

Efficient and Easy Parallel Implementation of Large Numerical Simulations

Florence Zara, Thierry Gautier, Rémi Revire

▶ **To cite this version:**

Florence Zara, Thierry Gautier, Rémi Revire. Efficient and Easy Parallel Implementation of Large Numerical Simulations. 10th EuroPVM (Parallel Virtual Machine)/MPI (Message Passing Interface), 2003, Venice, Italy. pp.663-666, 10.1007/978-3-540-39924-7_90 . inria-00537469

HAL Id: inria-00537469

https://inria.hal.science/inria-00537469

Submitted on 18 Nov 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.

Efficient and Easy Parallel Implementation of Large Numerical Simulations

Rémi Revire, Florence Zara, and Thierry Gautier
Projet APACHE* ID-IMAG, Antenne ENSIMAG, ZIRST
51, av. J. Kuntzmann, F38330 Montbonnot Saint Martin, France.
{remi.revire, florence.zara, thierry.gautier}@imag.fr

Abstract. This paper presents an efficient implementation of two large numerical simulations using a parallel programming environment called Athapascan. This library eases parallel implementations by managing communications and synchronisations. It provides facilities to adapt the schedule to efficiently map the application on the target architecture.

Keywords. Parallel Molecular Dynamics - Parallel Cloth Simulation - Parallel Programming Environment - Scheduling.

1 Introduction

In the past decade, computer science applications in simulation have become a key point for parallel programming environments. Many of these applications have been developed using MPI (Message Passing Interface) or PVM (Parallel Virtual Machine). Even if these low level libraries are efficient for fine grained applications, writing and debugging them remain difficult. Other works such as NAMD [3], a molecular dynamic simulation, implemented with an higher level programming library (Charm) have shown good speed-up. The Charm library integrates load balancing strategies within the runtime system. Despite the higher level of the programming interface, it does not provide an uniform portability across different parallel architectures.

In this paper, we present the implementation of a cloth simulation and a molecular dynamic simulation using Athapascan. This library is well suited for this kind of application for two main reasons. It eases applications implementation. For instance, an MPI version of our molecular dynamic application is about 100,000 lines of code while the Athapascan [7, 4] version is about 10,000 lines of code. Next, performances portability is made possible by adapting the scheduling algorithm to the specificities of the application and the target architecture.

The next section presents the two numerical simulations. Section 3 describes an overview of Athapascan and section 4 gives some implantation details and experimental results.

* Project funded by CNRS, INPG, INRIA, UJF. SAPPE was partly financed by the Rhône-Alpes region. This work is made in collaboration with François Faure (GRAVIR-IMAG), Jean-Louis Roch and Jean-Marc Vincent (ID-IMAG APACHE project).

2 Parallel Algorithms of two Numerical Simulations

The first numerical simulation, TUKTUT [2] is a molecular dynamics simulation. It aims at emulating the dynamic behaviour of multiple-particle systems to study mechanical and structural properties of proteins and other biological molecules. The second one, called Sappe [9] is a cloth simulation [1]. It provides a 3D and realistic modelling of dressed humans in real time. SAPPE is based on a physical model: a cloth is represented as a triangular mesh of particles linked up by springs emulate the material properties. The mesh topology describes how particles interact and exert forces on each other.

The loop iteration of each simulation is composed of two main parts: (1) Computation of forces that act on each particle or atom; (2) Computation of each particle or atom states (acceleration, velocity, position) by integrating the dynamic equations of the system. The two simulations differ only by the nature of the forces.

To design parallel algorithms for these two simulations, computations are partitioned in a set of tasks. Two techniques are used to obtain these partitions: a particle decomposition for SAPPE [9] and a domain decomposition for TUKTUT [2]. A particle decomposition consists in splitting the set of particles in several subsets, while a domain decomposition consists in splitting the simulation space. With both applications, the numerous interactions between particles leads to many data dependencies between computation tasks. Consequently, the distribution of tasks among distant processors induces non-trivial communication patterns. Tasks scheduling is thus a key point for performance.

3 Advantages of Athapascan for a Parallel Programming

This section is dedicated to the description of the Athapascan library. The programming interface is first briefly described before to present functionalities to handle the scheduling algorithm.

3.1 Overview of the Programming Interface

Parallelism is expressed using remote asynchronous procedure calls that create objects named *tasks*. Tasks communicate with each others using a virtual shared memory and synchronisations are deduced from the type of access made by the tasks on shared objects (read access, write access). The simplicity of Athapascan is mainly due to its sequential semantics: each read operation on a shared object returns the last value written as defined by the sequential order of the execution [4].

Athapascan programs are described independently of the target architecture and the chosen scheduling algorithm. Thus, the programmer can focus on algorithms letting the Athapascan runtime managing synchronisations and communications between tasks. Moreover the programming interface, described in [7], relies on two keywords only.

3.2 Scheduling Facilities for Efficient Executions

In order to get efficient executions, the scheduling should be adapted to the target application and architecture. The programmer can use general algorithms

already implemented or design its own specific scheduling strategy. These algorithms can take advantage of some specific scheduling attributes associated to tasks and shared data.

