Regular Paper

Parallel Cloth Simulation with Adaptive Mesh Refinement and Coarsening Using OpenMP on Fujitsu HPC2500

Alam Mujahid,† Koh Kakusho,†† Michihiko Minoh,††
Yasuhiko Nakashima,††† Shin-ichiro Mori† and Shinji Tomita†

In this paper, we propose a parallel implementation of simulating shape as well as force acting on cloth using particle based model on Fujitsu HPC2500 machine. Inherent degree of parallelism, iterative nature of particle based model and high computational cost make it feasible to employ the parallelization using OpenMP. Since each particle is linked with its neighbors only, parallel model just needs to resolve some reduction and synchronization issues. In many cases, some regions of cloth mesh are more responsible for simulation while other regions have insignificant role. The efficient solution is to adaptively increase or decrease the mesh resolution in particular regions. Adaptive mesh refinement combined with coarsening (AMRC) reduces the simulation time of sequential algorithm but faster application demands its parallel implementation. Mesh density varies during execution due to AMRC and requires the dynamic redistribution of load. For this purpose, a load balancing scheme based on 'Active-Lists' has been developed to simulate the cloth. This scheme is compared with the dynamic scheduling constructs available in OpenMP. Result describes that we have succeeded to parallelize the cloth simulation with a satisfactory efficiency.

1. Introduction

In the community of computer graphics, visual representation of cloth has been implemented by employing the mass-spring model, finite element model and particle-based model. It has been studied that shape (position) of cloth over the time in a model can be determined by using the force integration [5),6),17),19)] or energy minimization [1),10),16)] during animation. However, only visual information is not enough for virtual manipulation. User can only see the generated or animated images but cannot feel it. Inclusion of haptic feedback (force) to represent the mechanical behavior enhances the realism. Both aspects, shape and force, are required to represent the cloth in a virtual environment. We have considered, for the first time, the relation of force with shape deformation for soft objects like cloth [9),10)]. It holds the reality of shape, reality of force and reality of relation between force and shape. It utilizes the particle-based model, empirical data obtained from Kawabata Evaluation System (KES) and minimization algorithm.

Cloth has a variety of visual (drape) effects and a denser mesh can only represent its realistic model that needs a very high computational cost. In a cloth containing wrinkles or interacting with user, some regions of the mesh are more responsible for simulation while others play insignificant role. The efficient solution is to adaptively increase or decrease the mesh resolution in particular regions. This phenomenon introduces the concept of adaptive meshes in cloth simulation to control the computational time. Previous work [5),6),8),13),19)] has used the adaptive refinement in different ways and for different applications. Later Villard [17)] has modified the mechanical model that is more appropriate for adaptive refinement. However, adaptive coarsening is not considered. Volkov [18)] has introduced the refinement and simplification for cloth meshes and refinement is done by using the vertex insertion and edge swap but without adjusting the mechanical model. We have already reported the comparison of adaptive coarsening and adaptive refining [11)], which reflects that adaptive coarsening is better than refining. Later, a combination of adaptive mesh refining with coarsening (AMRC) is implemented that initiates simulation with finest mesh. Mesh density varies adaptively by adjusting the mechanical model accordingly during the course of simulation [12)].

Optimization of computational time is a challenging task in the realistic simulation of deforming cloth. A speed up in sequential algorithm is achieved by employing the AMRC but still simulation time is very large. So, cloth

† Graduate School of Informatics, Kyoto University
†† Academic Center for Computing and Media Studies, Kyoto University
††† Graduate School of Economics, Kyoto University

size cannot be increased for more reality and larger applications. The ultimate choice is a parallel simulator to further enhance the speed. OpenMP, a shared memory based model, helps user in implementing an easy parallel programming [3] for particle based model of cloth. Communication and synchronization overheads of OpenMP are discussed in Ref. 2), which limit the performance of a parallel program. A fast simulation for a flag has been done in Refs. 14), 15) but it is restricted to a small size cloth. R. Lario, et al. [8] have developed a parallel model of multi-level cloth using OpenMP in which coarser mesh is used for overall shape representation while finer mesh is applied for small-scale features of cloth to accelerate the convergence of optimization process. This technique is similar to multi-grid model proposed in Ref. 13) and un-necessarily refines or simplifies some regions of the cloth like Ref. 19), which may reduce the efficiency for very large mesh density.

Parallel programming for uniform density of a mesh is very easy and distribution of work among processors can be decided in advance. On the other hand, non-uniform mesh generated by AMRC adds complexity to parallel implementation. Therefore, extra efforts are required to divide the mesh such that each processor has approximately the same amount of work with minimum overheads. For this purpose, we propose a dynamic load balancing scheme based on 'Active-Lists'. When mesh density is changed by AMRC, lists of elements are created according to the new density of particle model and load is equally distributed among processors on the basis of these lists. The implementation of parallel algorithm with our load balancing scheme has successfully achieved the good performance.

2. Model Description

Kawabata Evaluation System (KES) [7], a fabric-testing device, has been globally used. KES data has been utilized in Refs. 1), 16) without considering the all conditions applicable during the KES measurement process. Cloth is a deformable object by stretching, bending and shearing to describe the basic properties of a cloth. KES characteristics describe the relation between force and shape of cloth, which are unidirectional (force to shape or vice versa) and hysteretic.

