

Parallel Simulation of Large Dynamic System on a PCs Cluster: Application to Cloth Simulation

Florence Zara, François Faure, Vincent Jean-Marc

▶ **To cite this version:**

Florence Zara, François Faure, Vincent Jean-Marc. Parallel Simulation of Large Dynamic System on a PCs Cluster: Application to Cloth Simulation. International Journal of Computers and Applications, 2004, 26 (3), 10.2316/Journal.202.2004.3.202-1464 . inria-00516888

HAL Id: inria-00516888

https://inria.hal.science/inria-00516888

Submitted on 13 Sep 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

PARALLEL SIMULATION OF LARGE DYNAMIC SYSTEM ON A PCS CLUSTER: APPLICATION TO CLOTH SIMULATION

F. Zara[1], F. Faure[2], J.-M. Vincent[1]

{Florence.Zara, Francois.Faure, Jean-Marc.Vincent}@imag.fr

[1] Laboratoire ID-IMAG, projet APACHE (CNRS, INPG, INRIA, UJF)

Antenne ENSIMAG, ZIRST, 51 av. J. Kuntzmann, 38330 Montbonnot St Martin

[2] Laboratoire GRAVIR-IMAG, équipe EVASION (CNRS, INPG, INRIA, UJF)

INRIA, 655 av. de l'Europe, 38330 Montbonnot St Martin

Abstract: In this paper, we first propose a strategy to parallelise the simulation of deformable objects based on a particle decomposition. We apply this strategy to cloth simulation and we describe the implementation of SAPPE, our parallel cloth simulation, using the Athapascan parallel programming language. We present experiments with about half a million particles on a cluster of PCs showing good speedups. Then we combine this parallel simulation with the Net Juggler multi-display visualisation system.

Key Words: Cluster of PCs - Large Dynamic System - Physical Model - Parallel Cloth Simulation - Coupled Clusters Applications - Multi-Display Visualisation.

1 Introduction

Progresses in hardware and software technologies make possible the use of parallel platforms and more specially PCs clusters in image synthesis. In this domain, deformable object

simulation is an essential topic of dynamic sceneries animation. Therefore, this kind of platforms make possible simulation and visualisation of complex scenes with massive data sets.

In this paper, we first propose a strategy to parallelise simulation of deformable objects based on a particle decomposition and we apply our method to cloth simulation. The main difficulty of such parallel simulation is the result of interactions between objects and several parts of the same object. The challenge is to provide a high performance implementation that scales well as the size of the complex scene increases.

SAPPE, our parallel cloth simulation, is implemented using Athapascan [1], a parallel programming language, enabling scalability and portability on any parallel platforms architecture (SMP, cluster of PCs, cluster of SMPs). Moreover this environment builds at runtime data-flow graph and generates the dependencies graph associated in order to distribute efficiently data among processors, using a scheduler to assign tasks onto processors. We simulate large simulation with about half a million particles and we obtain good speedups on a cluster of PCs.

The second interest of our application concerns its visualisation. Visualisation appears as an efficient way to analyse results of complex simulations. Immersive environments, like CAVEs [2], enhance the visualisation experience. They provide a high resolution and large surface display created by assembling multiple video projectors. These environments are classically powered by dedicated graphics supercomputers like SGI Onyx machines.

Today, the anatomy of super-computing is quickly and deeply changing. Clusters of

commodity components are becoming the leading choice architecture. They are scalable and modular with a high performance-price ratio. Clusters have proved efficient for classical (non interactive) intensive computations. Recently the availability of low cost high performance graphics cards have foster researches to use these architectures to drive immersive environments [3]. The first goal was to harness the power of multiple graphics cards distributed on different PCs. But the scalability and performance of PC clusters allow to go beyond only distributed graphics rendering. While some cluster nodes have graphics cards to power the immersive environment, complex simulations can take advantage of extra nodes to decrease their execution time and reach an update rate suitable for interactivity.

In this paper, we propose a coupling of the parallel cloth simulation with a multi-display visualisation on a PCs cluster. The distributed graphics rendering is made using Net Juggler [3] which enables an application to use the power of multiple graphics cards distributed on different PCs.

The paper is organised as follows. The next section presents principles of dynamical system modelling and parallelisation. Section 3 explains models and integration methods used for physical cloth simulation. Section 4 presents the parallelisation of our cloth simulation written in Athapascan and gives some experiments. Section 5 presents how we combine the parallel cloth simulation with a multi-displays visualisation. We conclude in section 6.

