s)</th>
<th>L1 cache miss rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Direct</td>
<td>Indirect</td>
</tr>
<tr>
<td>2.1$x_l_0$</td>
<td>0</td>
<td>0.530</td>
<td>1.501</td>
</tr>
<tr>
<td>2.1$x_l_0$</td>
<td>1</td>
<td>0.703</td>
<td>1.509</td>
</tr>
<tr>
<td>2.1$x_l_0$</td>
<td>2</td>
<td>1.186</td>
<td>2.048</td>
</tr>
<tr>
<td>3.1$x_l_0$</td>
<td>0</td>
<td>1.441</td>
<td>4.878</td>
</tr>
<tr>
<td>3.1$x_l_0$</td>
<td>1</td>
<td>2.213</td>
<td>5.154</td>
</tr>
<tr>
<td>3.1$x_l_0$</td>
<td>2</td>
<td>4.470</td>
<td>7.801</td>
</tr>
<tr>
<td>3.1$x_l_0$</td>
<td>3</td>
<td>10.760</td>
<td>14.584</td>
</tr>
</tbody>
</table>

4. Parallel performance studies

4.1. Status

In the explicit MPS algorithm of the hierarchical domain decomposition developed in this study, we advance development with respect to the following three points,

(a) dynamic load balance by domain decomposition,
(b) halo exchange of communication between neighboring processing elements, and
(c) increase in speed by tuning in a processing element,

with the aim of improving performance.

In the dynamic load balance of (a), to date, about a 5% difference in the number of particles between processing elements has been achieved using METIS. The dynamic load bal-

24
ance is a most important and difficult problems. One of the most difficult problems is the dis-
tribution of the fluid and wall particles for each processing element. The wall particles, which
are only required for the pressure calculation of (5), often account for 30% to 70% of the total
particles. Even if the number of particles is the same in all processing elements, the rate be-
tween the fluid and wall particles is often different. Since the calculation cost for the fluid
particles is quite a bit larger than the calculation cost of the wall particles, it is difficult to
maintain the load balance of calculation cost.

In the increase in speed by tuning in a processing element of (c), since the software pipe-
line and SIMD vectorization have become more effective due to the development of the “Di-
rect access” in Subsection 3.2, this contributes to a rise in achieved FLOPS of our software.

The halo exchange of (b) has been mostly completed at this time. In this section, the
parallel performance of the halo exchange is mainly confirmed. Therefore, we measure par-
allel performance for particle models using only fluid particles.

In this section, the FX10 at the University of Tokyo is used to measure parallel perfor-
mance with a six-dimensional mesh and torus “Tofu” interconnects. In the FX10, we can
configure “Parts” to specified processing elements using the function of the FX10 “FJM-
Pl_Topology_rank2xy”. Since we want to measure an ideal condition for the halo exchange,
grid-like domain decomposition, as shown in Fig. 10 (a), is adopted instead of a domain
decomposition by ParMETIS as shown in Fig. 10 (b). Table 2 shows the three-dimensional
configurations of the parallel processing elements.

![Domain decomposition](image)

(a) Grid-like domain decomposition  (b) Domain decompositon by ParMETIS

Figure 10: Domain decomposition

<table>
<thead>
<tr>
<th>Table 2: Three-dimensional configurations of parallel processing elements (PEs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEs</td>
</tr>
<tr>
<td>X</td>
</tr>
<tr>
<td>Y</td>
</tr>
<tr>
<td>Z</td>
</tr>
</tbody>
</table>

4.2. Parallel scalability

In this subsection, the strong scaling and the weak scaling of our parallel MPS algorithm are
measured. Strong scaling is defined as how the calculation time varies according to the num-
ber of processing elements for a fixed total problem size. Weak scaling is defined as how the
 calculation time varies according to the number of processing elements for a fixed problem
size per processing element. Strong scaling and weak scaling efficiency from m to n are re-
spectively defined as \((T_m*n)/(T_n*m)\) and \(T_m/T_n\), where \(m\) and \(n\) \((m < n)\) are the number of pro-
cesses and $T_a$ and $T_b$ are wall-clock time. Here, $m$ is the reference point, and $m=12$ in this section. The viscosity term in (3) is calculated using a second-order polynomial approximation, and the pressure gradient term in (6) is calculated using a first-order polynomial approximation in this section. By using their polynomial approximations, the viscosity and pressure gradient term become first-order accurate approximations.