Cloth model is comprised of $I \times J$ mesh of particles that uses the KES data as reference

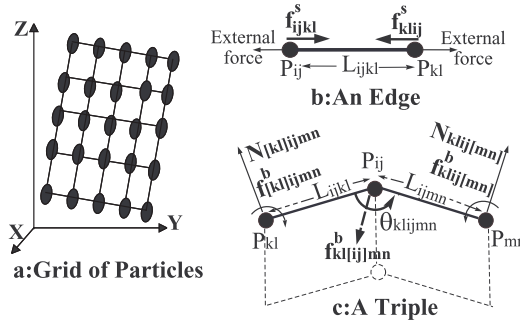


Fig. 1 Cloth model.

and energy minimization scheme is employed for simulation in this model. The crossing of warp and weft thread represents a particle. We have developed cloth model by considering three main assumptions. (1) Initial state at each simulation step works as internal variable that keeps the track of previous history. (2) KES curve works as boundary value. (3) All other hysteretic cycles lie within this boundary.

Two adjacent particles make an edge that reflects the stretching property. Similarly three adjacent particles make a triple, which is used to calculate the bending as shown in **Fig. 1**. $\mathbf{X_{ij}}$ is the position of a particle, f^s is the internal stretching force of an edge and f^b is the bending force for a triple. All other variables are function of above variables. There are three types of elements; particles, edges and triples defined in mathematical model. We are using the term element for a particle or an edge or a triple in this paper. Further detail of mathematical model is available in the work [9),10)].

Each element is characterized by energy/cost functions. We consider three factors, Motion and Gravitation (E^n), Stretching (E^s) and Bending (E^b), which are affecting the cloth. Since KES stretch curve is a relation from force to shape while KES bend curve describes the relation from shape to force, so stretching and bending properties are incorporated to have the relations in both directions. On the other hand Newton's law describes the bilateral relation between force and shape. Therefore, we are able to involve the force as well as shape at the same time. Other properties, for example shearing, are the same as stretch or bend and can be added in the model in the similar way. Each cost function is defined based on the KES data and represents the amount of violation from KES data. The cost is zero when the calculated data lies within or on the KES curve and

cost is increasing outside the KES curve. The total cost function is the sum of individual cost functions.

$$E = C_n E^n + C_s E^s + C_b E^b$$

Sequential flow of simulation is described as the pseudo code in Algorithm-I. The current variables are saved and gradient is computed numerically. The equilibrium state of the cloth is realized by employing the conjugate gradient minimization algorithm, which gives the new set of variables. Then cost is computed after updating the variables. This process continues until either the cost reaches the tolerable value or specified number of iterations are performed.

Algorithm-I : Simulation

```
for(Simulation Loop) {
  do{ Itr ++;     /* Minimization Loop */
      SaveVariables(position, force);
      ComputeGradient(clothMesh);
      ComputeMinima(clothMesh, gradient);
      UpdateVariables(position, force);
      ComputeCost(clothMesh);
  } while(Cost> Tolerance && Itr< MaxItr);
  DisplayCloth();
 } \* End Simulation */
```

3. Adaptive Meshes

The tradeoff between speed and realism for cloth simulation can be achieved by using the adaptive mesh size. Adaptive meshes represent the cloth region with minimum deformation (flat region) by coarser mesh and deformed region by denser mesh. Its implementation can be categorized as adaptive refinement or adaptive coarsening or combination of refinement and coarsening. We are implementing the adaptive refinement together with coarsening to make the cloth model more flexible and efficient. Its implementation requires to consider the mass conservation, force distribution and adjustment of the mechanical model.

Visualization of bending is more prominent than stretching in the cloth shape because stretching deforms the cloth along the line. Bending angle or curvature has been used for the mesh refinement in Refs. 6), 17) and we are also using the bending cost function as criterion in this regard. When bending cost function increases from a threshold value then mesh is refined. Similarly when bending cost function is smaller, coarsening takes place.

In cloth simulation with adaptive mesh refinement, initial mesh density is minimum and

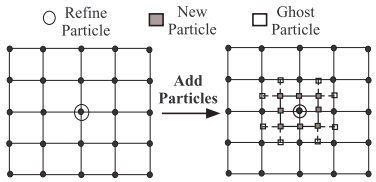


Fig. 2 Mesh refinement.

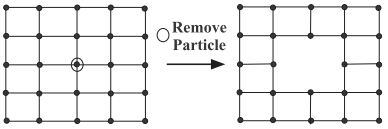


Fig. 3 Mesh coarsening.

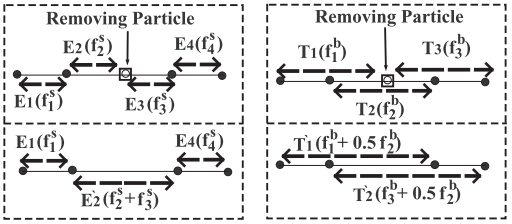


Fig. 4 Force adjustment for removing a particle.

new particles are added in some regions of cloth on demand. Refinement of a particle, adds eight new active particles and eight new ghost particles as shown in the **Fig. 2**. Ghost particles do not take part in simulation and are just used to maintain the topology of mesh. The length of a new edge becomes half and a new particle represents the 1/4 of the area as compared to coarser particle in the above level of refinement.