2 Model and Parallel Methods

2.1 Discrete Model Simulation

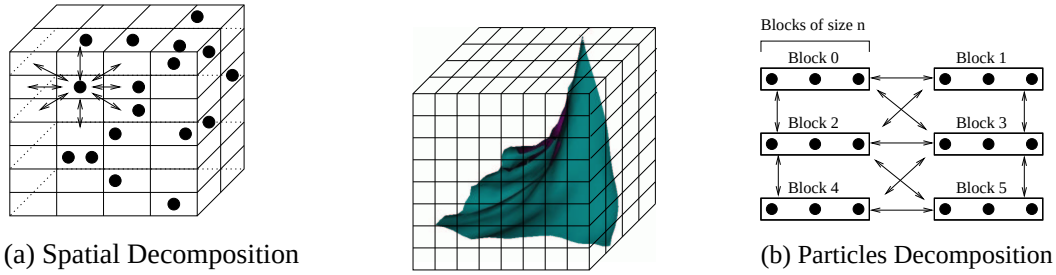
We focus essentially on simulation of deformable objects. They are represented by physical models and discretised into a mesh structure. Vertices of this mesh are called particles and the mesh topology describes how the particles interact and exert forces on each other. Each particle of the system is defined by its mass, position, velocity and forces applied to it due to its topological neighbouring.

A simulation of deformable object consists, at each time step, in the computation of a numerical solution for the equations of motion. We have: $F(t) = MX''(t)$ which is the fundamental equation of the dynamics that describes the relationship between the force F and the acceleration X'' at time t, with M a diagonal mass matrix. Therefore each simulation iteration is composed of two main parts. First, the computation of forces that act on each particle. Second, the update of each particle state (position, velocity, acceleration) by integrating the dynamic equations of the particle system. Then our goal is to decompose these computations to parallelise the simulation while taking into account the specificity of the PC cluster architecture.

2.2 Parallel Methods

The most common solution to parallelise an application is to divide it in several computation tasks and to execute them onto several processors. The major difficulty results from the dependencies between tasks due to shared data. In a simulation of deformable objects,

major dependencies come from forces computation. There exist essentially three kinds of parallel methods for computing the forces.



(a) Spatial Decomposition (b) Particles Decomposition

Figure 1: Spatial Decomposition *vs* Particle Decomposition

- **Spatial decomposition:** The physical space of the simulation is split into areas, each box being assigned to a processor (Fig. 1.a). Using this strategy, the particles forces present on a box are only computed using interactions from a neighbourhood of the box. The main drawback of this method concerns the computation load. As particles number is different between boxes, some processors could be more loaded than others.

- **Force decomposition:** This method splits the matrix of the forces between each pair of particles to balance the forces computation among the processors [4, 5]. But it does not take into account the locality of data needed for calculating these forces.

- **Particle decomposition:** This decomposition distributes particles among the processors. The object is split in several parts (the set of particles describing the object is divided in several subsets). Each processor manages a subset and computes the forces and particles states associated to this subset. The computation load of each

subset is approximately the same. Moreover, the forces tasks involving particles on different processors are mapped onto one of the processors that manages the particles, in order to balance the workload among the processors. Therefore with this strategy, all processors have approximately the same computation load. The size of the subset of particles represents the granularity of the parallel application. The parallelisation of our cloth simulation [6] follows this approach (Fig. 1.b).

These three methods have an arithmetic cost in $O(n/p)$ for a parallel simulation with n particles on p processors. Particle and spatial decompositions have a communication cost in $O(n/p)$ and force decomposition in $O(n/\sqrt{p})$.

3 Application to Cloth Simulation

In this part, we shortly describe the physical simulation of cloth. There is an interest in 3D animation of cloth simulation because it enables the realistic modelling of dressed humans. The principal stages of a cloth animation are: (a) The integration of an Ordinary Differential Equation (ODE), (b) The collision processing and (c) The rendering. This section focuses on the first stage.

3.1 Discrete Physical Model of Cloth

In computer animation, particles systems have proven to be an appropriate model for fast physically based simulation of deformable objects [7, 8, 9, 10, 11]. Cloth is modelled as a triangular mesh of n particles in space, connected by springs which enable to reproduce

realistic behaviour.