In the context of our numerical simulations, several scheduling algorithms have been implemented. In these algorithms, shared data are first distributed among execution nodes. Tasks are scheduled according to the mapping of their parameters using the Owner Compute Rule as for HPF Compiler [5]. We distinguish three strategies to distribute data. In the first one, data are Cyclically distributed (Cyclic). In the second one, the set of shared objects is recursively partitioned according to 3D position and estimated costs (attributes of shared objects). This strategy is called Orthogonal Recursive Bisection (ORB). In the third one, a data dependency graph is built and the Scotch partition library [6] is used to compute the mapping. These strategies only give the shared and data mapping. The execution according to this mapping is fully handled by Athapascan runtime.

4 Implementation of Applications and Results

The implementation of both applications in Athapascan is intuitive. We first declare a set of Athapascan shared objects representing either a geometric region of space and the associated atoms (TUKTUT), or a set of particles (SAPPE). Then, Athapascan tasks are iteratively created to compute forces between atoms or particles of each shared object. Finally the program creates tasks to integrate positions and velocities. This kind of shared decomposition makes the granularity of tasks easily adaptable by increasing or decreasing the number of atoms or particles encapsulated in shared objects. At runtime, Athapascan manages synchronisations and communications between tasks according to the schedule strategy.

SAPPE has been implemented and tested on a cluster of PCs composed of 120 mono-processor Pentium III running at 733MHz, with 256MBytes of main memory and interconnected through a switched 100Mbit/s network. The left part of figure 1 presents execution time for one iteration of the cloth simulation. Performances are obtained with a cyclic data mapping. We simulate system of one million of particles with a good speedup. Notice that Romero and Zapata have presented results of a parallel cloth simulation with collision detection for 3,520 particles on 8 processors [8] in 2000. This is three orders-of-magnitude less than our simulation size. Although they perform collision detection, we guess that with a similar detection we still be able to compute large simulation.

TUKTUT has been tested on a cluster of 10 SMPs dual Pentium III processors running at 866MHz with 512MBytes of main memory and interconnected through a switched Ethernet 100Mbit/s network. We report experiments using two molecular structures on figure 1. The first one has 11,615 atoms (called GPIK). The second one (an hydrated β-galactosidase, called BGLA) has 413,039 atoms. In these experiences, we compare two scheduling strategies, ORB and Scotch, on an average of ten iterations. We obtain a significant speed-up for each strategy. Also notice that the chosen scheduling has an important impact on the results that shows the importance of scheduling facilities of Athapascan.

# particles	#nodes	speedup
490,000	2	1.81
490,000	4	3.17
490,000	6	4.19
490,000	8	5.35
1,000,000	2	1.75
1,000,000	4	2.65
1,000,000	6	3.29

BGLA (413,039 atoms)

#nodes	ORB	Scotch
2	3.09	2.91
4	6.35	6.12
8	8.1	8.92

GPIK (11,615 atoms)

#nodes	ORB	Scotch
2	3.83	2.7
4	3.83	3.53
8	4.18	5.11

Fig. 1. SAPPE speed-up (left) on a cluster of PCs and TUKTUT speed-up (right) on a cluster of SMPs.

5 Conclusion

We have presented an efficient fine grain implementation of two numerical simulations (SAPPE and TUKTUT) with Athapascan. Experimental results confirm that the choice of a scheduling strategy is a key point for high performance. Athapascan is a parallel programming environment suited to this kind of applications. It offer 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 MPI or PVM.

References

1. D. Baraff and A. Witkin. Large steps in cloth simulation. In *Computer Graphics Proceedings, Annual Conference Series*, pages 43–54. SIGGRAPH, 1998.
2. P.-E. Bernard, T. Gautier, and D. Trystram. Large scale simulation of parallel molecular dynamics. In *Proceedings of Second Merged Symposium IPPS/SPDP*, San Juan, Puerto Rico, April 1999.
3. R.K. Brunner, J.C. Phillips, and Kale L.V. Scalable Molecular Dynamics for Large Biomolecular Systems. In *Proceedings of Supercomputing (SC) 2000, Dallas, TX*, November 2000.
4. F. Galilée, J.-L. Roch, G. Cavalheiro, and M. Doreille. Athapascan-1: On-line building data flow graph in a parallel language. In IEEE, editor, *Pact'98*, pages 88–95, Paris, France, October 1998.
5. High Performance Fortran Forum. High Performance Fortran language specification, version 1.0. Technical Report CRPC-TR92225, Houston, Tex., 1993.
6. F. Pellegrini and J. Roman. Experimental analysis of the dual recursive bipartitioning algorithm for static mapping. Technical Report 1038-96, 1996.
7. J.-L. Roch and *et al.* Athapascan: Api for asynchronous parallel programming. Technical Report RR-0276, INRIA Rhône-Alpes, projet APACHE, February 2003.
8. S. Romero, L.F. Romero, and E.L. Zapata. Fast cloth simulation with parallel computers. In *Euro-Par 2000*, pages 491–499, Munich, August 2000.
9. F. Zara, F. Faure, and J-M. Vincent. Physical cloth simulation on a pc cluster. In X. Pueyo D. Bartz and E. Reinhard, editors, *Fourth Eurographics Workshop on Parallel Graphics and Visualization 2002*, Blaubeuren, Germany, September 2002.