Coarsening is the reverse operation of the refinement and initial mesh for simulating cloth with adaptive coarsening has the maximum number of particles. It omits particles that have very small bending cost function. There are four edges and six triples linked with a particle, which are removed with a particle in coarsening process as shown in **Fig. 3**.

3.1 Adjusting Mechanical Model

KES characteristics, for both bend and stretch, are available for a specific size of cloth. These characteristics need normalization with respect to density of mesh and our model interpolates the KES curves accordingly. Therefore, it is feasible to use KES data for adaptive meshes.

In contrast to mass-spring model, calculation of bending angle in our model involves all three particles of a triple and permits to add or omit a particle from mesh. Elimination of a particle merges a triple and an edge in weft (or warp)

direction, for example see **Fig. 4**. Triple T_2 is merged into the triples T_1 and T_3 and force corresponding to T_2 is divided equally between T_1 and T_3. Similarly merging of edges E_2 and E_3, adds forces to resultant edge.

Refinement divides an edge into two edges with half length and space having four edges is represented by sixteen edges after refinement as shown in Fig. 2. Therefore, uniform force distribution over the area assigns $f^s/4$ to the finer edges. The bending parameters for refining can be adjusted in the similar manner as explained in above paragraph for coarsening but in opposite way.

Refinement reduces the area represented by a particle to 1/4, so mass should be reduced to 1/4 for finer particle to preserve the total mass. However, different masses show different response that should be tackled carefully. One way to get the same response, is to use the heavier mass as for coarser mesh and adjust it when calculating the stretching and bending. The other way is to take the lighter masses (like concept of mass density) as for finest mesh by considering that whole mesh is refined and coarser area contains more non-active particles. We are using the latter concept since it remains valid for adaptive coarsening.

3.2 Adaptive Mesh Refinement and Coarsening (AMRC)

We [11),12)] have already adopted the adaptive refining and coarsening separately and together to cloth simulation. Both adaptive refining and adaptive coarsening change the mesh density in one direction that may cause the inaccurate representation for the complex shape of cloth. Implementing adaptive refinement together with coarsening not only gives the optimum density of mesh but also makes the model more flexible and efficient. The initial cloth with flat shape is configured as maximum resolution for simulation. When it drapes over an object, deformation takes place around the boundary of object. Therefore, mesh needs coarsening to lower the density in the region away from the boundary of object.

In our simulation algorithm, omitted elements during coarsening are not completely deleted from data structure but those are just made inactive. This information is later used when refinement is required. Therefore, in case of refining, previously inactivated elements are made active in data structure. As we are neither deleting nor adding an element during the process of AMRC, the memory location remains unchanged. Mesh refinement and coarsening are employed accordingly if criteria for refining/coarsening are satisfied. This process continues throughout the simulation. Few possible mesh transitions as an example are shown in **Fig. 5**.

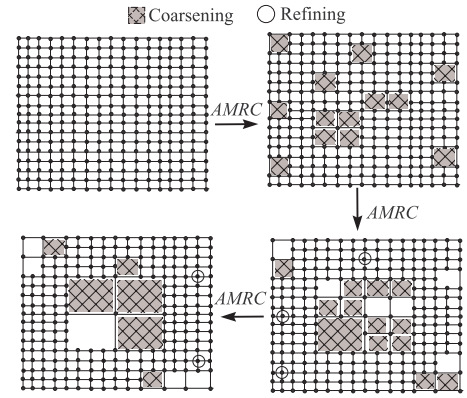


Fig. 5 Adaptive mesh refinement and coarsening.

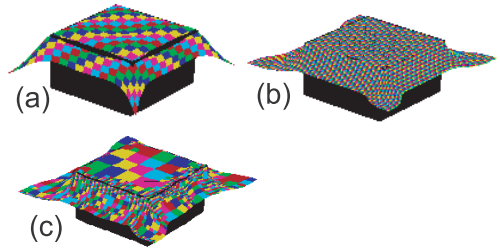


Fig. 6 Simulated image with (a) coarser mesh, (b) finer mesh, and (c) AMRC.

A sequential algorithm for cloth simulation is executed on a personal computer to observe the benefits of AMRC. First, a rectangular cloth draping over a box is simulated without utilizing the adaptive meshes. The cloth size of 77×77 particles is taken as maximum mesh density and 20×20 particles as minimum density. The simulated images with coarser and finer mesh are shown in **Fig. 6**-(a & b). Simulation with coarser mesh is faster but cloth is penetrating in the box and requires denser mesh for realistic simulation. On the other hand, simulation with finer mesh gives the best quality but needs maximum time.

Simulation with AMRC starts with maximum density and simulated image is shown in Fig. 6-(c). A test, whether to perform AMRC or not, and implementing AMRC need some computational cost. Therefore, test for AMRC is performed after three iterations by considering

Table 1 Comparison for different meshes.