Provot [12] proposes a mass-spring system for textiles with a rectangular mesh in which the particles are connected by structural springs to counteract tension, diagonal springs for shearing and interleaving springs for bending (see Fig. 2).

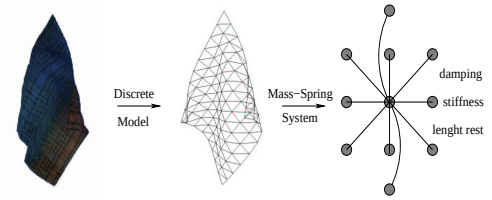


Figure 2: Triangular particles mesh connected by springs: Provot model [12]

3.2 Numerical Resolution

The acceleration of the i^{th} particle of the simulation is given by the basic Newton's law, $m_i x_i''(t) = \vec{f}_i(t)$, where $\vec{f}_i(t)$ is the force applied to this particle at time t and m_i its mass. If we define the diagonal matrix M by $M = diag(m_1, m_1, m_1, ..., m_N, m_N, m_N)$, where $m_1, ..., m_N$ are the masses of the particles, we have:

$$x''(t) = M^{-1} f(x(t), x'(t)) \tag{1}$$

The forces exerted at time t on each particle are due to springs forces (local forces) and to external forces like gravity, air-damping or wind (global forces). Consequently, the force applied on the i^{th} particle of the particle system is given by:

$$\vec{f}_i = \sum_{j|(i,j)\in E} [k_{ij}(l_{ij} - l_{0ij})\vec{u}_{ij} - \nu_{ij}(v_i - v_j)\vec{u}_{ij}] + m_i\vec{g} + \vec{F}_{external} \tag{2}$$

where E is the set of all edges in the particle system, k_{ij} and ν_{ij} are the stiffness and damping constants determining the spring properties, l_{ij} the rest lengths between the

particles i and j, l_{0ij} its lengths at time t_0, $\vec{g}$ the gravity constant, $\vec{u}_{ij} = \frac{x_i - x_j}{\|x_i - x_j\|}$ and v_i, v_j velocities projected to the u_{ij} direction.

Computing particle accelerations allows the update of positions and velocities using numerical integration. Many integration methods have been used in cloth simulation [13]. We invite the reader to refer to [14] written by Hauth and Etzmuss in 2001, which presents analysis to exploit special properties of the mechanics of deformable objects, and analyses stability for stiff equations.

3.2.1 *Explicit Methods*

The simplest method of integration is the explicit Euler's method. Time is discretised into slices of length h. The Fig. 3 shows the geometrical interpretation of the explicit Euler integration scheme. The solution approximation at time $t + h$ is given by the tangent of the solution at time t.

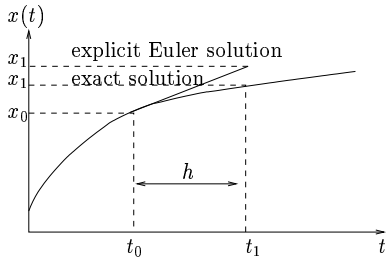


Figure 3: Geometrical interpretation of the explicit Euler integration scheme

The formulation of position x and velocity v at time t_0 and $t + h$ is given by:

$$\begin{cases} x'(t_0) = x'_0 \\ x(t_0) = x_0 \end{cases}, \quad \begin{cases} x'(t+h) = x'(t) + hx''(t+h) \\ x(t+h) = x(t) + hx'(t+h) \end{cases}$$

To obtain a stable simulation using this integration method, we have to limit the step size h. Indeed if we increase the springs damping and stiffness constants, we have to reduce h to maintain stability.

The leap-frog method is another explicit integration method which is time reversible because of the symmetric way in which it is defined, allowing energy conservation. This method is only second-order accurate, but it has been shown experimentally stable. It uses position x at time t and velocity x' at time $t - \frac{h}{2}$:

$$\begin{cases} x''(t_0) &= x''_0 \\ x'(t_0) &= x'_0 \\ x'(\frac{h}{2}) &= x'_0 + \frac{h}{2}x''_0 \\ x(t_0) &= x_0 \end{cases}, \quad \begin{cases} x''(t) &= \vec{f}(t)/m & (3) \\ x'(t+\frac{h}{2}) &= x'(t-\frac{h}{2}) + hx''(t) & (4) \\ x(t+h) &= x(t) + hx'(t+\frac{h}{2}) & (5) \end{cases}$$