Figure 11: Strong scaling of 314,572,800 particles

Figure 12: Weak scaling of 65,536 particles per a processing element
Figure 11 shows the strong scaling test result of 314,572,800 particles model where 640 x 960 x 512 fluid particles are arranged at equal distances. Fig. 12 shows the weak scaling test result of 65,536 particles model where 32 x 64 x 32 fluid particles are arranged at equal distances per processing element. In both the strong and weak scaling test results, patterns of grid-like domain decomposition according to configurations of PEs in Table 2 are adopted. We can see that the strong and weak scaling have high performances, as shown in Figs. 11 and 12. Therefore, we could argue that the part of the halo exchange of (b) has been mostly completed.

Table 3: Wall-clock time and performed efficiency to peak for \( r_c = 2.0 \times l_0 \)

<table>
<thead>
<tr>
<th></th>
<th>12 processing elements</th>
<th>4800 processing elements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (s)</td>
<td>Performed efficiency to peak (%)</td>
</tr>
<tr>
<td>Total calculation time</td>
<td>0.138</td>
<td>3.898</td>
</tr>
<tr>
<td>Viscosity term</td>
<td>0.024</td>
<td>9.657</td>
</tr>
<tr>
<td>Collision</td>
<td>0.009</td>
<td>5.830</td>
</tr>
<tr>
<td>Pressure</td>
<td>0.015</td>
<td>5.493</td>
</tr>
<tr>
<td>Pressure gradient term</td>
<td>0.033</td>
<td>5.154</td>
</tr>
<tr>
<td>Other</td>
<td>0.058</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Wall-clock time and performed efficiency to peak for \( r_c = 4.0 \times l_0 \)

<table>
<thead>
<tr>
<th></th>
<th>12 processing elements</th>
<th>4800 processing elements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (s)</td>
<td>Performed efficiency to peak (%)</td>
</tr>
<tr>
<td>Total calculation time</td>
<td>0.566</td>
<td>7.102</td>
</tr>
<tr>
<td>Viscosity term</td>
<td>0.150</td>
<td>11.214</td>
</tr>
<tr>
<td>Collision</td>
<td>0.053</td>
<td>7.474</td>
</tr>
<tr>
<td>Pressure</td>
<td>0.087</td>
<td>7.278</td>
</tr>
<tr>
<td>Pressure gradient term</td>
<td>0.185</td>
<td>7.087</td>
</tr>
<tr>
<td>Other</td>
<td>0.091</td>
<td></td>
</tr>
</tbody>
</table>

Tables 3 and 4 show the wall-clock time and performed efficiency to peak for \( r_c = 2.0 \times l_0 \) and \( r_c = 4.0 \times l_0 \), respectively. The performed efficiency to peak is defined as achieved FLOPS over a theoretical peak performance. The performed efficiency to peak was measured with a Fujitsu profiler of the FX10. The item Other in the left column is the halo exchange, the generation of communication tables, the assignment of particles to buckets and the arrangement of particle data. Since the viscosity term is calculated using the second polynomial approximation, the reuse of particle
In the viscosity term, since the loop of neighboring particles is high, the performed efficiency to peak becomes high. In the pressure gradient term, since the loop of neighboring particles is performed twice to obtain minimum pressure, it takes the longest for the calculation of the pressure gradient term.

In Table 4, even although the wall-clock time of 4800 processing elements is larger than the wall-clock time of 12 processing elements, the performed efficiency to peak of 4800 processing elements is larger than the performed efficiency to peak of 12 processing elements. Since the number of interior buckets of 4800 processing elements is more than that for 12 processing elements, the halo area of 4800 processing elements is larger than that for 12 processing elements. Therefore, the size of the MPS calculation of 4800 processing elements is larger than the size of the MPS calculation of 12 processing elements.

5. Tsunami run-up analysis for Ishinomaki City

5.1. Analysis conditions

In this research, we ran tsunami analyses, such as one on the inundation of the Ishinomaki urban area in which two 10-m diameter tanks were carried along by the tsunami resulting from the Great East Japan Earthquake that occurred on March 11, 2011. Figure 5.1 shows the location of the analysis area, Ishinomaki City. Figure 13 (d) shows the installation location and the drift location of one of the tanks. The land surface is generated from 5-m spacing digital elevation data from the Geospatial Information Authority of Japan. The sea depth is obtained from JTOPO30 (1-km spacing) and the river depth is assumed as 10 m. The three-dimensional buildings are generated from data prior to the Great East Japan Earthquake.

![Location of Miyagi Prefecture](image1)
![Ojika Peninsula](image2)
![Ishinomaki bay](image3)
![Trajectory of floating tanks in Ishinomaki urban area](image4)

Figure 13: Location of analysis area (from Google Map)
Since it is difficult to solve a shallow water equation for the area from the sea to dry land, solutions for a shallow water equation around 1.5 km off Ishinomaki bay is used in this study. If the same spacing particles were used from Ishinomaki bay to the Ishinomaki urban area, a large number of particles would be required. In this research, we target analyses of tens of millions to hundreds of millions of particles at a maximum due to computational resource issues. Therefore, we performed a zoom-up analysis between Ishinomaki bay and the coastal area (4.0 km × 3.5 km) and the Ishinomaki urban area (400 m × 550 m). Figures 13 (c) and (d) show Ishinomaki bay and the coastal area and the Ishinomaki urban area, respectively.