Mesh	Time [sec]	No.of Particles	Average Minimized Cost
Finer	2,940	5939	0.128716
AMRC	829	5939–1787	0.17316
Coarser	176	400	0.259187

that there is no abrupt change in the model in few iterations. In the beginning coarsening is dominant because most of the cloth is flat. Later high deformation occurs near the edges of the box that needs refinement. Simulation is slower at early iterations because of higher mesh density. However, a good speed up is achieved with acceptable quality.

The cost function is defined as the violation of cloth parameters from experimental (KES) data. The simulated cloth is seemed to be as the real one when cost function is zero. Thus, minimized value of cost function indicates the quality of simulation by considering that KES data corresponds to the best quality. Therefore, we can say that minimized cost value represents the quality. So, we compromise that the simulated cloth is tolerable in quality if the simulation prohibits objects to penetrate in the cloth. The processing time and average minimized cost for 200 iterations are given in **Table 1** for three types of simulation.

It is difficult to exactly define the tolerance of quality. The tolerance of quality depends on the demand of application or user. In Table 1, data is given for a fixed number of iterations to compare the time and minimized cost. It may not be so important for another user in a different application. For example, consider the case of animation where time is the key parameter. It is required to finish the simulation within a defined span of time even the minimized cost has larger value. We have also performed the simulation using AMRC, in which minimization terminates when time reaches the 500 seconds. It takes 165 iterations and average value of minimized cost is 0.227164, which are different from the data given in Table 1.

Visual observation in Fig. 6. shows that the object is penetrating in the cloth for simulation with coarser mesh, which is not acceptable. On the other hand, the object is not visible from the surface of cloth when finer mesh and AMRC is employed. Table 1 shows that finer mesh has the smallest value of average minimized cost that is closer to KES data and reflects the relatively best quality. In case of AMRC, there is no

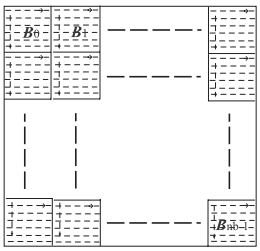


Fig. 7 Memory allocation in cyclic blocks.

object penetration and average cost value lies between the values for finer and coarser mesh. The minimized cost of AMRC can further be reduced by increasing the number of iteration but at the cost of computational time. Comparison among different simulations proves that AMRC gives a better combination of speed and quality.

4. Parallel Implementation

Minimizing the computational cost is the aim of parallel computing to achieve the more realistic simulation in different environments. Parallel implementation of cloth model is performed by using the shared memory model of OpenMP. When a parallel region is called in OpenMP, a master thread creates slave threads, iterations have to divide among processors and threads must synchronize at the end of parallel region. These additional costs are called ***Parallel Overhead*** and it increases with number of threads [2]. There is no guarantee even a code has been parallelized that its performance will be improved. Parallelization in wrong way may slow the program down. Some performance tuning is necessary to run the program acceptably fast. The over-all performance of OpenMP depends on the percentage of code that can be made parallel, granularity, load balancing, locality and synchronization among processors.

4.1 Synchronization

The cloth mesh is divided into blocks of 50×50 particles and memory is allocated in cyclic blocks as shown in **Fig. 7**. The block size is chosen without any specific reason but it should not be very small. This size may be increased or decreased depending on the application. Memory is allocated to particles, edges and triples in each block respectively. Each particle is connected to its neighbors through edges and triples, which reflects the sharing of data among neighbors.

In our model, there are three cost functions relating a particle to its eight neighbors along horizontal and vertical directions in cloth struc-

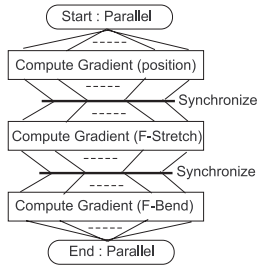


Fig. 8 Parallelization of **ComputeGradient()**.

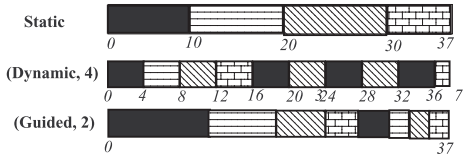


Fig. 9 Scheduling in OpenMP.

ture. These tasks may use the same data and cannot be computed simultaneously. This sharing must be resolved by employing the specific synchronization among processors that should keep the parallel overhead minimum. Therefore, we utilize the work-sharing directive of OpenMP to keep all tasks in one parallel region separated by event synchronization and parallelism is exploited within a task. **ComputeGradient()** is the most expensive function of our algorithm and its parallelization is shown in **Fig. 8**. Similar approach is applied to parallelize the rest of the functions.

4.2 Load Balancing in OpenMP

Adaptive mesh refinement and coarsening change the number of elements in the mesh. Variation in mesh density during execution, raises the necessity of dynamic load balancing. The ***schedule*** clause of OpenMP gives variety of options to specify the distribution of load among processors. An example of distributing 38 iterations among 4 processors by using static, dynamic and guided scheduling is shown in **Fig. 9**.

Static option of schedule clause simply divides the work into equal parts for all processors. As the distribution is decided at the start, it has the minimum overhead. Due to AMRC, some parts of the cloth have more elements while others have less elements for computation. As a result, faster processor has to wait for slower one, which degrades the performance.