3.2.2 *Implicit Euler's Method*

Baraff and Witkin [11] have shown that implicit integration methods allow the use of large time steps in cloth simulation without loss of stability. We define $\Delta x = x(t+h) - x(t)$,$\Delta x' = x'(t+h) - x'(t)$. By applying a Taylor series dynamical equation (1) becomes:

$$\left(M - h\frac{\partial f}{\partial x'} - h^2\frac{\partial f}{\partial x}\right)\Delta x' = hf(t) + h^2\frac{\partial f}{\partial x}x'(t)$$

which we have to solve to obtain $\Delta x'$ and then easily compute $x'(t+h) = x'(t) + \Delta x'$ and $x(t+h) = x(t) + hx'(t+h)$. To sum up, to use implicit Euler's method we have to: (a) value $f(t)$, (b) value $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial x'}$, (c) to build a linear sparse system, (d) to solve it in order to compute $\Delta x'$, (e) and then to update x and x'.

To solve the linear system, we apply the conjugate gradient method [11, 15] which easily uses the sparsity of the matrix, since it only addresses the matrix through its product with a vector. A complete description of our parallelisation of the conjugate gradient method is presented in [6]. In this paper, we focus essentially on explicit integration parallelisation. But we can note that explicit and implicit methods parallelisation have the same schemes parallelisation.

4 Parallel Cloth Simulation

In this part, we explain how we exploit the parallel programming environment Athapascan.

4.1 Related Works

Romero and Zapata [16] have detailed a solution for cloth and other non-rigid solid simulations on parallel computers. They have developed an application, which combines data parallelism with task parallelism. Inside an object, the redistribution and reordering of elements among the assigned processor, have been performed using domain decomposition methods. The preconditioned conjugate gradient algorithm has been parallelised following a strategy in which the successive parts of the vectors and the properly aligned rows of the matrices are distributed among the processors. Computations inside a processor have been performed using sequential BLAS libraries. They present results on 8 processors with 3,520 particles: computation time for this simulation is about $3s$ per step using implicit method with collision treatment.

4.2 Parallelisation with Athapascan

From the point of view of an Athapascan developer, the parallelism is expressed with a set of tasks interacting through a global memory [1]. The developer has a total control of the computation tasks and of the communicated objects granularity (shared variables). Programs are written in C++ using the following syntax. A shared variable v of type T will be declared $Shared < T > v$. A task is created using statement $Fork < Task > (param)$, where $Task$ is a structure which operator "()" implements the computing code and takes $param$ as parameters. Shared variables are passed with access rights information. For instance a read-only parameter is declared $Shared_r < p >$, a write-only parameter is declared $Shared_w < p >$ and a read-write parameter is declared $Shared_rw < p >$.

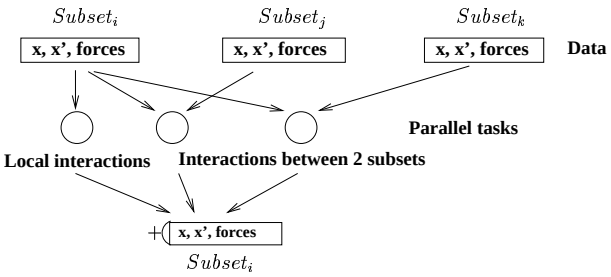


Figure 4: Forces computation task (2)

In our parallel cloth simulation based on particle decomposition, shared data are the subsets of particles resulting from the decomposition (Fig. 1.b) and tasks exist in two different kinds. The first kind of task compute particle states of an entire subset according to the integration schemes (position (5), velocity (4), acceleration (3)). Since these tasks do not need information from other particles subsets, they have no dependencies with others tasks. Hence, they do not require inter-node communication and can be computed efficiently in parallel.

The second kind of tasks calculate interactions between two adjacent subsets of particles (Fig. 4). Two subsets are said adjacent if there is at least one particle of one subset connected with another particle of the other subset (a subset being adjacent with itself). We create a task for each couple of adjacent subsets. This task is executed on one of the nodes where are placed the subsets. Hence, the computation of interactions between adjacent subsets placed on the same node (which includes intra-subset interactions) do not generate communication. However, for adjacent subsets placed on two different nodes, the computation first requires the communication of the positions of the distant subset before the forces can be computed. Note however that for efficiency reasons only the positions of the particles constituting the border are communicated. The main difference between the parallelisation of an explicit integration method and an implicit one comes from this second kind of tasks which are more frequent in implicit than in explicit.