5.2. Inflow and outflow boundary conditions

In this study, the inflow and outflow boundary conditions are set on the boundaries of an analysis area as shown in Fig. 14. In the case of the inflow boundary condition, wall particles arranged outside an analysis area are moved in the direction toward the inside of the analysis area. If the wall particles come within the analysis area, the wall particles are changed to fluid particles and new wall particles are generated behind the outermost wall particles. Additionally, all particles in an area higher than the water height on the boundaries and the inside area from the boundaries by $l_0$ are deleted. This procedure makes the water height around the boundary to be constant.

In the case of the outflow boundary condition, wall particles arranged outside an analysis area are treated as being static, and all the particles in an area higher than the water height on the boundaries and the area inside of the boundaries by $l_0$ are deleted. Ideally, although wall particles arranged outside an analysis area should be moved in the direction toward the outside of the analysis area; in the case of the outflow boundary condition, new wall particles cannot be generated behind the outermost wall particles. In our analyses, since both the anaseism and backwash of the tsunami occur several times, the inflow and outflow are often switched. Since specified boundaries do not always remain in an inflow or outflow state, an outflow boundary condition such as this has reluctantly been adopted in order to keep wall particles neatly arranged.

(a) Inflow boundary condition (b) Outflow boundary condition

Figure 14: Inflow and outflow boundary conditions
5.3. Ishinomaki tsunami analysis of coastal area

In this subsection, the tsunami run-up analysis for Ishinomaki bay and the coastal area is described. The analysis area is 4.0 km × 3.5 km. Let the initial spacing between particles be 1.0 m. In this analysis, the sound speed 200 m/s is adopted such that Mach number becomes less than 0.1. Time increment in each time step is determined such that Courant number becomes 0.1.

The inflow and outflow data provided by the Tsunami Engineering Laboratory of Tohoku University and KOKUSAID KOGYO CO., LTD is set on the bottom (south) side of Ishinomaki bay and coastal area in Fig. 15. The inflow and outflow data are the result of a simulation of the Great East Japan Earthquake obtained by solving a shallow water equation. Particle wall is put on the other side of Ishinomaki bay and coastal area in Fig. 15. In the analysis, the standard Dirichlet and Neumann boundary conditions of pressure are explicitly set for the free surface and the others, respectively.

About 7 days were required for this analysis of 800 sec using 96 processing elements of the FX10 at the University of Tokyo. The maximum number of particles was 260 million. The wall-clock time of one time step is an average of 5.5 sec. Figure 15 is a transparent visualization created by the particle-based rendering developed by Tanaka et al. [23,24,25]. In Fig. 15, transparency over the sea and river is increased and transparency over dry land is decreased.

Since the tsunami caused by the Great East Japan Earthquake came through from south of the Ojika Peninsula, the tsunami struck Ishinomaki bay southeast to northwest as shown in Fig. 13 (b). On the other hand, we know that the tanks were carried from the southwest to northeast in the Ishinomaki urban area as shown in Fig. 13 (d). The first reason for this is that the tsunami was turned east by breakwaters in Ishinomaki bay. The second reason is that the tsunami was going upstream along the former Kitakami river and overflow from the river. These phenomena were simulated and the results are shown in Fig. 15 (c) and Fig. 16 (b). In this analysis, we have succeeded in demonstrating the flow direction of the tsunami for the Ishinomaki urban area by means of detailed three-dimensional land surface data.

![Figure 15: Tsunami run-up analyses for Ishinomaki bay and coastal area](image)
5.4. Ishinomaki tsunami analysis with two floating objects in the urban area

We performed a zoom-up analysis between Ishinomaki bay and the coastal area (4.0 km × 3.5 km) and the Ishinomaki urban area (400 m × 550 m). In this subsection, the event whereby two 10-m diameter tanks were carried along by the tsunami is simulated. The two tanks are regarded as rigid bodies. This analysis is a problem of weak coupling due to the interplay between rigid bodies and fluid using the procedure in [3]. In [3], the rigid bodies are represented by particles with fixed relative configurations and the rigid body particles are calculated using the same procedure with fluid particles. Translations and rotations of rigid bodies are calculated from rigid body particles. Finally, the positions of the rigid body particles are calculated and replaced by the motions of the rigid bodies. Since a fragmentation phenomenon is not simulated in our research, the restraint of the two tanks is only released at 60 seconds. The boundary conditions for the Ishinomaki urban area in Fig. 16 (b) are generated along the red lines in Fig. 16 (a) for the zoom-up analysis. In this analysis, values interpolated from particle data for grid positions arranged at equal distances on the boundary lines of the Ishinomaki urban area in Fig. 16 (b) are used as the inflow and outflow conditions.