In dynamic scheduling, a fixed amount of job is allocated to a processor on first come first served basis. When a thread finishes its work, it goes to the system runtime manager of

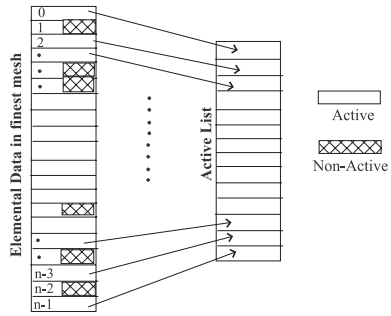


Fig. 10 Active list creation.

OpenMP and asks for a next job. The fastest thread never has to wait for the slowest thread to compute more than fixed amount of job. In this scheme, job distribution is not known in advance. Job is allocated to a processor at runtime, which incurs more overhead as compared to static scheduling.

Guided schedule starts with larger job size and job size decreases exponentially until it reaches the minimum size (given by user). It is similar to dynamic scheduling except that next job size is not fixed. This scheme not only decides the job allocation to a processor at run time but also calculates the size of next job, which is equal to the remaining work divided by the number of processors. Due to this reason, it is more costly scheme than dynamic scheduling. Its efficiency depends on the combination of job size and number of processors.

4.3 Load Balancing Using Active Lists

We propose the distribution of data on the basis of active elements in the mesh. Elements are stored in data structure for finest mesh and when coarsening omits an element its status is set to be non-active. In case of refining, status of an element is changed from non-active to active. The lists (one dimensional arrays) are created for active elements as shown in **Fig. 10**. These lists contain the pointers to active elements. The non-active elements are removed from the list after coarsening and active elements are inserted in the list when refining takes place. There are three types of elements (particles, edges and triples) that need to make three active lists. Therefore, three threads are employed to update three lists and each list in sequential way. The total number of elements in each list are distributed among processors like static scheduling. The above procedure is elaborated by an example as shown in **Fig. 11**. In this example, there are maximum 20 ele-

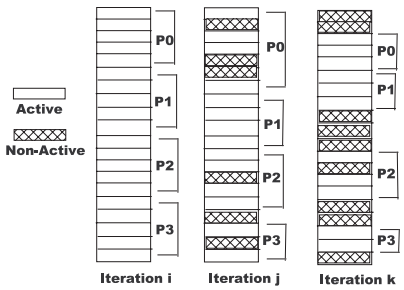


Fig. 11 Work distribution among processors.

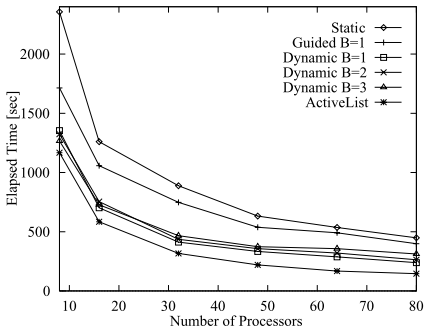


Fig. 12 Performance evaluation.

ments initially and only active elements are redistributed among four processors at different iterations.

5. Experimental Results

Parallel algorithm based on OpenMP is designed to simulate the cloth. Cloth draping over an object is simulated using the Fujitsu PRIMEPOWER HPC2500 machine. The available HPC2500 server has 96 processors, which are distributed over 12 system boards. Each system board has eight SPARC64 V processors (1.3 GHz) and 16 GB of memory. All these boards are connected with each other through a crossbar network with capacity of 133 GB/s. We are performing all experiments using batch mode, which ensures that the number of threads and processors is equal. Source code is compiled using the C-compiler **fcc** with KOMP, Kfast_GP2 = 2, V9, Klargepage=2 and Khardbarrier options.

5.1 Cloth Draping Over Box

We have simulated the flat rectangular cloth with $1,000 \times 1,000$ particles, staying just over the surface of a box as initial state. The variables forces, acceleration, velocity and cost functions have zero initial values. Simulation begins by allowing the cloth to drape over a box under gravity. Cloth deforms as simulation progresses. The AMRC is invoked only three times and number of particles is reduced from 1,000,000 to 45,995 during 100 iterations. The performance evaluation in terms of computational time and number of processors is expressed by the graph as shown in **Fig. 12**. All the given times are for 100 iterations.

Static scheduling is a simplest scheme with minimum parallel overhead. It has the poorest performance due to un-even distribution of load among processors as shown in Fig. 12.

Guided scheduling is done with minimum size equal to one block. Job size becomes very small at the end, specially for large number of processors and more synchronization cost is needed. As a consequence, its performance lies between static and dynamic scheduling.

We are employing the dynamic scheduling with minimum job-size equal to one, two and three blocks to determine the effect of size. The choice of job size is a compromise between quality of load balancing and synchronization costs. The load balancing improves as job-size decreases because fastest processor has to wait for less time at the end. On the other hand, small size of job increases the number of job allocations at run time. Synchronization cost increases because one synchronization per job allocation is required. Results in Fig. 12 show that the simulation with smaller job size is better for large number of processors and the simulation with larger job size is better for small number of processors. However, difference in performance corresponding to the number of blocks is not too big in our simulation.