Thanks to Athapascan, the code for computing the interactions between two subsets does not depend on the placement of those subsets. Tasks programmers just do not have to consider where the data are actually placed. They take in parameters the positions of the two subsets, the list of interacting particles and a reference to the resulting forces. The positions are passed as read-only shared variables (Shared_r), whereas the forces are passed with "cumulative write" (Shared_cw) access rights.

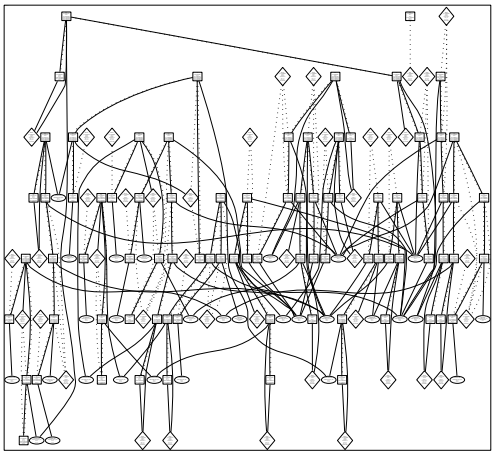


Figure 5: Data-flow graph built at runtime

Cumulative writes are very useful for associative and commutative operations which is the case here since the force applied to a particle is the sum of the elementary interaction forces with its connected particles. Elementary forces can then be computed in parallel by the above mentioned tasks, potentially on different nodes, and are automatically added by the Athapascan runtime. Without this facility, computing the sum would be particularly inefficient since it would lead to a lot of dependencies.

At runtime, Athapascan builds the data-flow graph that expresses the accesses into the global memory made by the tasks. This graph is built by interpreting shared and task statements [17]. For example, Fig. 5 shows the graph built for an execution of the application for four subsets of particles. Rectangles, lozenges and circles represent respectively "accesses to shared data", "shared data" and "tasks".

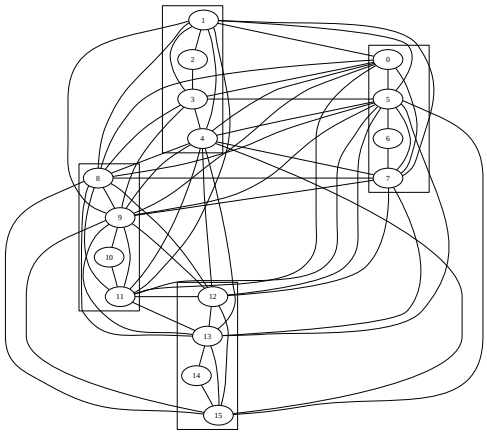


Figure 6: Dependencies graph with its partitioning generated by Athapascan

Given this data-flow graph, Athapascan generates the dependencies graph partitioned onto processors. A scheduling algorithm is then applied to map the tasks and the data onto the processors and the memory locations according to information given in our implementation. Fig. 6 shows the dependencies graph and its partition onto 4 processors (16 tasks mapped on 4 processors).

4.3 Experimentations on a PCs Cluster

Leapfrog and implicit Euler's integration methods have been implemented and tested on a cluster of PCs composed of 190 mono-processor Pentium III at 733MHz with 256M-Bytes of main memory interconnected with a switched 100M-bit/s network. Computation times are shown in Fig. 7.