![Location of Ishinomaki urban area](image1.png)

(a) Location of Ishinomaki urban area

![Extraction of boundaries](image2.png)

(b) Extraction of boundaries

Figure 16: Location of Ishinomaki urban area

The analysis area is 550 m × 400 m. Let the initial spacing between particles be 0.5 m. In this analysis, the sound speed 200 m/s is adopted such that Mach number becomes less than 0.1. Time increment in each time step is determined such that Courant number becomes 0.1.

About 6 hours were required for this analysis of 200 seconds using 600 processing elements of the FX10 at the University of Tokyo. The maximum number of particles was 40 million. The wall-clock time of one time step is on average 0.25 seconds. The Courant number is 0.1. If the initial spacing between particles were changed to 0.2 m under the same conditions, about 30 days would be required for this analysis of 200 seconds using 32 processing elements of the CX400 at Kyushu University. The maximum number of particles reached 390 million particles. The wall-clock time of one time step is an average of 11.7 seconds. Figure 17 is a visualization made using transparency points for fluid particles and opacity surfaces for structural objects such as tanks and buildings using ParaView. Figure 18 is the sectional side view obtained by opacity point sprite.
From Fig. 17 (b), we can see that the tsunami is running up along an avenue from south to north and a prefectural road from west to east. Figures 17 (a), (b), (c), and (d) show the contracted flow phenomena generated by the rapid flooding of the tsunami among buildings. Figures 17 (c) and (d) show that the two tanks are carried along at high speed. The flow direction of the two tanks in Figs. 17 (c) and (d) is the same as the direction of one tank carried from southwest to northeast on the day of the Great East Japan Earthquake in Fig. 13 (d). In this analysis, we succeeded in demonstrating the flow direction of the tanks in the Ishinomaki urban area by the use of detailed three-dimensional buildings data, although the tanks are moving out of the analysis area.
6. Conclusion

In this paper, a distributed parallel explicit MPS algorithm of hierarchical domain decomposition was developed. The development offers two improved points from the distributed parallel explicit MPS algorithm of single domain decomposition [1].

The first improved point is the hierarchical domain decomposition of two levels. Since the units of the problem become smaller, the locality of data increases. Additionally, once a “Part” is decomposed into “Subdomains”, even calculation algorithms that are difficult to parallelize can be automatically parallelized by “Subdomains”. Dynamic load balance among “Subdomains” (the second hierarchy level) is automatically performed by the “dynamic schedules” of OpenMP.

The second improved point is that particle data are managed by buckets. The read in memories becomes “Direct access” instead of “Indirect access” by linked lists, which leads to the decrease of the L1 cache miss rate. Additionally, since neighboring particles in the analysis area are stored in neighboring position in memories, when the target particles remain in the same bucket, the same neighboring particles data can be used again (there is no need to re-store in the L1 cache). Even if the target particles move to a neighboring bucket, two-thirds of the particle data remain stored in the L1 cache.

By these improvements, with ideal particle arrangements and domain decompositions, the strong scaling efficiencies from 12 PEs to 4800 PEs in the cases of $r_e = 2.0 \times l_0$ and $r_e = 4.0 \times l_0$ are 0.823 and 0.931, respectively. The weak scaling efficiencies from 12 PEs to 4800 PEs in the cases of $r_e = 2.0 \times l_0$ and $r_e = 4.0 \times l_0$ are 0.952 and 0.957, respectively. Performed efficiency to peak of 4800 PEs in the cases of $r_e = 2.0 \times l_0$ and $r_e = 4.0 \times l_0$ are 3.810% and 7.246%, respectively.

Finally, we applied our developing method to tsunami analyses, such as one on the inundation of the Ishinomaki urban area by the tsunami resulting from the Great East Japan Earthquake.

Acknowledgements

This research was financially supported as the JST CREST project “Development of a Numerical Library based on Hierarchical Domain Decomposition for Post Petascale Simulation”. The computational resource of the Fujitsu FX10 was awarded by the “Large-scale HPC Challenge” Project, Information Technology Center, the University of Tokyo. This research was supported in part by the results of the HPCI and JHPCN Systems Research Projects. (Project ID hp120232 / jh130031). The authors also wish to thank Professor Kentaro Imai and Professor Shunichi Koshimura of Tohoku University and KOKUSAI KOGYO CO., LTD. for the tsunami inlet boundary conditions, the City of Ishinomaki for the map data, and all the members of the ADVENTURE project for their cooperation.

References