Using Active-Lists scheme, all processors require to compute the equal number of active elements. As a result, its performance is the best one as compared to static, guided and dynamic scheduling schemes.

The computational time for cloth simulation by Active-Lists scheme is elaborated in **Table 2** corresponding to different number of processors for further analysis. It is observed that speed up first increases and then decreases. It gives the best performance for 8, 16 and 32 processors. In case of up to 32 processors, parallel overhead is very small as compared to total elapsed time. Furthermore, each processor has large enough percentage of total work to compute, which matches with the size of cache memory. On the other hand, performance slows down with larger number of processors. Therefore, it is required to investigate that why perfor-

Table 2 Timing analysis.

No. of processors	Seq. overhead [sec]	Para. overhead [sec]	Time [sec]	Speed Up
1	11.5	0.000	10,459.70	1.00
2	11.5	0.024	5,039.45	2.07
4	11.5	0.050	2,498.97	4.18
8	11.5	0.080	1,167.61	8.96
16	11.5	0.131	583.80	17.91
32	11.5	0.205	316.86	33.01
48	11.5	0.382	220.16	47.50
64	11.5	0.457	168.06	62.24
80	11.5	0.588	145.58	71.85

mance limitation occurs. The main reasons are sequential and parallel overheads. These overheads become gradually obvious as the number of processors increases over 48.

Sequential overhead in Table 2 is the time required to compute the sequential part of the code that is approximately 10 seconds and time needed to create the active lists, which is equal to 1.5 seconds. As there are three active lists for particles, edges and triples, always three threads are used to update these lists.

Parallel overhead can be factorized into thread creation, synchronization among processors and event synchronization costs. These costs increase with the number of processors (see Table 2).

If the cost of sequential and parallel overheads is deducted from the total elapsed time then result shows the linear speed up. For example, consider the case of 80 processors used in simulation. The net time spent by 80 processors in parallel computation is $(145.58 - 11.5 - 0.588) = 133.4$ seconds. In case of using one processor, parallel execution part spending $(10459.7 - 11.5) = 10448.2$ seconds might be executed in parallel. So, ideally this time would be reduced to $(10448.2/80) = 130.6$ seconds if we employ the 80 processors. These two values (i.e., 133.4 and 130.6 seconds) are approximately same. This fact means that Active-List scheme works well to achieve the best load balancing and the crossbar network of HPC2500 has not yet become a bottleneck.

5.2 Cloth Draping Over Spheres

A cloth draping over a sphere is simulated with the same initial shape and conditions as used above for a box. Result in **Table 3** describes the linear speed up, which indicates that we are able to use objects different from a box in cloth simulation. In both cases, for a box and a sphere, deformation occurs only in specified region of cloth. Therefore, another experiment

Table 3 Timing for cloth draping over spheres.

No. of processors	Time [sec] 1 sphere	Time [sec] 3 spheres
2	3,664.59	3,865.34
4	1,864.35	2,004.75
8	939.43	1,010.12
16	478.73	515.42
32	233:87	249.21
64	123.68	128.23
80	100.77	104.72

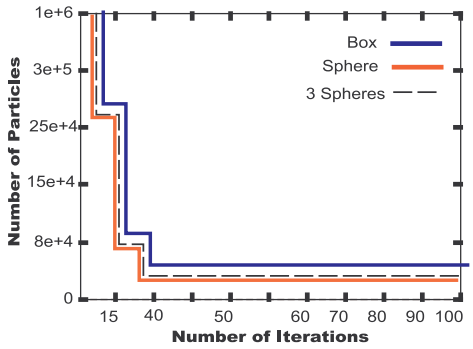


Fig. 13 Invocations of AMRC.

is performed for the case of a cloth draping over three small spheres. These spheres are placed under the cloth near top-left, top-right and mid-bottom of cloth respectively. As a result, the cloth deforms in various regions. Table 3 also shows the liner efficiency in computational time for the case of three spheres. The AMRC is invoked seven times during the course of 100 iterations. These invocations occur at different iteration numbers and produce different density for one and three spheres. The numbers of particles reduce from 1,000,000 to 30,592 and 32,832 for one and three spheres respectively (See **Fig. 13**). As AMRC is not invoked frequently, its cost is smaller as compared to the total simulation cost.

6. Discussion

6.1 Regular Versus Adaptive Meshes

Both regular and adaptive meshes have their own merits and demerits depending on the application. No doubt, implementation of regular meshes is easier. It does not require load balancing but there is a possibility that some regions are un-necessarily refined or coarsened. The overall efficiency depends on the chosen density of mesh. Choosing one level coarser mesh can boost the time efficiency at the cost of quality. Regular meshes are better than adaptive meshes when AMRC occurs frequently and

most area of the mesh needs higher density.

Adaptive meshes are better than regular meshes when the decrease in total computation time is larger as compared to the time needed for AMRC and load balancing. Moreover, its efficiency can be enhanced by parallel implementation of AMRC. Infact, in our examples of cloth draping over a box, a sphere and three spheres, AMRC is invoked only three or seven times per 100 iterations and very vast area of mesh uses coarser density.