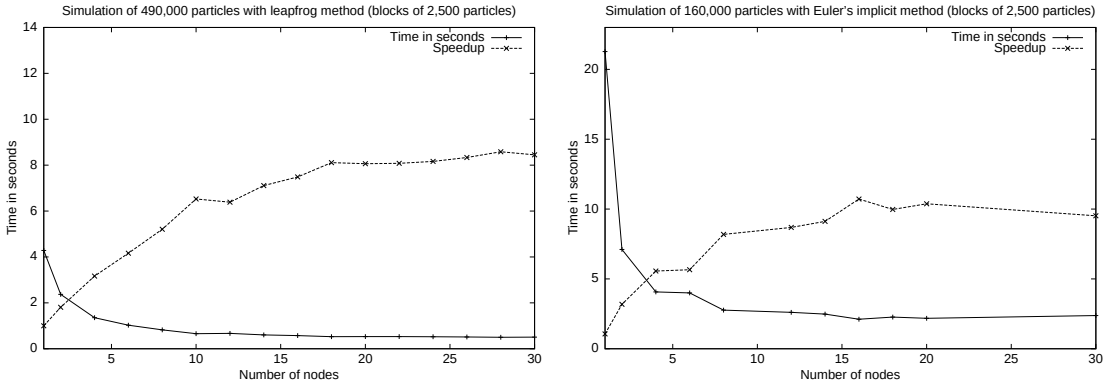


Figure 7: Performances obtained for 490,000 particles using the leapfrog integration method (left) and for 160,000 particles using the implicit Euler's integration method (right) for one iteration with a size of subsets of 2,500 particles

The sequential time for the leapfrog integration method is about $4.28s$. The parallel time on two nodes is $2.36s$ (speedup of 1.81) and $0.57s$ on sixteen nodes (speedup of 7.51). For the implicit method, we surprisingly obtain a speedup of around three on two nodes ($21.27s$ is sequential and $7.10s$ on two nodes) which can be explained by the large amount of memory needed by the tasks which exceed the available physical memory of a single node.

As we can see, for both methods, the computation time decreases significantly up to sixteen processors before the gain begins to stagnate. As the calculation time decreases, the

overhead due to the runtime becomes more and more significant. This overhead includes the computation of the data-flow graph, the mapping of data and tasks on processors and the communication of shared data. To estimate this overhead, we run the same application with empty tasks (the tasks are executed, the data are transfered, but no computation is performed). The difference shows a near to optimal speedup which means that the tasks are well distributed. Hence, to obtain better speedup, some improvements are required to diminish this overhead. A new version of Athapascan, which is being developed, should soon give us even better results. Among other, the construction of the data-flow graph and the mapping onto the processors will be done only once. This is possible in our case since it is the same at each iteration. We can also expect better speedups for larger clothes on more powerful clusters, which we could not experiment on our cluster due to memory constraints.

5 Improving Reactivity by a Parallel Visualisation

For the rendering of our cloth simulation, we use the Net Juggler library [18, 3]. Net Juggler is based on VR Juggler [19], a platform for virtual reality applications. It enables an application to use the power of multiple graphics boards distributed on several PCs. It parallelises graphics rendering computations for multi-display environments by replicating the rendering program on each node and using MPI (Message Passing Interface) to ensure that copies are consistent and displayed images are synchronised.

In order to combine our parallel simulation with parallel rendering, we have made

two programs [20, 21]. The first one computes positions of particles using the parallel programming environment Athapascan (explained in section 4.2). The second program makes the rendering of the cloth using Net Juggler. These two programs communicate together using a TCP connection. Each process of the first program sends its computed positions to a specific computer of the graphic cluster and then this computer broadcasts positions to others. The Fig. 8 shows this coupling.

To obtain real time animation (Fig. 9) , we have to manage that these two parallel programs have different execution times: simulation iteration step, integration method step and display frequency. Consequently, positions computed by the parallel simulation are only sent to the visualisation for multiple iteration of $0.04s$. The model built is then completely asynchronous.

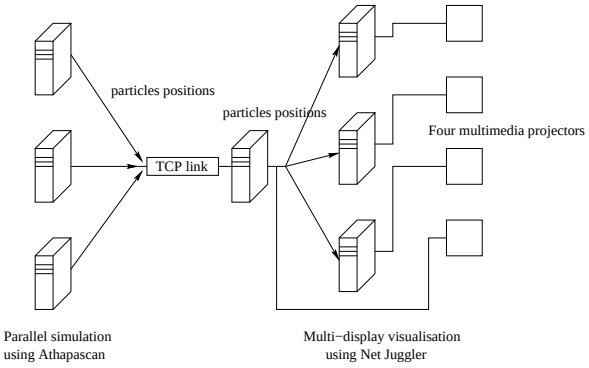


Figure 8: Coupling the parallel simulation with a multi-display visualisation.

6 Conclusion

We have presented the problem of parallelising the simulation of deformable objects using physical models. Different integration methods for the equations of motion have been presented. Two of them, namely the leapfrog and the Euler's implicit methods have been implemented with a parallelisation based on a particle decomposition. This kind of decomposition enables to obtain the same computation load on each processors of the cluster.