6.2 Scalability

Scalability should be considered in terms of problem size and multiprocessor organization. Cloth with $1,000 \times 1,000$ particles needs around 1.4 GB of memory, which is so smaller than total memory 12×16 GB of HPC2500 with 96 processors. Therefore, problem size can be enlarged and algorithm will perform well for any OpenMP platform due to coarser granularity. Total processing time is of course proportional to the problem size and efficiency of the machine other than HPC2500.

In general for all parallel codes, increasing the number of processors has to face two problems: decrease in granularity and increase in parallel overhead. The performance of our algorithm will go down for a very large scale system because of sequential implementation of AMRC and updating Active-Lists. As can be seen in Table 2, the sequential time is 11.5 seconds. Even if AMRC and HPC2500 interconnection network work well in case of 1,000 processors, the computational time would remain at $11.5 + (10459.7 - 11.5)/1,000 = 21.95$, thus the speed up is 476.5. It reflects that sequential cost has dominated the parallel cost for more than 1,000 processors.

Parallel programming is relatively easier for shared memory computers but these do not scale well to hundreds of processors (1,000 processors are unrealistic), while distributed systems are scaled to thousands of computers. Employing a very large set of CPUs to get the real time simulation, requires switching from shared memory systems to cluster-based systems. Such distributed systems need mesh repartitioning and boundary exchange.

The inter-cluster communication is usually expensive, so it is required to reduce the amount of data movement. For this purpose, two dimensional block distribution is a promising candidate for our application since it can minimize the boundary between clusters. As for data migration scheme, diffusion algorithm may be a good choice that moves data only between neighbors. Indeed, we have already implemented the similar scheme in our parallelizing compiler [4]. It is also possible to perform mesh generation for AMRC in parallel on a distributed mesh. By applying these techniques, we can port our algorithm to a distributed system.

7. Conclusion and Future Work

We have successfully implemented the parallel algorithm in OpenMP for simulating the realistic force as well as shape of cloth with adaptive meshes. AMRC reduces the sequential time but increases the complexity of parallel programming. So, we have realized the effective load balancing based on the Active-Lists. Comparison with the scheduling schemes available in OpenMP reflects the good speedup achieved by our parallel model.

In this simulation, the crossbar network of HPC2500 has not yet become a performance bottleneck because it provides sufficient capability for the simulation size of 1.4 GB with $1,000 \times 1,000$ particles. In case of bigger mesh size problems, we need the effective memory allocation by which the data transfer through crossbar network might be minimized as much as possible. This memory allocation is related to the logical-physical memory address transformation done by the HPC2500 operating system. This memory allocation scheme may influence on the Active-Lists scheme for load balancing.

We have simulated the cloth containing the $1,000 \times 1,000$ particles, draping over a box, a sphere and three spheres, that is large enough for real size applications. The computational cost is still away from real time simulation. This algorithm can be employed to complex applications (for example, dress on human body) after achieving the real time simulation that is our task for future.

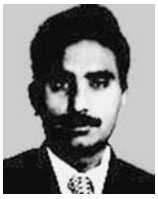
Currently visualization part is a negligible task in comparison with simulation part because simulation is computationally expensive and non-real time simulation has been performed at this stage. On the other hand, we also have a research project on parallel visualization of simulation results. So, we could utilize the results of this work if we could achieve the real time simulation of cloth with force feedback.

Acknowledgments Part of this work was supported by the Grant-in-Aid for Scientific Research (S)#16100001 and (B)#13480083 from JSPS.

References

1) Breen, D.E., et al.: Predicting the drape of woven cloth using interacting particles, *Computer Graphics* (*SIGGRAPH Proceedings*), pp.365–372 (1994).
2) Bull, J.M.: Measuring synchronization and scheduling overheads in OpenMP, *1st European Workshop on OpenMP*, Lund, Sweden (1999).
3) Chandra, R., et al.: *Parallel programming in OpenMP*, Morgan Kaufmann Publishers (2001).
4) Goto, S., et al.: Optimized code generation for heterogeneous computing environment using parallelizing compiler TINPAR, *Proc. Int'l Conf. on Parallel Architecture and Compilation Techniques*, pp.426–433 (1998).
5) Howlett, P. and Hewitt, W.T.: Mass-spring simulation using adaptive non-active points, *Computer Graphics Forum*, Vol.17, No.3 (1998).
6) Hutchinson, D., Preston, M. and Hewitt, W.T.: Adaptive refinement for mass/spring simulations, *7th Eurographics Workshop on Animation and Simulation*, pp.31–45 (1996).
7) Kawabata, S.: *The standardization and analysis of hand evaluation*, The Textile Machinery Society of Japan (1980).
8) Lario, R., et al.: Rapid parallelization of a multilevel cloth simulator using OpenMP, *3rd European Workshop on OpenMP*, Spain (2001).
9) Mujahid, A., et al.: A new approach towards modeling the virtual cloth based on realistic force and shape, *8th Int'l Conference on Virtual Systems and MultiMedia*, Gyeongju, Korea, pp.399–408 (2002).
10) Mujahid, A., et al.: Modeling virtual cloth to display realistic shape and force based on physical data, *Journal of System, Control and Information, Japan*, Vol.47, No.4, pp.183–191 (2003).
11) Mujahid, A., et al.: Comparison between adaptive refinement and adaptive coarsening for simulating realistic force and shape of virtual cloth, *MMU Int'l Symposium on Information and Communications Technologies*, Malaysia, pp.147–150 (2003).
12) Mujahid, A., et al.: Simulating realistic force and shape of virtual cloth with adaptive meshes and its parallel implementation in OpenMP, *IASTED Int'l Conf. on Parallel and Distributed Computing and Networks* (*PDCN 2004*), Austria (2004).
13) Ng, H.N., Grimsdale, R.I. and Allen, W.G.: A system for modeling and visualization of cloth material, *Computer and Graphics*, Vol.19, No.3, pp.423–430 (1995).
14) Romero, S., Romero, L.F. and Zapta, E.L.: Fast cloth simulation with parallel computers, *6th Int'l Euro-Par Conference* (*Euro-Par2000*), *Munich, Germany*, pp.491–499 (2000).
15) Romero, S., Romero, L.F. and Zapta, E.L.: Approaching real time cloth simulation using parallelism, *16th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation, Lausanne, Switzerland* (2000).
16) Sakaguchi, Y., et al.: Party: A numerical calculation method for a dynamically deformable cloth model, *System and Computers in Japan*, Vol.26, No.8, pp.75–87 (1995).
17) Villard, J. and Borouchaki, H.: Adaptive meshing for cloth animation, *11th Int'l Meshing Roundtable*, pp.243–252 (2002).
18) Volkov, V. and Ling, L.: Adaptive local refinement and simplification of cloth meshes, *1st Int'l Conference on Information Technology and Application, Australia* (2002).
19) Zhang, D. and Yuen, M.M.F.: Cloth simulation using multilevel meshes, *Computer and Graphics*, Vol.25, No.3, pp.383–389 (2001).