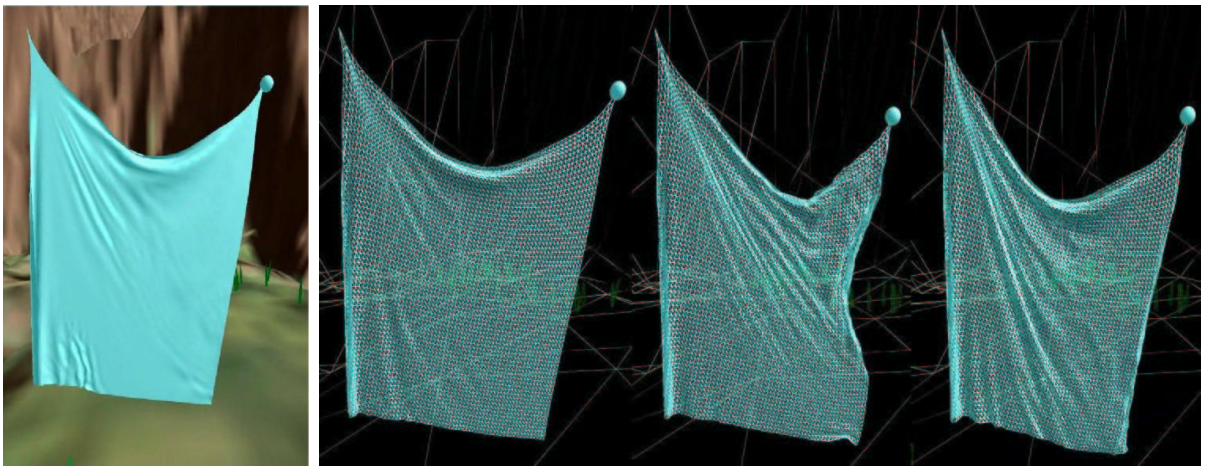


Figure 9: Pieces of cloth of 490 particles

We wrote the application using the Athapascan programming language which relieves us from the management of data distribution and tasks synchronisation. Indeed Athapascan is a parallel programming environment suited to this kind of application. It offers facilities to adapt the scheduling strategy to the specificities of the applications and the target architecture. Furthermore the high level programming interface helps to implement applications in an easier way than with communication libraries like MPI. We obtain good speedups up to sixteen processors.

We also described how we couple this parallel simulation with the Net Juggler multi-display visualisation system. It enables our application to use the power of multiple graphics cards distributed on different processors of a cluster. The model built is totally asynchronous. Any synchronisation is made between the simulation and the visualisation. Particles positions are sent so as to ensure real time.

References

[1] J.-L. Roch, T. Gautier, and R. Revire. Athapascan: Api for asynchronous parallel programming user's guide. Technical Report RR-0276, INRIA Rhône-Alpes, projet APACHE, February 2003.

[2] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. The Cave Audio VIsual Experience Automatic Virtual Environement. *Communication of the ACM*, 35(6):64–72, 1992.

[3] J. Allard, V. Gouranton, L. Lecointre, E. Melin, and B. Raffin. Net juggler: Running VR juggler with multiple displays on a commodity component cluster. In *IEEE VR*, pages 275–276, Orlando, USA, March 2002.

[4] R. K. Brunner, J. C. Phillips, and L. V. Kale. Scalable molecular dynamics for large biomolecular systems. In *Proceedings of Supercomputing (SC) 2000*, Dallas, Texas, USA, November 2000.

[5] P-E. Bernard, T. Gautier, and D. Trystram. Large scale simulation of parallel molecular dynamics. In *Proceedings of Second Merged Symposium IPPS/SPDP 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing*, San Juan, Puerto Rico, April 1999.

[6] F. Zara, F. Faure, and J-M. Vincent. Physical cloth simulation on a PC cluster. In *Proceedings of the Fourth Eurographics Workshop on Parallel Graphics and Visualization*, pages 105–112, Blaubeuren, Germany, September 2002. Eurographics Association.

[7] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable

models: Recovering 3d shape and nonrigid motion. *Artificial Intelligence*, 36:91–123, November 1988.

[8] D. Breen, D. House, and P. Getto. A particle-based model for simulating the draping behavior of woven cloth. *Textile Research Journal, Vol. 64*, 11:663–685, November 1994.