(Received January 31, 2004)
(Accepted May 29, 2004)



Alam Mujahid received his M.Sc. and M.Phil. degrees in Electronics from Quaid-e-Azam University, Islamabad, Pakistan in 1991 and 1994 respectively. He was affiliated with Informatics Complex, Islamabd as Scientific Officer and Senior Scientific Officer from 1993 to 1998. He is currently post-graduate student in the Graduate School of Informatics, Kyoto University. His interested research fields are computer graphics and parallel processing.

Koh Kakusho received his B.E. degree in electrical engineering from Nagoya University in 1988, and his M.E. and Ph.D. degrees in communication engineering from Osaka University in 1990 and 1993 respectively. From 1992 to 1994, he was a Fellow of JSPS, and spent a year from 1993 at the Robotics Laboratory, Stanford University as a Visiting Scholar. From 1994 to 1997, he was a Research Associate at the Institute of Scientific and Industrial Research, Osaka University. He is currently an Associate Professor at the Academic Center for Computing and Media Studies, Kyoto University. His primary research interest is information media technology especially for human-computer communication using visual media. He is a member of IEEE, ACM, IEICE, IPSJ and JSAI.

Michihiko Minoh is a professor at Academic Center for Computing and Media Studies, Kyoto University, Japan. He received the B.Eng., M.Eng. and D.Eng. degrees in Information Science from Kyoto University, in 1978, 1980 and 1983 respectively. His research interest includes image processing, artificial intelligence and multimedia applications. He is currently a leader of Trans-pacific Interactive Distance Education (TIDE) project. He is a member of IPSJ, IEICEJ, IEEE, and ACM.

Yasuhiko Nakashima was born in 1963. He received the B.E., M.E. and Ph.D. degree in Computer Engineering from Kyoto University, Japan in 1986, 1988 and 1998 respectively. He was a computer architect at Computer and System Architecture Department, FUJITSU Limited in 1988–1999. Since 1999, he is an Associate Professor in Graduate School of Economics, Kyoto University. His research interests include processor architecture, emulation, CMOS circuit design, and evolutionary computation. He is a member of IEEE CS, ACM and IPSJ.

Shin-ichiro Mori was born in 1963. He received his B.E. in electronics from Kumamoto Univ. in 1987 and M.E. and Ph.D. in computer science from Kyushu Univ. in 1989 and 1995 respectively. From 1992 to 1995, he was a research associate in the Faculty of Engineering, Kyoto University. Since 1995, he has been an associate professor in the Department of Computer Science, Kyoto University. His research interests include computer architecture, parallel processing and visualization. He is a member of IEEE CS, ACM, EUROGRAPHICS, IEICE and IPSJ.

Shinji Tomita was born in 1945. He received his B.E., M.E. and Ph.D. degrees in Electronics from Kyoto University in 1968, 1970, and 1973 respectively. He was a research associate from 1973 to 1978 and an associate professor from 1978 to 1986 in the Department of Computer Science, Kyoto University. From 1986 to 1991, he was a professor in the Department of Information Systems, Kyushu University. From 1991 to 1998, he was a professor in the Faculty of Engineering, Kyoto University. Since 1998, he has been a professor in the Graduate School of Informatics, Kyoto University. His current interests include computer architecture and parallel processing. He is a member of IEEE, ACM, IEICE and IPSJ. He was a board member of IPSJ directors in 1995, 1996, 1998 and 1999. He is a fellow of IEICE and IPSJ.