[9] J. Louchet, X. Provot, and D. Crochemore. Evolutionary identification of cloth animation models. In Dimitri Terzopoulos and Daniel Thalmann, editors, *Computer Animation and Simulation'95*, pages 44–54. Springer-Verlag, 1995.

[10] B. Eberhardt, A. Weber, and W. Strasser. A fast, flexible, particle-system model for cloth. *IEEE Computer Graphics and Applications*, 16:52–59, 1996.

[11] D. Baraff and A. Witkin. Large Steps in Cloth Simulation. In Michael Cohen, editor, *SIGGRAPH'98 Conference Proceedings*, Annual Conference Series, pages 43–54, USA, 1998. ACM SIGGRAPH, Addison Wesley.

[12] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface '95*, pages 147–154, 1995.

[13] M. Hauth, O. Etzmuss, B. Eberhardt, R. Klein, R. Sarlette, M. Sattler, K. Daubert, and J. Kautz. Cloth animation and rendering. In *EUROGRAPHICS 2002*, Saarbrucken, Germany, September 2002. The Eurographics Association. Tutorial.

[14] M. Hauth and O. Etzmuss. A high performance solver for the animation of deformable object using advanced numerical methods. In *European Associ-*

ation for Computer Graphics (EUROGRAPHICS'2001), Manchester, UK, September 2001. ACM.

[15] P. Volino and N. Magnenat-Thalmann. Implementing Fast Cloth Simulation with Collision Response. In *CGI'00 Computer Graphics International*, Geneva, June 2000.

[16] S. Romero, L. F. Romero, and E. L. Zapata. Fast Cloth Simulation with Parallel Computers. In *Proceedings of the 6th International Euro-Par Conference on Parallel Processing*, volume 1900 of *Lecture Notes in Computer Science*, pages 491–499, Munich, Germany, August/September 2000. Springer-Verlag Heidelberg.

[17] F. Galilée, J.-L. Roch, G. Cavalheiro, and M. Doreille. Athapascan-1: Online building data flow graph in a parallel language. In IEEE, editor, *Pact'98*, Paris, France, oct 1998.

[18] J. Allard, L. Lecointre, V. Gouranton, E. Melin, and B. Raffin. Net Juggler Guide. Technical Report RR-LIFO-2001-02, LIFO, Orléans, France, June 2001.

[19] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira. Vr juggler: A virtual platform for virtual reality application development. In *IEEE VR 2001*, Yokohama, Japan, March 2001.

[20] F. Zara, J-M. Vincent, and F. Faure. Coupling Parallel Simulation and Parallel Visualization on PC Clusters. In *Commodity Cluster for Virtual Reality 2003, VR 2003 Workshop*, Los Angeles, USA, March 2003.

[21] J. Allard, B. Raffin, and F. Zara. Cou-

pling Parallel Simulation and Multi-Display Visualization on a PC Cluster. In *International Conference on Parallel and Distributed, Euro-Par 2003*, Klagenfurt, Austria, August 2003.

Bibliographies



Florence Zara is a Ph.D. student in Computer Science at ID-IMAG ("Informatique et Distribution") laboratory, Polytechnic National Institute of Grenoble, France. Her research interests are about parallel physically-based animation of complex scenes on cluster.

François Faure completed a Ph.D. thesis in Computer Science at University Joseph Fourier, Grenoble, France in 1997. He previously studied Mechanical Engineering at the Ecole Normale Suprieure de Cachan. His research interests are about physically-based animation of complex scenes. Francois is an Assistant Professor at GRAVIR computer graphics lab, Joseph Fourier University.





Jean-Marc Vincent is assistant professor in computer science at University of Grenoble. He received the aggregation in mathematics in 1986 and a thesis is Computer Science in 1990 from University of Paris XI

(FRANCE). In 1991, he joined the INRIA-IMAG APACHE project at the University of Grenoble in the performance evaluation group. He is a member of the "Informatique et Distribution" laboratory.

His research interests concerns stochastic modelling and analysis of massively parallel or distributed computer systems. It includes : fundamental studies on dynamics of stochastic discrete event systems (Markovian models, (max,+)-algebra, stochastic ordering); software simulation techniques with application to high speed networks (rare event estimation, quality of service...); measurement software tools that provide aggregated or disaggregated informations on the behaviour of parallel or distributed program executions (software tracers, statistical on-line analysers...).

The application fields of such researches concerns scientific computations, parallel efficient algorithms, with a specific focus on simulation of complex systems such as physical systems.